

GigaDevice Semiconductor Inc.

GD32H7xx

Arm[®] Cortex[®]-M7 32-bit MCU

**Firmware Library
User Guide**

Revision 1.4

(Jan. 2025)

Table of Contents

Table of Contents	1
List of Figures	6
List of Tables	7
1. Introduction	51
1.1. Rules of User Manual and Firmware Library	51
1.1.1. Peripherals.....	51
1.1.2. Naming rules.....	53
2. Firmware Library Overview	54
2.1. File Structure of Firmware Library	54
2.1.1. Examples Folder	56
2.1.2. Firmware Folder	56
2.1.3. Template Folder	56
2.1.4. Utilities Folder	58
2.2. File descriptions of Firmware Library	59
3. Firmware Library of Standard Peripherals	60
3.1. Overview of Firmware Library of Standard Peripherals.....	60
3.2. ADC	60
3.2.1. Descriptions of Peripheral registers.....	60
3.2.2. Descriptions of Peripheral functions	61
3.3. CAN	100
3.3.1. Descriptions of Peripheral registers.....	100
3.3.2. Descriptions of Peripheral functions	101
3.4. CAU	146
3.4.1. Descriptions of Peripheral registers.....	146
3.4.2. Descriptions of Peripheral functions	146
3.5. CMP	173
3.5.1. Descriptions of Peripheral registers.....	174
3.5.2. Descriptions of Peripheral functions	174
3.6. CPDM.....	188
3.6.1. Descriptions of Peripheral registers.....	188
3.6.2. Descriptions of Peripheral functions	189
3.7. CRC	194
3.7.1. Descriptions of Peripheral registers.....	195

3.7.2.	Descriptions of Peripheral functions	195
3.8.	CTC	203
3.8.1.	Descriptions of Peripheral registers	203
3.8.2.	Descriptions of Peripheral functions	203
3.9.	DAC	216
3.9.1.	Peripheral register description	216
3.9.2.	Descriptions of Peripheral functions	217
3.10.	DBG	237
3.10.1.	Descriptions of Peripheral registers	237
3.10.2.	Descriptions of Peripheral functions	237
3.11.	DCI	243
3.11.1.	Descriptions of Peripheral registers	243
3.11.2.	Descriptions of Peripheral functions	243
3.12.	DMA / DMAMUX	260
3.12.1.	Descriptions of Peripheral registers	261
3.12.2.	Descriptions of Peripheral functions	262
3.13.	EDOUT	318
3.13.1.	Descriptions of Peripheral registers	318
3.13.2.	Descriptions of Peripheral functions	319
3.14.	EFUSE	325
3.14.1.	Descriptions of Peripheral registers	325
3.14.2.	Descriptions of Peripheral functions	326
3.15.	ENET	338
3.15.1.	Descriptions of Peripheral registers	339
3.15.2.	Descriptions of Peripheral functions	341
3.16.	EXMC	437
3.16.1.	Descriptions of Peripheral registers	437
3.16.2.	Descriptions of Peripheral functions	437
3.17.	EXTI	463
3.17.1.	Descriptions of Peripheral registers	464
3.17.2.	Descriptions of Peripheral functions	464
3.18.	FAC	472
3.18.1.	Descriptions of Peripheral registers	472
3.18.2.	Descriptions of Peripheral functions	472
3.19.	FMC	491
3.19.1.	Descriptions of Peripheral registers	491
3.19.2.	Descriptions of Peripheral functions	492
3.20.	FWDGT	528
3.20.1.	Descriptions of Peripheral registers	528

3.20.2.	Descriptions of Peripheral functions	529
3.21.	GPIO	534
3.21.1.	Descriptions of Peripheral registers	534
3.21.2.	Descriptions of Peripheral functions	534
3.22.	HAU	546
3.22.1.	Descriptions of Peripheral registers	546
3.22.2.	Descriptions of Peripheral functions	547
3.23.	HPDF	567
3.23.1.	Descriptions of Peripheral registers	567
3.23.2.	Descriptions of Peripheral functions	567
3.24.	HWSEM	626
3.24.1.	Descriptions of Peripheral registers	626
3.24.2.	Descriptions of Peripheral functions	626
3.25.	I2C	636
3.25.1.	Descriptions of Peripheral registers	636
3.25.2.	Descriptions of Peripheral functions	637
3.26.	IPA.....	676
3.26.1.	Descriptions of Peripheral registers	676
3.26.2.	Descriptions of Peripheral functions	677
3.27.	LPDTS	701
3.27.1.	Descriptions of Peripheral registers	701
3.27.2.	Descriptions of Peripheral functions	701
3.28.	MDIO	711
3.28.1.	Descriptions of Peripheral registers	711
3.28.2.	Descriptions of Peripheral functions	711
3.29.	MDMA.....	725
3.29.1.	Descriptions of Peripheral registers	725
3.29.2.	Descriptions of Peripheral functions	726
3.30.	OSPI	756
3.30.1.	Descriptions of Peripheral registers	756
3.30.2.	Descriptions of Peripheral functions	757
3.31.	OSPIM	797
3.31.1.	Descriptions of Peripheral registers	797
3.31.2.	Descriptions of Peripheral functions	797
3.32.	MISC.....	804
3.32.1.	Descriptions of Peripheral registers	804
3.32.2.	Descriptions of Peripheral functions	805
3.33.	PMU.....	816
3.33.1.	Descriptions of Peripheral registers	816

3.33.2.	Descriptions of Peripheral functions	817
3.34.	RAMECCMU	835
3.34.1.	Descriptions of Peripheral registers	836
3.34.2.	Descriptions of Peripheral functions	836
3.35.	RCU	850
3.35.1.	Descriptions of Peripheral registers	850
3.35.2.	Descriptions of Peripheral functions	852
3.36.	RSPDIF	912
3.36.1.	Descriptions of Peripheral registers	912
3.36.2.	Descriptions of Peripheral functions	913
3.37.	RTC	930
3.37.1.	Descriptions of Peripheral registers	930
3.37.2.	Descriptions of Peripheral functions	931
3.38.	RTDEC	957
3.38.1.	Descriptions of Peripheral registers	957
3.38.2.	Descriptions of Peripheral functions	958
3.39.	SAI	969
3.39.1.	Descriptions of Peripheral registers	969
3.39.2.	Descriptions of Peripheral functions	970
3.40.	SDIO	997
3.40.1.	Descriptions of Peripheral registers	998
3.40.2.	Descriptions of Peripheral functions	998
3.41.	SPI	1040
3.41.1.	Descriptions of Peripheral registers	1040
3.41.2.	Descriptions of Peripheral functions	1041
3.42.	SYSCFG	1085
3.42.1.	Descriptions of Peripheral registers	1085
3.42.2.	Descriptions of Peripheral functions	1086
3.43.	TIMER	1108
3.43.1.	Descriptions of Peripheral registers	1109
3.43.2.	Descriptions of Peripheral functions	1110
3.44.	TLI	1199
3.44.1.	Descriptions of Peripheral registers	1200
3.44.2.	Descriptions of Peripheral functions	1200
3.45.	TMU	1219
3.45.1.	Descriptions of Peripheral registers	1219
3.45.2.	Descriptions of Peripheral functions	1219
3.46.	TRIGSEL	1228
3.46.1.	Descriptions of Peripheral registers	1228

3.46.2.	Descriptions of Peripheral functions	1230
3.47.	TRNG.....	1238
3.47.1.	Descriptions of Peripheral registers	1238
3.47.2.	Descriptions of Peripheral functions	1239
3.48.	USART.....	1259
3.48.1.	Descriptions of Peripheral registers	1260
3.48.2.	Descriptions of Peripheral functions	1260
3.49.	VREF	1313
3.49.1.	Descriptions of Peripheral registers	1313
3.49.2.	Descriptions of Peripheral functions	1313
3.50.	WWDGT.....	1318
3.50.1.	Descriptions of Peripheral registers	1318
3.50.2.	Descriptions of Peripheral functions	1318
4.	Revision history	1323

List of Figures

Figure 2-1. File structure of firmware library of GD32H7xx	55
Figure 2-2. Select peripheral example files	57
Figure 2-3. Copy the peripheral example files	57
Figure 2-4. Open the project file	57
Figure 2-5. Configure project files	58
Figure 2-6. Compile-debug-download	58

List of Tables

Table 1-1. Peripherals	51
Table 2-1. Function descriptions of Firmware Library	59
Table 3-1. Peripheral function format of Firmware Library	60
Table 3-2. ADC Registers	60
Table 3-3. ADC firmware function	61
Table 3-4. Function adc_deinit	63
Table 3-5. Function adc_clock_config	63
Table 3-6. Function adc_special_function_config	65
Table 3-7. Function adc_data_alignment_config	66
Table 3-8. Function adc_enable	66
Table 3-9. Function adc_disable	67
Table 3-10. Function adc_calibration_mode_config	67
Table 3-11. Function adc_calibration_number	68
Table 3-12. Function adc_calibration_enable	69
Table 3-13. Function adc_resolution_config	69
Table 3-14. Function adc_internal_channel_config	70
Table 3-15. Function adc_dma_mode_enable	71
Table 3-16. Function adc_dma_mode_disable	71
Table 3-17. Function adc_dma_request_after_last_enable	72
Table 3-18. Function adc_dma_request_after_last_disable	72
Table 3-19. Function adc_hpdf_mode_enable	73
Table 3-20. Function adc_hpdf_mode_disable	73
Table 3-21. Function adc_discontinuous_mode_config	74
Table 3-22. Function adc_channel_length_config	75
Table 3-23. Function adc_regular_channel_config	75
Table 3-24. Function adc_inserted_channel_config	76
Table 3-25. Function adc_inserted_channel_offset_config	77
Table 3-26. Function adc_channel_differential_mode_config	78
Table 3-27. Function adc_external_trigger_config	78
Table 3-28. Function adc_software_trigger_enable	79
Table 3-29. Function adc_end_of_conversion_config	80
Table 3-30. Function adc_regular_data_read	81
Table 3-31. Function adc_inserted_data_read	81
Table 3-32. Function adc_watchdog0_single_channel_enable	82
Table 3-33. Function adc_watchdog0_group_channel_enable	83
Table 3-34. Function adc_watchdog0_disable	83
Table 3-35. Function adc_watchdog1_channel_config	84
Table 3-36. Function adc_watchdog2_channel_config	85
Table 3-37. Function adc_watchdog1_disable	85
Table 3-38. Function adc_watchdog2_disable	86

Table 3-39. Function <code>adc_watchdog0_threshold_config</code>	86
Table 3-40. Function <code>adc_watchdog1_threshold_config</code>	87
Table 3-41. Function <code>adc_watchdog2_threshold_config</code>	88
Table 3-42. Function <code>adc_oversample_mode_config</code>	88
Table 3-43. Function <code>adc_oversample_mode_enable</code>	90
Table 3-44. Function <code>adc_oversample_mode_disable</code>	90
Table 3-45. Function <code>adc_flag_get</code>	91
Table 3-46. Function <code>adc_flag_clear</code>	92
Table 3-47. Function <code>adc_interrupt_enable</code>	92
Table 3-48. Function <code>adc_interrupt_disable</code>	93
Table 3-49. Function <code>adc_interrupt_flag_get</code>	94
Table 3-50. Function <code>adc_interrupt_flag_clear</code>	95
Table 3-51. Function <code>adc_sync_mode_config</code>	95
Table 3-52. Function <code>adc_interrupt_disable</code>	96
Table 3-53. Function <code>adc_sync_dma_config</code>	97
Table 3-54. Function <code>adc_sync_dma_request_after_last_enable</code>	98
Table 3-55. Function <code>adc_sync_dma_request_after_last_disable</code>	98
Table 3-56. Function <code>adc_sync_master_adc_regular_data0_read</code>	99
Table 3-57. Function <code>adc_sync_slave_adc_regular_data0_read</code>	99
Table 3-58. Function <code>adc_sync_regular_data1_read</code>	100
Table 3-59. CAN Registers	100
Table 3-60. CAN firmware function	102
Table 3-61. Structure <code>can_error_counter_struct</code>	103
Table 3-62. Structure <code>can_parameter_struct</code>	103
Table 3-63. Structure <code>can_mailbox_descriptor_struct</code>	104
Table 3-64. Structure <code>can_rx_fifo_struct</code>	104
Table 3-65. Structure <code>can_fd_parameter_struct</code>	105
Table 3-66. Structure <code>can_rx_fifo_id_filter_struct</code>	105
Table 3-67. Structure <code>can_fifo_parameter_struct</code>	105
Table 3-68. Structure <code>can_pn_mode_filter_struct</code>	105
Table 3-69. Structure <code>can_pn_mode_config_struct</code>	106
Table 3-70. Structure <code>can_crc_struct</code>	106
Table 3-71. Enum <code>can_interrupt_enum</code>	106
Table 3-72. Enum <code>can_flag_enum</code>	108
Table 3-73. Enum <code>can_interrupt_flag_enum</code>	110
Table 3-74. Enum <code>can_operation_modes_enum</code>	112
Table 3-75. Enum <code>can_struct_type_enum</code>	112
Table 3-76. Enum <code>can_error_state_enum</code>	113
Table 3-77. Function <code>can_deinit</code>	113
Table 3-78. Function <code>can_software_reset</code>	114
Table 3-79. Function <code>can_init</code>	114
Table 3-80. Function <code>can_struct_para_init</code>	115
Table 3-81. Function <code>can_private_filter_config</code>	115
Table 3-82. Function <code>can_operation_mode_enter</code>	116

Table 3-83. Function can_operation_mode_get.....	117
Table 3-84. Function can_inactive_mode_exit.....	117
Table 3-85. Function can_pn_mode_exit	118
Table 3-86. Function can_fd_config.....	118
Table 3-87. Function can_bitrate_switch_enable	119
Table 3-88. Function can_bitrate_switch_disable	119
Table 3-89. Function can_tdc_get	120
Table 3-90. Function can_tdc_enable	120
Table 3-91. Function can_tdc_disable	121
Table 3-92. Function can_rx_fifo_config	121
Table 3-93. Function can_rx_fifo_filter_table_config.....	122
Table 3-94. Function can_rx_fifo_read	123
Table 3-95. Function can_rx_fifo_filter_matching_number_get	123
Table 3-96. Function can_rx_fifo_clear	124
Table 3-97. Function can_ram_address_get	124
Table 3-98. Function can_mailbox_config	125
Table 3-99. Function can_mailbox_transmit_abort.....	126
Table 3-100. Function can_mailbox_transmit_inactive	126
Table 3-101. Function can_mailbox_receive_data_read	127
Table 3-102. Function can_mailbox_receive_lock.....	128
Table 3-103. Function can_mailbox_receive_unlock	128
Table 3-104. Function can_mailbox_receive_inactive	129
Table 3-105. Function can_mailbox_code_get.....	129
Table 3-106. Function can_error_counter_config	130
Table 3-107. Function can_error_counter_get	130
Table 3-108. Function can_error_state_get	131
Table 3-109. Function can_crc_get	132
Table 3-110. Function can_pn_mode_config.....	132
Table 3-111. Function can_pn_mode_filter_config.....	133
Table 3-112. Function can_pn_mode_num_of_match_get	134
Table 3-113. Function can_pn_mode_data_read	134
Table 3-114. Function can_self_reception_enable	135
Table 3-115. Function can_self_reception_disable	135
Table 3-116. Function can_transmit_abort_enable	136
Table 3-117. Function can_transmit_abort_disable	136
Table 3-118. Function can_auto_busoff_recovery_enable	137
Table 3-119. Function can_auto_busoff_recovery_disable	137
Table 3-120. Function can_time_sync_enable	138
Table 3-121. Function can_time_sync_disable	138
Table 3-122. Function can_edge_filter_mode_enable	139
Table 3-123. Function can_edge_filter_mode_disable	139
Table 3-124. Function can_ped_mode_enable.....	140
Table 3-125. Function can_ped_mode_disable.....	140
Table 3-126. Function can_arbitration_delay_bits_config	141

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

Table 3-171. Function cau_des_cbc.....	170
Table 3-172. Function cau_flag_get	171
Table 3-173. Function cau_interrupt_enable	172
Table 3-174. Function cau_interrupt_disable	172
Table 3-175. Function cau_interrupt_flag_get.....	173
Table 3-176. CMP registers	174
Table 3-177. CMP firmware function	174
Table 3-178. Enum cmp_enum	175
Table 3-179. Function cmp_deinit	175
Table 3-180. Function cmp_mode_init.....	175
Table 3-181. Function cmp_noninverting_input_select	177
Table 3-182. Function cmp_output_init.....	177
Table 3-183. Function cmp_output_mux_config	178
Table 3-184. Function cmp_outputblank_init.....	179
Table 3-185. Function cmp_enable.....	180
Table 3-186. Function cmp_disable	180
Table 3-187. Function cmp_window_enable.....	181
Table 3-188. Function cmp_window_disable	181
Table 3-189. Function cmp_lock_enable.....	182
Table 3-190. Function cmp_voltage_scaler_enable	182
Table 3-191. Function cmp_voltage_scaler_disable	183
Table 3-192. Function cmp_scaler_bridge_enable.....	183
Table 3-193. Function cmp_scaler_bridge_disable.....	184
Table 3-194. Function cmp_output_level_get	184
Table 3-195. Function cmp_flag_get.....	185
Table 3-196. Function cmp_flag_clear	185
Table 3-197. Function cmp_interrupt_enable.....	186
Table 3-198. Function cmp_interrupt_disable.....	187
Table 3-199. Function cmp_interrupt_flag_get	187
Table 3-200. Function cmp_interrupt_flag_clear	188
Table 3-201. CPDM Registers	189
Table 3-202. CPDM firmware function	189
Table 3-203. Enum cpdm_output_phase_enum.....	189
Table 3-204. Function cpdm_enable	190
Table 3-205. Function cpdm_disable	191
Table 3-206. Function cpdm_delayline_sample_enable.....	191
Table 3-207. Function cpdm_delayline_sample_disable.....	192
Table 3-208. Function cpdm_output_clock_phase_select.....	192
Table 3-209. Function cpdm_delayline_length_valid_flag_get	193
Table 3-210. Function cpdm_delayline_length_get.....	193
Table 3-211. Function cpdm_clock_output.....	194
Table 3-212. CRC Registers	195
Table 3-213. CRC firmware function	195
Table 3-214. Function crc_deinit	195

Table 3-215. Function <code>crc_reverse_output_data_enable</code>	196
Table 3-216. Function <code>crc_reverse_output_data_disable</code>	196
Table 3-217. Function <code>crc_data_register_reset</code>	197
Table 3-218. Function <code>crc_data_register_read</code>	197
Table 3-219. Function <code>crc_free_data_register_read</code>	198
Table 3-220. Function <code>crc_free_data_register_write</code>	198
Table 3-221. Function <code>crc_init_data_register_write</code>	199
Table 3-222. Function <code>crc_input_data_reverse_config</code>	199
Table 3-223. Function <code>crc_polynomial_size_set</code>	200
Table 3-224. Function <code>crc_polynomial_set</code>	201
Table 3-225. Function <code>crc_single_data_calculate</code>	201
Table 3-226. Function <code>crc_block_data_calculate</code>	202
Table 3-227. CTC Registers	203
Table 3-228. CTC firmware function.....	203
Table 3-229. Function <code>ctc_deinit</code>	204
Table 3-230. Function <code>ctc_counter_enable</code>	204
Table 3-231. Function <code>ctc_counter_disable</code>	205
Table 3-232. Function <code>ctc_irc48m_trim_value_config</code>	205
Table 3-233. Function <code>ctc_software_refsource_pulse_generate</code>	206
Table 3-234. Function <code>ctc_hardware_trim_mode_config</code>	206
Table 3-235. Function <code>ctc_refsource_polarity_config</code>	207
Table 3-236. Function <code>ctc_refsource_signal_select</code>	207
Table 3-237. Function <code>ctc_refsource_prescaler_config</code>	208
Table 3-238. Function <code>ctc_clock_limit_value_config</code>	209
Table 3-239. Function <code>ctc_counter_reload_value_config</code>	209
Table 3-240. Function <code>ctc_counter_capture_value_read</code>	210
Table 3-241. Function <code>ctc_counter_direction_read</code>	210
Table 3-242. Function <code>ctc_counter_reload_value_read</code>	211
Table 3-243. Function <code>ctc_irc48m_trim_value_read</code>	211
Table 3-244. Function <code>ctc_flag_get</code>	212
Table 3-245. Function <code>ctc_flag_clear</code>	213
Table 3-246. Function <code>ctc_interrupt_enable</code>	213
Table 3-247. Function <code>ctc_interrupt_disable</code>	214
Table 3-248. Function <code>ctc_interrupt_flag_get</code>	214
Table 3-249. Function <code>ctc_interrupt_flag_clear</code>	215
Table 3-250. DAC Registers	216
Table 3-251. DAC firmware functions	217
Table 3-252. Function <code>dac_deinit</code>	218
Table 3-253. Function <code>dac_enable</code>	218
Table 3-254. Function <code>dac_disable</code>	219
Table 3-255. Function <code>dac_dma_enable</code>	219
Table 3-256. Function <code>dac_dma_disable</code>	220
Table 3-257. Function <code>dac_mode_config</code>	220
Table 3-258. Function <code>dac_trimming_value_get</code>	221

Table 3-259. Function <code>dac_trimming_value_set</code>	222
Table 3-260. Function <code>dac_trimming_enable</code>	223
Table 3-261. Function <code>dac_output_value_get</code>	223
Table 3-262. Function <code>dac_data_set</code>	224
Table 3-263. Function <code>dac_trigger_enable</code>	225
Table 3-264. Function <code>dac_trigger_disable</code>	225
Table 3-265. Function <code>dac_trigger_source_config</code>	226
Table 3-266. Function <code>dac_software_trigger_enable</code>	226
Table 3-267. Function <code>dac_wave_mode_config</code>	227
Table 3-268. Function <code>dac_lfsr_noise_config</code>	228
Table 3-269. Function <code>dac_triangle_noise_config</code>	228
Table 3-270. Function <code>dac_concurrent_enable</code>	229
Table 3-271. Function <code>dac_concurrent_disable</code>	230
Table 3-272. Function <code>dac_concurrent_software_trigger_enable</code>	230
Table 3-273. Function <code>dac_concurrent_data_set</code>	231
Table 3-274. Function <code>dac_sample_keep_mode_config</code>	231
Table 3-275. Function <code>dac_flag_get</code>	232
Table 3-276. Function <code>dac_flag_clear</code>	233
Table 3-277. Function <code>dac_interrupt_enable</code>	233
Table 3-278. Function <code>dac_interrupt_disable</code>	234
Table 3-279. Function <code>dac_interrupt_flag_get</code>	235
Table 3-280. Function <code>dac_interrupt_flag_clear</code>	235
Table 3-281. DBG Registers.....	237
Table 3-282. DBG firmware function	237
Table 3-283. Enum <code>dbg_periph_enum</code>	237
Table 3-284. Function <code>dbg_deinit</code>	238
Table 3-285. Function <code>dbg_id_get</code>	238
Table 3-286. Function <code>dbg_low_power_enable</code>	239
Table 3-287. Function <code>dbg_low_power_disable</code>	239
Table 3-288. Function <code>dbg_periph_enable</code>	240
Table 3-289. Function <code>dbg_periph_disable</code>	241
Table 3-290. Function <code>dbg_trace_pin_enable</code>	241
Table 3-291. Function <code>dbg_trace_pin_disable</code>	242
Table 3-292. Function <code>dbg_trace_pin_mode_set</code>	242
Table 3-293. DCI Registers.....	243
Table 3-294. DCI firmware function	244
Table 3-295. Structure <code>dc_i_parameter_struct</code>	244
Table 3-296. Function <code>dc_i_deinit</code>	245
Table 3-297. Function <code>dc_i_init</code>	245
Table 3-298. Function <code>dc_i_enable</code>	246
Table 3-299. Function <code>dc_i_disable</code>	247
Table 3-300. Function <code>dc_i_capture_enable</code>	247
Table 3-301. Function <code>dc_i_capture_disable</code>	248
Table 3-302. Function <code>dc_i_external_vsync_enable</code>	248

Table 3-303. Function dci_external_vsync_disable	249
Table 3-304. Function dci_automatic_error_correction_enable	249
Table 3-305. Function dci_automatic_error_correction_disable	250
Table 3-306. Function dci_jpeg_enable	250
Table 3-307. Function dci_jpeg_disable	251
Table 3-308. Function dci_crop_window_enable	251
Table 3-309. Function dci_crop_window_disable	252
Table 3-310. Function dci_crop_window_config	252
Table 3-311. Function dci_embedded_sync_enable	253
Table 3-312. Function dci_embedded_sync_disable	253
Table 3-313. Function dci_ccir_enable	254
Table 3-314. Function dci_ccir_disable	254
Table 3-315. Function dci_ccir_mode_select	255
Table 3-316. Function dci_sync_codes_config	255
Table 3-317. Function dci_sync_codes_unmask_config	256
Table 3-318. Function dci_data_read	257
Table 3-319. Function dci_flag_get	257
Table 3-320. Function dci_interrupt_enable	258
Table 3-321. Function dci_interrupt_disable	258
Table 3-322. Function dci_interrupt_flag_get	259
Table 3-323. Function dci_interrupt_flag_clear	260
Table 3-324. DMA Registers	261
Table 3-325. DMAMUX Registers	261
Table 3-326. DMA firmware function	262
Table 3-327. DMAMUX firmware function	263
Table 3-328. Structure dma_multi_data_parameter_struct	263
Table 3-329. Structure dma_single_data_parameter_struct	264
Table 3-330. Structure dmamux_sync_parameter_struct	264
Table 3-331. Structure dmamux_gen_parameter_struct	265
Table 3-332. Enum dma_channel_enum	265
Table 3-333. Enum dmamux_multiplexer_channel_enum	265
Table 3-334. Enum dmamux_generator_channel_enum	266
Table 3-335. Enum dmamux_interrupt_enum	266
Table 3-336. Enum dmamux_flag_enum	267
Table 3-337. Enum dmamux_interrupt_flag_enum	269
Table 3-338. Function dma_deinit	270
Table 3-339. Function dma_single_data_para_struct_init	270
Table 3-340. Function dma_multi_data_para_struct_init	271
Table 3-341. Function dma_single_data_mode_init	272
Table 3-342. Function dma_multi_data_mode_init	273
Table 3-343. Function dma_periph_address_config	274
Table 3-344. Function dma_memory_address_config	274
Table 3-345. Function dma_transfer_number_config	275
Table 3-346. Function dma_transfer_number_get	276

Table 3-347. Function dma_priority_config	276
Table 3-348. Function dma_memory_burst_beats_config	277
Table 3-349. Function dma_periph_burst_beats_config	278
Table 3-350. Function dma_memory_width_config	279
Table 3-351. Function dma_periph_width_config	280
Table 3-352. Function dma_memory_address_generation_config	280
Table 3-353. Function dma_peripheral_address_generation_config	281
Table 3-354. Function dma_circulation_enable	282
Table 3-355. Function dma_circulation_disable	282
Table 3-356. Function dma_channel_enable	283
Table 3-357. Function dma_channel_disable	284
Table 3-358. Function dma_transfer_direction_config	284
Table 3-359. Function dma_switch_buffer_mode_config	285
Table 3-360. Function dma_using_memory_get	286
Table 3-361. Function dma_switch_buffer_mode_enable	286
Table 3-362. Function dma_switch_buffer_mode_disable	287
Table 3-363. Function dma_fifo_status_get	288
Table 3-364. Function dma_flag_get	288
Table 3-365. Function dma_flag_clear	289
Table 3-366. Function dma_interrupt_enable	290
Table 3-367. Function dma_interrupt_disable	291
Table 3-368. Function dma_interrupt_flag_get	291
Table 3-369. Function dma_interrupt_flag_clear	292
Table 3-370. Function dmamux_sync_struct_para_init	293
Table 3-371. Function dmamux_synchronization_init	294
Table 3-372. Function dmamux_synchronization_enable	294
Table 3-373. Function dmamux_synchronization_disable	295
Table 3-374. Function dmamux_event_generation_enable	295
Table 3-375. Function dmamux_event_generation_disable	296
Table 3-376. Function dmamux_gen_struct_para_init	297
Table 3-377. Function dmamux_request_generator_init	297
Table 3-378. Function dmamux_request_generator_channel_enable	298
Table 3-379. Function dmamux_request_generator_channel_disable	298
Table 3-380. Function dmamux_synchronization_polarity_config	299
Table 3-381. Function dmamux_request_forward_number_config	300
Table 3-382. Function dmamux_sync_id_config	300
Table 3-383. Function dmamux_request_id_config	302
Table 3-384. Function dmamux_trigger_polarity_config	312
Table 3-385. Function dmamux_request_generate_number_config	312
Table 3-386. Function dmamux_trigger_id_config	313
Table 3-387. Function dmamux_flag_get	315
Table 3-388. Function dmamux_flag_clear	316
Table 3-389. Function dmamux_interrupt_enable	316
Table 3-390. Function dmamux_interrupt_disable	317

Table 3-391. Function dmamux_interrupt_flag_get	317
Table 3-392. Function dmamux_interrupt_flag_clear	318
Table 3-393. EDOUT Registers	319
Table 3-394. EDOUT firmware function	319
Table 3-395. Function edout_deinit	319
Table 3-396. Function edout_init	320
Table 3-397. Function edout_enable	320
Table 3-398. Function edout_disable	321
Table 3-399. Function edout_polarity_config	321
Table 3-400. Function edout_max_location_value_config	322
Table 3-401. Function edout_output_counter_update	322
Table 3-402. Function edout_current_location_config	323
Table 3-403. Function edout_current_location_get	323
Table 3-404. Function edout_z_output_mode_config	324
Table 3-405. Function edout_z_output_start_loc_and_width_config	325
Table 3-406. EFUSE Registers	325
Table 3-407. EFUSE firmware function	326
Table 3-408. Enum efuse_system_para_size_enum	326
Table 3-409. Enum efuse_system_para_index_enum	327
Table 3-410. Enum efuse_state_enum	327
Table 3-411. Enum efuse_interrupt_flag_enum	327
Table 3-412. Function efuse_read	327
Table 3-413. Function efuse_program	328
Table 3-414. Function efuse_user_control_write	329
Table 3-415. Function efuse_mcu_reserved_write	329
Table 3-416. Function efuse_dp_write	330
Table 3-417. Function efuse_aes_key_write	331
Table 3-418. Function efuse_user_data_write	331
Table 3-419. Function efuse_aes_key_crc_get	332
Table 3-420. Function efuse_monitor_program_voltage_enable	332
Table 3-421. Function efuse_monitor_program_voltage_disable	333
Table 3-422. Function efuse_monitor_program_voltage_get	333
Table 3-423. Function efuse_ldo_ready_get	334
Table 3-424. Function efuse_flag_get	334
Table 3-425. Function efuse_flag_clear	335
Table 3-426. Function efuse_interrupt_enable	336
Table 3-427. Function efuse_interrupt_disable	336
Table 3-428. Function efuse_interrupt_flag_get	337
Table 3-429. Function efuse_interrupt_flag_clear	338
Table 3-430. ENET Registers	339
Table 3-431. ENET firmware function	341
Table 3-432. Structure enet_initpara_struct	344
Table 3-433. Structure enet_descriptors_struct	345
Table 3-434. Structure enet_ptp_systime_struct	345

Table 3-435. Enum enet_flag_enum	345
Table 3-436. Enum enet_flag_clear_enum	347
Table 3-437. Enum enet_int_enum	348
Table 3-438. Enum enet_int_flag_enum	349
Table 3-439. Enum enet_int_flag_clear_enum	351
Table 3-440. Enum enet_desc_reg_enum	351
Table 3-441. Enum enet_msc_counter_enum	352
Table 3-442. Enum enet_option_enum	352
Table 3-443. Enum enet_mediamode_enum	353
Table 3-444. Enum enet_chksumconf_enum	353
Table 3-445. Enum enet_frmrecept_enum	353
Table 3-446. Enum enet_registers_type_enum	354
Table 3-447. Enum enet_dmadirection_enum	354
Table 3-448. Enum enet_phydirection_enum	354
Table 3-449. Enum enet_regdirection_enum	354
Table 3-450. Enum enet_macaddress_enum	354
Table 3-451. Enum enet_descstate_enum	355
Table 3-452. Enum enet_msc_preset_enum	355
Table 3-453. Function enet_deinit	355
Table 3-454. Function enet_initpara_config	356
Table 3-455. Function enet_init	359
Table 3-456. Function enet_software_reset	361
Table 3-457. Function enet_rxframe_size_get	361
Table 3-458. Function enet_descriptors_chain_init	362
Table 3-459. Function enet_descriptors_ring_init	363
Table 3-460. Function enet_frame_receive	364
Table 3-461. Function enet_frame_transmit	364
Table 3-462. Function enet_transmit_checksum_config	365
Table 3-463. Function enet_enable	366
Table 3-464. Function enet_disable	366
Table 3-465. Function enet_mac_address_set	367
Table 3-466. Function enet_mac_address_get	368
Table 3-467. Function enet_flag_get	369
Table 3-468. Function enet_flag_clear	371
Table 3-469. Function enet_interrupt_enable	372
Table 3-470. Function enet_interrupt_disable	374
Table 3-471. Function enet_interrupt_flag_get	375
Table 3-472. Function enet_interrupt_flag_clear	377
Table 3-473. Function enet_tx_enable	379
Table 3-474. Function enet_tx_disable	379
Table 3-475. Function enet_rx_enable	380
Table 3-476. Function enet_rx_disable	380
Table 3-477. Function enet_registers_get	381
Table 3-478. Function enet_debug_status_get	382

Table 3-479. Function enet_address_filter_enable.....	383
Table 3-480. Function enet_address_filter_disable	384
Table 3-481. Function enet_address_filter_config	384
Table 3-482. Function enet_phy_config	386
Table 3-483. Function enet_phy_write_read.....	386
Table 3-484. Function enet_phyloopback_enable	387
Table 3-485. Function enet_phyloopback_disable	388
Table 3-486. Function enet_forward_feature_enable.....	388
Table 3-487. Function enet_forward_feature_disable.....	389
Table 3-488. Function enet_fliter_feature_enable	390
Table 3-489. Function enet_fliter_feature_disable	391
Table 3-490. Function enet_pauseframe_generate	392
Table 3-491. Function enet_pauseframe_detect_config	392
Table 3-492. Function enet_pauseframe_config.....	393
Table 3-493. Function enet_flowcontrol_threshold_config	394
Table 3-494. Function enet_flowcontrol_feature_enable	395
Table 3-495. Function enet_flowcontrol_feature_disable	396
Table 3-496. Function enet_dmaprocess_state_get	397
Table 3-497. Function enet_dmaprocess_resume.....	398
Table 3-498. Function enet_rxprocess_check_recovery.....	399
Table 3-499. Function enet_txfifo_flush.....	399
Table 3-500. Function enet_current_desc_address_get	400
Table 3-501. Function enet_desc_information_get	401
Table 3-502. Function enet_missed_frame_counter_get	402
Table 3-503. Function enet_desc_flag_get	403
Table 3-504. Function enet_desc_flag_set	404
Table 3-505. Function enet_desc_flag_clear	405
Table 3-506. Function enet_rx_desc_immediate_receive_complete_interrupt.....	406
Table 3-507. Function enet_rx_desc_delay_receive_complete_interrupt.....	407
Table 3-508. Function enet_rxframe_drop	407
Table 3-509. Function enet_dma_feature_enable.....	408
Table 3-510. Function enet_dma_feature_disable.....	409
Table 3-511. Function enet_rx_desc_enhanced_status_get.....	409
Table 3-512. Function enet_desc_select_enhanced_mode	410
Table 3-513. Function enet_ptp_enhanced_descriptors_chain_init	411
Table 3-514. Function enet_ptp_enhanced_descriptors_ring_init.....	411
Table 3-515. Function enet_ptpframe_receive_enhanced_mode	412
Table 3-516. Function enet_ptpframe_transmit_enhanced_mode	413
Table 3-517. Function enet_desc_select_normal_mode	414
Table 3-518. Function enet_ptp_normal_descriptors_chain_init	414
Table 3-519. Function enet_ptp_normal_descriptors_ring_init	415
Table 3-520. Function enet_ptpframe_receive_normal_mode	416
Table 3-521. Function enet_ptpframe_transmit_normal_mode	417
Table 3-522. Function enet_wum_filter_register_pointer_reset	418

Table 3-523. Function enet_wum_filter_config	418
Table 3-524. Function enet_wum_feature_enable	419
Table 3-525. Function enet_wum_feature_disable	420
Table 3-526. Function enet_msc_counters_reset.....	420
Table 3-527. Function enet_msc_feature_enable	421
Table 3-528. Function enet_msc_feature_disable	421
Table 3-529. Function enet_msc_counters_preset_config	422
Table 3-530. Function enet_msc_counters_get	423
Table 3-531. Function enet_ptp_subsecond_2_nanosecond	424
Table 3-532. Function enet_ptp_nanosecond_2_subsecond	424
Table 3-533. Function enet_ptp_feature_enable	425
Table 3-534. Function enet_ptp_feature_disable	426
Table 3-535. Function enet_ptp_timestamp_function_config	427
Table 3-536. Function enet_ptp_subsecond_increment_config	428
Table 3-537. Function enet_ptp_timestamp_addend_config	429
Table 3-538. Function enet_ptp_timestamp_update_config	430
Table 3-539. Function enet_ptp_expected_time_config.....	430
Table 3-540. Function enet_ptp_system_time_get.....	431
Table 3-541. Function enet_ptp_pps_output_frequency_config	432
Table 3-542. Function enet_ptp_start	433
Table 3-543. Function enet_ptp_finecorrection_adjfreq	434
Table 3-544. Function enet_ptp_coarsecorrection_systime_update	434
Table 3-545. Function enet_ptp_finecorrection_settime	435
Table 3-546. Function enet_ptp_flag_get	436
Table 3-547. Function enet_initpara_reset.....	436
Table 3-548. EXMC Registers	437
Table 3-549. EXMC firmware function.....	438
Table 3-550. Structure exmc_norsram_timing_parameter_struct	439
Table 3-551. Structure exmc_norsram_parameter_struct.....	439
Table 3-552. Structure exmc_nand_timing_parameter_struct	439
Table 3-553. Structure exmc_nand_parameter_struct.....	440
Table 3-554. Structure exmc_sdram_timing_parameter_struct.....	440
Table 3-555. Structure exmc_sdram_parameter_struct	440
Table 3-556. Structure exmc_sdram_command_parameter_struct	441
Table 3-557. Function exmc_norsram_deinit	441
Table 3-558. Function exmc_norsram_struct_para_init.....	442
Table 3-559. Function exmc_norsram_init.....	442
Table 3-560. Function exmc_norsram_enable	444
Table 3-561. Function exmc_norsram_disable	444
Table 3-562. Function exmc_nand_deinit	445
Table 3-563. Function exmc_nand_struct_para_init	445
Table 3-564. Function exmc_nand_init.....	446
Table 3-565. Function exmc_nand_enable	447
Table 3-566. Function exmc_nand_disable	447

Table 3-567. Function <code>exmc_sdram_deinit</code>	448
Table 3-568. Function <code>exmc_sdram_struct_para_init</code>	448
Table 3-569. Function <code>exmc_sdram_init</code>	449
Table 3-570. Function <code>exmc_norsram_sdram_remap_config</code>	450
Table 3-571. Function <code>exmc_norsram_sdram_remap_get</code>	451
Table 3-572. Function <code>exmc_norsram_consecutive_clock_config</code>	451
Table 3-573. Function <code>exmc_norsram_page_size_config</code>	452
Table 3-574. Function <code>exmc_nand_ecc_config</code>	453
Table 3-575. Function <code>exmc_ecc_get</code>	453
Table 3-576. Function <code>exmc_sdram_readsampler_enable</code>	454
Table 3-577. Function <code>exmc_sdram_readsampler_disable</code>	454
Table 3-578. Function <code>exmc_sdram_readsampler_config</code>	455
Table 3-579. Function <code>exmc_sdram_command_config</code>	455
Table 3-580. Function <code>exmc_sdram_refresh_count_set</code>	456
Table 3-581. Function <code>exmc_sdram_autorefresh_number_set</code>	457
Table 3-582. Function <code>exmc_sdram_write_protection_config</code>	457
Table 3-583. Function <code>exmc_sdram_bankstatus_get</code>	458
Table 3-584. Function <code>exmc_flag_get</code>	458
Table 3-585. Function <code>exmc_flag_clear</code>	459
Table 3-586. Function <code>exmc_interrupt_enable</code>	460
Table 3-587. Function <code>exmc_interrupt_disable</code>	461
Table 3-588. Function <code>exmc_interrupt_flag_get</code>	462
Table 3-589. Function <code>exmc_interrupt_flag_clear</code>	463
Table 3-590. EXTI Registers	464
Table 3-591. EXTI firmware function	464
Table 3-592. Enum <code>exti_line_enum</code>	464
Table 3-593. Enum <code>exti_mode_enum</code>	465
Table 3-594. Enum <code>exti_trig_type_enum</code>	466
Table 3-595. Function <code>exti_deinit</code>	466
Table 3-596. Function <code>exti_init</code>	466
Table 3-597. Function <code>exti_interrupt_enable</code>	467
Table 3-598. Function <code>exti_interrupt_disable</code>	467
Table 3-599. Function <code>exti_event_enable</code>	468
Table 3-600. Function <code>exti_event_disable</code>	468
Table 3-601. Function <code>exti_software_interrupt_enable</code>	469
Table 3-602. Function <code>exti_software_interrupt_disable</code>	469
Table 3-603. Function <code>exti_flag_get</code>	470
Table 3-604. Function <code>exti_flag_clear</code>	470
Table 3-605. Function <code>exti_interrupt_flag_get</code>	471
Table 3-606. Function <code>exti_interrupt_flag_clear</code>	471
Table 3-607. FAC Registers	472
Table 3-608. FAC firmware function	472
Table 3-609. Structure <code>fac_parameter_struct</code>	473
Table 3-610. Structure <code>fac_fixed_data_preload_struct</code>	474

Table 3-611. Structure fac_float_data_preload_struct	474
Table 3-612. Function fac_deinit	474
Table 3-613. Function fac_struct_para_init	475
Table 3-614. Function fac_fixed_data_preload_init	475
Table 3-615. Function fac_float_data_preload_init	476
Table 3-616. Function fac_init	477
Table 3-617. Function fac_preload	477
Table 3-618. Function fac_float_buffer_preload	478
Table 3-619. Function fac_float_preload	478
Table 3-620. Function fac_float_preload	479
Table 3-621. Function fac_init	479
Table 3-622. Function fac_clip_config	480
Table 3-623. Function fac_init	480
Table 3-624. Function fac_float_disable	481
Table 3-625. Function fac_dma_enable	481
Table 3-626. Function fac_dma_enable	482
Table 3-627. Function fac_dma_enable	482
Table 3-628. Function fac_dma_enable	483
Table 3-629. Function fac_dma_enable	483
Table 3-630. Function fac_dma_enable	484
Table 3-631. Function fac_start	485
Table 3-632. Function fac_start	485
Table 3-633. Function fac_start	486
Table 3-634. Function fac_fixed_data_write	486
Table 3-635. Function fac_fixed_data_write	487
Table 3-636. Function fac_fixed_data_write	487
Table 3-637. Function fac_fixed_data_write	488
Table 3-638. Function fac_interrupt_enable	488
Table 3-639. Function fac_interrupt_disable	489
Table 3-640. Function fac_interrupt_flag_get	489
Table 3-641. Function fac_flag_get	490
Table 3-642. FMC Registers	491
Table 3-643. FMC firmware function	492
Table 3-644. Enum fmc_state_enum	493
Table 3-645. Enum fmc_flag_enum	494
Table 3-646. Enum fmc_interrupt_flag_enum	494
Table 3-647. Enum fmc_interrupt_enum	495
Table 3-648. Function fmc_unlock	495
Table 3-649. Function fmc_lock	496
Table 3-650. Function fmc_page_erase	496
Table 3-651. Function fmc_mass_erase	497
Table 3-652. Function fmc_protection_removed_mass_erase	497
Table 3-653. Function fmc_word_program	498
Table 3-654. Function fmc_doubleword_program	498

Table 3-655. Function <code>fmc_check_programming_area_enable</code>	499
Table 3-656. Function <code>fmc_check_programming_area_disable</code>	499
Table 3-657. Function <code>ob_unlock</code>	500
Table 3-658. Function <code>ob_lock</code>	500
Table 3-659. Function <code>ob_start</code>	501
Table 3-660. Function <code>ob_factory_value_config</code>	501
Table 3-661. Function <code>ob_secure_access_mode_enable</code>	502
Table 3-662. Function <code>ob_secure_access_mode_disable</code>	503
Table 3-663. Function <code>ob_security_protection_config</code>	503
Table 3-664. Function <code>ob_bor_threshold_config</code>	504
Table 3-665. Function <code>ob_low_power_config</code>	505
Table 3-666. Function <code>ob_tcm_ecc_config</code>	506
Table 3-667. Function <code>ob_iospeed_optimize_config</code>	507
Table 3-668. Function <code>ob_tcm_shared_ram_config</code>	508
Table 3-669. Function <code>ob_data_program</code>	509
Table 3-670. Function <code>ob_boot_address_config</code>	509
Table 3-671. Function <code>ob_dcrp_area_config</code>	510
Table 3-672. Function <code>ob_secure_area_config</code>	511
Table 3-673. Function <code>ob_write_protection_enable</code>	512
Table 3-674. Function <code>ob_write_protection_disable</code>	513
Table 3-675. Function <code>ob_secure_mode_get</code>	514
Table 3-676. Function <code>ob_secure_mode_get</code>	514
Table 3-677. Function <code>ob_bor_threshold_get</code>	515
Table 3-678. Function <code>ob_low_power_get</code>	515
Table 3-679. Function <code>ob_tcm_ecc_get</code>	517
Table 3-680. Function <code>ob_secure_mode_get</code>	518
Table 3-681. Function <code>ob_tcm_shared_ram_size_get</code>	518
Table 3-682. Function <code>ob_data_get</code>	519
Table 3-683. Function <code>ob_boot_address_get</code>	520
Table 3-684. Function <code>ob_dcrp_area_get</code>	520
Table 3-685. Function <code>ob_secure_area_get</code>	521
Table 3-686. Function <code>ob_write_protection_get</code>	522
Table 3-687. Function <code>fmc_no_rtdec_config</code>	522
Table 3-688. Function <code>fmc_aes_iv_config</code>	523
Table 3-689. Function <code>fmc_flash_ecc_get</code>	523
Table 3-690. Function <code>fmc_no_rtdec_get</code>	524
Table 3-691. Function <code>fmc_aes_iv_get</code>	524
Table 3-692. Function <code>fmc_pid_get</code>	525
Table 3-693. Function <code>fmc_flag_get</code>	525
Table 3-694. Function <code>fmc_flag_clear</code>	526
Table 3-695. Function <code>fmc_interrupt_enable</code>	526
Table 3-696. Function <code>fmc_interrupt_disable</code>	527
Table 3-697. Function <code>fmc_interrupt_flag_get</code>	527
Table 3-698. Function <code>fmc_interrupt_flag_clear</code>	528

Table 3-699. FWDGT Registers	528
Table 3-700. FWDGT firmware function.....	529
Table 3-701. Function fwdgt_write_enable	529
Table 3-702. Function fwdgt_write_disable	530
Table 3-703. Function fwdgt_enable	530
Table 3-704. Function fwdgt_prescaler_value_config	531
Table 3-705. Function fwdgt_reload_value_config.....	531
Table 3-706. Function fwdgt_window_value_config	532
Table 3-707. Function fwdgt_counter_reload.....	532
Table 3-708. Function fwdgt_config.....	533
Table 3-709. Function fwdgt_flag_get.....	533
Table 3-710. GPIO Registers.....	534
Table 3-711. GPIO firmware function	535
Table 3-712. Function gpio_deinit	535
Table 3-713. Function gpio_mode_set.....	536
Table 3-714. Function gpio_output_options_set	537
Table 3-715. Function gpio_bit_set	538
Table 3-716. Function gpio_bit_reset	538
Table 3-717. Function gpio_bit_write.....	539
Table 3-718. Function gpio_port_write	539
Table 3-719. Function gpio_input_filter_set	540
Table 3-720. Function gpio_input_bit_get.....	541
Table 3-721. Function gpio_input_port_get.....	541
Table 3-722. Function gpio_output_bit_get	542
Table 3-723. Function gpio_output_port_get	543
Table 3-724. Function gpio_af_set	543
Table 3-725. Function gpio_pin_lock.....	544
Table 3-726. Function gpio_bit_toggle	545
Table 3-727. Function gpio_port_toggle	546
Table 3-728. HAU Registers	546
Table 3-729. HAU firmware function	547
Table 3-730. Structure hau_init_parameter_struct.....	548
Table 3-731. Structure hau_digest_parameter_struct	548
Table 3-732. Structure hau_context_parameter_struct	548
Table 3-733. Function hau_deinit	548
Table 3-734. Function hau_init.....	549
Table 3-735. Function hau_init_struct_para_init	550
Table 3-736. Function hau_reset	550
Table 3-737. Function hau_last_word_validbits_num_config	551
Table 3-738. Function hau_data_write	551
Table 3-739. Function hau_infifo_words_num_get	552
Table 3-740. Function hau_digest_read	552
Table 3-741. Function hau_digest_calculation_enable	553
Table 3-742. Function hau_multiple_single_dma_config.....	553

Table 3-743. Function hau_dma_enable.....	554
Table 3-744. Function hau_dma_disable.....	554
Table 3-745. Function hau_context_struct_para_init.....	555
Table 3-746. Function hau_context_save.....	555
Table 3-747. Function hau_context_restore	556
Table 3-748. Function hau_hash_sha_1.....	556
Table 3-749. Function hau_hmac_sha_1.....	557
Table 3-750. Function hau_hash_sha_224.....	558
Table 3-751. Function hau_hmac_sha_224.....	559
Table 3-752. Function hau_hash_sha_256.....	559
Table 3-753. Function hau_hmac_sha_256.....	560
Table 3-754. Function hau_hash_md5.....	561
Table 3-755. Function hau_hmac_md5.....	562
Table 3-756. Function hau_flag_get	563
Table 3-757. Function hau_flag_clear	563
Table 3-758. Function hau_interrupt_enable	564
Table 3-759. Function hau_interrupt_disable.....	564
Table 3-760. Function hau_interrupt_flag_get	565
Table 3-761. Function hau_interrupt_flag_clear	566
Table 3-762. HPDF Registers.....	567
Table 3-763. HPDF firmware function	567
Table 3-764. Structure hpdf_channel_parameter_struct.....	570
Table 3-765. Structure hpdf_filter_parameter_struct.....	570
Table 3-766. Structure hpdf_rc_parameter_struct.....	571
Table 3-767. Structure hpdf_ic_parameter_struct.....	571
Table 3-768. Enum hpdf_channel_enum	571
Table 3-769. Enum hpdf_filter_enum	571
Table 3-770. Enum hpdf_flag_enum.....	572
Table 3-771. Enum hpdf_interrput_flag_enum.....	574
Table 3-772. Enum hpdf_interrput_enum.....	575
Table 3-773. Function hpdf_deinit	575
Table 3-774. Function hpdf_channel_struct_para_init	576
Table 3-775. Function hpdf_filter_struct_para_init	576
Table 3-776. Function hpdf_rc_struct_para_init.....	577
Table 3-777. Function hpdf_ic_struct_para_init	577
Table 3-778. Function hpdf_enable	578
Table 3-779. Function hpdf_disable	578
Table 3-780. Function hpdf_channel_init.....	579
Table 3-781. Function hpdf_filter_init.....	580
Table 3-782. Function hpdf_rc_init.....	581
Table 3-783. Function hpdf_ic_init	582
Table 3-784. Function hpdf_clock_output_config	582
Table 3-785. Function hpdf_clock_output_source_config	583
Table 3-786. Function hpdf_clock_output_duty_mode_disable	584

Table 3-787. Function <code>hpdf_clock_output_duty_mode_enable</code>	584
Table 3-788. Function <code>hpdf_clock_output_divider_config</code>	585
Table 3-789. Function <code>hpdf_channel_enable</code>	585
Table 3-790. Function <code>hpdf_channel_disable</code>	586
Table 3-791. Function <code>hpdf_spi_clock_source_config</code>	586
Table 3-792. Function <code>hpdf_serial_interface_type_config</code>	587
Table 3-793. Function <code>hpdf_malfunction_monitor_disable</code>	587
Table 3-794. Function <code>hpdf_malfunction_monitor_enable</code>	588
Table 3-795. Function <code>hpdf_clock_loss_disable</code>	588
Table 3-796. Function <code>hpdf_clock_loss_enable</code>	589
Table 3-797. Function <code>hpdf_channel_pin_redirection_disable</code>	590
Table 3-798. Function <code>hpdf_channel_pin_redirection_enable</code>	590
Table 3-799. Function <code>hpdf_channel_multiplexer_config</code>	591
Table 3-800. Function <code>hpdf_data_pack_mode_config</code>	591
Table 3-801. Function <code>hpdf_data_right_bit_shift_config</code>	592
Table 3-802. Function <code>hpdf_calibration_offset_config</code>	592
Table 3-803. Function <code>hpdf_malfunction_break_signal_config</code>	593
Table 3-804. Function <code>hpdf_malfunction_counter_config</code>	594
Table 3-805. Function <code>hpdf_write_parallel_data_standard_mode</code>	594
Table 3-806. Function <code>hpdf_write_parallel_data_interleaved_mode</code>	595
Table 3-807. Function <code>hpdf_write_parallel_data_dual_mode</code>	595
Table 3-808. Function <code>hpdf_pulse_skip_update</code>	596
Table 3-809. Function <code>hpdf_pulse_skip_read</code>	597
Table 3-810. Function <code>hpdf_filter_enable</code>	597
Table 3-811. Function <code>hpdf_filter_disable</code>	598
Table 3-812. Function <code>hpdf_filter_config</code>	598
Table 3-813. Function <code>hpdf_integrator_oversample</code>	599
Table 3-814. Function <code>hpdf_threshold_monitor_filter_config</code>	600
Table 3-815. Function <code>hpdf_threshold_monitor_filter_read_data</code>	600
Table 3-816. Function <code>hpdf_threshold_monitor_fast_mode_disable</code>	601
Table 3-817. Function <code>hpdf_threshold_monitor_fast_mode_enable</code>	601
Table 3-818. Function <code>hpdf_threshold_monitor_channel</code>	602
Table 3-819. Function <code>hpdf_threshold_monitor_high_threshold</code>	602
Table 3-820. Function <code>hpdf_threshold_monitor_low_threshold</code>	603
Table 3-821. Function <code>hpdf_high_threshold_break_signal</code>	604
Table 3-822. Function <code>hpdf_low_threshold_break_signal</code>	604
Table 3-823. Function <code>hpdf_extremes_monitor_channel</code>	605
Table 3-824. Function <code>hpdf_extremes_monitor_maximum_get</code>	606
Table 3-825. Function <code>hpdf_extremes_monitor_minimum_get</code>	606
Table 3-826. Function <code>hpdf_conversion_time_get</code>	607
Table 3-827. Function <code>hpdf_rc_continuous_disable</code>	607
Table 3-828. Function <code>hpdf_rc_continuous_enable</code>	608
Table 3-829. Function <code>hpdf_rc_start_by_software</code>	608
Table 3-830. Function <code>hpdf_rc_syn_disable</code>	609

Table 3-831. Function <code>hpdf_rc_syn_enable</code>	609
Table 3-832. Function <code>hpdf_rc_dma_disable</code>	610
Table 3-833. Function <code>hpdf_rc_dma_enable</code>	610
Table 3-834. Function <code>hpdf_rc_channel_config</code>	611
Table 3-835. Function <code>hpdf_rc_fast_mode_disable</code>	611
Table 3-836. Function <code>hpdf_rc_fast_mode_enable</code>	612
Table 3-837. Function <code>hpdf_rc_data_get</code>	612
Table 3-838. Function <code>hpdf_rc_channel_get</code>	613
Table 3-839. Function <code>hpdf_ic_start_by_software</code>	613
Table 3-840. Function <code>hpdf_ic_syn_disable</code>	614
Table 3-841. Function <code>hpdf_ic_syn_enable</code>	614
Table 3-842. Function <code>hpdf_ic_dma_disable</code>	615
Table 3-843. Function <code>hpdf_ic_dma_enable</code>	615
Table 3-844. Function <code>hpdf_ic_scan_mode_disable</code>	616
Table 3-845. Function <code>hpdf_ic_scan_mode_enable</code>	616
Table 3-846. Function <code>hpdf_ic_trigger_signal_disable</code>	617
Table 3-847. Function <code>hpdf_ic_trigger_signal_config</code>	617
Table 3-848. Function <code>hpdf_ic_channel_config</code>	619
Table 3-849. Function <code>hpdf_ic_data_get</code>	619
Table 3-850. Function <code>hpdf_ic_channel_get</code>	620
Table 3-851. Function <code>hpdf_flag_get</code>	620
Table 3-852. Function <code>hpdf_flag_clear</code>	621
Table 3-853. Function <code>hpdf_interrupt_enable</code>	622
Table 3-854. Function <code>hpdf_interrupt_disable</code>	623
Table 3-855. Function <code>hpdf_interrupt_flag_get</code>	624
Table 3-856. Function <code>hpdf_interrupt_flag_clear</code>	625
Table 3-857. HWSEM Registers	626
Table 3-858. HWSEM firmware function	626
Table 3-859. Enum <code>hwsem_semaphore_enum</code>	627
Table 3-860. Function <code>hwsem_lock_set</code>	628
Table 3-861. Function <code>hwsem_lock_release</code>	628
Table 3-862. Function <code>hwsem_lock_by_reading</code>	629
Table 3-863. Function <code>hwsem_unlock_all</code>	630
Table 3-864. Function <code>hwsem_process_id_get</code>	630
Table 3-865. Function <code>hwsem_master_id_get</code>	631
Table 3-866. Function <code>hwsem_lock_status_get</code>	631
Table 3-867. Function <code>hwsem_key_set</code>	632
Table 3-868. Function <code>hwsem_key_get</code>	632
Table 3-869. Function <code>hwsem_flag_get</code>	633
Table 3-870. Function <code>hwsem_flag_clear</code>	634
Table 3-871. Function <code>hwsem_interrupt_flag_get</code>	634
Table 3-872. Function <code>hwsem_interrupt_flag_clear</code>	635
Table 3-873. Function <code>hwsem_interrupt_enable</code>	635
Table 3-874. Function <code>hwsem_interrupt_disable</code>	636

Table 3-875. I2C Registers	636
Table 3-876. I2C firmware function.....	637
Table 3-877. Enum i2c_interrupt_flag_enum	639
Table 3-878. Function i2c_deinit	639
Table 3-879. Function i2c_timing_config	640
Table 3-880. Function i2c_digital_noise_filter_config	640
Table 3-881. Function i2c_analog_noise_filter_enable	641
Table 3-882. Function i2c_analog_noise_filter_disable	642
Table 3-883. Function i2c_master_clock_config	642
Table 3-884. Function i2c_master_addressing	643
Table 3-885. Function i2c_address10_header_enable.....	644
Table 3-886. Function i2c_address10_header_disable.....	644
Table 3-887. Function i2c_address10_enable	645
Table 3-888. Function i2c_address10_disable	645
Table 3-889. Function i2c_automatic_end_enable	646
Table 3-890. Function i2c_automatic_end_disable	646
Table 3-891. Function i2c_slave_response_to_gcall_enable	647
Table 3-892. Function i2c_slave_response_to_gcall_disable	647
Table 3-893. Function i2c_stretch_scl_low_enable	648
Table 3-894. Function i2c_stretch_scl_low_disable	648
Table 3-895. Function i2c_address_config	649
Table 3-896. Function i2c_address_bit_compare_config	649
Table 3-897. Function i2c_address_disable	650
Table 3-898. Function i2c_second_address_config.....	651
Table 3-899. Function i2c_second_address_disable	652
Table 3-900. Function i2c_receved_address_get	652
Table 3-901. Function i2c_slave_byte_control_enable.....	653
Table 3-902. Function i2c_slave_byte_control_disable.....	653
Table 3-903. Function i2c_nack_enable	654
Table 3-904. Function i2c_nack_disable	654
Table 3-905. Function i2c_wakeup_from_deepsleep_enable.....	655
Table 3-906. Function i2c_wakeup_from_deepsleep_disable.....	655
Table 3-907. Function i2c_enable	656
Table 3-908. Function i2c_disable	656
Table 3-909. Function i2c_start_on_bus	657
Table 3-910. Function i2c_stop_on_bus.....	657
Table 3-911. Function i2c_data_transmit.....	658
Table 3-912. Function i2c_data_receive	658
Table 3-913. Function i2c_reload_enable.....	659
Table 3-914. Function i2c_reload_disable.....	660
Table 3-915. Function i2c_transfer_byte_number_config.....	660
Table 3-916. Function i2c_dma_enable	661
Table 3-917. Function i2c_dma_disable	661
Table 3-918. Function i2c_pec_transfer	662

Table 3-919. Function i2c_pec_enable.....	662
Table 3-920. Function i2c_pec_disable.....	663
Table 3-921. Function i2c_pec_value_get	663
Table 3-922. Function i2c_smbus_alert_enable	664
Table 3-923. Function i2c_smbus_alert_disable.....	664
Table 3-924. Function i2c_smbus_default_addr_enable	665
Table 3-925. Function i2c_smbus_default_addr_disable	665
Table 3-926. Function i2c_smbus_host_addr_enable	666
Table 3-927. Function i2c_smbus_host_addr_disable	666
Table 3-928. Function i2c_extented_clock_timeout_enable	667
Table 3-929. Function i2c_extented_clock_timeout_disable	667
Table 3-930. Function i2c_clock_timeout_enable.....	668
Table 3-931. Function i2c_clock_timeout_disable.....	668
Table 3-932. Function i2c_bus_timeout_b_config.....	669
Table 3-933. Function i2c_bus_timeout_a_config	670
Table 3-934. Function i2c_idle_clock_timeout_config.....	670
Table 3-935. Function i2c_flag_get.....	671
Table 3-936. Function i2c_flag_clear.....	672
Table 3-937. Function i2c_interrupt_enable.....	672
Table 3-938. Function i2c_interrupt_disable	673
Table 3-939. Function i2c_interrupt_flag_get.....	674
Table 3-940. Function i2c_interrupt_flag_clear	675
Table 3-941. IPA Registers	676
Table 3-942. IPA firmware function.....	677
Table3-943. Structure ipa_foreground_parameter_struct.....	678
Table 3-944. Structure ipa_background_parameter_struct	678
Table 3-945. Structure ipa_destination_parameter_struct.....	679
Table 3-946. Structure ipa_conversion_parameter_struct.....	679
Table 3-947. Enum ipa_dpf_enum	680
Table 3-948. Enum ipa_colorspace_enum	680
Table 3-949. Function ipa_deinit.....	680
Table 3-950. Function ipa_transfer_enable.....	681
Table 3-951. Function ipa_transfer_hangup_enable	681
Table 3-952. Function ipa_transfer_hangup_disable	682
Table 3-953. Function ipa_transfer_stop_enable.....	682
Table 3-954. Function ipa_transfer_stop_disable	683
Table 3-955. Function ipa_foreground_lut_loading_enable	683
Table 3-956. Function ipa_background_lut_loading_enable	684
Table 3-957. Function ipa_pixel_format_convert_mode_set.....	684
Table 3-958. Function ipa_foreground_interlace_mode_enable.....	685
Table 3-959. Function ipa_foreground_interlace_mode_disable.....	685
Table 3-960. Function ipa_foreground_struct_para_init	686
Table 3-961. Function ipa_foreground_init.....	686
Table 3-962. Function ipa_background_struct_para_init.....	687



Table 3-963. Function ipa_background_init	688
Table 3-964. Function ipa_destination_struct_para_init	689
Table 3-965. Function ipa_destination_init	689
Table 3-966. Function ipa_foreground_lut_init	691
Table 3-967. Function ipa_background_lut_init	691
Table 3-968. Function ipa_line_mark_config	692
Table 3-969. Function ipa_inter_timer_config	693
Table 3-970. Function ipa_interval_clock_num_config	693
Table 3-971. Function ipa_color_conversion_struct_para_init	694
Table 3-972. Function ipa_color_conversion_config	694
Table 3-973. Function ipa_foreground_scaling_config	695
Table 3-974. Function ipa_destination_scaling_config	696
Table 3-975. Function ipa_flag_get	697
Table 3-976. Function ipa_flag_clear	698
Table 3-977. Function ipa_interrupt_enable	698
Table 3-978. Function ipa_interrupt_disable	699
Table 3-979. Function ipa_interrupt_flag_get	699
Table 3-980. Function ipa_interrupt_flag_clear	700
Table 3-981. LPDTS Registers	701
Table 3-982. LPDTS firmware function	701
Table3-983. Structure lpdts_parameter_struct	702
Table3-984. Function lpdts_deinit	702
Table3-985. Function lpdts_struct_para_init	703
Table3-986. Function lpdts_init	703
Table3-987. Function lpdts_enable	704
Table3-988. Function lpdts_disable	704
Table3-989. Function lpdts_soft_trigger_enable	705
Table3-990. Function lpdts_soft_trigger_disable	705
Table3-991. Function lpdts_high_threshold_set	706
Table3-992. Function lpdts_low_threshold_set	706
Table3-993. Function lpdts_ref_clock_source_config	707
Table3-994. Function lpdts_temperature_get	707
Table3-995. Function lpdts_flag_get	708
Table3-996. Function lpdts_interrupt_enable	708
Table3-997. Function lpdts_interrupt_disable	709
Table3-998. Function lpdts_interrupt_flag_get	710
Table3-999. Function lpdts_interrupt_flag_clear	710
Table 3-1000. MDIO Registers	711
Table 3-1001. MDIO firmware function	711
Table 3-1002. Function mdio_deinit	712
Table 3-1003. Function mdio_software_reset	713
Table 3-1004. Function mdio_init	713
Table 3-1005. Function mdio_phy_length_config	714
Table 3-1006. Function mdio_soft_phyadr_set	715

Table 3-1007. Function mdio_framefield_phyadr_config.....	715
Table 3-1008. Function mdio_framefield_devadd_config	716
Table 3-1009. Function mdio_phy_pin_read.....	716
Table 3-1010. Function mdio_timeout_config	717
Table 3-1011. Function mdio_timeout_enable.....	717
Table 3-1012. Function mdio_timeout_disable	718
Table 3-1013. Function mdio_op_receive	718
Table 3-1014. Function mdio_phyadr_receive	719
Table 3-1015. Function mdio_devadd_receive.....	719
Table 3-1016. Function mdio_ta_receive.....	720
Table 3-1017. Function mdio_data_receive	720
Table 3-1018. Function mdio_address_receive	721
Table 3-1019. Function mdio_data_transmit	721
Table 3-1020. Function mdio_flag_get.....	722
Table 3-1021. Function mdio_flag_clear	723
Table 3-1022. Function mdio_interrupt_enable	724
Table 3-1023. Function mdio_interrupt_disable	724
Table 3-1024. MDMA Registers	725
Table 3-1025. MDMA firmware function	726
Table 3-1026. Structure mdma_parameter_struct	727
Table 3-1027. Structure mdma_multi_block_parameter_struct	728
Table 3-1028. Structure mdma_link_node_parameter_struct	728
Table 3-1029. Enum mdma_add_update_dir_enum	728
Table 3-1030. Enum mdma_channel_enum	729
Table 3-1031. Function mdma_deinit.....	729
Table 3-1032. Function mdma_channel_deinit.....	730
Table 3-1033. Function mdma_para_struct_init.....	730
Table 3-1034. Function mdma_multi_block_para_struct_init	731
Table 3-1035. Function mdma_link_node_para_struct_init.....	731
Table 3-1036. Function mdma_init	732
Table 3-1037. Function mdma_buffer_block_mode_config.....	733
Table 3-1038. Function mdma_multi_block_mode_config	734
Table 3-1039. Function mdma_node_create.....	735
Table 3-1040. Function mdma_node_add	735
Table 3-1041. Function mdma_node_delete	736
Table 3-1042. Function mdma_destination_address_config.....	737
Table 3-1043. Function mdma_source_address_config	737
Table 3-1044. Function mdma_destination_bus_config.....	738
Table 3-1045. Function mdma_source_bus_config	738
Table 3-1046. Function mdma_priority_config.....	739
Table 3-1047. Function mdma_endianness_config.....	740
Table 3-1048. Function mdma_alignment_config.....	741
Table 3-1049. Function mdma_source_burst_beats_config.....	742
Table 3-1050. Function mdma_destination_burst_beats_config.....	743

Table 3-1051. Function mdma_source_width_config	744
Table 3-1052. Function mdma_destination_width_config	744
Table 3-1053. Function mdma_source_increment_config	745
Table 3-1054. Function mdma_destination_increment_config	746
Table 3-1055. Function mdma_channel_bufferable_write_enable	747
Table 3-1056. Function mdma_channel_bufferable_write_disable	748
Table 3-1057. Function mdma_channel_software_request_enable	748
Table 3-1058. Function mdma_channel_enable.....	749
Table 3-1059. Function mdma_channel_disable.....	749
Table 3-1060. Function mdma_transfer_error_direction_get	750
Table 3-1061. Function mdma_transfer_error_address_get.....	751
Table 3-1062. Function mdma_flag_get.....	751
Table 3-1063. Function mdma_flag_clear	752
Table 3-1064. Function mdma_interrupt_enable	753
Table 3-1065. Function mdma_interrupt_disable	754
Table 3-1066. Function mdma_interrupt_flag_get.....	754
Table 3-1067. Function mdma_interrupt_flag_clear.....	755
Table 3-1068. OSPI Registers	756
Table 3-1069. OSPI firmware function	757
Table3-1070. Structure ospi_parameter_struct.....	758
Table3-1071. Structure ospi_regular_cmd_struct	759
Table3-1072. Structure ospi_autopolling_struct.....	760
Table 3-1073. Enum ospi_interrupt_flag_enum	760
Table3-1074. Function ospi_deinit	761
Table3-1075. Function ospi_struct_init	761
Table3-1076. Function ospi_init.....	762
Table3-1077. Function ospi_enable	762
Table3-1078. Function ospi_disable	763
Table3-1079. Function ospi_device_memory_type_config	763
Table3-1080. Function ospi_device_memory_size_config	764
Table3-1081. Function ospi_functional_mode_config.....	765
Table3-1082. Function ospi_status_polling_config	765
Table3-1083. Function ospi_status_mask_config	766
Table3-1084. Function ospi_status_match_config.....	767
Table3-1085. Function ospi_interval_cycle_config	767
Table3-1086. Function ospi_fifo_level_config.....	768
Table3-1087. Function ospi_chip_select_high_cycle_config.....	768
Table3-1088. Function ospi_prescaler_config	769
Table3-1089. Function ospi_dummy_cycles_config.....	770
Table3-1090. Function ospi_delay_hold_cycle_config.....	770
Table3-1091. Function ospi_sample_shift_config.....	771
Table3-1092. Function ospi_data_length_config.....	771
Table3-1093. Function ospi_instruction_config	772
Table3-1094. Function ospi_address_config	773

Table3-1095. Function <code>ospi_alternate_bytes_config</code>	774
Table3-1096. Function <code>ospi_data_config</code>	776
Table3-1097. Function <code>ospi_data_transmit</code>	776
Table3-1098. Function <code>ospi_data_receive</code>	777
Table3-1099. Function <code>ospi_dma_enable</code>	777
Table3-1100. Function <code>ospi_dma_disable</code>	778
Table3-1101. Function <code>ospi_wrap_size_config</code>	778
Table3-1102. Function <code>ospi_wrap_instruction_config</code>	779
Table3-1103. Function <code>ospi_wrap_address_config</code>	780
Table3-1104. Function <code>ospi_wrap_alternate_bytes_config</code>	781
Table3-1105. Function <code>ospi_wrap_data_config</code>	783
Table3-1106. Function <code>ospi_wrap_dummy_cycles_config</code>	783
Table3-1107. Function <code>ospi_wrap_delay_hold_cycle_config</code>	784
Table3-1108. Function <code>ospi_wrap_sample_shift_config</code>	785
Table3-1109. Function <code>ospi_write_instruction_config</code>	785
Table3-1110. Function <code>ospi_write_address_config</code>	786
Table3-1111. Function <code>ospi_write_alternate_bytes_config</code>	788
Table3-1112. Function <code>ospi_write_data_config</code>	789
Table3-1113. Function <code>ospi_write_dummy_cycles_config</code>	790
Table3-1114. Function <code>ospi_write_dummy_cycles_config</code>	790
Table3-1115. Function <code>ospi_transmit</code>	791
Table3-1116. Function <code>ospi_receive</code>	791
Table3-1117. Function <code>ospi_autopolling_mode</code>	792
Table3-1118. Function <code>ospi_interrupt_enable</code>	793
Table3-1119. Function <code>ospi_interrupt_disable</code>	793
Table3-1120. Function <code>ospi_fifo_level_get</code>	794
Table3-1121. Function <code>ospi_flag_get</code>	794
Table3-1122. Function <code>ospi_flag_clear</code>	795
Table3-1123. Function <code>ospi_interrupt_flag_get</code>	796
Table3-1124. Function <code>ospi_interrupt_flag_clear</code>	796
Table 3-1125. OSPIM Registers	797
Table 3-1126. OSPIM firmware function	797
Table3-1127. Function <code>ospim_deinit</code>	798
Table3-1128. Function <code>ospim_port_sck_config</code>	798
Table3-1129. Function <code>ospim_port_sck_source_select</code>	799
Table3-1130. Function <code>ospim_port_csn_config</code>	799
Table3-1131. Function <code>ospim_port_csn_source_select</code>	800
Table3-1132. Function <code>ospim_port_io3_0_config</code>	801
Table3-1133. Function <code>ospim_port_io3_0_source_select</code>	801
Table3-1134. Function <code>ospim_port_io7_4_config</code>	802
Table3-1135. Function <code>ospim_port_io7_4_source_select</code>	803
Table 3-1136. NVIC Registers	804
Table 3-1137. SysTick Registers	805
Table 3-1138. MISC firmware function	805

Table 3-1139. Structure mpu_region_init_struct.....	806
Table 3-1140. Enum IRQn_Type	806
Table 3-1141. Function nvic_priority_group_set	810
Table 3-1142. Function nvic_irq_enable.....	811
Table 3-1143. Function nvic_irq_disable.....	812
Table 3-1144. Function nvic_vector_table_set.....	812
Table 3-1145. Function system_lowpower_set	813
Table 3-1146. Function system_lowpower_reset.....	813
Table 3-1147. Function systick_clksource_set	814
Table 3-1148. Function mpu_region_struct_para_init	814
Table 3-1149. Function mpu_region_config	815
Table 3-1150. Function mpu_region_enable.....	816
Table 3-1151. PMU Registers.....	816
Table 3-1152. PMU firmware function	817
Table 3-1153. Function pmu_deinit	818
Table 3-1154. Function pmu_lvd_select.....	818
Table 3-1155. Function pmu_lvd_enable.....	819
Table 3-1156. Function pmu_lvd_disable.....	819
Table 3-1157. Function pmu_avd_select.....	820
Table 3-1158. Function pmu_avd_enable.....	820
Table 3-1159. Function pmu_avd_disable.....	821
Table 3-1160. Function pmu_cvd_enable.....	821
Table 3-1161. Function pmu_cvd_disable.....	822
Table 3-1162. Function pmu_ldo_output_select.....	822
Table 3-1163. Function pmu_slido_output_select.....	823
Table 3-1164. Function pmu_vbat_charging_select.....	823
Table 3-1165. Function pmu_vbat_charging_enable	824
Table 3-1166. Function pmu_vbat_charging_disable	824
Table 3-1167. Function pmu_vbat_temp_monitor_enable	825
Table 3-1168. Function pmu_vbat_temp_monitor_disable	825
Table 3-1169. Function pmu_usb_regulator_enable	826
Table 3-1170. Function pmu_usb_regulator_disable	826
Table 3-1171. Function pmu_usb_voltage_detector_enable	827
Table 3-1172. Function pmu_usb_voltage_detector_disable	827
Table 3-1173. Function pmu_smps_ldo_supply_config.....	828
Table 3-1174. Function pmu_to_sleepmode	829
Table 3-1175. Function pmu_to_deepsleepmode	829
Table 3-1176. Function pmu_to_standbymode	830
Table 3-1177. Function pmu_wakeup_pin_enable.....	830
Table 3-1178. Function pmu_wakeup_pin_disable.....	831
Table 3-1179. Function pmu_backup_write_enable	831
Table 3-1180. Function pmu_backup_write_disable	832
Table 3-1181. Function pmu_backup_voltage_stabilizer_enable	832
Table 3-1182. Function pmu_backup_voltage_stabilizer_disable	833

Table 3-1183. Function pmu_enter_deepsleep_wait_time_config	833
Table 3-1184. Function pmu_exit_deepsleep_wait_time_config	834
Table 3-1185. Function pmu_flag_get	834
Table 3-1186. Function pmu_flag_clear	835
Table 3-1187. RAMECCMU Registers	836
Table 3-1188. RAMECCMU firmware function	836
Table 3-1189. Enum rameccmu_monitor_enum	837
Table 3-1190. Function rameccmu_deinit	837
Table 3-1191. Function rameccmu_monitor_failing_address_get	838
Table 3-1192. Function rameccmu_monitor_failing_data_low_bits_get	839
Table 3-1193. Function rameccmu_monitor_failing_data_high_bits_get	839
Table 3-1194. Function rameccmu_monitor_failing_ecc_error_code_get	840
Table 3-1195. Function rameccmu_global_interrupt_enable	841
Table 3-1196. Function rameccmu_global_interrupt_disable	842
Table 3-1197. Function rameccmu_monitor_interrupt_enable	843
Table 3-1198. Function rameccmu_monitor_interrupt_disable	844
Table 3-1199. Function rameccmu_monitor_flag_get	845
Table 3-1200. Function rameccmu_monitor_flag_clear	846
Table 3-1201. Function rameccmu_monitor_interrupt_flag_get	848
Table 3-1202. Function rameccmu_monitor_interrupt_flag_clear	849
Table 3-1203. RCU Registers	850
Table 3-1204. RCU firmware function	852
Table 3-1205. Enum rcu_periph_enum	854
Table 3-1206. Enum rcu_periph_sleep_enum	857
Table 3-1207. Enum rcu_periph_reset_enum	860
Table 3-1208. Enum rcu_flag_enum	862
Table 3-1209. Enum rcu_int_flag_enum	863
Table 3-1210. Enum rcu_int_flag_clear_enum	864
Table 3-1211. Enum rcu_int_enum	864
Table 3-1212. Enum rcu_osci_type_enum	865
Table 3-1213. Enum rcu_clock_freq_enum	865
Table 3-1214. Enum usart_idx_enum	866
Table 3-1215. Enum i2c_idx_enum	866
Table 3-1216. Enum can_idx_enum	866
Table 3-1217. Enum sai_idx_enum	866
Table 3-1218. Enum sai2b_idx_enum	867
Table 3-1219. Enum adc_idx_enum	867
Table 3-1220. Enum usbhs_idx_enum	867
Table 3-1221. Enum pll_idx_enum	867
Table 3-1222. Enum sdio_idx_enum	867
Table 3-1223. Enum spi_idx_enum	867
Table 3-1224. Function rcu_deinit	868
Table 3-1225. Function rcu_periph_clock_enable	868
Table 3-1226. Function rcu_periph_clock_disable	869

Table 3-1227. Function rcu_periph_clock_sleep_enable	869
Table 3-1228. Function rcu_periph_clock_sleep_disable	870
Table 3-1229. Function rcu_periph_reset_enable	870
Table 3-1230. Function rcu_periph_reset_disable	871
Table 3-1231. Function rcu_bkp_reset_enable	871
Table 3-1232. Function rcu_bkp_reset_disable	872
Table 3-1233. Function rcu_system_clock_source_config	872
Table 3-1234. Function rcu_system_clock_source_get	873
Table 3-1235. Function rcu_ahb_clock_config	873
Table 3-1236. Function rcu_apb1_clock_config	874
Table 3-1237. Function rcu_apb2_clock_config	874
Table 3-1238. Function rcu_apb3_clock_config	875
Table 3-1239. Function rcu_apb4_clock_config	875
Table 3-1240. Function rcu_ckout0_config	876
Table 3-1241. Function rcu_ckout1_config	877
Table 3-1242. Function rcu_pll_input_output_clock_range_config	878
Table 3-1243. Function rcu_pll_fractional_config	879
Table 3-1244. Function rcu_pll_fractional_latch_enable	879
Table 3-1245. Function rcu_pll_fractional_latch_disable	880
Table 3-1246. Function rcu_pll_source_config	880
Table 3-1247. Function rcu_pll0_config	881
Table 3-1248. Function rcu_pll1_config	882
Table 3-1249. Function rcu_pll2_config	882
Table 3-1250. Function rcu_pll_clock_output_enable	883
Table 3-1251. Function rcu_pll_clock_output_disable	884
Table 3-1252. Function rcu_pllusb0_config	885
Table 3-1253. Function rcu_pllusb1_config	886
Table 3-1254. Function rcu_rtc_clock_config	886
Table 3-1255. Function rcu_rtc_div_config	887
Table 3-1256. Function rcu_ck48m_clock_config	888
Table 3-1257. Function rcu_pll48m_clock_config	888
Table 3-1258. Function rcu_irc64mdiv_clock_config	889
Table 3-1259. Function rcu_irc64mdiv_freq_get	889
Table 3-1260. Function rcu_timer_clock_prescaler_config	890
Table 3-1261. Function rcu_spi_clock_config	890
Table 3-1262. Function rcu_sdio_clock_config	891
Table 3-1263. Function rcu_deepsleep_wakeup_sys_clock_config	892
Table 3-1264. Function rcu_tli_clock_div_config	893
Table 3-1265. Function rcu_usart_clock_config	893
Table 3-1266. Function rcu_i2c_clock_config	894
Table 3-1267. Function rcu_can_clock_config	895
Table 3-1268. Function rcu_adc_clock_config	895
Table 3-1269. Function rcu_sai_clock_config	896
Table 3-1270. Function rcu_sai2_block_clock_config	896

Table 3-1271. Function rcu_rspdif_clock_config	897
Table 3-1272. Function rcu_exmc_clock_config	898
Table 3-1273. Function rcu_hpdf_clock_config	899
Table 3-1274. Function rcu_per_clock_config	899
Table 3-1275. Function rcu_usbhs_pll1qpsc_config	900
Table 3-1276. Function rcu_usb48m_clock_config	900
Table 3-1277. Function rcu_usbhs_clock_config	901
Table 3-1278. Function rcu_usbhs_clock_selection_enable	902
Table 3-1279. Function rcu_usbhs_clock_selection_disable	902
Table 3-1280. Function rcu_lxtal_drive_capability_config	903
Table 3-1281. Function rcu_osci_stab_wait	903
Table 3-1282. Function rcu_osci_on	904
Table 3-1283. Function rcu_osci_off	904
Table 3-1284. Function rcu_osci_bypass_mode_enable	905
Table 3-1285. Function rcu_osci_bypass_mode_disable	905
Table 3-1286. Function rcu_irc64m_adjust_value_set	906
Table 3-1287. Function rcu_lpirc4m_adjust_value_set	906
Table 3-1288. Function rcu_hxtal_clock_monitor_enable	907
Table 3-1289. Function rcu_hxtal_clock_monitor_disable	907
Table 3-1290. Function rcu_lxtal_clock_monitor_enable	908
Table 3-1291. Function rcu_lxtal_clock_monitor_disable	908
Table 3-1292. Function rcu_clock_freq_get	909
Table 3-1293. Function rcu_flag_get	909
Table 3-1294. Function rcu_all_reset_flag_clear	910
Table 3-1295. Function rcu_interrupt_enable	910
Table 3-1296. Function rcu_interrupt_disable	911
Table 3-1297. Function rcu_interrupt_flag_get	911
Table 3-1298. Function rcu_interrupt_flag_clear	912
Table 3-1299. RSPDIF Registers	912
Table 3-1300. RSPDIF firmware function	913
Table 3-1301. Structure rspdif_parameter_struct	913
Table 3-1302. Structure rspdif_data_struct	914
Table 3-1303. Function rspdif_deinit	915
Table 3-1304. Function rspdif_struct_para_init	915
Table 3-1305. Function rspdif_init	916
Table 3-1306. Function rspdif_enable	917
Table 3-1307. Function rspdif_disable	917
Table 3-1308. Function rspdif_symbol_clock_enable	918
Table 3-1309. Function rspdif_symbol_clock_disable	918
Table 3-1310. Function rspdif_backup_symbol_clock_enable	919
Table 3-1311. Function rspdif_backup_symbol_clock_disable	919
Table 3-1312. Function rspdif_dma_enable	920
Table 3-1313. Function rspdif_dma_disable	920
Table 3-1314. Function rspdif_control_buffer_dma_enable	921

Table 3-1315. Function <code>rspdif_control_buffer_dma_disable</code>	921
Table 3-1316. Function <code>rspdif_data_read</code>	922
Table 3-1317. Function <code>rspdif_duration_of_symbols_get</code>	922
Table 3-1318. Function <code>rspdif_user_data_get</code>	923
Table 3-1319. Function <code>rspdif_channel_status_get</code>	923
Table 3-1320. Function <code>rspdif_start_block_status_get</code>	924
Table 3-1321. Function <code>rspdif_low_threshold_get</code>	924
Table 3-1322. Function <code>rspdif_high_threshold_get</code>	925
Table 3-1323. Function <code>rspdif_flag_get</code>	925
Table 3-1324. Function <code>rspdif_flag_clear</code>	926
Table 3-1325. Function <code>rspdif_interrupt_enable</code>	927
Table 3-1326. Function <code>rspdif_interrupt_disable</code>	927
Table 3-1327. Function <code>rspdif_interrupt_flag_get</code>	928
Table 3-1328. Function <code>rspdif_interrupt_flag_clear</code>	929
Table 3-1329. RTC Registers	930
Table 3-1330. RTC firmware function	931
Table 3-1331. Structure <code>rtc_parameter_struct</code>	932
Table 3-1332. Structure <code>rtc_alarm_struct</code>	933
Table 3-1333. Structure <code>rtc_timestamp_struct</code>	933
Table 3-1334. Structure <code>rtc_tamper_struct</code>	933
Table 3-1335. Function <code>rtc_deinit</code>	934
Table 3-1336. Function <code>rtc_init</code>	934
Table 3-1337. Function <code>rtc_init_mode_enter</code>	935
Table 3-1338. Function <code>rtc_init_mode_exit</code>	936
Table 3-1339. Function <code>rtc_register_sync_wait</code>	936
Table 3-1340. Function <code>rtc_current_time_get</code>	937
Table 3-1341. Function <code>rtc_subsecond_get</code>	937
Table 3-1342. Function <code>rtc_alarm_config</code>	938
Table 3-1343. Function <code>rtc_alarm_subsecond_config</code>	938
Table 3-1344. Function <code>rtc_alarm_enable</code>	940
Table 3-1345. Function <code>rtc_alarm_disable</code>	940
Table 3-1346. Function <code>rtc_alarm_get</code>	941
Table 3-1347. Function <code>rtc_alarm_subsecond_get</code>	941
Table 3-1348. Function <code>rtc_timestamp_enable</code>	942
Table 3-1349. Function <code>rtc_timestamp_disable</code>	942
Table 3-1350. Function <code>rtc_timestamp_internalevent_config</code>	943
Table 3-1351. Function <code>rtc_timestamp_get</code>	943
Table 3-1352. Function <code>rtc_timestamp_subsecond_get</code>	944
Table 3-1353. Function <code>rtc_tamper_enable</code>	944
Table 3-1354. Function <code>rtc_tamper_disable</code>	945
Table 3-1355. Function <code>rtc_output_pin_select</code>	945
Table 3-1356. Function <code>rtc_alarm_output_config</code>	946
Table 3-1357. Function <code>rtc_calibration_output_config</code>	947
Table 3-1358. Function <code>rtc_hour_adjust</code>	947

Table 3-1359. Function rtc_second_adjust.....	948
Table 3-1360. Function rtc_bypass_shadow_enable	948
Table 3-1361. Function rtc_bypass_shadow_disable	949
Table 3-1362. Function rtc_refclock_detection_enable.....	949
Table 3-1363. Function rtc_refclock_detection_disable.....	950
Table 3-1364. Function rtc_wakeup_enable	950
Table 3-1365. Function rtc_wakeup_disable	951
Table 3-1366. Function rtc_wakeup_clock_set	951
Table 3-1367. Function rtc_wakeup_timer_set	952
Table 3-1368. Function rtc_wakeup_timer_get	953
Table 3-1369. Function rtc_smooth_calibration_config.....	953
Table 3-1370. Function rtc_interrupt_enable.....	954
Table 3-1371. Function rtc_interrupt_disable.....	955
Table 3-1372. Function rtc_flag_get.....	955
Table 3-1373. Function rtc_flag_clear.....	956
Table 3-1374. RTDEC Registers	957
Table 3-1375. RTDEC firmware function	958
Table 3-1376. Structure rtdec_parameter_struct	958
Table 3-1377. Function rtdec_deinit.....	958
Table 3-1378. Function rtdec_struct_para_init	959
Table 3-1379. Function rtdec_init	960
Table 3-1380. Function rtdec_config.....	960
Table 3-1381. Function rtdec_lock	961
Table 3-1382. Function rtdec_addr_init.....	962
Table 3-1383. Function rtdec_nonce_init.....	962
Table 3-1384. Function rtdec_key_init.....	963
Table 3-1385. Function rtdec_key_crc_get.....	964
Table 3-1386. Function rtdec_enable	964
Table 3-1387. Function rtdec_disable	965
Table 3-1388. Function rtdec_flag_get	965
Table 3-1389. Function rtdec_flag_clear	966
Table 3-1390. Function rtdec_interrupt_enable	967
Table 3-1391. Function rtdec_interrupt_disable	967
Table 3-1392. Function rtdec_interrupt_flag_get.....	968
Table 3-1393. Function rtdec_interrupt_flag_clear.....	969
Table 3-1394. SAI Registers.....	969
Table 3-1395. SAI firmware function	970
Table 3-1396. Structure sai_parameter_struct	971
Table 3-1397. Structure sai_frame_parameter_struct.....	972
Table 3-1398. Structure sai_slot_parameter_struct	972
Table 3-1399. Enum sai_fifo_state_enum	972
Table 3-1400. Function sai_deinit.....	972
Table 3-1401. Function sai_struct_para_init	973
Table 3-1402. Function sai_frame_struct_para_init	973

Table 3-1403. Function sai_slot_struct_para_init.....	974
Table 3-1404. Function sai_init	974
Table 3-1405. Function sai_frame_init.....	975
Table 3-1406. Function sai_slot_init	976
Table 3-1407. Function sai_enable.....	977
Table 3-1408. Function sai_disable	978
Table 3-1409. Function sai_sdoutput_config	978
Table 3-1410. Function sai_monomode_config	979
Table 3-1411. Function sai_companing_config	980
Table 3-1412. Function sai_mute_enable.....	980
Table 3-1413. Function sai_mute_disable.....	981
Table 3-1414. Function sai_mute_value_config.....	982
Table 3-1415. Function sai_mute_count_config	982
Table 3-1416. Function sai_data_transmit	983
Table 3-1417. Function sai_data_receive	984
Table 3-1418. Function sai_fifo_status_get	984
Table 3-1419. Function sai_fifo_flush.....	985
Table 3-1420. Function sai_dma_enable	985
Table 3-1421. Function sai_dma_disable	986
Table 3-1422. Function sai_sync_input_config	987
Table 3-1423. Function sai_sync_output_config	987
Table 3-1424. Function sai_pdm_enable.....	988
Table 3-1425. Function sai_pdm_disable.....	988
Table 3-1426. Function sai_pdm_microphone_num_config.....	989
Table 3-1427. Function sai_pdm_delay_config.....	990
Table 3-1428. Function sai_pdm_clk0_enable	991
Table 3-1429. Function sai_pdm_clk0_disable	991
Table 3-1430. Function sai_pdm_clk1_enable	992
Table 3-1431. Function sai_pdm_clk1_disable	992
Table 3-1432. Function sai_interrupt_enable	993
Table 3-1433. Function sai_interrupt_disable	993
Table 3-1434. Function sai_interrupt_flag_get.....	994
Table 3-1435. Function sai_interrupt_flag_clear.....	995
Table 3-1436. Function sai_flag_get	996
Table 3-1437. Function sai_flag_clear	997
Table 3-1438. SDIO Registers.....	998
Table 3-1439. SDIO firmware function	998
Table 3-1440. Function sdio_deinit	1000
Table 3-1441. Function sdio_clock_config	1001
Table 3-1442. Function sdio_clock_receive_set.....	1001
Table 3-1443. Function sdio_hardware_clock_enable.....	1002
Table 3-1444. Function sdio_hardware_clock_disable.....	1003
Table 3-1445. Function sdio_bus_mode_set.....	1003
Table 3-1446. Function sdio_bus_speed_set.....	1004

Table 3-1447. Function sdio_data_rate_set	1005
Table 3-1448. Function sdio_direction_polarity_set	1005
Table 3-1449. Function sdio_power_state_set.....	1006
Table 3-1450. Function sdio_power_state_get	1007
Table 3-1451. Function sdio_command_response_config	1007
Table 3-1452. Function sdio_wait_type_set.....	1008
Table 3-1453. Function sdio_trans_start_enable	1009
Table 3-1454. Function sdio_trans_start_disable.....	1009
Table 3-1455. Function sdio_trans_stop_enable	1010
Table 3-1456. Function sdio_trans_stop_disable	1010
Table 3-1457. Function sdio_csm_enable.....	1011
Table 3-1458. Function sdio_csm_disable.....	1012
Table 3-1459. Function sdio_command_index_get.....	1012
Table 3-1460. Function sdio_response_get.....	1013
Table 3-1461. Function sdio_hold_enable	1013
Table 3-1462. Function sdio_hold_disable	1014
Table 3-1463. Function sdio_suspend_enable.....	1014
Table 3-1464. Function sdio_suspend_disable.....	1015
Table 3-1465. Function sdio_data_config	1016
Table 3-1466. Function sdio_data_transfer_config	1017
Table 3-1467. Function sdio_dsm_enable	1018
Table 3-1468. Function sdio_dsm_disable	1018
Table 3-1469. Function sdio_data_write.....	1019
Table 3-1470. Function sdio_data_read	1020
Table 3-1471. Function sdio_data_counter_get.....	1020
Table 3-1472. Function sdio_fifo_reset_enable	1021
Table 3-1473. Function sdio_fifo_reset_disable	1021
Table 3-1474. Function sdio_idma_set.....	1022
Table 3-1475. Function sdio_idma_buffer0_address_set	1022
Table 3-1476. Function sdio_idma_buffer1_address_set	1023
Table 3-1477. Function sdio_buffer_selection_get	1024
Table 3-1478. Function sdio_idma_buffer_select.....	1024
Table 3-1479. Function sdio_idma_enable	1025
Table 3-1480. Function sdio_idma_disable	1025
Table 3-1481. Function sdio_flag_get.....	1026
Table 3-1482. Function sdio_flag_clear.....	1028
Table 3-1483. Function sdio_interrupt_enable.....	1029
Table 3-1484. Function sdio_interrupt_disable.....	1030
Table 3-1485. Function sdio_interrupt_flag_get	1031
Table 3-1486. Function sdio_interrupt_flag_clear	1033
Table 3-1487. Function sdio_voltage_switch_enable	1035
Table 3-1488. Function sdio_voltage_switch_disable	1035
Table 3-1489. Function sdio_voltage_switch_sequence_enable.....	1036
Table 3-1490. Function sdio_voltage_switch_sequence_disable	1036

Table 3-1491. Function sdio_boot_mode_set	1037
Table 3-1492. Function sdio_boot_ack_enable.....	1037
Table 3-1493. Function sdio_boot_ack_disable.....	1038
Table 3-1494. Function sdio_boot_acktimeout_set	1038
Table 3-1495. Function sdio_boot_enable	1039
Table 3-1496. Function sdio_boot_disable	1039
Table 3-1497. SPI/I2S Registers	1040
Table 3-1498. SPI/I2S firmware function.....	1041
Table 3-1499. spi_parameter_struct.....	1043
Table 3-1500. Function spi_i2s_deinit	1043
Table 3-1501. Function spi_struct_para_init	1044
Table 3-1502. Function spi_init	1044
Table 3-1503. Function spi_enable.....	1045
Table 3-1504. Function spi_disable.....	1046
Table 3-1505. Function i2s_init	1046
Table 3-1506. Function i2s_psc_config	1047
Table 3-1507. Function i2s_enable	1048
Table 3-1508. Function i2s_disable	1049
Table 3-1509. Function spi_io_config	1049
Table 3-1510. Function spi_nss_idleness_delay_set.....	1050
Table 3-1511. Function spi_data_frame_delay_set	1051
Table 3-1512. Function spi_master_receive_clock_delay_set.....	1051
Table 3-1513. Function spi_slave_receive_clock_delay_set.....	1052
Table 3-1514. Function spi_master_receive_clock_delay_clear.....	1052
Table 3-1515. Function spi_slave_receive_clock_delay_clear	1053
Table 3-1516. Function spi_nss_output_control	1053
Table 3-1517. Function spi_nss_polarity_set.....	1054
Table 3-1518. Function spi_nss_output_enable	1055
Table 3-1519. Function spi_nss_output_disable	1055
Table 3-1520. Function spi_nss_internal_high	1056
Table 3-1521. Function spi_nss_internal_low	1056
Table 3-1522. Function spi_dma_enable.....	1057
Table 3-1523. Function spi_dma_disable.....	1057
Table 3-1524. Function spi_i2s_data_frame_size_config	1058
Table 3-1525. Function spi_i2s_data_transmit	1059
Table 3-1526. Function spi_i2s_data_receive	1059
Table 3-1527. Function spi_bidirectional_transfer_config	1060
Table 3-1528. Function spi_master_transfer_start.....	1060
Table 3-1529. Function spi_current_data_num_config	1061
Table 3-1530. Function spi_reload_data_num_config.....	1061
Table 3-1531. Function spi_crc_polynomial_set.....	1062
Table 3-1532. Function spi_crc_polynomial_get	1063
Table 3-1533. Function spi_crc_length_config	1063
Table 3-1534. Function spi_crc_on	1064

Table 3-1535. Function spi_crc_off	1064
Table 3-1536. Function spi_crc_get	1065
Table 3-1537. Function spi_crc_full_size_enable.....	1065
Table 3-1538. Function spi_crc_full_size_disable.....	1066
Table 3-1539. Function spi_tcr_init_pattern	1066
Table 3-1540. Function spi_rcrc_init_pattern	1067
Table 3-1541. Function spi_ti_mode_enable	1068
Table 3-1542. Function spi_ti_mode_disable	1068
Table 3-1543. Function spi_quad_enable.....	1069
Table 3-1544. Function spi_quad_disable.....	1069
Table 3-1545. Function spi_quad_write_enable.....	1070
Table 3-1546. Function spi_quad_read_enable.....	1070
Table 3-1547. Function spi_quad_io23_output_enable	1071
Table 3-1548. Function spi_quad_io23_output_disable	1071
Table 3-1549. Function spi_underrun_operation.....	1072
Table 3-1550. Function spi_underrun_config.....	1072
Table 3-1551. Function spi_underrun_data_config.....	1073
Table 3-1552. Function spi_suspend_mode_config	1074
Table 3-1553. Function spi_suspend_request	1074
Table 3-1554. Function spi_related_ios_af_enable	1075
Table 3-1555. Function spi_related_ios_af_disable	1075
Table 3-1556. Function spi_af_gpio_control	1076
Table 3-1557. Function spi_i2s_interrupt_enable.....	1076
Table 3-1558. Function spi_i2s_interrupt_disable.....	1077
Table 3-1559. Function spi_i2s_interrupt_flag_get	1078
Table 3-1560. Function spi_i2s_flag_get	1079
Table 3-1561. Function spi_i2s_flag_clear.....	1080
Table 3-1562. Function spi_i2s_rxfifo_plevel_get	1081
Table 3-1563. Function spi_i2s_remain_data_num_get	1082
Table 3-1564. Function spi_fifo_threshold_level_set.....	1082
Table 3-1565. Function spi_word_access_enable	1083
Table 3-1566. Function spi_word_access_disable	1083
Table 3-1567. Function spi_byte_access_enable	1084
Table 3-1568. Function spi_byte_access_disable	1084
Table 3-1569. SYSCFG Registers	1085
Table 3-1570. SYSCFG firmware function	1086
Table 3-1571. Enum timer_channel_input_enum	1086
Table 3-1572. Function syscfg_deinit	1093
Table 3-1573. Function syscfg_i2c_fast_mode_plus_enable.....	1093
Table 3-1574. Function syscfg_i2c_fast_mode_plus_disable.....	1094
Table 3-1575. Function syscfg_analog_switch_enable	1095
Table 3-1576. Function syscfg_analog_switch_disable.....	1096
Table 3-1577. Function syscfg_enet_phy_interface_config	1096
Table 3-1578. Function syscfg_exti_line_config.....	1097

Table 3-1579. Function syscfg_lockup_enable	1098
Table 3-1580. Function syscfg_lockup_disable	1098
Table 3-1581. Function syscfg_timer_input_source_select	1099
Table 3-1582. Function syscfg_compensation_config	1100
Table 3-1583. Function syscfg_io_low_voltage_speed_optimization_enable	1100
Table 3-1584. Function syscfg_io_low_voltage_speed_optimization_disable	1101
Table 3-1585. Function syscfg_pnmos_compensation_code_set	1101
Table 3-1586. Function syscfg_secure_sram_size_set	1102
Table 3-1587. Function syscfg_secure_sram_size_get	1103
Table 3-1588. Function syscfg_bootmode_get	1103
Table 3-1589. Function syscfg_tcm_wait_state_enable	1104
Table 3-1590. Function syscfg_tcm_wait_state_disable	1104
Table 3-1591. Function syscfg_fpu_interrupt_enable	1105
Table 3-1592. Function syscfg_fpu_interrupt_disable	1106
Table 3-1593. Function syscfg_compensation_flag_get	1106
Table 3-1594. Function syscfg_cpu_cache_status_get	1107
Table 3-1595. Function syscfg_brownout_reset_threshold_level_get	1108
Table 3-1596. TIMERx Registers	1109
Table 3-1597. TIMERx firmware function	1110
Table 3-1598. Structure timer_parameter_struct	1113
Table 3-1599. Structure timer_break_parameter_struct	1114
Table 3-1600. Structure timer_oc_parameter_struct	1115
Table 3-1601. Structure timer_omc_parameter_struct	1115
Table 3-1602. Structure timer_ic_parameter_struct	1115
Table 3-1603. Structure timer_free_complementary_parameter_struct	1116
Table 3-1604. Function timer_deinit	1116
Table 3-1605. Function timer_struct_para_init	1116
Table 3-1606. Function timer_init	1117
Table 3-1607. Function timer_enable	1118
Table 3-1608. Function timer_disable	1118
Table 3-1609. Function timer_auto_reload_shadow_enable	1119
Table 3-1610. Function timer_auto_reload_shadow_disable	1120
Table 3-1611. Function timer_update_event_enable	1120
Table 3-1612. Function timer_update_event_disable	1121
Table 3-1613. Function timer_counter_alignment	1121
Table 3-1614. Function timer_counter_up_direction	1122
Table 3-1615. Function timer_counter_down_direction	1122
Table 3-1616. Function timer_prescaler_config	1123
Table 3-1617. Function timer_repetition_value_config	1124
Table 3-1618. Function timer_runtime_repetition_value_read	1124
Table 3-1619. Function timer_autoreload_value_config	1125
Table 3-1620. Function timer_autoreload_value_read	1126
Table 3-1621. Function timer_counter_value_config	1126
Table 3-1622. Function timer_counter_read	1127

Table 3-1623. Function timer_prescaler_read	1128
Table 3-1624. Function timer_single_pulse_mode_config	1128
Table 3-1625. Function timer_delayable_single_pulse_mode_config	1129
Table 3-1626. Function timer_update_source_config	1130
Table 3-1627. Function timer_dma_enable	1131
Table 3-1628. Function timer_dma_disable	1131
Table 3-1629. Function timer_channel_dma_request_source_select	1132
Table 3-1630. Function timer_dma_transfer_config	1133
Table 3-1631. Function timer_event_software_generate	1136
Table 3-1632. Function timer_break_struct_para_init	1137
Table 3-1633. Function timer_break_config	1138
Table 3-1634. Function timer_break_enable	1139
Table 3-1635. Function timer_break_disable	1139
Table 3-1636. Function timer_automatic_output_enable	1140
Table 3-1637. Function timer_automatic_output_disable	1141
Table 3-1638. Function timer_primary_output_config	1141
Table 3-1639. Function timer_channel_control_shadow_config	1142
Table 3-1640. Function timer_channel_control_shadow_update_config	1142
Table 3-1641. Function timer_channel_output_struct_para_init	1143
Table 3-1642. Function timer_channel_output_config	1144
Table 3-1643. Function timer_channel_output_mode_config	1145
Table 3-1644. Function timer_channel_output_pulse_value_config	1146
Table 3-1645. Function timer_channel_output_shadow_config	1147
Table 3-1646. Function timer_channel_output_clear_config	1148
Table 3-1647. Function timer_channel_output_polarity_config	1149
Table 3-1648. Function timer_channel_complementary_output_polarity_config	1150
Table 3-1649. Function timer_channel_output_state_config	1151
Table 3-1650. Function timer_channel_complementary_output_state_config	1152
Table 3-1651. Function timer_channel_input_struct_para_init	1153
Table 3-1652. Function timer_input_capture_config	1153
Table 3-1653. Function timer_channel_input_capture_prescaler_config	1154
Table 3-1654. Function timer_channel_capture_value_register_read	1155
Table 3-1655. Function timer_input_pwm_capture_config	1156
Table 3-1656. Function timer_hall_mode_config	1157
Table 3-1657. Function timer_multi_mode_channel_output_parameter_struct_init	1158
Table 3-1658. Function timer_multi_mode_channel_output_config	1158
Table 3-1659. Function timer_multi_mode_channel_mode_config	1159
Table 3-1660. Function timer_input_trigger_source_select	1160
Table 3-1661. Function timer_master_output0_trigger_source_select	1162
Table 3-1662. Function timer_master_output1_trigger_source_select	1163
Table 3-1663. Function timer_slave_mode_select	1164
Table 3-1664. Function timer_master_slave_mode_config	1165
Table 3-1665. Function timer_external_trigger_config	1165
Table 3-1666. Function timer_quadrature_decoder_mode_config	1166

Table 3-1667. Function timer_non_quadrature_decoder_mode_config.....	1168
Table 3-1668. Function timer_internal_clock_config	1168
Table 3-1669. Function timer_internal_trigger_as_external_clock_config	1169
Table 3-1670. Function timer_external_trigger_as_external_clock_config	1170
Table 3-1671. Function timer_external_clock_mode0_config	1171
Table 3-1672. Function timer_external_clock_mode1_config	1172
Table 3-1673. Function timer_external_clock_mode1_disable	1173
Table 3-1674. Function timer_write_chxval_register_config.....	1174
Table 3-1675. Function timer_output_value_selection_config	1175
Table 3-1676. Function timer_commutation_control_shadow_register_config.....	1175
Table 3-1677. Function timer_output_match_pulse_select	1176
Table 3-1678. Function timer_channel_composite_pwm_mode_config	1177
Table 3-1679. Function timer_channel_composite_pwm_mode_output_pulse_value_config	1178
Table 3-1680. Function timer_channel_additional_compare_value_config	1179
Table 3-1681. Function timer_channel_additional_output_shadow_config	1179
Table 3-1682. Function timer_channel_additional_compare_value_read	1180
Table 3-1683. Function timer_break_external_source_config	1181
Table 3-1684. Function timer_break_external_polarity_config	1182
Table 3-1685. Function timer_break_lock_config.....	1183
Table 3-1686. Function timer_break_lock_release_config	1184
Table 3-1687. Function timer_channel_break_control_config	1184
Table 3-1688. Function timer_channel_dead_time_config	1185
Table 3-1689. Function timer_free_complementary_struct_para_init	1186
Table 3-1690. Function timer_channel_free_complementary_config	1187
Table 3-1691. Function timer_watchdog_value_config	1187
Table 3-1692. Function timer_watchdog_value_read	1188
Table 3-1693. Function timer_decoder_disconnection_detection_config	1189
Table 3-1694. Function timer_decoder_jump_detection_config	1189
Table 3-1695. Function timer_upif_backup_config	1190
Table 3-1696. Function timer_upifbu_bit_get.....	1191
Table 3-1697. Function timer_flag_get	1191
Table 3-1698. Function timer_flag_clear	1193
Table 3-1699. Function timer_interrupt_enable	1194
Table 3-1700. Function timer_interrupt_disable	1195
Table 3-1701. Function timer_interrupt_flag_get.....	1197
Table 3-1702. Function timer_interrupt_flag_clear.....	1198
Table 3-1703. TLI Registers	1200
Table 3-1704. TLI firmware function.....	1200
Table3-1705. Structure tli_parameter_struct	1201
Table3-1706. Structure tli_layer_parameter_struct	1202
Table3-1707. Structure tli_layer_lut_parameter_struct	1202
Table 3-1708. Function tli_deinit.....	1203
Table 3-1709. Function tli_struct_para_init	1203
Table 3-1710. Function tli_init	1204

Table 3-1711. Function tli_dither_config	1205
Table 3-1712. Function tli_enable	1205
Table 3-1713. Function tli_disable	1206
Table 3-1714. Function tli_reload_config	1206
Table 3-1715. Function tli_layer_struct_para_init	1207
Table 3-1716. Function tli_layer_init	1208
Table 3-1717. Function tli_layer_window_offset_modify	1209
Table 3-1718. Function tli_lut_struct_para_init	1210
Table 3-1719. Function tli_lut_init	1210
Table 3-1720. Function tli_color_key_init	1211
Table 3-1721. Function tli_layer_enable	1212
Table 3-1722. Function tli_layer_disable	1212
Table 3-1723. Function tli_color_key_enable	1213
Table 3-1724. Function tli_color_key_disable	1213
Table 3-1725. Function tli_lut_enable	1214
Table 3-1726. Function tli_lut_disable	1214
Table 3-1727. Function tli_line_mark_set	1215
Table 3-1728. Function tli_current_pos_get	1215
Table 3-1729. Function tli_interrupt_enable	1216
Table 3-1730. Function tli_interrupt_disable	1216
Table 3-1731. Function tli_interrupt_flag_get	1217
Table 3-1732. Function tli_interrupt_flag_clear	1217
Table 3-1733. Function tli_flag_get	1218
Table 3-1734. TMU Registers	1219
Table 3-1735. TMU firmware function	1219
Table 3-1736. Structure tmu_parameter_struct	1220
Table 3-1737. Function tmu_deinit	1220
Table 3-1738. Function tmu_struct_para_init	1221
Table 3-1739. Function tmu_init	1221
Table 3-1740. Function tmu_read_interrupt_enable	1222
Table 3-1741. Function tmu_read_interrupt_disable	1222
Table 3-1742. Function tmu_dma_read_enable	1223
Table 3-1743. Function tmu_dma_read_disable	1223
Table 3-1744. Function tmu_dma_write_enable	1224
Table 3-1745. Function tmu_dma_write_disable	1224
Table 3-1746. Function tmu_one_q31_write	1225
Table 3-1747. Function tmu_two_q31_write	1225
Table 3-1748. Function tmu_two_q15_write	1226
Table 3-1749. Function tmu_one_q31_read	1227
Table 3-1750. Function tmu_two_q31_read	1227
Table 3-1751. Function tmu_two_q15_read	1228
Table 3-1752. TRIGSEL Registers	1229
Table 3-1753. TRIGSEL firmware function	1230
Table 3-1754. Enum trigselsource_enum	1230

Table 3-1755. Enum trigsel_periph_enum	1234
Table 3-1756. Function trigsel_deinit.....	1236
Table 3-1757. Function trigsel_init	1236
Table 3-1758. Function trigsel_trigger_source_get	1237
Table 3-1759. Function trigsel_register_lock_set.....	1237
Table 3-1760. Function trigsel_register_lock_get	1238
Table 3-1761. TRNG Registers	1238
Table 3-1762. TRNG firmware function.....	1239
Table 3-1763. Enum trng_inmod_enum.....	1239
Table 3-1764. Enum trng_outmod_enum	1240
Table 3-1765. Enum trng_modsel_enum.....	1240
Table 3-1766. Enum trng_flag_enum	1240
Table 3-1767. Enum trng_int_flag_enum.....	1240
Table 3-1768. Function trng_deinit.....	1240
Table 3-1769. Function trng_enable.....	1241
Table 3-1770. Function trng_disable.....	1241
Table 3-1771. Function trng_lock	1242
Table 3-1772. Function trng_mode_config	1242
Table 3-1773. Function trng_postprocessing_enable	1243
Table 3-1774. Function trng_postprocessing_disable	1244
Table 3-1775. Function trng_conditioning_enable	1245
Table 3-1776. Function trng_conditioning_disable	1245
Table 3-1777. Function trng_disable.....	1246
Table 3-1778. Function trng_conditioning_output_bitwidth.....	1247
Table 3-1779. Function trng_replace_test_enable.....	1248
Table 3-1780. Function trng_replace_test_disable.....	1248
Table 3-1781. Function trng_hash_init_enable	1249
Table 3-1782. Function trng_hash_init_disable	1250
Table 3-1783. Function trng_powermode_config	1250
Table 3-1784. Function trng_disable.....	1251
Table 3-1785. Function trng_clockerror_detection_enable	1252
Table 3-1786. Function trng_clockerror_detection_disable	1252
Table 3-1787. Function trng_get_true_random_data	1253
Table 3-1788. Function trng_conditioning_reset_enable	1254
Table 3-1789. Function trng_conditioning_reset_disable	1254
Table 3-1790. Function trng_conditioning_algo_config.....	1255
Table 3-1791. Function trng_health_tests_config	1256
Table 3-1792. Function trng_flag_get	1257
Table 3-1793. Function trng_interrupt_enable	1257
Table 3-1794. Function trng_interrupt_disable	1258
Table 3-1795. Function trng_interrupt_flag_get.....	1258
Table 3-1796. Function trng_interrupt_flag_clear	1259
Table 3-1797. USART Registers	1260
Table 3-1798. USART firmware function	1260

Table 3-1799. Enum usart_flag_enum	1262
Table 3-1800. Enum usart_interrupt_flag_enum	1263
Table 3-1801. Enum usart_interrupt_enum	1264
Table 3-1802. Enum usart_invert_enum	1265
Table 3-1803. Function usart_deinit	1265
Table 3-1804. Function usart_baudrate_set	1265
Table 3-1805. Function usart_parity_config	1266
Table 3-1806. Function usart_word_length_set	1267
Table 3-1807. Function usart_stop_bit_set	1267
Table 3-1808. Function usart_enable	1268
Table 3-1809. Function usart_disable	1269
Table 3-1810. Function usart_transmit_config	1269
Table 3-1811. Function usart_receive_config	1270
Table 3-1812. Function usart_data_first_config	1270
Table 3-1813. Function usart_invert_config	1271
Table 3-1814. Function usart_overrun_enable	1272
Table 3-1815. Function usart_overrun_disable	1272
Table 3-1816. Function usart_oversample_config	1273
Table 3-1817. Function usart_sample_bit_config	1273
Table 3-1818. Function usart_receiver_timeout_enable	1274
Table 3-1819. Function usart_receiver_timeout_disable	1274
Table 3-1820. Function usart_receiver_timeout_threshold_config	1275
Table 3-1821. Function usart_data_transmit	1276
Table 3-1822. Function usart_data_receive	1276
Table 3-1823. Function usart_command_enable	1277
Table 3-1824. Function usart_address_0_match_mode_enable	1277
Table 3-1825. Function usart_address_0_match_mode_disable	1278
Table 3-1826. Function usart_address_1_match_mode_enable	1278
Table 3-1827. Function usart_address_1_match_mode_disable	1279
Table 3-1828. Function usart_address_0_config	1280
Table 3-1829. Function usart_address_1_config	1280
Table 3-1830. Function usart_address_0_detection_mode_config	1281
Table 3-1831. Function usart_address_1_detection_mode_config	1281
Table 3-1832. Function usart_mute_mode_enable	1282
Table 3-1833. Function usart_mute_mode_disable	1283
Table 3-1834. Function usart_mute_mode_wakeup_config	1283
Table 3-1835. Function usart_lin_mode_enable	1284
Table 3-1836. Function usart_lin_mode_disable	1284
Table 3-1837. Function usart_lin_break_detection_length_config	1285
Table 3-1838. Function usart_halfduplex_enable	1285
Table 3-1839. Function usart_halfduplex_disable	1286
Table 3-1840. Function usart_clock_enable	1287
Table 3-1841. Function usart_clock_disable	1287
Table 3-1842. Function usart_synchronous_clock_config	1288

Table 3-1843. Function usart_guard_time_config	1288
Table 3-1844. Function usart_smartcard_mode_enable	1289
Table 3-1845. Function usart_smartcard_mode_disable	1289
Table 3-1846. Function usart_smartcard_mode_nack_enable.....	1290
Table 3-1847. Function usart_smartcard_mode_nack_disable	1290
Table 3-1848. Function usart_smartcard_mode_early_nack_enable.....	1291
Table 3-1849. Function usart_smartcard_mode_early_nack_disable.....	1291
Table 3-1850. Function usart_smartcard_autoretry_config.....	1292
Table 3-1851. Function usart_block_length_config	1293
Table 3-1852. Function usart_irda_mode_enable.....	1293
Table 3-1853. Function usart_irda_mode_disable.....	1294
Table 3-1854. Function usart_prescaler_config.....	1294
Table 3-1855. Function usart_irda_lowpower_config	1295
Table 3-1856. Function usart_hardware_flow_rts_config	1295
Table 3-1857. Function usart_hardware_flow_cts_config	1296
Table 3-1858. Function usart_hardware_flow_coherence_config	1297
Table 3-1859. Function usart_rs485_driver_enable	1297
Table 3-1860. Function usart_rs485_driver_disable	1298
Table 3-1861. Function usart_driver_asserttime_config	1298
Table 3-1862. Function usart_driver_deasserttime_config.....	1299
Table 3-1863. Function usart_depolarity_config	1300
Table 3-1864. Function usart_dma_receive_config	1300
Table 3-1865. Function usart_dma_transmit_config.....	1301
Table 3-1866. Function usart_reception_error_dma_disable.....	1302
Table 3-1867. Function usart_reception_error_dma_enable.....	1302
Table 3-1868. Function usart_wakeup_enable	1303
Table 3-1869. Function usart_wakeup_disable	1303
Table 3-1870. Function usart_wakeup_mode_config	1304
Table 3-1871. Function usart_fifo_enable	1304
Table 3-1872. Function usart_fifo_disable.....	1305
Table 3-1873. Function usart_transmit_fifo_threshold_config	1305
Table 3-1874. Function usart_receive_fifo_threshold_config	1306
Table 3-1875. Function usart_receive_fifo_counter_number.....	1307
Table 3-1876. Function usart_flag_get	1308
Table 3-1877. Function usart_flag_clear	1308
Table 3-1878. Function usart_interrupt_enable	1309
Table 3-1879. Function usart_interrupt_disable	1310
Table 3-1880. Function usart_interrupt_flag_get.....	1311
Table 3-1881. Function usart_interrupt_flag_clear.....	1311
Table 3-1882. VREF Registers	1313
Table 3-1883. VREF firmware function	1313
Table 3-1884. Function vref_deinit	1313
Table 3-1885. Function vref_enable	1314
Table 3-1886. Function vref_disable	1314



Table 3-1887. Function vref_high_impedance_mode_enable	1315
Table 3-1888. Function vref_high_impedance_mode_disable	1315
Table 3-1889. Function vref_status_get	1316
Table 3-1890. Function vref_voltage_select	1316
Table 3-1891. Function vref_calib_value_set	1317
Table 3-1892. Function vref_calib_value_get	1317
Table 3-1893. WWDGT Registers	1318
Table 3-1894. WWDGT firmware function	1318
Table 3-1895. Function wwdgt_deinit	1319
Table 3-1896. Function wwdgt_enable	1319
Table 3-1897. Function wwdgt_counter_update	1320
Table 3-1898. Function wwdgt_config	1320
Table 3-1899. Function wwdgt_interrupt_enable	1321
Table 3-1900. Function wwdgt_flag_get	1321
Table 3-1901. Function wwdgt_flag_clear	1322
Table 4-1. Revision history	1323

1. Introduction

This manual introduces firmware library of GD32H7xx 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 GD32H7xx devices are involved in the package. The peripheral driving code and firmware examples on evaluation board are also included in firmware library. Users need not learn each peripherals in details and it's easy to apply a peripheral by using the firmware library. Using firmware library can greatly reduce programming time, thereby reducing development costs.

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

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

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

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

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

1.1. Rules of User Manual and Firmware Library

1.1.1. Peripherals

Table 1-1. Peripherals

Peripherals	Descriptions
ADC	Analog-to-digital converter
CAN	Controller area network
CAU	Cryptographic Acceleration Unit
CMP	Comparator
CPDM	Clock phase delay module

Peripherals	Descriptions
CRC	CRC calculation unit
CTC	Clock trim controller
DAC	Digital-to-analog converter
DBG	Debug
DCI	Digital camera interface
DMA	Direct memory access controller
DMAMUX	DMA request multiplexer
EDOUT	Encoder Divided-Output controller
EFUSE	Electronic fuse
ENET	Ethernet
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
HAU	Hash Acceleration Unit
HPDF	High-Performance Digital Filter
HWSEM	Hardware semaphore
I2C	Inter-integrated circuit interface
IPA	Image processing accelerator
LPDTS	Low power digital temperature sensor
MDIO	Management data input / output
MDMA	Master direct memory access controller
OSPI	Octal-SPI interface
OSPIM	OSPI I/O manager
MISC	Nested Vectored Interrupt Controller
PMU	Power management unit
RAMECCMU	RAM ECC monitor unit
RCU	Reset and clock unit
RSPDIF	Reset and clock unit
RTC	Reset and clock unit
RTDEC	Real-time decryption
SAI	Serial Audio Interface
SDIO	Secure digital input/output interface
SPI/I2S	Secure digital input/output interface
SYSCFG	System configuration
TIMER	TIMER
TLI	TFT-LCD interface
TMU	Trigonometric Math Unit
TRIGSEL	Trigger selection controller

Peripherals	Descriptions
TRNG	True random number generator
USART	Universal synchronous / asynchronous receiver / transmitter
VREF	VREF
WWDGT	Window watchdog timer

1.1.2. Naming rules

The firmware library naming rules are shown as below:

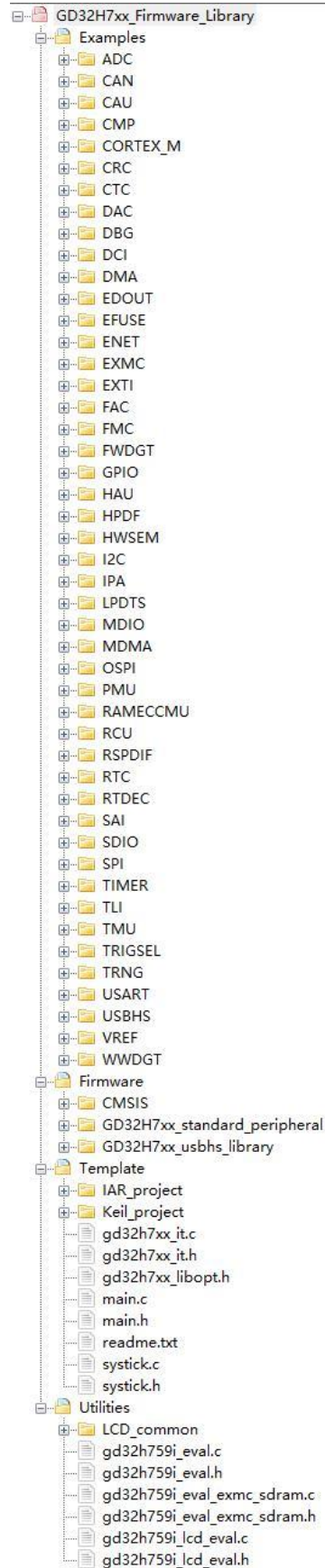
- The peripherals are shortened in XXX format, such as: ADC. More shorten information of peripherals refer to [Peripherals](#);
- The name of sourcefile and header file are started with “gd32h7xx_”, such as: gd32h7xx_adc.h;
- The constants used only in one file should be defined in the used file; the 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

GD32H7xx_Firmware_Library, the file structure is shown as below:

Figure 2-1. File structure of firmware library of GD32H7xx



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;
- `gd32h7xx_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);
- `gd32h7xx_it.c`: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- `gd32h7xx_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 GD32H7xx and system configuration file;
- GD32H7xx_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.

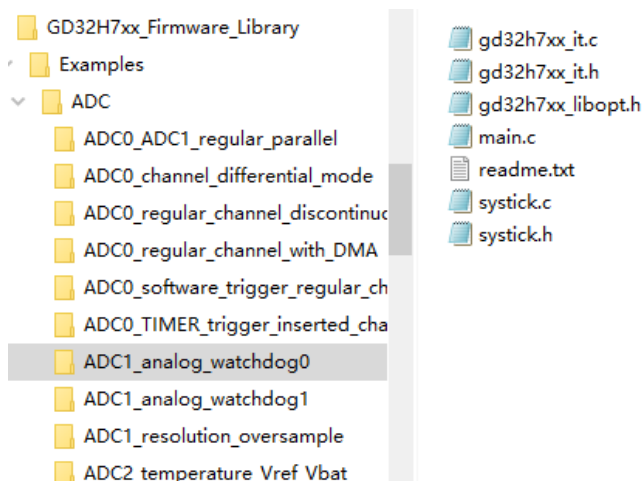
2.1.3. Template Folder

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:

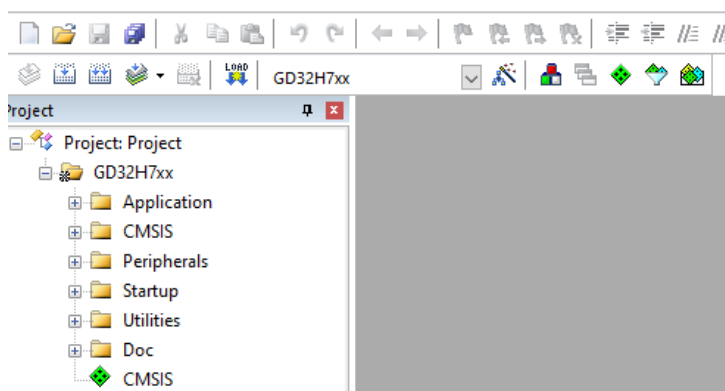
Figure 2-3. Copy the peripheral example files



Open a project

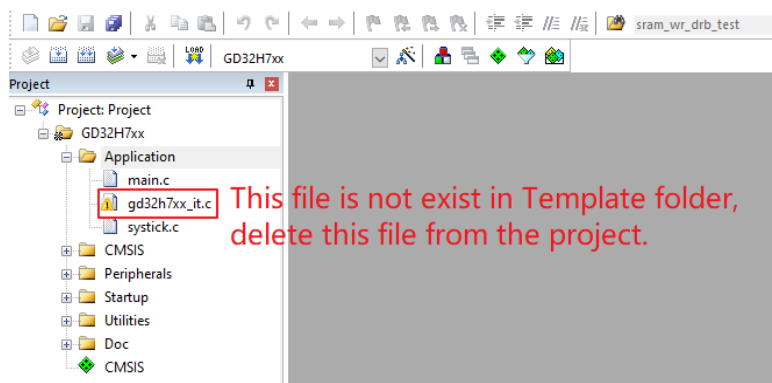
GD provides project in Keil and IAR, users can open project in different IDEs according to their need, such as ”Keil_project”, open \Template\Keil_project\Project.uvprojx, shown as below:

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:

- gd32h759i_eval.h is related header file of the evaluation board about running the firmware examples;
- gd32h759i_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
gd32h7xx_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.
gd32h7xx_it.h	Header file, including all the prototypes of interrupt service routines.
gd32h7xx_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.
gd32h7xx_xxx.h	The header file of peripheral xxx, including functions about peripheral xxx, and the variables used for functions.
gd32h7xx_xxx.c	The C source file for driving peripheral xxx.
systick.h	The header file of systick.c, including prototypes of systick configuration function and delay function.
systick.c	The source file about systick configuration function and delay function.
readme.txt	Description document about how to configure and how to use the firmware example.

3. Firmware Library of Standard Peripherals

3.1. Overview of Firmware Library of Standard Peripherals

The description format of firmware functions are shown as below:

Table 3-1. Peripheral function format of Firmware Library

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

3.2. ADC

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

3.2.1. Descriptions of Peripheral registers

ADC registers are listed in the table shown as below:

Table 3-2. ADC Registers

Registers	Descriptions
ADC_STAT	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 regular sequence register x(x=0..8)

Registers	Descriptions
ADC_ISQx	ADC inserted sequence register x(x=0..2)
ADC_IDATAx	ADC inserted data register x(x=0..3)
ADC_RDATA	ADC regular 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 regular data register 0
ADC_SYNCDATA1	ADC sync regular 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 regular conversion
adc_dma_request_after_last_disable	the DMA engine is disabled after the end of transfer signal from DMA controller is detected
adc_hpdf_mode_enable	enable hpdf mode
adc_hpdf_mode_disable	disable hpdf mode
adc_discontinuous_mode_config	configure ADC discontinuous mode

Function name	Function description
adc_channel_length_config	configure the length of regular channel group or inserted channel group
adc_regular_channel_config	configure ADC regular channel
adc_inserted_channel_config	configure ADC inserted channel
adc_inserted_channel_offset_config	configure ADC inserted channel offset
adc_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_regular_data_read	read ADC regular group data register
adc_inserted_data_read	read ADC inserted group 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

Function name	Function description
adc_sync_master_adc_regular_data0_read	read ADC sync master adc regular data register 0
adc_sync_slave_adc_regular_data0_read	read ADC sync slave adc regular data register 0
adc_sync_regular_data1_read	read ADC sync regular 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_HCL K_DIV2	ADC sync clock mode HCLK div2
ADC_CLK_SYNC_HCL K_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
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

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

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

adc_special_function_config

The description of adc_special_function_config is shown as below:

Table 3-6. Function adc_special_function_config

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

Example:

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

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

adc_data_alignment_config

The description of adc_data_alignment_config is shown as below:

Table 3-7. Function `adc_data_alignment_config`

Function name	<code>adc_data_alignment_config</code>
Function prototype	<code>void adc_data_alignment_config(uint32_t adc_periph, uint32_t data_alignment);</code>
Function descriptions	configure ADCx data alignment
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>data_alignment</code>	data alignment select
<code>ADC_DATAALIGN_RIGHT</code>	LSB alignment
<code>ADC_DATAALIGN_LEFT</code>	MSB alignment
Output parameter{out}	
-	-
Return value	
-	-

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`

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

Example:

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

adc_disable

The description of adc_disable is shown as below:

Table 3-9. Function adc_disable

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

Example:

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

adc_calibration_mode_config

The description of adc_calibration_mode_config is shown as below:

Table 3-10. Function adc_calibration_mode_config

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

ADC_CALIBRATION_OFFSET	ADC calibration offset mode
Output parameter{out}	
-	-
Return value	
-	-

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

Function name	adc_calibration_number
Function prototype	void adc_calibration_number(uint32_t adc_periph, uint32_t clb_num);
Function descriptions	configure ADC calibration number
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
clb_num	calibration number
ADC_CALIBRATION_NUMBER1	calibrate once
ADC_CALIBRATION_NUMBER2	calibrate twice
ADC_CALIBRATION_NUMBER4	calibrate 4 times
ADC_CALIBRATION_NUMBER8	calibrate 8 times
ADC_CALIBRATION_NUMBER16	calibrate 16 times
ADC_CALIBRATION_NUMBER32	calibrate 32 times
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 calibration number */

adc_calibration_number(ADC0, ADC_CALIBRATION_NUM1);
```

adc_calibration_enable

The description of adc_calibration_enable is shown as below:

Table 3-12. Function adc_calibration_enable

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

Example:

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

adc_calibration_enable(ADC0);
```

adc_resolution_config

The description of adc_resolution_config is shown as below:

Table 3-13. Function adc_resolution_config

Function name	adc_resolution_config
Function prototype	void adc_resolution_config(uint32_t adc_periph, uint32_t resolution);
Function descriptions	configure ADC resolution
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
Input parameter{in}	
resolution	ADC resolution
ADC_RESOLUTION_14B	14-bit ADC resolution, for ADC0/ADC1

<i>ADC_RESOLUTION_12B</i>	12-bit ADC resolution, for all ADCs
<i>ADC_RESOLUTION_10B</i>	10-bit ADC resolution, for all ADCs
<i>ADC_RESOLUTION_8B</i>	8-bit ADC resolution, for all ADCs
<i>ADC_RESOLUTION_6B</i>	6-bit ADC resolution, only for ADC2
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

adc_internal_channel_config

The description of `adc_internal_channel_config` is shown as below:

Table 3-14. Function `adc_internal_channel_config`

Function name	<code>adc_internal_channel_config</code>
Function prototype	<code>void adc_internal_channel_config(uint32_t internal_channel, ControlStatus newvalue);</code>
Function descriptions	enable or disable ADC internal channels
Precondition	-
The called functions	-
Input parameter{in}	
internal_channel	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
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function

Output parameter{out}	
-	-
Return value	
-	-

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

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

Example:

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

```
adc_dma_mode_enable(ADC0);
```

adc_dma_mode_disable

The description of adc_dma_mode_disable is shown as below:

Table 3-16. Function adc_dma_mode_disable

Function name	adc_dma_mode_disable
Function prototype	void adc_dma_mode_disable(uint32_t adc_periph);
Function descriptions	disable ADC DMA request
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral

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

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

Function name	adc_dma_request_after_last_enable
Function prototype	void adc_dma_request_after_last_enable(uint32_t adc_periph);
Function descriptions	when DMA=1, the DMA engine issues a request at end of each regular conversion
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:

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

Function name	adc_dma_request_after_last_disable
Function prototype	void adc_dma_request_after_last_disable(uint32_t adc_periph);
Function descriptions	the DMA engine is disabled after the end of transfer signal from DMA controller is detected
Precondition	-

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

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

Function name	adc_hpdf_mode_enable
Function prototype	void adc_hpdf_mode_enable(uint32_t adc_periph);
Function descriptions	enable hpdf mode
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:

```
/* 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

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

Function descriptions	disable hpdf mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
Output parameter{out}	
-	-
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_channel_group, uint32_t length);
Function descriptions	configure ADC discontinuous mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
<i>ADC_REGULAR_CHANNEL</i>	regular channel group
<i>ADC_INSERTED_CHANNEL</i>	inserted channel group
<i>ADC_CHANNEL_DISCON_DISABLE</i>	disable discontinuous mode of regular and inserted channel
Input parameter{in}	
length	number of conversions in discontinuous mode, the number can be 1..8 for regular channel, the number has no effect for inserted channel
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* configure ADC0 regular channel group discontinuous mode */
```

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

adc_channel_length_config

The description of adc_channel_length_config is shown as below:

Table 3-22. Function adc_channel_length_config

Function name	adc_channel_length_config
Function prototype	void adc_channel_length_config(uint32_t adc_periph, uint8_t adc_channel_group, uint32_t length);
Function descriptions	configure the length of regular channel group or inserted channel group
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
length	the length of the channel, regular channel 1-16, inserted channel 1-4
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

adc_regular_channel_config

The description of adc_regular_channel_config is shown as below:

Table 3-23. Function adc_regular_channel_config

Function name	adc_regular_channel_config
Function prototype	void adc_regular_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t

	adc_channel, uint32_t sample_time);
Function descriptions	configure ADC regular channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
Input parameter{in}	
rank	the regular group sequence rank, this parameter must be between 0 to 15
Input parameter{in}	
adc_channel	the selected ADC channel
<i>ADC_CHANNEL_x</i> (x=0..20)	ADC Channelx
Input parameter{in}	
sample_time	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
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
adc_regular_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

Function name	adc_inserted_channel_config
Function prototype	void adc_inserted_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
Function descriptions	configure ADC inserted channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
Input parameter{in}	
rank	the inserted group sequencer rank, this parameter must be between 0 to 3
Input parameter{in}	

adc_channel	the selected ADC channel
<i>ADC_CHANNEL_x</i> (x=0..20)	ADC Channelx
Input parameter{in}	
sample_time	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
Output parameter{out}	
-	-
Return value	
-	-

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

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

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

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

Function name	adc_external_trigger_config
Function prototype	void adc_external_trigger_config(uint32_t adc_periph, uint8_t adc_channel_group, uint32_t trigger_mode)
Function descriptions	configure ADC external trigger

Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
<i>ADC_REGULAR_CHANNEL</i>	regular channel group
<i>ADC_INSERTED_CHANNEL</i>	inserted channel group
Input parameter{in}	
trigger_mode	external trigger mode
<i>EXTERNAL_TRIGGER_DISABLE</i>	external trigger disable
<i>EXTERNAL_TRIGGER_RISING</i>	rising edge of external trigger
<i>EXTERNAL_TRIGGER_FALLING</i>	falling edge of external trigger
<i>EXTERNAL_TRIGGER_RISING_FALLING</i>	rising and falling edge of external trigger
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config ADC0 inserted channel group 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

Function name	adc_software_trigger_enable
Function prototype	void adc_software_trigger_enable(uint32_t adc_periph, uint8_t adc_channel_group);
Function descriptions	enable ADC software trigger
Precondition	-
The called functions	-
Input parameter{in}	

adc_periph	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
<i>ADC_REGULAR_CHANNEL</i>	regular channel group
<i>ADC_INSERTED_CHANNEL</i>	inserted channel group
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC0 regular channel group software trigger */
```

```
adc_software_trigger_enable(ADC0, ADC_REGULAR_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

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

Example:

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

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

adc_regular_data_read

The description of adc_regular_data_read is shown as below:

Table 3-30. Function adc_regular_data_read

Function name	adc_regular_data_read
Function prototype	uint32_t adc_regular_data_read(uint32_t adc_periph);
Function descriptions	read ADC regular group data register
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
uint32_t	ADC conversion value (0~0xFFFF)

Example:

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

```
uint32_t adc_value = 0;
```

```
adc_value = adc_regular_data_read(ADC0);
```

adc_inserted_data_read

The description of adc_inserted_data_read is shown as below:

Table 3-31. Function adc_inserted_data_read

Function name	adc_inserted_data_read
Function prototype	uint32_t adc_inserted_data_read(uint32_t adc_periph, uint8_t inserted_channel);
Function descriptions	read ADC inserted group data register
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
Input parameter{in}	
inserted_channel	insert channel select
ADC_INSERTED_	inserted channelx, x=0,1,2,3

<i>CHANNEL_x</i> (<i>x</i> =0..3)	
Output parameter{out}	
-	-
Return value	
uint32_t	ADC conversion value (0~0xFFFF)

Example:

```
/* read ADC0 inserted group 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}	
adc_periph	ADC peripheral
<i>ADCx</i> (<i>x</i> =0, 1, 2)	ADC peripheral selection
Input parameter{in}	
adc_channel	the selected ADC channel
<i>ADC_CHANNEL_x</i> (<i>x</i> =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_channel_group);</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_channel_group</code>	the channel group use analog watchdog
<code>ADC_REGULAR_CHANNEL</code>	regular channel group
<code>ADC_INSERTED_CHANNEL</code>	inserted channel group
<code>ADC_REGULAR_INSERTED_CHANNEL</code>	both regular and inserted group
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

`adc_watchdog0_disable`

The description of `adc_watchdog0_disable` is shown as below:

Table 3-34. Function `adc_watchdog0_disable`

Function name	<code>adc_watchdog0_disable</code>
Function prototype	<code>void adc_watchdog0_disable(uint32_t adc_periph);</code>
Function descriptions	disable ADC analog watchdog 0
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_periph</code>	ADC peripheral
<code>ADCx(x=0,1,2)</code>	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}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
Input parameter{in}	
selection_channel	the channel use analog watchdog 1
ADC_AWD1_2_SELECTION_CHANNEL_x (x=0..20), ADC_AWD1_2_SELECTION_CHANNEL_ALL	ADC channel analog watchdog 1/2 selection
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

adc_watchdog2_channel_config

The description of adc_watchdog2_channel_config is shown as below:

Table 3-36. Function adc_watchdog2_channel_config

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

Function name	adc_watchdog1_disable
Function prototype	void adc_watchdog1_disable(uint32_t adc_periph);
Function descriptions	disable ADC analog watchdog 1
Precondition	-

The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	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

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

Function name	adc_watchdog0_threshold_config
Function prototype	void adc_watchdog0_threshold_config(uint32_t adc_periph , uint32_t low_threshold , uint32_t high_threshold);

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

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

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

Return value	
-	-

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

Function name	adc_watchdog2_threshold_config
Function prototype	void adc_watchdog2_threshold_config(uint32_t adc_periph , uint32_t low_threshold , uint32_t high_threshold);
Function descriptions	configure ADC analog watchdog 2 threshold
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
Input parameter{in}	
low_threshold	analog watchdog 2 low threshold, 0..0xFFFF for ADC0/ADC1, 0..0xFF for ADC2
Input parameter{in}	
high_threshold	analog watchdog 2 high threshold, 0..0xFFFF for ADC0/ADC1, 0..0xFF for ADC2
Output parameter{out}	
-	-
Return value	
-	-

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

Function name	adc_oversample_mode_config
Function prototype	void adc_oversample_mode_config(uint32_t adc_periph, uint32_t mode, uint16_t shift, uint16_t ratio);

Function descriptions	configure ADC oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
Input parameter{in}	
mode	ADC oversampling mode
<i>ADC_OVERSAMPLING _ALL_CONVERT</i>	all oversampled conversions for a channel are done consecutively after a trigger
<i>ADC_OVERSAMPLING _ONE_CONVERT</i>	each oversampled conversion for a channel needs a trigger
Input parameter{in}	
shift	ADC oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_NONE</i>	no oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_1B</i>	1-bit oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_2B</i>	2-bit oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_3B</i>	3-bit oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_4B</i>	4-bit oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_5B</i>	5-bit oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_6B</i>	6-bit oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_7B</i>	7-bit oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_8B</i>	8-bit oversampling shift
<i>ADC_OVERSAMPLING _SHIFT_9B</i>	9-bit oversampling shift, available for ADC0/ADC1
<i>ADC_OVERSAMPLING _SHIFT_10B</i>	10-bit oversampling shift, available for ADC0/ADC1
<i>ADC_OVERSAMPLING _SHIFT_11B</i>	11-bit oversampling shift, available for ADC0/ADC1
Input parameter{in}	
ratio	ADC oversampling ratio, 0..1023 corresponding 1x~1024x for ADC0/ADC1, 0..255 corresponding 1x~256x for ADC2
Output parameter{out}	

-	-
Return value	
-	-

Example:

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

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

adc_oversample_mode_enable

The description of adc_oversample_mode_enable is shown as below:

Table 3-43. Function adc_oversample_mode_enable

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

Example:

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

```
adc_oversample_mode_enable (ADC0);
```

adc_oversample_mode_disable

The description of adc_oversample_mode_disable is shown as below:

Table 3-44. Function adc_oversample_mode_disable

Function name	adc_oversample_mode_disable
Function prototype	void adc_oversample_mode_disable(uint32_t adc_periph);
Function descriptions	disable ADC oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral

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

Example:

```
/* disable ADC0 oversample mode */
adc_oversample_mode_disable (ADC0);
```

adc_flag_get

The description of adc_flag_get is shown as below:

Table 3-45. Function adc_flag_get

Function name	adc_flag_get
Function prototype	FlagStatus adc_flag_get(uint32_t adc_periph, uint32_t flag);
Function descriptions	get the ADC flag
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
Input parameter{in}	
flag	the adc flag bits
ADC_FLAG_WDE0	analog watchdog 0 event flag
ADC_FLAG_EOC	end of group conversion flag
ADC_FLAG_EOIC	end of inserted group conversion flag
ADC_FLAG_STIC	start flag of inserted channel group
ADC_FLAG_STRC	start flag of regular channel group
ADC_FLAG_ROVF	regular data register overflow flag
ADC_FLAG_WDE1	analog watchdog 1 event flag
ADC_FLAG_WDE2	analog watchdog 2 event flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the ADC0 analog watchdog 0 flag bits */
FlagStatus flag_value;
flag_value = adc_flag_get(ADC0, ADC_FLAG_WDE0);
```

adc_flag_clear

The description of adc_flag_clear is shown as below:

Table 3-46. Function adc_flag_clear

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

Example:

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

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

adc_interrupt_enable

The description of adc_interrupt_enable is shown as below:

Table 3-47. Function adc_interrupt_enable

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

ADCx(x=0,1,2)	ADC peripheral selection
Input parameter{in}	
adc_interrupt	the adc interrupt
ADC_INT_WDE0	analog watchdog 0 interrupt
ADC_INT_EOC	end of group conversion interrupt
ADC_INT_EOIC	end of inserted group conversion interrupt
ADC_INT_ROVF	regular data register overflow interrupt
ADC_INT_WDE1	analog watchdog 1 interrupt
ADC_INT_WDE2	analog watchdog 2 interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

adc_interrupt_disable

The description of adc_interrupt_disable is shown as below:

Table 3-48. Function adc_interrupt_disable

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

Example:

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

adc_interrupt_flag_get

The description of adc_interrupt_flag_get is shown as below:

Table 3-49. Function adc_interrupt_flag_get

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

Example:

```
/* get the ADC0 analog watchdog 0 interrupt bits*/
FlagStatus flag_value;
flag_value = adc_interrupt_flag_get(ADC0, ADC_INT_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`

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

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	<code>adc_sync_mode_config</code>
Function prototype	<code>void adc_sync_mode_config(uint32_t sync_mode);</code>
Function descriptions	configure the ADC sync mode
Precondition	-
The called functions	-

Input parameter{in}	
sync_mode	ADC sync mode
<i>ADC_SYNC_MODE_INDEPENDENT</i>	all the ADCs work independently
<i>ADC_DUAL_REGULAR_PARALLEL_INSERTED_PARALLEL</i>	ADC0 and ADC1 work in combined regular parallel & inserted parallel mode
<i>ADC_DUAL_REGULAR_PARALLEL_INSERTED_ROTATION</i>	ADC0 and ADC1 work in combined regular parallel & trigger rotation mode
<i>ADC_DUAL_INSERTED_PARALLEL</i>	ADC0 and ADC1 work in inserted parallel mode
<i>ADC_DUAL_REGULAR_PARALLEL</i>	ADC0 and ADC1 work in regular parallel mode
<i>ADC_DUAL_REGULAR_FOLLOW_UP</i>	ADC0 and ADC1 work in follow-up mode
<i>ADC_DUAL_INSERTED_TRIGGER_ROTATION</i>	ADC0 and ADC1 work in trigger rotation mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
adc_sync_mode_config (ADC_DUAL_REGULAR_PARALLEL_INSERTED_PARALLEL);
```

adc_sync_delay_config

The description of adc_sync_delay_config is shown as below:

Table 3-52. Function adc_interrupt_disable

Function name	adc_sync_delay_config
Function prototype	void adc_sync_delay_config(uint32_t sample_delay);

Function descriptions	configure the delay between 2 sampling phases in ADC sync modes
Precondition	-
The called functions	-
Input parameter{in}	
sample_delay	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
Output parameter{out}	
-	-
Return value	
-	-

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

Function name	adc_sync_dma_config
Function prototype	void adc_sync_dma_config(uint32_t dma_mode);
Function descriptions	configure ADC sync DMA mode selection
Precondition	-
The called functions	-
Input parameter{in}	
dma_mode	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
Output parameter{out}	
-	-
Return value	
-	-

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

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

Example:

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

```
adc_sync_dma_request_after_last_enable();
```

adc_sync_dma_request_after_last_disable

The description of adc_sync_dma_request_after_last_disable is shown as below:

Table 3-55. Function adc_sync_dma_request_after_last_disable

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

-	-
---	---

Example:

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

```
adc_sync_dma_request_after_last_disable();
```

adc_sync_master_adc_regular_data0_read

The description of adc_sync_master_adc_regular_data0_read is shown as below:

Table 3-56. Function adc_sync_master_adc_regular_data0_read

Function name	adc_sync_master_adc_regular_data0_read
Function prototype	uint16_t adc_sync_master_adc_regular_data0_read (void);
Function descriptions	read ADC sync master adc regular data register 0
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	sync regular data 0

Example:

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

```
adc_sync_master_adc_regular_data0_read ();
```

adc_sync_slave_adc_regular_data0_read

The description of adc_sync_slave_adc_regular_data0_read is shown as below:

Table 3-57. Function adc_sync_slave_adc_regular_data0_read

Function name	adc_sync_slave_adc_regular_data0_read
Function prototype	uint16_t adc_sync_slave_adc_regular_data0_read (void);
Function descriptions	read ADC sync slave adc regular data register 0
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

uint16_t	sync regular data 0
----------	---------------------

Example:

```
/* read ADC sync slave adc regular data register0 */
```

```
adc_sync_slave_adc_regular_data0_read ();
```

adc_sync_regular_data1_read

The description of adc_sync_regular_data1_read is shown as below:

Table 3-58. Function adc_sync_regular_data1_read

Function name	adc_sync_regular_data1_read
Function prototype	uint32_t adc_sync_regular_data1_read(void);
Function descriptions	read ADC sync regular data register 1
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	sync regular data 1

Example:

```
/* read ADC sync regular data register 1 */
```

```
adc_sync_regular_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

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

3.3.2. Descriptions of Peripheral functions

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

Table 3-60. 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
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

Function name	Function description
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
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
data	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
irate_switch_enable	data bit rate switch
mailbox_data_size	mailbox data size
tdc_enable	trnasmitter delay compensation enable
tdc_offset	trnasmitter delay compensation offset
prescaler	baudrate prescaler
resync_jump_width	resynchronization jump width
prop_time_segment	propagation time segment
time_segment_1	time segment 1
time_segment_2	time segment 2

Structure can_rx_fifo_id_filter_struct

Table 3-66. 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

Member name	Function description
CAN_INT_MB1	mailbox 1 interrupt
CAN_INT_MB2	mailbox 2 interrupt
CAN_INT_MB3	mailbox 3 interrupt
CAN_INT_MB4	mailbox 4 interrupt
CAN_INT_MB5	mailbox 5 interrupt
CAN_INT_MB6	mailbox 6 interrupt
CAN_INT_MB7	mailbox 7 interrupt
CAN_INT_MB8	mailbox 8 interrupt
CAN_INT_MB9	mailbox 9 interrupt
CAN_INT_MB10	mailbox 10 interrupt
CAN_INT_MB11	mailbox 11 interrupt
CAN_INT_MB12	mailbox 12 interrupt
CAN_INT_MB13	mailbox 13 interrupt
CAN_INT_MB14	mailbox 14 interrupt
CAN_INT_MB15	mailbox 15 interrupt
CAN_INT_MB16	mailbox 16 interrupt
CAN_INT_MB17	mailbox 17 interrupt
CAN_INT_MB18	mailbox 18 interrupt
CAN_INT_MB19	mailbox 19 interrupt
CAN_INT_MB20	mailbox 20 interrupt
CAN_INT_MB21	mailbox 21 interrupt
CAN_INT_MB22	mailbox 22 interrupt
CAN_INT_MB23	mailbox 23 interrupt
CAN_INT_MB24	mailbox 24 interrupt
CAN_INT_MB25	mailbox 25 interrupt
CAN_INT_MB26	mailbox 26 interrupt
CAN_INT_MB27	mailbox 27 interrupt
CAN_INT_MB28	mailbox 28 interrupt
CAN_INT_MB29	mailbox 29 interrupt
CAN_INT_MB30	mailbox 30 interrupt
CAN_INT_MB31	mailbox 31 interrupt
CAN_INT_FIFO_AVAILABLE	fifo available interrupt
CAN_INT_FIFO_WARNING	fifo warning interrupt
CAN_INT_FIFO_OVERFLOW	fifo overflow interrupt
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_P N	Pretended Networking state flag
CAN_FLAG_SOFT_ RST	software reset flag
CAN_FLAG_ERR_S UMMARY	error summary flag
CAN_FLAG_BUSO FF	bus off flag
CAN_FLAG_RECEI VING	receiving state flag
CAN_FLAG_TRAN SMITTING	transmitting state flag
CAN_FLAG_IDLE	IDLE state flag
CAN_FLAG_RX_W ARNING	receive warning flag
CAN_FLAG_TX_W ARNING	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 RR	ACK error flag
CAN_FLAG_BIT_D OMINANT_ERR	bit dominant error flag
CAN_FLAG_BIT_R ECESSIVE_ERR	bit recessive error flag
CAN_FLAG_SYNC_ ERR	synchronization flag
CAN_FLAG_BUSO FF_RECOVERY	bus off recovery flag
CAN_FLAG_ERR_S UMMARY_FD	FD error summary flag
CAN_FLAG_ERR_ OVERRUN	error overrun flag
CAN_FLAG_STUFF _ERR_FD	stuff error in FD data phase flag

Member name	Function description
CAN_FLAG_FORM_ERR_FD	form error in FD data phase flag
CAN_FLAG_CRC_ERR_FD	CRC error in FD data phase flag
CAN_FLAG_BIT_DOMINANT_ERR_FD	bit dominant error in FD data phase flag
CAN_FLAG_BIT_RECESSIVE_ERR_FD	bit recessive error in FD data phase flag
CAN_FLAG_MB0	mailbox 0 flag
CAN_FLAG_MB1	mailbox 1 flag
CAN_FLAG_MB2	mailbox 2 flag
CAN_FLAG_MB3	mailbox 3 flag
CAN_FLAG_MB4	mailbox 4 flag
CAN_FLAG_MB5	mailbox 5 flag
CAN_FLAG_MB6	mailbox 6 flag
CAN_FLAG_MB7	mailbox 7 flag
CAN_FLAG_MB8	mailbox 8 flag
CAN_FLAG_MB9	mailbox 9 flag
CAN_FLAG_MB10	mailbox 10 flag
CAN_FLAG_MB11	mailbox 11 flag
CAN_FLAG_MB12	mailbox 12 flag
CAN_FLAG_MB13	mailbox 13 flag
CAN_FLAG_MB14	mailbox 14 flag
CAN_FLAG_MB15	mailbox 15 flag
CAN_FLAG_MB16	mailbox 16 flag
CAN_FLAG_MB17	mailbox 17 flag
CAN_FLAG_MB18	mailbox 18 flag
CAN_FLAG_MB19	mailbox 19 flag
CAN_FLAG_MB20	mailbox 20 flag
CAN_FLAG_MB21	mailbox 21 flag
CAN_FLAG_MB22	mailbox 22 flag
CAN_FLAG_MB23	mailbox 23 flag
CAN_FLAG_MB24	mailbox 24 flag
CAN_FLAG_MB25	mailbox 25 flag
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

Member name	Function description
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_ERROR_SUMMARY_FD	fd error summary interrupt flag
CAN_INT_FLAG_MB0	mailbox 0 interrupt flag
CAN_INT_FLAG_MB1	mailbox 1 interrupt flag
CAN_INT_FLAG_MB2	mailbox 2 interrupt flag
CAN_INT_FLAG_MB3	mailbox 3 interrupt flag
CAN_INT_FLAG_MB4	mailbox 4 interrupt flag
CAN_INT_FLAG_MB5	mailbox 5 interrupt flag

Member name	Function description
CAN_INT_FLAG_M B6	mailbox 6 interrupt flag
CAN_INT_FLAG_M B7	mailbox 7 interrupt flag
CAN_INT_FLAG_M B8	mailbox 8 interrupt flag
CAN_INT_FLAG_M B9	mailbox 9 interrupt flag
CAN_INT_FLAG_M B10	mailbox 10 interrupt flag
CAN_INT_FLAG_M B11	mailbox 11 interrupt flag
CAN_INT_FLAG_M B12	mailbox 12 interrupt flag
CAN_INT_FLAG_M B13	mailbox 13 interrupt flag
CAN_INT_FLAG_M B14	mailbox 14 interrupt flag
CAN_INT_FLAG_M B15	mailbox 15 interrupt flag
CAN_INT_FLAG_M B16	mailbox 16 interrupt flag
CAN_INT_FLAG_M B17	mailbox 17 interrupt flag
CAN_INT_FLAG_M B18	mailbox 18 interrupt flag
CAN_INT_FLAG_M B19	mailbox 19 interrupt flag
CAN_INT_FLAG_M B20	mailbox 20 interrupt flag
CAN_INT_FLAG_M B21	mailbox 21 interrupt flag
CAN_INT_FLAG_M B22	mailbox 22 interrupt flag
CAN_INT_FLAG_M B23	mailbox 23 interrupt flag
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

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

Enum can_struct_type_enum

Table 3-75. Enum can_struct_type_enum

Member name	Function description
CAN_INIT_STRUCT	CAN initialize parameters struct
CAN_FD_INIT_STR	CAN FD parameters struct

UCT	
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}	
can_periph	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
Input parameter{in}	
can_parameter_init	Refers to Table 3-62. Structure can_parameter_struct
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
can_parameter_struct can_parameter;
```

```
ErrStatus err;
```

```
.....
```

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

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

can_struct_para_init

The description of can_struct_para_init is shown as below:

Table 3-80. Function can_struct_para_init

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

Example:

```
can_parameter_struct can_parameter;
```

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

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

can_private_filter_config

The description of can_private_filter_config is shown as below:

Table 3-81. Function can_private_filter_config

Function name	can_private_filter_config
Function prototype	void can_private_filter_config(uint32_t can_periph, uint32_t index, uint32_t filter_data);
Function descriptions	configure receive fifo/mailbox private filter
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral

$CANx(x=0,1,2)$	CAN peripheral selection
Input parameter{in}	
index	mailbox index
$0..31$	CAN mailbox index selection
Input parameter{in}	
filter_data	filter data to configure
$0..0xFFFFFFFF$	filter data
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

can_operation_mode_enter

The description of can_operation_mode_enter is shown as below:

Table 3-82. Function can_operation_mode_enter

Function name	can_operation_mode_enter
Function prototype	ErrStatus can_operation_mode_enter(uint32_t can_periph, can_operation_modes_enum mode);
Function descriptions	enter the corresponding mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
$CANx(x=0,1,2)$	CAN peripheral selection
Input parameter{in}	
mode	Refers to enum Table 3-74. Enum can_operation_modes_enum
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus err;

/* CAN0 enter normal mode */
err = can_operation_mode_enter(CAN0, CAN_NORMAL_MODE);
```

can_operation_mode_get

The description of can_operation_mode_get is shown as below:

Table 3-83. Function can_operation_mode_get

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

Example:

```
can_operation_modes_enum mode;

/* get CAN0 mode*/

mode = can_operation_mode_get(CAN0);
```

can_inactive_mode_exit

The description of can_inactive_mode_exit is shown as below:

Table 3-84. Function can_inactive_mode_exit

Function name	can_inactive_mode_exit
Function prototype	ErrStatus can_inactive_mode_exit(uint32_t can_periph);
Function descriptions	exit inactive mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```

ErrStatus err;

/* CAN0 exit INACTIVE mode */

err = can_inactive_mode_exit(CAN0);

```

can_pn_mode_exit

The description of can_pn_mode_exit is shown as below:

Table 3-85. Function can_pn_mode_exit

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

Example:

```

ErrStatus err;

/* CAN0 exit PN mode */

err = can_pn_mode_exit(CAN0);

```

can_fd_config

The description of can_fd_config is shown as below:

Table 3-86. 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
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Input parameter{in}	
can_fd_para_init	Refers to structure Table 3-65. Structure can fd parameter struct

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_fd_parameter_struct fd_parameter;
```

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

```
.....
```

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

can_bitrate_switch_enable

The description of can_bitrate_switch_enable is shown as below:

Table 3-87. 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

Function name	can_bitrate_switch_disable
Function prototype	void can_bitrate_switch_disable(uint32_t can_periph);
Function descriptions	disable bit rate switching
Precondition	-
The called functions	-

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

Example:

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

can_tdc_get

The description of can_tdc_get is shown as below:

Table 3-89. Function can_tdc_get

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

Example:

```
uint32_t tdc;

/* get transmitter delay compensation value */
tdc = can_tdc_get(CAN0);
```

can_tdc_enable

The description of can_tdc_enable is shown as below:

Table 3-90. Function can_tdc_enable

Function name	can_tdc_enable
Function prototype	void can_tdc_enable(uint32_t can_periph);
Function descriptions	enable transmitter delay compensation

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

Example:

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

```
can_tdc_enable(CAN0);
```

can_tdc_disable

The description of can_tdc_disable is shown as below:

Table 3-91. Function can_tdc_disable

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

Example:

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

```
can_tdc_disable(CAN0);
```

can_rx_fifo_config

The description of can_rx_fifo_config is shown as below:

Table 3-92. Function can_rx_fifo_config

Function name	can_rx_fifo_config
Function prototype	void can_rx_fifo_config(uint32_t can_periph, can_fifo_parameter_struct

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

Example:

```
can_fifo_parameter_struct fifo_struct;

/* configure rx FIFO */

.....

can_rx_fifo_config(CAN0, &fifo_struct);
```

can_rx_fifo_filter_table_config

The description of can_rx_fifo_filter_table_config is shown as below:

Table 3-93. Function can_rx_fifo_filter_table_config

Function name	can_rx_fifo_filter_table_config
Function prototype	void can_rx_fifo_filter_table_config(uint32_t can_periph, can_rx_fifo_id_filter_struct id_filter_table[]);
Function descriptions	configure rx FIFO filter table
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
Input parameter{in}	
id_filter_table	Refers to structure Table 3-66. Structure can_rx_fifo_id_filter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

can_rx_fifo_id_filter_struct id_filter_table[104];

/* configure rx FIFO filter table */

.....

can_rx_fifo_filter_table_config(CAN0, id_filter_table);

```

can_rx_fifo_read

The description of can_rx_fifo_read is shown as below:

Table 3-94. Function can_rx_fifo_read

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

Example:

```

can_rx_fifo_struct rx_fifo;

/* read rx FIFO data */

can_rx_fifo_read(CAN0, &rx_fifo);

```

can_rx_fifo_filter_matching_number_get

The description of can_rx_fifo_filter_matching_number_get is shown as below:

Table 3-95. Function can_rx_fifo_filter_matching_number_get

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

-	-
Return value	
uint32_t	0-416

Example:

```
uint32_t number;
```

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

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

can_rx_fifo_clear

The description of can_rx_fifo_clear is shown as below:

Table 3-96. Function can_rx_fifo_clear

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

Example:

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

```
can_rx_fifo_clear(CAN0);
```

can_ram_address_get

The description of can_ram_address_get is shown as below:

Table 3-97. 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

<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Input parameter{in}	
index	mailbox index
<i>0-31</i>	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
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Input parameter{in}	
index	mailbox index
<i>0-31</i>	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure Table 3-63. Structure can_mailbox_descriptor_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_mailbox_descriptor_struct transmit_message;
```

```
.....
```

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

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

can_mailbox_transmit_abort

The description of can_mailbox_transmit_abort is shown as below:

Table 3-99. 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
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Input parameter{in}	
index	mailbox index
<i>0-31</i>	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure Table 3-63. Structure can mailbox descriptor struct
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus err;
```

```
can_mailbox_descriptor_struct receive_message;
```

```
.....
```

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

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

can_mailbox_receive_lock

The description of can_mailbox_receive_lock is shown as below:

Table 3-102. Function can_mailbox_receive_lock

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

Example:

```
/* lock the receive mailbox 0 */
can_mailbox_receive_lock(CAN0, 0);
```

can_mailbox_receive_unlock

The description of can_mailbox_receive_unlock is shown as below:

Table 3-103. Function can_mailbox_receive_unlock

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

Example:

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

can_mailbox_receive_inactive

The description of can_mailbox_receive_inactive is shown as below:

Table 3-104. Function can_mailbox_receive_inactive

Function name	can_mailbox_receive_inactive
Function prototype	void can_mailbox_receive_inactive(uint32_t can_periph, uint32_t index);
Function descriptions	inactive the receive mailbox
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
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 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

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

uint32_t	0-0xF
----------	-------

Example:

```
uint32_t code;

/* get mailbox code value */

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

can_error_counter_config

The description of can_error_counter_config is shown as below:

Table 3-106. Function can_error_counter_config

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

Example:

```
can_error_counter_struct err_struct;

.....

/* configure error counter */

can_error_counter_config(CAN0, &err_struct);
```

can_error_counter_get

The description of can_error_counter_get is shown as below:

Table 3-107. Function can_error_counter_get

Function name	can_error_counter_get
Function prototype	void can_error_counter_get(uint32_t can_periph, can_error_counter_struct *errcnt_struct);

Function descriptions	get error count
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
errcnt_struct	Refers to structure Table 3-61. Structure can error counter struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_error_counter_struct err_struct;

/* get error count */

can_error_counter_get(CAN0, &err_struct);
```

can_error_state_get

The description of can_error_state_get is shown as below:

Table 3-108. Function can_error_state_get

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

Example:

```
can_error_state_enum error_state;

/* get error state indicator */

error_state = can_error_state_get(CAN0);
```

can_crc_get

The description of can_crc_get is shown as below:

Table 3-109. Function can_crc_get

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

Example:

```
can_crc_struct crc_struct;

/* get mailbox CRC value */

can_crc_get(CAN0, &crc_struct);
```

can_pn_mode_config

The description of can_pn_mode_config is shown as below:

Table 3-110. Function can_pn_mode_config

Function name	can_pn_mode_config
Function prototype	void can_pn_mode_config(uint32_t can_periph, can_pn_mode_config_struct *pnmod_config);
Function descriptions	configure Pretended Networking mode parameter
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
Return value	
pnmod_config	Refers to structure Table 3-69. Structure can_pn_mode_config_struct
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
can_pn_mode_config_struct pn_struct;

.....

/* configure Pretended Networking mode parameter */

can_pn_mode_config(CAN0, &pn_struct);
```

can_pn_mode_filter_config

The description of can_pn_mode_filter_config is shown as below:

Table 3-111. 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 Table 3-68. Structure can_pn_mode_filter_struct
Return value	
filter	Refers to structure Table 3-68. Structure can_pn_mode_filter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_pn_mode_filter_struct pn_filter[2];

.....

/* configure pn mode filter */

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

can_pn_mode_num_of_match_get

The description of can_pn_mode_num_of_match_get is shown as below:

Table 3-112. Function can_pn_mode_num_of_match_get

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

Example:

```
int32_t counter;
```

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

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

can_pn_mode_data_read

The description of can_pn_mode_data_read is shown as below:

Table 3-113. Function can_pn_mode_data_read

Function name	can_pn_mode_data_read
Function prototype	void can_pn_mode_data_read(uint32_t can_periph, uint32_t index, can_mailbox_descriptor_struct *mdpara);
Function descriptions	get matching message
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure Table 3-63. Structure can mailbox descriptor struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_mailbox_descriptor_struct mb_para;

/* get matching message */

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

can_self_reception_enable

The description of can_self_reception_enable is shown as below:

Table 3-114. 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
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
can_edge_filter_mode_disable(CAN0);
```

can_ped_mode_enable

The description of can_ped_mode_enable is shown as below:

Table 3-124. 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
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
can_ped_mode_enable(CAN0);
```

can_ped_mode_disable

The description of can_ped_mode_disable is shown as below:

Table 3-125. 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
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
can_ped_mode_disable(CAN0);
```

can_arbitration_delay_bits_config

The description of can_arbitration_delay_bits_config is shown as below:

Table 3-126. 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
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Input parameter{in}	
delay_bits	delay bits
<i>0-31</i>	delay bits selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

can_bsp_mode_config

The description of can_bsp_mode_config is shown as below:

Table 3-127. 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
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
sampling_mode	bsp sample mode
CAN_BSP_MODE_ON E_SAMPLE	one sample for received bit
CAN_BSP_MODE_TR HEE_SAMPLES	three samples for received bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

can_flag_get

The description of can_flag_get is shown as below:

Table 3-128. Function can_flag_get

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

Example:

```
FlagStatus flag;
```

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

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

can_flag_clear

The description of can_flag_clear is shown as below:

Table 3-129. Function can_flag_clear

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

Example:

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

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

can_interrupt_enable

The description of can_interrupt_enable is shown as below:

Table 3-130. Function can_interrupt_enable

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

Input parameter{in}	
interrupt	Refers to enum Table 3-71. Enum can_interrupt_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CAN bus off interrupt */
```

```
can_interrupt_enable(CAN0, CAN_INT_BUSOFF);
```

can_interrupt_disable

The description of can_interrupt_disable is shown as below:

Table 3-131. Function can_interrupt_disable

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

Example:

```
/* disable CAN bus off interrupt */
```

```
can_interrupt_disable(CAN0, CAN_INT_BUSOFF);
```

can_interrupt_flag_get

The description of can_interrupt_flag_get is shown as below:

Table 3-132. Function can_interrupt_flag_get

Function name	can_interrupt_flag_get
Function prototype	FlagStatus can_interrupt_flag_get(uint32_t can_periph,

	can_interrupt_flag_enum int_flag);
Function descriptions	get CAN interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
Input parameter{in}	
int_flag	Refers to enum Table 3-73. Enum can_interrupt_flag_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus int_flag;
```

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

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

can_interrupt_flag_clear

The description of can_interrupt_flag_clear is shown as below:

Table 3-133. 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
<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
Input parameter{in}	
int_flag	Refers to enum Table 3-73. Enum can_interrupt_flag_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

3.4. CAU

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

3.4.1. Descriptions of Peripheral registers

CAU registers are listed in the table shown as below:

Table 3-134. CAU Registers

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

3.4.2. Descriptions of Peripheral functions

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

Table 3-135. CAU firmware function

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

Structure cau_key_parameter_struct

Table 3-136. Structure cau_key_parameter_struct

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

Structure cau_iv_parameter_struct

Table 3-137. Structure cau_iv_parameter_struct

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

Structure cau_context_parameter_struct

Table 3-138. Structure cau_context_parameter_struct

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

Structure cau_parameter_struct

Table 3-139. Structure cau_parameter_struct

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

cau_deinit

The description of cau_deinit is shown as below:

Table 3-140. Function cau_deinit

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

Example:

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

cau_struct_para_init

The description of cau_struct_para_init is shown as below:

Table 3-141. Function cau_struct_para_init

Function name	cau_struct_para_init
Function prototype	void cau_struct_para_init(cau_parameter_struct *cau_parameter)
Function descriptions	initialize the CAU encrypt and decrypt parameter struct with the default values

Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Return value	
-	-

Example:

```
cau_parameter_struct text;
```

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

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

cau_key_struct_para_init

The description of cau_key_struct_para_init is shown as below:

Table 3-142. Function cau_key_struct_para_init

Function name	cau_key_struct_para_init
Function prototype	void cau_key_struct_para_init(cau_key_parameter_struct *key_initpara)
Function descriptions	initialize the key parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
key_initpara	refer to structure Table 3-136. Structure cau_key_parameter_struct
Return value	
-	-

Example:

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

```
cau_key_parameter_struct key_initpara;
```

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

cau_iv_struct_para_init

The description of cau_iv_struct_para_init is shown as below:

Table 3-143. Function cau_iv_struct_para_init

Function name	cau_iv_struct_para_init
Function prototype	void cau_iv_struct_para_init(cau_iv_parameter_struct *iv_initpara)

Function descriptions	initialize the vectors parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
iv_initpara	refer to structure Table 3-137. Structure cau_iv parameter struct
Return value	
-	-

Example:

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

```
cau_iv_parameter_struct iv_initpara;
```

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

cau_context_struct_para_init

The description of cau_context_struct_para_init is shown as below:

Table 3-144. Function cau_context_struct_para_init

Function name	cau_context_struct_para_init
Function prototype	void cau_context_struct_para_init(cau_context_parameter_struct *cau_context)
Function descriptions	initialize the context parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
cau_context	refer to structure Table 3-138. Structure cau_context parameter struct
Return value	
-	-

Example:

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

```
cau_context_parameter_struct context_initpara;
```

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

cau_enable

The description of cau_enable is shown as below:

Table 3-145. Function cau_enable

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

Example:

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

```
cau_enable();
```

cau_disable

The description of cau_disable is shown as below:

Table 3-146. Function cau_disable

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

Example:

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

```
cau_disable();
```

cau_dma_enable

The description of cau_dma_enable is shown as below:

Table 3-147. Function cau_dma_enable

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

Example:

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

cau_dma_disable

The description of cau_dma_disable is shown as below:

Table 3-148. Function cau_dma_disable

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

Example:

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

cau_init

The description of cau_init is shown as below:

Table 3-149. Function cau_init

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

<i>T</i>	
<i>CAU_SWAPPING_16BIT</i>	half-word swapping
<i>T</i>	
<i>CAU_SWAPPING_8BIT</i>	bytes swapping
<i>CAU_SWAPPING_1BIT</i>	bit swapping
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

cau_aes_key_select

The description of cau_aes_key_select is shown as below:

Table 3-150. Function cau_aes_key_select

Function name	cau_aes_key_select
Function prototype	void cau_aes_key_select(uint32_t key_selection);
Function descriptions	configure key selection
Precondition	-
The called functions	-
Input parameter{in}	
key_selection	key source selection when aes mode
<i>CAU_KEY</i>	use the key from CAU register
<i>CAU_EFUSE_KEY</i>	use the key from EFUSE
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the aes key source as CAU register */
```

```
cau_aes_key_select(CAU_KEY);
```

cau_aes_keysize_config

The description of cau_aes_keysize_config is shown as below:

Table 3-151. Function cau_aes_keysize_config

Function name	cau_aes_keysize_config
----------------------	------------------------

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

Example:

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

cau_key_init

The description of cau_key_init is shown as below:

Table 3-152. Function cau_key_init

Function name	cau_key_init
Function prototype	void cau_key_init(cau_key_parameter_struct* key_initpara);
Function descriptions	initialize the key parameters
Precondition	-
The called functions	-
Input parameter{in}	
key_initpara	refer to structure Table 3-136. Structure cau_key_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

cau_iv_init

The description of cau_iv_init is shown as below:

Table 3-153. Function cau_iv_init

Function name	cau_iv_init
Function prototype	void cau_iv_init(cau_iv_parameter_struct* iv_initpara);
Function descriptions	initialize the vectors parameters
Precondition	-
The called functions	-
Input parameter{in}	
iv_initpara	refer to structure Table 3-137. Structure cau_iv_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the vectors parameters */

cau_iv_parameter_struct iv_initpara;

cau_iv_init(&iv_initpara);

```

cau_phase_config

The description of cau_phase_config is shown as below:

Table 3-154. Function cau_phase_config

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

Example:

```

/* select prepare phase */

```

```
cau_phase_config(CAU_PREPARE_PHASE);
```

cau_fifo_flush

The description of cau_fifo_flush is shown as below:

Table 3-155. Function cau_fifo_flush

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

Example:

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

```
cau_fifo_flush( );
```

cau_enable_state_get

The description of cau_enable_state_get is shown as below:

Table 3-156. Function cau_enable_state_get

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

Example:

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

```
ControlStatus state = DISABLE;
```

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

cau_data_write

The description of cau_data_write is shown as below:

Table 3-157. Function cau_data_write

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

Example:

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

```
cau_data_write(0x10);
```

cau_data_read

The description of cau_data_read is shown as below:

Table 3-158. Function cau_data_read

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

Example:

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

```
uint32_t data;
```



```
data = cau_data_read( );
```

cau_context_save

The description of cau_context_save is shown as below:

Table 3-159. Function cau_context_save

Function name	cau_context_save
Function prototype	void cau_context_save(cau_context_parameter_struct *cau_context, cau_key_parameter_struct* key_initpara)
Function descriptions	save context before context switching
Precondition	-
The called functions	-
Input parameter{in}	
key_initpara	refer to structure Table 3-136. Structure cau_key_parameter_struct
Output parameter{out}	
cau_context	refer to structure Table 3-138. Structure cau_context_parameter_struct
Return value	
-	-

Example:

```
cau_context_parameter_struct context;

cau_key_parameter_struct key;

cau_parameter_struct cau_parameter;

uint32_t keyaddr;

.....

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

cau_key_struct_para_init(&key);

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

keyaddr += 4U;

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

/* save context before context switching */

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

cau_context_restore

The description of cau_context_restore is shown as below:

Table 3-160. Function cau_context_restore

Function name	cau_context_restore
Function prototype	void cau_context_restore(cau_context_parameter_struct *cau_context);
Function descriptions	restore context after context switching
Precondition	-
The called functions	-
Input parameter{in}	
cau_context	refer to structure Table 3-138. Structure cau context parameter struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
cau_context_parameter_struct context;

.....

cau_context_save(&context, &key);

.....

/* restore context after context switching */

cau_context_restore (&context);
```

cau_aes_ecb

The description of cau_aes_ecb is shown as below:

Table 3-161. Function cau_aes_ecb

Function name	cau_aes_ecb
Function prototype	ErrStatus cau_aes_ecb(cau_parameter_struct *cau_parameter, uint8_t *output);
Function descriptions	encrypt and decrypt using AES in ECB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau parameter struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;
```

```
uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

key_addr = key_select[i];

key_size = keysize[i];

text.alg_dir    = CAU_ENCRYPT;

text.key        = key_addr;

text.key_size   = key_size;

text.input      = plaintext;

text.in_length  = TEXT_SIZE;

/* encryption in ECB mode */

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

cau_aes_cbc

The description of cau_aes_cbc is shown as below:

Table 3-162. Function cau_aes_cbc

Function name	cau_aes_cbc
Function prototype	ErrStatus cau_aes_cbc(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using AES in CBC mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

key_addr = key_select[i];
```

```

key_size = keysize[i];

text.alg_dir    = CAU_ENCRYPT;

text.key        = key_addr;

text.key_size   = key_size;

text.iv         = vectors;

text.input      = plaintext;

text.in_length  = TEXT_SIZE;

/* encryption in CBC mode */

status = cau_aes_cbc(&text, encrypt_result);

```

cau_aes_ctr

The description of cau_aes_ctr is shown as below:

Table 3-163. Function cau_aes_ctr

Function name	cau_aes_ctr
Function prototype	ErrStatus cau_aes_ctr(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using AES in CTR mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```

cau_parameter_struct text;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

key_addr = key_select[i];

key_size = keysize[i];

text.alg_dir    = CAU_ENCRYPT;

text.key        = key_addr;

```

```

text.key_size = key_size;

text.iv       = vectors;

text.input    = plaintext;

text.in_length = TEXT_SIZE;

/* encryption in CTR mode */

status = cau_aes_ctr(&text, encrypt_result);

```

cau_aes_cfb

The description of cau_aes_cfb is shown as below:

Table 3-164. Function cau_aes_cfb

Function name	cau_aes_cfb
Function prototype	ErrStatus cau_aes_cfb(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using AES in CFB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```

cau_parameter_struct cau_cfb_parameter;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

/* encryption in CFB mode */

cau_cfb_parameter.alg_dir = CAU_ENCRYPT;

cau_cfb_parameter.key      = (uint8_t *)key_128;

cau_cfb_parameter.key_size = KEY_SIZE;

cau_cfb_parameter.iv       = (uint8_t *)vectors;

cau_cfb_parameter.iv_size  = IV_SIZE;

cau_cfb_parameter.input    = (uint8_t *)plaintext;

```

```

cau_cfb_parameter.in_length = PLAINTEXT_SIZE;

status = cau_aes_cfb(&cau_cfb_parameter, encrypt_result);

```

cau_aes_ofb

The description of cau_aes_ofb is shown as below:

Table 3-165. Function cau_aes_ofb

Function name	cau_aes_ofb
Function prototype	ErrStatus cau_aes_ofb(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using AES in OFB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```

cau_parameter_struct cau_ofb_parameter;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

/* encryption in OFB mode */

cau_ofb_parameter.alg_dir    = CAU_ENCRYPT;

cau_ofb_parameter.key        = (uint8_t *)key_128;

cau_ofb_parameter.key_size   = KEY_SIZE;

cau_ofb_parameter.iv         = (uint8_t *)vectors;

cau_ofb_parameter.iv_size    = IV_SIZE;

cau_ofb_parameter.input      = (uint8_t *)plaintext;

cau_ofb_parameter.in_length = PLAINTEXT_SIZE;

status = cau_aes_ofb(&cau_ofb_parameter, encrypt_result);

```

cau_aes_gcm

The description of cau_aes_gcm is shown as below:

Table 3-166. Function cau_aes_gcm

Function name	cau_aes_gcm
Function prototype	ErrStatus cau_aes_gcm(cau_parameter_struct *cau_parameter, uint8_t *output, uint8_t *tag)
Function descriptions	encrypt and decrypt using AES in GCM mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Output parameter{out}	
tag	pointer to the returned tag buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```

cau_parameter_struct cau_gcm_parameter;

uint8_t encrypt_result[TEXT_SIZE];

uint8_t gcm_tag[GCM_TAG_SIZE];

ErrStatus status;

.....

/* encryption in GCM mode */

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

```

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

cau_aes_ccm

The description of cau_aes_ccm is shown as below:

Table 3-167. Function cau_aes_ccm

Function name	cau_aes_ccm
Function prototype	ErrStatus cau_aes_ccm(cau_parameter_struct *cau_parameter, uint8_t *output, uint8_t tag[], uint32_t tag_size, uint8_t aad_buf[])
Function descriptions	encrypt and decrypt using AES in CCM mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Input parameter{in}	
tag_size	tag size (in bytes)
Output parameter{out}	
output	pointer to the returned buffer
Output parameter{out}	
tag	pointer to the returned tag buffer
Output parameter{out}	
aad_buf	pointer to the user buffer used when formatting aad block
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct cau_ccm_parameter;

uint8_t encrypt_result[TEXT_SIZE];

uint8_t ccm_tag[CCM_TAG_SIZE];

uint8_t aad_buf[AAD_SIZE + 21];

ErrStatus status;

.....

/* encryption in CCM mode */

cau_ccm_parameter.alg_dir    = CAU_ENCRYPT;

cau_ccm_parameter.key        = (uint8_t *)ccm_key_128;

cau_ccm_parameter.key_size   = KEY_SIZE;

cau_ccm_parameter.iv         = (uint8_t *)ccm_vectors;
```



```

cau_ccm_parameter.iv_size    = CCM_IV_SIZE;

cau_ccm_parameter.input      = (uint8_t *)plaintext;

cau_ccm_parameter.in_length  = PLAINTEXT_SIZE;

cau_ccm_parameter.aad        = (uint8_t *)aadmessage;

cau_ccm_parameter.aad_size   = AAD_SIZE;

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

```

cau_tdes_ecb

The description of cau_tdes_ecb is shown as below:

Table 3-168. Function cau_tdes_ecb

Function name	cau_tdes_ecb
Function prototype	ErrStatus cau_tdes_ecb(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using TDES in ECB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```

cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

text.alg_dir    = CAU_ENCRYPT;

text.key        = tdes_key;

text.input      = plaintext;

text.in_length  = DATA_SIZE;

/* encryption in ECB mode */

status = cau_tdes_ecb(&text, encrypt_result);

```

cau_tdes_cbc

The description of cau_tdes_cbc is shown as below:

Table 3-169. Function cau_tdes_cbc

Function name	cau_tdes_cbc
Function prototype	ErrStatus cau_tdes_cbc(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using TDES in CBC mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

text.alg_dir    = CAU_ENCRYPT;

text.key        = tdes_key;

text.iv         = vectors;

text.input      = plaintext;

text.in_length = DATA_SIZE;

/* encryption in CBC mode */

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

cau_des_ecb

The description of cau_des_ecb is shown as below:

Table 3-170. Function cau_des_ecb

Function name	cau_des_ecb
Function prototype	ErrStatus cau_des_ecb(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using DES in ECB mode

Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

text.alg_dir   = CAU_ENCRYPT;
text.key       = des_key;
text.input     = plaintext;
text.in_length = DATA_SIZE;

/* encryption in ECB mode */

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

cau_des_cbc

The description of cau_des_cbc is shown as below:

Table 3-171. Function cau_des_cbc

Function name	cau_des_cbc
Function prototype	ErrStatus cau_des_cbc(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using DES in CBC mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	refer to structure Table 3-139. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```

cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

text.alg_dir    = CAU_ENCRYPT;

text.key        = des_key;

text.iv         = vectors;

text.input      = plaintext;

text.in_length = DATA_SIZE;

/* encryption in CBC mode */

status = cau_des_cbc(&text, encrypt_result);

```

cau_flag_get

The description of cau_flag_get is shown as below:

Table 3-172. Function cau_flag_get

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

Example:

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

FlagStatus status;

status = cau_flag_get (CAU_FLAG_INFIFO_EMPTY);
```

cau_interrupt_enable

The description of cau_interrupt_enable is shown as below:

Table 3-173. Function cau_interrupt_enable

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

Example:

```
/* enable cau interrupt */

cau_interrupt_enable (CAU_INT_INFIFO);
```

cau_interrupt_disable

The description of cau_interrupt_disable is shown as below:

Table 3-174. Function cau_interrupt_disable

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

-	-
Return value	
-	-

Example:

```
/* disable cau interrupt */
```

```
cau_interrupt_disable(CAU_INT_INFIFO);
```

cau_interrupt_flag_get

The description of cau_interrupt_flag_get is shown as below:

Table 3-175. Function cau_interrupt_flag_get

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

Example:

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

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

3.5. CMP

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

3.5.1. Descriptions of Peripheral registers

CMP registers are listed in the table shown as below:

Table 3-176. CMP registers

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

3.5.2. Descriptions of Peripheral functions

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

Table 3-177. 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 3-178. 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 3-179. 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 Table 3-178. Enum cmp_enum
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 3-180. 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 Table 3-178. Enum cmp_enum
Input parameter{in}	
operating_mode	operating mode
CMP_MODE_HIGHSP	high speed mode

<i>EED</i>	
<i>CMP_MODE_MIDDLE SPEED</i>	medium speed mode
<i>CMP_MODE_VERYLO WSPEED</i>	very-low speed mode
Input parameter{in}	
inverting_input	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
Input parameter{in}	
output_hysteresis	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-181. Function cmp_noninverting_input_select

Function name	cmp_noninverting_input_select
Function prototype	void cmp_noninverting_input_select(cmp_enum cmp_periph, uint32_t noninverting_input);
Function descriptions	CMP noninverting input select
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
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
Output parameter{out}	
-	-
Return value	
-	-

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 3-182. Function cmp_output_init

Function name	cmp_output_init
Function prototype	void cmp_output_init(cmp_enum cmp_periph, uint32_t output_polarity);
Function descriptions	CMP output init
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
output_polarity	CMP output polarity
CMP_OUTPUT_POLARITY_INVERTED	output is inverted

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

Example:

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

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

cmp_output_mux_config

The description of cmp_output_mux_config is shown as below:

Table 3-183. Function cmp_output_mux_config

Function name	cmp_output_mux_config
Function prototype	void cmp_output_mux_config(cmp_enum cmp_periph, uint32_t cmp_output_sel);
Function descriptions	config comparator output port
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
cmp_output_sel	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

0	
CMP_AFSE_GPIO_PK1	CMP alternate GPIO PK1
1	
CMP_AFSE_GPIO_PK2	CMP alternate GPIO PK2
2	
Output parameter{out}	
-	-
Return value	
-	-

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 3-184. Function cmp_outputblank_init

Function name	cmp_blanking_init
Function prototype	void cmp_blanking_init(cmp_enum cmp_periph, uint32_t blanking_source_selection);
Function descriptions	CMP output blanking function init
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
blanking_source_selection	blanking source selection
CMP_BLANKING_NONE	CMP no blanking source
CMP_BLANKING_TIMER0_OC0	CMP TIMER0_CH0 output compare signal selected as blanking source
CMP_BLANKING_TIMER1_OC2	CMP TIMER1_CH2 output compare signal selected as blanking source
CMP_BLANKING_TIMER2_OC2	CMP TIMER2_CH2 output compare signal selected as blanking source
CMP_BLANKING_TIMER2_OC3	CMP TIMER2_CH3 output compare signal selected as blanking source
CMP_BLANKING_TIMER7_OC0	CMP TIMER7_CH0 output compare signal selected as blanking source
CMP_BLANKING_TIMER14_OC0	CMP TIMER14_CH0 output compare signal selected as blanking source

<i>ER14_OC0</i>	
Output parameter{out}	
-	-
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 3-185. 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 Table 3-178. Enum cmp_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
cmp_enable(CMP0);
```

cmp_disable

The description of cmp_disable is shown as below:

Table 3-186. 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 Table 3-178. Enum cmp_enum

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 3-187. Function cmp_window_enable

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

Example:

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

cmp_window_disable

The description of cmp_window_disable is shown as below:

Table 3-188. Function cmp_window_disable

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

-	-
Return value	
-	-

Example:

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

```
cmp_window_disable();
```

cmp_lock_enable

The description of cmp_lock_enable is shown as below:

Table 3-189. Function cmp_lock_enable

Function name	cmp_lock_enable
Function prototype	void cmp_lock_enable(cmp_enum cmp_periph);
Function descriptions	lock the comparator
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
cmp_lock_enable(CMP0);
```

cmp_voltage_scaler_enable

The description of cmp_voltage_scaler_enable is shown as below:

Table 3-190. Function cmp_voltage_scaler_enable

Function name	cmp_voltage_scaler_enable
Function prototype	void cmp_voltage_scaler_enable(cmp_enum cmp_periph);
Function descriptions	enable the voltage scaler
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Output parameter{out}	
-	-

Return value	
-	-

Example:

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

```
cmp_voltage_scaler_enable(CMP0);
```

cmp_voltage_scaler_disable

The description of cmp_voltage_scaler_disable is shown as below:

Table 3-191. Function cmp_voltage_scaler_disable

Function name	cmp_voltage_scaler_disable
Function prototype	void cmp_voltage_scaler_disable(cmp_enum cmp_periph);
Function descriptions	disable comparator the voltage scaler
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CMP0 the voltage scaler */
```

```
cmp_voltage_scaler_disable(CMP0);
```

cmp_scaler_bridge_enable

The description of cmp_scaler_bridge_enable is shown as below:

Table 3-192. Function cmp_scaler_bridge_enable

Function name	cmp_scaler_bridge_enable
Function prototype	void cmp_scaler_bridge_enable(cmp_enum cmp_periph);
Function descriptions	enable the scaler bridge
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable CMP0 the scaler bridge */
```

```
cmp_scaler_bridge_enable(CMP0);
```

cmp_scaler_bridge_disable

The description of cmp_scaler_bridge_disable is shown as below:

Table 3-193. Function cmp_scaler_bridge_disable

Function name	cmp_scaler_bridge_disable
Function prototype	void cmp_scaler_bridge_disable(cmp_enum cmp_periph);
Function descriptions	disable the scaler bridge
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
cmp_scaler_bridge_disable(CMP0);
```

cmp_output_level_get

The description of cmp_output_level_get is shown as below:

Table 3-194. Function cmp_output_level_get

Function name	cmp_output_level_get
Function prototype	uint32_t cmp_output_level_get(uint32_t cmp_periph);
Function descriptions	get output level
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Output parameter{out}	
-	-
Return value	
uint32_t	the output level

<i>CMP_OUTPUTLEVEL_</i> <i>HIGH</i>	comparator output high
<i>CMP_OUTPUTLEVEL_</i> <i>LOW</i>	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 3-195. Function cmp_flag_get

Function name	cmp_flag_get
Function prototype	FlagStatus cmp_flag_get(cmp_enum cmp_periph, uint32_t flag);
Function descriptions	get CMP flag
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
flag	CMP interrupt flag
<i>CMP_FLAG_COMPAR</i> <i>E</i>	CMP compare flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

FlagStatus flag_value;

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

cmp_flag_clear

The description of cmp_flag_clear is shown as below:

Table 3-196. Function cmp_flag_clear

Function name	cmp_flag_clear
Function prototype	void cmp_flag_clear(cmp_enum cmp_periph, uint32_t flag);

Function descriptions	clear CMP flag
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
flag	CMP interrupt flag
CMP_FLAG_COMPARE	CMP compare flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the CMP0 interrupt flag bit */
cmp_flag_clear(CMP0, CMP_FLAG_COMPARE);
```

cmp_interrupt_enable

The description of cmp_interrupt_enable is shown as below:

Table 3-197. Function cmp_interrupt_enable

Function name	cmp_interrupt_enable
Function prototype	void cmp_interrupt_enable(cmp_enum cmp_periph, uint32_t interrupt);
Function descriptions	enable CMP interrupt
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
interrupt	CMP interrupt
CMP_INT_COMPARE	CMP compare interrupt
Output parameter{out}	
-	-
Return value	
-	-

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 3-198. Function cmp_interrupt_disable

Function name	cmp_interrupt_disable
Function prototype	void cmp_interrupt_disable(cmp_enum cmp_periph, uint32_t interrupt);
Function descriptions	disable CMP interrupt
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
interrupt	CMP interrupt
CMP_INT_COMPARE	CMP compare interrupt
Output parameter{out}	
-	-
Return value	
-	-

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 3-199. Function cmp_interrupt_flag_get

Function name	cmp_interrupt_flag_get
Function prototype	FlagStatus cmp_interrupt_flag_get(cmp_enum cmp_periph, uint32_t flag);
Function descriptions	get CMP interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
flag	CMP interrupt flag
CMP_INT_FLAG_COMPARE	CMP compare interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

FlagStatus flag_value;

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

cmp_interrupt_flag_clear

The description of cmp_interrupt_flag_clear is shown as below:

Table 3-200. Function cmp_interrupt_flag_clear

Function name	cmp_interrupt_flag_clear
Function prototype	void cmp_interrupt_flag_clear(cmp_enum cmp_periph, uint32_t flag);
Function descriptions	clear CMP interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum Table 3-178. Enum cmp_enum
Input parameter{in}	
flag	CMP interrupt flag
CMP_INT_FLAG_COMPARE	CMP compare interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

cmp_interrupt_flag_clear(CMP0, CMP_INT_FLAG_COMPARE );
```

3.6. CPDM

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

3.6.1. Descriptions of Peripheral registers

CPDM registers are listed in the table shown as below:

Table 3-201. CPDM Registers

Registers	Descriptions
CPDM_CTL	CPDM control register
CPDM_CFG	CPDM configuration register

3.6.2. Descriptions of Peripheral functions

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

Table 3-202. CPDM firmware function

Function name	Function description
cpdm_enable	enable CPDM
cpdm_disable	disable CPDM
cpdm_delayline_sample_enable	enable CPDM delay line sample module
cpdm_delayline_sample_disable	disable CPDM delay line sample module
cpdm_output_clock_phase_select	select CPDM output clock phase
cpdm_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 3-203. 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	output clock phase = input clock + 6 * UNIT delay

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

cpdm_enable

The description of cpdm_enable is shown as below:

Table 3-204. Function cpdm_enable

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

Example:

```

/* enable CPDM */

cpdm_enable(CPDM_SDIO0);

```

cpdm_disable

The description of cpdm_disable is shown as below:

Table 3-205. Function cpdm_disable

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

Example:

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

cpdm_delayline_sample_enable

The description of cpdm_delayline_sample_enable is shown as below:

Table 3-206. Function cpdm_delayline_sample_enable

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

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 3-207. Function cpdm_delayline_sample_disable

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

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 3-208. Function cpdm_output_clock_phase_select

Function name	cpdm_output_clock_phase_select
Function prototype	void cpdm_output_clock_phase_select(uint32_t cpdm_periph, cpdm_output_phase_enum output_clock_phase);
Function descriptions	select CPDM output clock phase
Precondition	-
The called functions	-
Input parameter{in}	
cpdm_periph	the clock phase delay module of SDIO
CPDM_SDIOx	x = 0,1
Input parameter{in}	
output_clock_phase	the output clock phase, refer to Table 3-203. Enum cpdm_output_phase_enum
Output parameter{out}	
-	-
Return value	
-	-

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 3-209. Function cpdm_delayline_length_valid_flag_get

Function name	cpdm_delayline_length_valid_flag_get
Function prototype	FlagStatus cpdm_delayline_length_valid_flag_get(uint32_t cpdm_periph);
Function descriptions	get delay line length valid flag
Precondition	-
The called functions	-
Input parameter{in}	
cpdm_periph	the clock phase delay module of SDIO
<i>CPDM_SDIOx</i>	x = 0,1
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-210. Function cpdm_delayline_length_get

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

Return value	
uint16_t	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 3-211. Function cpdm_clock_output

Function name	cpdm_clock_output
Function prototype	void cpdm_clock_output(uint32_t cpdm_periph, cpdm_output_phase_enum output_clock_phase);
Function descriptions	configure CPDM clock output
Precondition	-
The called functions	-
Input parameter{in}	
cpdm_periph	the clock phase delay module of SDIO
CPDM_SDIOx	x = 0,1
Input parameter{in}	
output_clock_phase	the output clock phase, refer to Table 3-203. Enum cpdm_output_phase_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CPDM clock output */

cpdm_clock_output(CPDM_SDIO0, CPDM_OUTPUT_PHASE_SELECTION_1);
```

3.7. CRC

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. The CRC registers are listed in chapter [3.7.1](#), the CRC firmware functions are introduced in chapter [3.7.2](#).

3.7.1. Descriptions of Peripheral registers

CRC registers are listed in the table shown as below:

Table 3-212. CRC Registers

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

3.7.2. Descriptions of Peripheral functions

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

Table 3-213. CRC firmware function

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

crc_deinit

The description of crc_deinit is shown as below:

Table 3-214. Function crc_deinit

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

Output parameter{out}	
-	-
Return value	
-	-

Example:

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

crc_reverse_output_data_enable

The description of crc_reverse_output_data_enable is shown as below:

Table 3-215. Function crc_reverse_output_data_enable

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

Example:

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

crc_reverse_output_data_disable

The description of crc_reverse_output_data_disable is shown as below:

Table 3-216. Function crc_reverse_output_data_disable

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

-	-
Return value	
-	-

Example:

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

```
crc_reverse_output_data_disable();
```

crc_data_register_reset

The description of crc_data_register_reset is shown as below:

Table 3-217. Function crc_data_register_reset

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

Example:

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

```
crc_data_register_reset();
```

crc_data_register_read

The description of crc_data_register_read is shown as below:

Table 3-218. 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 3-219. Function `crc_free_data_register_read`

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

Example:

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

```
uint8_t crc_value = 0;
crc_value = crc_free_data_register_read();
```

crc_free_data_register_write

The description of `crc_free_data_register_write` is shown as below:

Table 3-220. Function `crc_free_data_register_write`

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

-	-
Return value	
-	-

Example:

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

crc_init_data_register_write

The description of crc_init_data_register_write is shown as below:

Table 3-221. 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_input_data_reverse_config

The description of crc_input_data_reverse_config is shown as below:

Table 3-222. 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}	
data_reverse	specify input data reverse function
CRC_INPUT_DATA_N OT	input data is not reversed

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

Example:

```
/* configure the crc input data */
crc_input_data_reverse_config(CRC_INPUT_DATA_WORD);
```

crc_polynomial_size_set

The description of `crc_polynomial_size_set` is shown as below:

Table 3-223. Function `crc_polynomial_size_set`

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

Example:

```
/* configure the CRC polynomial size*/
crc_polynomial_size_set(CRC_CTL_PS_7);
```

crc_polynomial_set

The description of `crc_polynomial_set` is shown as below:

Table 3-224. Function `crc_polynomial_set`

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

Example:

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

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

`crc_single_data_calculate`

The description of `crc_single_data_calculate` is shown as below:

Table 3-225. Function `crc_single_data_calculate`

Function name	<code>crc_single_data_calculate</code>
Function prototype	<code>uint32_t crc_single_data_calculate(uint32_t sdata, uint8_t data_format);</code>
Function descriptions	CRC calculate single data
Precondition	-
The called functions	-
Input parameter{in}	
sdata	specify 32-bit data
Input parameter{in}	
data_format	input data format
<code>INPUT_FORMAT_WORD</code>	input data in word format
<code>INPUT_FORMAT_HALFWORD</code>	input data in half-word format
<code>INPUT_FORMAT_BYTE</code>	input data in byte format
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit CRC calculate value (0-0xFFFFFFFF)

Example:

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

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

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

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

crc_block_data_calculate

The description of `crc_block_data_calculate` is shown as below:

Table 3-226. Function `crc_block_data_calculate`

Function name	<code>crc_block_data_calculate</code>
Function prototype	<code>uint32_t crc_block_data_calculate(void *array, uint32_t size, uint8_t data_format);</code>
Function descriptions	CRC calculate a data array
Precondition	-
The called functions	-
Input parameter{in}	
array	array of the input data array
Input parameter{in}	
size	size of the array
Input parameter{in}	
data_format	input data format
<code>INPUT_FORMAT_WORD</code>	input data in word format
<code>INPUT_FORMAT_HALFWORD</code>	input data in half-word format
<code>INPUT_FORMAT_BYTE</code>	input data in byte format
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit CRC calculate value (0-0xFFFFFFFF)

Example:

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

```
#define BUFFER_SIZE    6
```

```
uint32_t valcrc = 0;
```

```
static const uint32_t data_buffer[BUFFER_SIZE] = {
```

```
0x00001111, 0x00002222, 0x00003333, 0x00004444, 0x00005555, 0x00006666};
```

```
valcrc    =    crc_block_data_calculate((uint32_t *)    data_buffer,    BUFFER_SIZE,
```

```
INPUT_FORMAT_WORD);
```

3.8. 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 [3.8.1](#), the CTC firmware functions are introduced in chapter [3.8.2](#).

3.8.1. Descriptions of Peripheral registers

CTC registers are listed in the table shown as below:

Table 3-227. 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

3.8.2. Descriptions of Peripheral functions

CTC registers are listed in the table shown as below:

Table 3-228. 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

Function name	Function description
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 3-229. 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 3-230. Function ctc_counter_enable

Function name	ctc_counter_enable
Function prototype	void ctc_counter_enable (void);
Function descriptions	enable CTC counter
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* enable CTC trim counter */
```

```
ctc_counter_enable ();
```

ctc_counter_disable

The description of ctc_counter_disable is shown as below:

Table 3-231. Function ctc_counter_disable

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

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 3-232. Function ctc_irc48m_trim_value_config

Function name	ctc_irc48m_trim_value_config
Function prototype	void ctc_irc48m_trim_value_config(uint8_t trim_value);
Function descriptions	configure the IRC48M trim value
Precondition	-
The called functions	-
Input parameter{in}	
trim_value	0~63
Output parameter{out}	
-	-

Return value	
-	-

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 3-233. Function ctc_software_refsource_pulse_generate

Function name	ctc_software_refsource_pulse_generate
Function prototype	void ctc_software_refsource_pulse_generate (void);
Function descriptions	generate software reference source sync pulse
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-234. Function ctc_hardware_trim_mode_config

Function name	ctc_hardware_trim_mode_config
Function prototype	void ctc_hardware_trim_mode_config(uint32_t hardmode);
Function descriptions	configure hardware automatically trim mode
Precondition	-
The called functions	-
Input parameter{in}	
hardmode	hardware automatically trim mode enable or disable
<i>CTC_HARDWARE_TRIM_MODE_ENABLE</i>	hardware automatically trim mode enable
<i>CTC_HARDWARE_TRIM_MODE_DISABLE</i>	hardware automatically trim mode disable

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

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 3-235. Function ctc_refsource_polarity_config

Function name	ctc_refsource_polarity_config
Function prototype	void ctc_refsource_polarity_config(uint32_t polarity);
Function descriptions	configure reference signal source polarity
Precondition	-
The called functions	-
Input parameter{in}	
polarity	reference signal source polarity
<i>CTC_REFSOURCE_POLARITY_FALLING</i>	reference signal source polarity is falling edge
<i>CTC_REFSOURCE_POLARITY_RISING</i>	reference signal source polarity is rising edge
Output parameter{out}	
-	-
Return value	
-	-

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 3-236. Function ctc_refsource_signal_select

Function name	ctc_refsource_signal_select
Function prototype	void ctc_refsource_signal_select(uint32_t refs);
Function descriptions	select reference signal source

Precondition	-
The called functions	-
Input parameter{in}	
refs	reference signal source
<i>CTC_REFSOURCE_GPIO</i>	GPIO is selected
<i>CTC_REFSOURCE_LXTAL</i>	LXTAL is selected
Output parameter{out}	
-	-
Return value	
-	-

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 3-237. Function ctc_refsource_prescaler_config

Function name	ctc_refsource_prescaler_config
Function prototype	void ctc_refsource_prescaler_config(uint32_t prescaler);
Function descriptions	configure reference signal source prescaler
Precondition	-
The called functions	-
Input parameter{in}	
prescaler	Prescaler factor
<i>CTC_REFSOURCE_PSC_OFF</i>	reference signal not divided
<i>CTC_REFSOURCE_PSC_DIV2</i>	reference signal divided by 2
<i>CTC_REFSOURCE_PSC_DIV4</i>	reference signal divided by 4
<i>CTC_REFSOURCE_PSC_DIV8</i>	reference signal divided by 8
<i>CTC_REFSOURCE_PSC_DIV16</i>	reference signal divided by 16
<i>CTC_REFSOURCE_PSC_DIV32</i>	reference signal divided by 32
<i>CTC_REFSOURCE_PSC_DIV64</i>	reference signal divided by 64

<i>CTC_REFSOURCE_P</i> <i>SC_DIV128</i>	reference signal divided by 128
Output parameter{out}	
-	-
Return value	
-	-

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 3-238. Function ctc_clock_limit_value_config

Function name	ctc_clock_limit_value_config
Function prototype	void ctc_clock_limit_value_config(uint8_t limit_value);
Function descriptions	configure clock trim base limit value
Precondition	-
The called functions	-
Input parameter{in}	
limit_value	0x00 - 0xFF
Output parameter{out}	
-	-
Return value	
-	-

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 3-239. Function ctc_counter_reload_value_config

Function name	ctc_counter_reload_value_config
Function prototype	void ctc_counter_reload_value_config(uint16_t reload_value);
Function descriptions	configure CTC counter reload value
Precondition	-
The called functions	-
Input parameter{in}	

reload_value	0x0000 - 0xFFFF
Output parameter{out}	
-	-
Return value	
-	-

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 3-240. Function ctc_counter_capture_value_read

Function name	ctc_counter_capture_value_read
Function prototype	uint16_t ctc_counter_capture_value_read(void);
Function descriptions	read CTC counter capture value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	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 3-241. Function ctc_counter_direction_read

Function name	ctc_counter_direction_read
Function prototype	FlagStatus ctc_counter_direction_read(void);
Function descriptions	read CTC trim counter direction
Precondition	-
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-242. Function ctc_counter_reload_value_read

Function name	ctc_counter_reload_value_read
Function prototype	uint16_t ctc_counter_reload_value_read(void);
Function descriptions	read CTC counter reload value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	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 3-243. Function ctc_irc48m_trim_value_read

Function name	ctc_irc48m_trim_value_read
Function prototype	uint8_t ctc_irc48m_trim_value_read(void);
Function descriptions	read the IRC48M trim value
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	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 3-244. Function ctc_flag_get

Function name	ctc_flag_get
Function prototype	FlagStatus ctc_flag_get(uint32_t flag);
Function descriptions	get CTC status flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	CTC status flag
CTC_FLAG_CKOK	clock trim OK flag
CTC_FLAG_CKWARN	clock trim warning flag
CTC_FLAG_ERR	error flag
CTC_FLAG_EREFP	expect reference flag
CTC_FLAG_CKERR	clock trim error bit
CTC_FLAG_REFMIS	reference sync pulse miss flag
CTC_FLAG_TRIMERR	trim value error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-245. Function `ctc_flag_clear`

Function name	<code>ctc_flag_clear</code>
Function prototype	<code>void ctc_flag_clear (uint32_t flag);</code>
Function descriptions	clear CTC status flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	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_EREf</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	
-	-

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 3-246. Function `ctc_interrupt_enable`

Function name	<code>ctc_interrupt_enable</code>
Function prototype	<code>void ctc_interrupt_enable(uint32_t interrupt);</code>
Function descriptions	enable the CTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	CTC interrupt
<code>CTC_INT_CKOK</code>	clock trim OK interrupt
<code>CTC_INT_CKWARN</code>	clock trim warning interrupt
<code>CTC_INT_ERR</code>	error interrupt
<code>CTC_INT_EREf</code>	expect reference interrupt
Output parameter{out}	
-	-
Return value	

-	-
---	---

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 3-247. Function ctc_interrupt_disable

Function name	ctc_interrupt_disable
Function prototype	void ctc_interrupt_disable(uint32_t interrupt);
Function descriptions	disable the CTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-248. Function ctc_interrupt_flag_get

Function name	ctc_interrupt_flag_get
Function prototype	FlagStatus ctc_interrupt_flag_get(uint32_t int_flag);
Function descriptions	get CTC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	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
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-249. Function ctc_interrupt_flag_clear

Function name	ctc_interrupt_flag_clear
Function prototype	void ctc_interrupt_flag_clear(uint32_t int_flag);
Function descriptions	clear CTC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	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_TRIM ERR</i>	trim value error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear CTC interrupt flag status */
```

```
ctc_interrupt_flag_clear (CTC_INT_FLAG_CKOK);
```

3.9. DAC

The Digital-to-analog converter converts 12-bit digital data to a voltage on the external pins. The DAC registers are listed in chapter [3.9.1](#) the DAC firmware functions are introduced in chapter [3.9.2](#).

3.9.1. Peripheral register description

DAC registers are listed in the table shown as below:

Table 3-250. DAC Registers

Register	Descriptions
DAC_CTL0	DACx control register 0
DAC_SWT	DACx software trigger register
DAC_OUT0_R12DH	DACx_OUT0 12-bit right-aligned data holding register
DAC_OUT0_L12DH	DACx_OUT0 12-bit left-aligned data holding register
DAC_OUT0_R8DH	DACx_OUT0 8-bit right-aligned data holding register
DAC_OUT1_R12DH	DACx_OUT1 12-bit right-aligned data holding register
DAC_OUT1_L12DH	DACx_OUT1 12-bit left-aligned data holding register
DAC_OUT1_R8DH	DACx_OUT1 8-bit right-aligned data holding register
DACC_R12DH	DACx concurrent mode 12-bit right-aligned data holding register
DACC_L12DH	DACx concurrent mode 12-bit left-aligned data holding register
DACC_R8DH	DACx concurrent mode 8-bit right-aligned data holding register
DAC_OUT0_DO	DACx_OUT0 data output register
DAC_OUT1_DO	DACx_OUT1 data output register
DAC_STAT0	DACx status register 0
DAC_CALR	DACx calibration register
DAC_MDCR	DACx mode control register
DAC_SKSTR0	DACx sample and keep sample time register 0
DAC_SKSTR1	DACx sample and keep sample time register 1
DAC_SKKTR	DACx sample and keep time register

Register	Descriptions
DAC_SKRTR	DACx sample and keep refresh time register

3.9.2. Descriptions of Peripheral functions

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

Table 3-251. DAC firmware functions

Function name	Function description
<code>dac_deinit</code>	deinitialize DAC
<code>dac_enable</code>	enable DAC
<code>dac_disable</code>	disable DAC
<code>dac_dma_enable</code>	enable DAC DMA function
<code>dac_dma_disable</code>	disable DAC DMA function
<code>dac_mode_config</code>	configure DAC mode
<code>dac_trimming_value_get</code>	get the DACx trimming value
<code>dac_trimming_value_set</code>	set the DACx trimming value
<code>dac_trimming_enable</code>	enable the DACx trimming
<code>dac_output_value_get</code>	get DAC output value
<code>dac_data_set</code>	set DAC data holding register value
<code>dac_trigger_enable</code>	enable DAC trigger
<code>dac_trigger_disable</code>	disable DAC trigger
<code>dac_trigger_source_config</code>	configure DAC trigger source
<code>dac_software_trigger_enable</code>	enable DAC software trigger
<code>dac_wave_mode_config</code>	configure DAC wave mode
<code>dac_lfsr_noise_config</code>	configure DAC LFSR noise mode
<code>dac_triangle_noise_config</code>	configure DAC triangle noise mode
<code>dac_concurrent_enable</code>	enable DAC concurrent mode
<code>dac_concurrent_disable</code>	disable DAC concurrent mode
<code>dac_concurrent_software_trigger_enable</code>	enable DAC concurrent software trigger
<code>dac_concurrent_data_set</code>	set DAC concurrent mode data holding register value
<code>dac_sample_keep_mode_config</code>	set DAC sample and keep time value
<code>dac_flag_get</code>	get DAC flag
<code>dac_flag_clear</code>	clear DAC flag
<code>dac_interrupt_enable</code>	enable DAC interrupt
<code>dac_interrupt_disable</code>	disable DAC interrupt
<code>dac_interrupt_flag_get</code>	get DAC interrupt flag
<code>dac_interrupt_flag_clear</code>	clear DAC interrupt flag

`dac_deinit`

The description of `dac_deinit` is shown as below:

Table 3-252. Function dac_deinit

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

Example:

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

dac_enable

The description of dac_enable is shown as below:

Table 3-253. Function dac_enable

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

Example:

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

dac_disable

The description of dac_disable is shown as below:

Table 3-254. Function dac_disable

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

Example:

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

dac_dma_enable

The description of dac_dma_enable is shown as below:

Table 3-255. Function dac_dma_enable

Function name	dac_dma_enable
Function prototype	void dac_dma_enable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	enable DAC DMA function
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
Output parameter{out}	
-	-
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 3-256. Function dac_dma_disable

Function name	dac_dma_disable
Function prototype	void dac_dma_disable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	disable DAC DMA function
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	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 3-257. Function dac_mode_config

Function name	dac_mode_config
Function prototype	void dac_mode_config(uint32_t dac_periph, uint32_t dac_out, uint32_t mode);
Function descriptions	configure DAC mode
Precondition	-
The called functions	-
Input parameter{in}	

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

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 3-258. Function `dac_trimming_value_get`

Function name	<code>dac_trimming_value_get</code>
Function prototype	<code>void dac_trimming_value_get(uint32_t dac_periph, uint32_t dac_out,);</code>
Function descriptions	get the DACx trimming value
Precondition	-
The called functions	-
Input parameter{in}	

dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
Output parameter{out}	
-	-
Return value	
UInt32_t	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 3-259. Function dac_trimming_value_set

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

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 3-260. Function dac_trimming_enable

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

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 3-261. Function dac_output_value_get

Function name	dac_output_value_get
Function prototype	uint16_t dac_output_value_get(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	get DAC output value
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
Output parameter{out}	
-	-
Return value	

uint16_t	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 3-262. Function `dac_data_set`

Function name	<code>dac_data_set</code>
Function prototype	<code>void dac_data_set(uint32_t dac_periph, uint8_t dac_out, uint32_t dac_align, uint16_t data);</code>
Function descriptions	set DAC data holding register value
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
Input parameter{in}	
dac_align	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
Input parameter{in}	
data	data to be loaded (0~4095)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC0_OUT0 data holding register value */
```

```
dac_data_set(DAC0, DAC_OUT0, DAC_ALIGN_8B_R, 0xFF);
```

dac_trigger_enable

The description of `dac_trigger_enable` is shown as below:

Table 3-263. Function `dac_trigger_enable`

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

Example:

```
/* enable DAC0_OUT0 trigger */
dac_trigger_enable(DAC0, DAC_OUT0);
```

`dac_trigger_disable`

The description of `dac_trigger_disable` is shown as below:

Table 3-264. Function `dac_trigger_disable`

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

Example:

```
/* disable DAC0_OUT0 trigger */
```

```
dac_trigger_disable(DAC0, DAC_OUT0);
```

dac_trigger_source_config

The description of `dac_trigger_source_config` is shown as below:

Table 3-265. Function `dac_trigger_source_config`

Function name	<code>dac_trigger_source_config</code>
Function prototype	<code>void dac_trigger_source_config(uint32_t dac_periph, uint8_t dac_out, uint32_t triggersource);</code>
Function descriptions	configure DAC trigger source
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
Input parameter{in}	
triggersource	external trigger of DAC
<i>DAC_TRIGGER_EXTERNAL</i>	external trigger selected by TRIGSEL
<i>DAC_TRIGGER_SOFTWARE</i>	software trigger
Output parameter{out}	
-	-
Return value	
-	-

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 3-266. Function `dac_software_trigger_enable`

Function name	<code>dac_software_trigger_enable</code>
Function prototype	<code>void dac_software_trigger_enable(uint32_t dac_periph, uint8_t dac_out);</code>
Function descriptions	enable DAC software trigger

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

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 3-267. Function dac_wave_mode_config

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

-	-
---	---

Example:

```
/* configure DAC0_OUT0 wave mode */
dac_wave_mode_config(DAC0, DAC_OUT0, DAC_WAVE_DISABLE);
```

dac_lfsr_noise_config

The description of dac_lfsr_noise_config is shown as below:

Table 3-268. Function dac_lfsr_noise_config

Function name	dac_lfsr_noise_config
Function prototype	void dac_lfsr_noise_config(uint32_t dac_periph, uint8_t dac_out, uint32_t unmask_bits);
Function descriptions	configure DAC LFSR noise mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
Input parameter{in}	
unmask_bits	LFSR noise unmask bits
<i>DAC_LFSR_BIT0</i>	unmask the LFSR bit0
<i>DAC_LFSR_BITSx_0</i>	unmask the LFSR bits [x:0] (x = 1,2,3..11)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DAC0_OUT0 LFSR noise mode */
dac_lfsr_noise_config(DAC0, DAC_OUT0, DAC_LFSR_BIT0);
```

dac_triangle_noise_config

The description of dac_triangle_noise_config is shown as below:

Table 3-269. Function dac_triangle_noise_config

Function name	dac_triangle_noise_config
Function prototype	void dac_triangle_noise_config(uint32_t dac_periph, uint8_t dac_out,

	uint32_t amplitude);
Function descriptions	configure DAC triangle noise mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
Input parameter{in}	
amplitude	the amplitude of the triangle
<i>DAC_TRIANGLE_AMPLIT</i> <i>UDE_x</i>	$x = 2^n - 1$ (n = 1..12)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DAC0_OUT0 triangle noise mode */
```

```
dac_triangle_noise_config(DAC0, DAC_OUT0, DAC_TRIANGLE_AMPLITUDE_1);
```

dac_concurrent_enable

The description of dac_concurrent_enable is shown as below:

Table 3-270. Function dac_concurrent_enable

Function name	dac_concurrent_enable
Function prototype	void dac_concurrent_enable(uint32_t dac_periph);
Function descriptions	enable DAC concurrent mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC0 concurrent mode */
```

```
dac_concurrent_enable(DAC0);
```

dac_concurrent_disable

The description of `dac_concurrent_disable` is shown as below:

Table 3-271. Function `dac_concurrent_disable`

Function name	<code>dac_concurrent_disable</code>
Function prototype	<code>void dac_concurrent_disable(uint32_t dac_periph);</code>
Function descriptions	disable DAC concurrent mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DAC0 concurrent mode */
```

```
dac_concurrent_disable(DAC0);
```

dac_concurrent_software_trigger_enable

The description of `dac_concurrent_software_trigger_enable` is shown as below:

Table 3-272. Function `dac_concurrent_software_trigger_enable`

Function name	<code>dac_concurrent_software_trigger_enable</code>
Function prototype	<code>void dac_concurrent_software_trigger_enable(uint32_t dac_periph);</code>
Function descriptions	enable DAC concurrent software trigger
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
Output parameter{out}	
-	-
Return value	
-	-

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 3-273. 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}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_align	DAC data alignment mode
<i>DAC_ALIGN_8B_R</i>	8-bit right-aligned data
<i>DAC_ALIGN_12B_R</i>	12-bit right-aligned data
<i>DAC_ALIGN_12B_L</i>	12-bit left-aligned data
Input parameter{in}	
data0	DACx_OUT0 data to be loaded (0~4095)
Input parameter{in}	
data1	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 3-274. 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}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
Input parameter{in}	
sample_time	DAC sample time
Input parameter{in}	
keep_time	DAC keep time
Input parameter{in}	
refresh_time	DAC refresh time
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC0_OUT0 sample and keep time value */
```

```
dac_sample_keep_mode_config (DAC0, DAC_OUT0, 1, 1, 1);
```

dac_flag_get

The description of dac_flag_get is shown as below:

Table 3-275. Function dac_flag_get

Function name	dac_flag_get
Function prototype	FlagStatus dac_flag_get(uint32_t dac_periph, uint32_t flag);
Function descriptions	get DAC flag
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
flag	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

Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-276. Function dac_flag_clear

Function name	dac_flag_clear
Function prototype	void dac_flag_clear(uint32_t dac_periph, uint32_t flag);
Function descriptions	clear DAC flag
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
flag	DAC flag
DAC_FLAG_DDUDR0	DACx_OUT0 DMA underrun flag
DAC_FLAG_DDUDR1	DACx_OUT1 DMA underrun flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DAC0 flag */
```

```
dac_flag_clear(DAC0, DAC_FLAG_DDUDR0);
```

dac_interrupt_enable

The description of dac_interrupt_enable is shown as below:

Table 3-277. Function dac_interrupt_enable

Function name	dac_interrupt_enable
Function prototype	void dac_interrupt_enable(uint32_t dac_periph, uint32_t interrupt);

Function descriptions	enable DAC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
interrupt	the DAC interrupt
<i>DAC_INT_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt
<i>DAC_INT_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC0 interrupt */
```

```
dac_interrupt_enable (DAC0, DAC_INT_DDUDRIE0);
```

dac_interrupt_disable

The description of dac_interrupt_disable is shown as below:

Table 3-278. Function dac_interrupt_disable

Function name	dac_interrupt_disable
Function prototype	void dac_interrupt_disable(uint32_t dac_periph, uint32_t interrupt);
Function descriptions	disable DAC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
interrupt	the DAC interrupt
<i>DAC_INT_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt
<i>DAC_INT_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt
Output parameter{out}	
-	-
Return value	
-	-

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 3-279. Function `dac_interrupt_flag_get`

Function name	<code>dac_interrupt_flag_get</code>
Function prototype	<code>FlagStatus dac_interrupt_flag_get(uint32_t dac_periph, uint32_t int_flag);</code>
Function descriptions	get DAC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
int_flag	DAC interrupt flag
<i>DAC_INT_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt flag
<i>DAC_INT_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-280. Function `dac_interrupt_flag_clear`

Function name	<code>dac_interrupt_flag_clear</code>
Function prototype	<code>Void dac_interrupt_flag_clear(uint32_t dac_periph, uint32_t int_flag);</code>
Function descriptions	clear DAC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
Input parameter{in}	
int_flag	DAC interrupt flag

<i>DAC_INT_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt flag
<i>DAC_INT_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DAC0 interrupt flag */
```

```
dac_interrupt_flag_clear(DAC0, DAC_INT_FLAG_DDUDR0);
```

3.10. DBG

The DBG hold unit helps debugger to debug power saving mode. The DBG registers are listed in chapter [3.10.1](#). the DBG firmware functions are introduced in chapter [3.10.2](#).

3.10.1. Descriptions of Peripheral registers

DBG registers are listed in the table shown as below:

Table 3-281. 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

3.10.2. Descriptions of Peripheral functions

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

Table 3-282. 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 3-283. 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

Member name	Function description
DBG_I2Cx_HOLD	hold I2Cx (x=0,1,2,3) smbus when core is halted
DBG_CANx_HOLD	hold CANx (x=0,1,2) counter when core is halted
DBG_RTC_HOLD	hold RTC calendar and wakeup counter when core is halted

dbg_deinit

The description of dbg_deinit is shown as below:

Table 3-284. 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 3-285. 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 3-286. Function dbg_low_power_enable

Function name	dbg_low_power_enable
Function prototype	void dbg_low_power_enable(uint32_t dbg_low_power);
Function descriptions	enable low power behavior when the mcu is in debug mode
Precondition	-
The called functions	-
Input parameter{in}	
dbg_low_power	low power mode
DBG_LOW_POWER_SLEEP	keep debugger connection during sleep mode
DBG_LOW_POWER_DEEPSLEEP	keep debugger connection during deepsleep mode
DBG_LOW_POWER_STANDBY	keep debugger connection during standby mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
dbg_low_power_enable(DBG_LOW_POWER_SLEEP);
```

dbg_low_power_disable

The description of dbg_low_power_disable is shown as below:

Table 3-287. Function dbg_low_power_disable

Function name	dbg_low_power_disable
Function prototype	void dbg_low_power_disable(uint32_t dbg_low_power);
Function descriptions	disable low power behavior when the mcu is in debug mode
Precondition	-
The called functions	-
Input parameter{in}	
dbg_low_power	low power mode

<i>DBG_LOW_POWER_SLEEP</i>	do not keep debugger connection during sleep mode
<i>DBG_LOW_POWER_DEEPSLEEP</i>	do not keep debugger connection during deepsleep mode
<i>DBG_LOW_POWER_STANDBY</i>	do not keep debugger connection during standby mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
dbg_low_power_disable(DBG_LOW_POWER_SLEEP);
```

dbg_periph_enable

The description of dbg_periph_enable is shown as below:

Table 3-288. Function dbg_periph_enable

Function name	dbg_periph_enable
Function prototype	void dbg_periph_enable(dbg_periph_enum dbg_periph);
Function descriptions	Enable peripheral behavior when the mcu is in debug mode
Precondition	-
The called functions	-
Input parameter{in}	
dbg_periph	Peripheral refer to Table 3-283. Enum dbg_periph_enum
<i>DBG_FWDGT_HOLD</i>	debug FWDGT kept when core is halted
<i>DBG_WWDGT_HOLD</i>	debug WWDGT kept when core is halted
<i>DBG_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
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
dbg_periph_enable(DBG_TIMER1_HOLD);
```

dbg_periph_disable

The description of dbg_periph_disable is shown as below:

Table 3-289. Function dbg_periph_disable

Function name	dbg_periph_disable
Function prototype	void dbg_periph_disable(dbg_periph_enum dbg_periph);
Function descriptions	Disable peripheral behavior when the mcu is in debug mode
Precondition	-
The called functions	-
Input parameter{in}	
dbg_periph	peripheral refer to Table 3-283. Enum dbg_periph_enum
<i>DBG_FWDGT_HOLD</i>	debug FWDGT kept when core is halted
<i>DBG_WWDGT_HOLD</i>	debug WWDGT kept when core is halted
<i>DBG_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
Output parameter{out}	
-	-
Return value	
-	-

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 3-290. Function dbg_trace_pin_enable

Function name	dbg_trace_pin_enable
Function prototype	void dbg_trace_pin_enable(void);
Function descriptions	Enable trace pin assignment
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable trace pin assignment */
```

```
dbg_trace_pin_enable();
```

dbg_trace_pin_disable

The description of dbg_trace_pin_disable is shown as below:

Table 3-291. Function dbg_trace_pin_disable

Function name	dbg_trace_pin_disable
Function prototype	void dbg_trace_pin_disable(void);
Function descriptions	Disable trace pin assignment
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable trace pin assignment */
```

```
dbg_trace_pin_disable();
```

dbg_trace_pin_mode_set

The description of dbg_trace_pin_mode_set is shown as below:

Table 3-292. Function dbg_trace_pin_mode_set

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

<code>TRACE_MODE_SYNC</code> <code>_DATASIZE_2</code>	trace pin used for sync mode and data size is 2
<code>TRACE_MODE_SYNC</code> <code>_DATASIZE_4</code>	trace pin used for sync mode and data size is 4
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
dbg_trace_pin_mode_set(TRACE_MODE_ASYNC);
```

3.11. DCI

The DCI is a parallel interface able to capture video or picture from a camera via Digital Camera Interface. The DCI registers are listed in chapter [3.11.1](#). the DCI firmware functions are introduced in chapter [3.11.2](#).

3.11.1. Descriptions of Peripheral registers

DCI registers are listed in the table shown as below:

Table 3-293. DCI Registers

Registers	Descriptions
DCI_CTL	DCI control register
DCI_STAT0	DCI status register 0
DCI_STAT1	DCI status register 1
DCI_INTEN	DCI interrupt enable register
DCI_INTF	DCI interrupt flag register
DCI_INTC	DCI interrupt clear register
DCI_SC	DCI synchronization codes register
DCI_SCUMSK	DCI synchronization codes unmask register
DCI_CWSPOS	DCI cropping window start position register
DCI_CWSZ	DCI cropping window size register
DCI_DATA	DCI data register

3.11.2. Descriptions of Peripheral functions

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

Table 3-294. DCI firmware function

Function name	Function description
dc_i_deinit	DCI deinit
dc_i_init	initialize DCI registers
dc_i_enable	enable DCI function
dc_i_disable	disble DCI function
dc_i_capture_enable	enable DCI capture
dc_i_capture_disable	disble DCI capture
dc_i_external_vsync_enable	enable DCI external vsync in CCIR progressive mode
dc_i_external_vsync_disable	disable DCI external vsync in CCIR progressive mode
dc_i_automatic_error_correction_enable	enable DCI automatic error correction in CCIR interlaced mode
dc_i_automatic_error_correction_disable	disable DCI automatic error correction in CCIR interlaced mode
dc_i_jpeg_enable	enable DCI jpeg mode
dc_i_jpeg_disable	disble DCI jpeg mode
dc_i_crop_window_enable	enable cropping window function
dc_i_crop_window_disable	disble cropping window function
dc_i_ccir_enable	CCIR enable
dc_i_ccir_disable	CCIR disable
dc_i_ccir_mode_select	CCIR mode select
dc_i_crop_window_config	config DCI cropping window
dc_i_embedded_sync_enable	enable embedded synchronous mode
dc_i_embedded_sync_disable	disble embedded synchronous mode
dc_i_sync_codes_config	config synchronous codes in embedded synchronous mode
dc_i_sync_codes_unmask_config	config synchronous codes unmask in embedded synchronous mode
dc_i_data_read	read DCI data register
dc_i_flag_get	get specified flag
dc_i_interrupt_enable	enable specified DCI interrupt
dc_i_interrupt_disable	disable specified DCI interrupt
dc_i_interrupt_flag_get	get specified interrupt flag
dc_i_interrupt_flag_clear	clear specified interrupt flag

Structure dc_i_parameter_struct

Table 3-295. Structure dc_i_parameter_struct

Member name	Function description
capture_mode	DCI capture mode: DCI_CAPTURE_MODE_CONTINUOUS / DCI_CAPTURE_MODE_SNAPSHOT
clock_polarity	clock polarity selection: DCI_CK_POLARITY_FALLING / DCI_CK_POLARITY_RISING

hsync_polarity	horizontal polarity selection: DCI_HSYNC_POLARITY_LOW / DCI_HSYNC_POLARITY_HIGH
vsync_polarity	vertical polarity selection: DCI_VSYNC_POLARITY_LOW / DCI_VSYNC_POLARITY_HIGH
frame_rate	frame capture rate: DCI_FRAME_RATE_ALL / DCI_FRAME_RATE_1_2 / DCI_FRAME_RATE_1_4
interface_format	digital camera interface format: DCI_INTERFACE_FORMAT_8BITS / DCI_INTERFACE_FORMAT_10BITS / DCI_INTERFACE_FORMAT_12BITS / DCI_INTERFACE_FORMAT_14BITS

dc_i_deinit

The description of dc_i_deinit is shown as below:

Table 3-296. Function dc_i_deinit

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

Example:

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

```
dc_i_deinit();
```

dc_i_init

The description of dc_i_init is shown as below:

Table 3-297. Function dc_i_init

Function name	dc_i_init
Function prototype	void dc_i_init(dc_i_parameter_struct* dc_i_struct);
Function descriptions	initialize DCI registers
Precondition	-
The called functions	-
Input parameter{in}	

dci_struct	address of DCI parameter initialization struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize DCI registers */

dci_parameter_struct dci_struct;

dci_struct.capture_mode = DCI_CAPTURE_MODE_CONTINUOUS;

dci_struct.clock_polarity = DCI_CK_POLARITY_RISING;

dci_struct.hsync_polarity = DCI_HSYNC_POLARITY_LOW;

dci_struct.vsync_polarity = DCI_VSYNC_POLARITY_LOW;

dci_struct.frame_rate = DCI_FRAME_RATE_ALL;

dci_struct.interface_format = DCI_INTERFACE_FORMAT_8BITS;

dci_init(&dci_struct);

```

dci_enable

The description of dci_enable is shown as below:

Table 3-298. Function dci_enable

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

Example:

```

/* enable DCI function */

dci_enable();

```

dc_i_disable

The description of dc_i_disable is shown as below:

Table 3-299. Function dc_i_disable

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

Example:

```
/* disable DCI function */
dc_i_disable();
```

dc_i_capture_enable

The description of dc_i_capture_enable is shown as below:

Table 3-300. Function dc_i_capture_enable

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

Example:

```
/* enable DCI capture */
dc_i_capture_enable();
```


dc_i_capture_disable

The description of dc_i_capture_disable is shown as below:

Table 3-301. Function dc_i_capture_disable

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

Example:

```
/* disable DCI capture */
dc_i_capture_disable();
```

dc_i_external_vsync_enable

The description of dc_i_external_vsync_enable is shown as below:

Table 3-302. Function dc_i_external_vsync_enable

Function name	dc_i_external_vsync_enable
Function prototype	void dc_i_external_vsync_enable (void);
Function descriptions	enable DCI external vsync in CCIR progressive mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DCI external vsync in CCIR progressive mode */
dc_i_external_vsync_enable();
```

dc_i_external_vsync_disable

The description of dc_i_external_vsync_disable is shown as below:

Table 3-303. Function dc_i_external_vsync_disable

Function name	dc_i_external_vsync_disable
Function prototype	void dc_i_external_vsync_disable (void);
Function descriptions	disable DCI external vsync in CCIR progressive mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DCI external vsync in CCIR progressive mode */
dc_i_external_vsync_disable();
```

dc_i_automatic_error_correction_enable

The description of dc_i_automatic_error_correction_enable is shown as below:

Table 3-304. Function dc_i_automatic_error_correction_enable

Function name	dc_i_automatic_error_correction_enable
Function prototype	void dc_i_automatic_error_correction_enable(void);
Function descriptions	enable DCI automatic error correction in CCIR interlaced mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DCI automatic error correction in CCIR interlaced mode */
dc_i_automatic_error_correction_enable();
```

dc_i_automati_c_error_correction_disable

The description of dc_i_automati_c_error_correction_disable is shown as below:

Table 3-305. Function dc_i_automati_c_error_correction_disable

Function name	dc_i_automati_c_error_correction_disable
Function prototype	void dc_i_automati_c_error_correction_disable(void);
Function descriptions	disable DCI automatic error correction in CCIR interlaced mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DCI automatic error correction in CCIR interlaced mode */
dc_i_automati_c_error_correction_disable();
```

dc_i_peg_enable

The description of dc_i_peg_enable is shown as below:

Table 3-306. Function dc_i_peg_enable

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

Example:

```
/* enable DCI jpeg mode */
dc_i_peg_enable();
```

dcj_jpeg_disable

The description of dcj_jpeg_disable is shown as below:

Table 3-307. Function dcj_jpeg_disable

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

Example:

```
/* disable DCI jpeg mode */
dcj_jpeg_disable();
```

dcj_crop_window_enable

The description of dcj_crop_window_enable is shown as below:

Table 3-308. Function dcj_crop_window_enable

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

Example:

```
/* enable cropping window function */
dcj_crop_window_enable();
```

dc_i_crop_window_disable

The description of dc_i_crop_window_disable is shown as below:

Table 3-309. Function dc_i_crop_window_disable

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

Example:

```
/* disable cropping window function */
dc_i_crop_window_disable();
```

dc_i_crop_window_config

The description of dc_i_crop_window_config is shown as below:

Table 3-310. Function dc_i_crop_window_config

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

Example:

```
/* config DCI cropping window */

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

dc_i_embedded_sync_enable

The description of dc_i_embedded_sync_enable is shown as below:

Table 3-311. Function dc_i_embedded_sync_enable

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

Example:

```
/* enable embedded synchronous mode */

dc_i_embedded_sync_enable();
```

dc_i_embedded_sync_disable

The description of dc_i_embedded_sync_disable is shown as below:

Table 3-312. Function dc_i_embedded_sync_disable

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

Example:

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

```
dc_i_embedded_sync_disable();
```

dc_i_ccir_enable

The description of dc_i_ccir_enable is shown as below:

Table 3-313. Function dc_i_ccir_enable

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

Example:

```
/* CCIR mode enable */
```

```
dc_i_ccir_enable();
```

dc_i_ccir_disable

The description of dc_i_ccir_disable is shown as below:

Table 3-314. Function dc_i_ccir_disable

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

Example:

```
/* CCIR mode disable */
```

```
dcir_ccir_disable();
```

dcir_ccir_mode_select

The description of `dcir_ccir_mode_select` is shown as below:

Table 3-315. Function `dcir_ccir_mode_select`

Function name	<code>dcir_ccir_mode_select</code>
Function prototype	<code>void dcir_ccir_mode_select(uint32_t ccir_mode);</code>
Function descriptions	CCIR mode select
Precondition	-
The called functions	-
Input parameter{in}	
ccir_mode	CCIR mode
<code>CCIR_PROGRESSIVE_MODE</code>	CCIR Progressive mode
<code>CCIR_INTERLACE_MODE</code>	CCIR Interlace mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* CCIR mode disable */
dcir_ccir_mode_select(CCIR_Interlace_Mode);
```

dcir_sync_codes_config

The description of `dcir_sync_codes_config` is shown as below:

Table 3-316. Function `dcir_sync_codes_config`

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

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

Example:

```
/* config synchronous codes in embedded synchronous mode */
```

```
dc_i_sync_codes_config(0x10, 0x10, 0x20, 0x20);
```

dc_i_sync_codes_unmask_config

The description of dc_i_sync_codes_unmask_config is shown as below:

Table 3-317. Function dc_i_sync_codes_unmask_config

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

Example:

```
/* config synchronous codes unmask in embedded synchronous mode */
```

```
dc_i_sync_codes_unmask_config(0x10, 0x10, 0x20, 0x20);
```

dc_i_data_read

The description of dc_i_data_read is shown as below:

Table 3-318. Function dci_data_read

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

Example:

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

dci_flag_get

The description of dci_flag_get is shown as below:

Table 3-319. Function dci_flag_get

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

Example:

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

```
FlagStatus status = dci_flag_get(DCI_FLAG_HS);
```

dc_i_interrupt_enable

The description of dc_i_interrupt_enable is shown as below:

Table 3-320. Function dc_i_interrupt_enable

Function name	dc_i_interrupt_enable
Function prototype	void dc_i_interrupt_enable(uint32_t interrupt);
Function descriptions	enable specified DCI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	DCI interrupt
<i>DCI_INT_EF</i>	end of frame interrupt
<i>DCI_INT_OVR</i>	FIFO overrun interrupt
<i>DCI_INT_ESE</i>	embedded synchronous error interrupt
<i>DCI_INT_VSYNC</i>	vsync interrupt
<i>DCI_INT_EL</i>	end of line interrupt
<i>DCI_INT_F0</i>	CCIR field 0 interrupt
<i>DCI_INT_F1</i>	CCIR field 1 interrupt
<i>DCI_INT_COF</i>	CCIR change of field interrupt
<i>DCI_INT_CCE</i>	CCIR error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable specified DCI interrupt */
```

```
dc_i_interrupt_enable(DCI_INT_EF);
```

dc_i_interrupt_disable

The description of dc_i_interrupt_disable is shown as below:

Table 3-321. Function dc_i_interrupt_disable

Function name	dc_i_interrupt_disable
Function prototype	void dc_i_interrupt_disable(uint32_t interrupt);
Function descriptions	disable specified DCI interrupt
Precondition	-
The called functions	-
Input parameter{in}	

interrupt	DCI interrupt
<i>DCI_INT_EF</i>	end of frame interrupt
<i>DCI_INT_OVR</i>	FIFO overrun interrupt
<i>DCI_INT_ESE</i>	embedded synchronous error interrupt
<i>DCI_INT_VSYNC</i>	vsync interrupt
<i>DCI_INT_EL</i>	end of line interrupt
<i>DCI_INT_F0</i>	CCIR field 0 interrupt
<i>DCI_INT_F1</i>	CCIR field 1 interrupt
<i>DCI_INT_COF</i>	CCIR change of field interrupt
<i>DCI_INT_CCE</i>	CCIR error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
dc_i_interrupt_disable(DCI_INT_EF);
```

dc_i_interrupt_flag_get

The description of dc_i_interrupt_flag_get is shown as below:

Table 3-322. Function dc_i_interrupt_flag_get

Function name	dc_i_interrupt_flag_get
Function prototype	FlagStatus dc_i_interrupt_flag_get(uint32_t interrupt);
Function descriptions	get specified interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	DCI interrupt
<i>DCI_INT_FLAG_EF</i>	end of frame interrupt flag
<i>DCI_INT_FLAG_OVR</i>	FIFO overrun interrupt flag
<i>DCI_INT_FLAG_ESE</i>	embedded synchronous error interrupt flag
<i>DCI_INT_FLAG_VSYN C</i>	vsync interrupt flag
<i>DCI_INT_FLAG_EL</i>	end of line interrupt flag
<i>DCI_INT_FLAG_F0</i>	CCIR field 0 interrupt flag
<i>DCI_INT_FLAG_F1</i>	CCIR field 1 interrupt flag
<i>DCI_INT_FLAG_COF</i>	CCIR change of field interrupt flag
<i>DCI_INT_FLAG_CCE</i>	CCIR error interrupt flag
Output parameter{out}	
-	-

Return value	
FlagStatus	FlagStatus: SET or RESET

Example:

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

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

dc_i_interrupt_flag_clear

The description of dc_i_interrupt_flag_clear is shown as below:

Table 3-323. Function dc_i_interrupt_flag_clear

Function name	dc_i_interrupt_flag_clear
Function prototype	void dc_i_interrupt_flag_clear(uint32_t interrupt);
Function descriptions	clear specified interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	DCI interrupt
DCI_INT_FLAG_EF	end of frame interrupt flag
DCI_INT_FLAG_OVR	FIFO overrun interrupt flag
DCI_INT_FLAG_ESE	embedded synchronous error interrupt flag
DCI_INT_FLAG_VSYN C	vsync interrupt flag
DCI_INT_FLAG_EL	end of line interrupt flag
DCI_INT_FLAG_COF	CCIR change of field interrupt flag
DCI_INT_FLAG_CCE	CCIR error interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear specified interrupt flag */
```

```
dc_i_interrupt_flag_clear (DCI_INT_FLAG_EF);
```

3.12. 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 [3.12.1](#), the DMA firmware functions are introduced in chapter [3.12.2](#).

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 [3.12.1](#), the DMAMUX firmware functions are introduced in chapter [3.12.2](#).

3.12.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

Table 3-324. 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
DMA_CHxM1ADDR (x=0..7)	Channel x memory 1 base address register
DMA_CHxFCTL	Channel x FIFO control register

DMAMUX registers are listed in the table shown as below:

Table 3-325. 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	Request generator channel interrupt flag clear register

3.12.2. Descriptions of Peripheral functions

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

Table 3-326. DMA firmware function

Function name	Function description
<code>dma_deinit</code>	deinitialize DMA registers of a channel
<code>dma_single_data_para_struct_init</code>	initialize the DMA single data mode parameters struct with the default values
<code>dma_multi_data_para_struct_init</code>	initialize the DMA multi data mode parameters struct with the default values
<code>dma_single_data_mode_init</code>	initialize DMA single data mode
<code>dma_multi_data_mode_init</code>	initialize DMA multi data mode
<code>dma_periph_address_config</code>	configure DMA peripheral base address
<code>dma_memory_address_config</code>	configure DMA memory base address
<code>dma_transfer_number_config</code>	configure the number of remaining data to be transferred by the DMA
<code>dma_transfer_number_get</code>	get the number of remaining data to be transferred by the DMA
<code>dma_priority_config</code>	configure priority level of DMA channel
<code>dma_memory_burst_beats_config</code>	configure transfer burst beats of memory
<code>dma_periph_burst_beats_config</code>	configure transfer burst beats of peripheral
<code>dma_memory_width_config</code>	configure transfer data size of memory
<code>dma_periph_width_config</code>	configure transfer data size of peripheral
<code>dma_memory_address_generation_config</code>	configure memory address generation algorithm
<code>dma_peripheral_address_generation_config</code>	configure peripheral address generation algorithm
<code>dma_circulation_enable</code>	enable DMA circulation mode
<code>dma_circulation_disable</code>	disable DMA circulation mode
<code>dma_channel_enable</code>	enable DMA channel
<code>dma_channel_disable</code>	disable DMA channel
<code>dma_transfer_direction_config</code>	configure the direction of data transfer on the channel
<code>dma_switch_buffer_mode_config</code>	configure DMA switch buffer mode
<code>dma_using_memory_get</code>	get DMA using memory
<code>dma_switch_buffer_mode_enable</code>	enable DMA switch buffer mode
<code>dma_switch_buffer_mode_disable</code>	disable DMA switch buffer mode
<code>dma_fifo_status_get</code>	get DMA FIFO status
<code>dma_flag_get</code>	get DMA flag
<code>dma_flag_clear</code>	clear DMA flag
<code>dma_interrupt_enable</code>	enable DMA interrupt
<code>dma_interrupt_disable</code>	disable DMA interrupt
<code>dma_interrupt_flag_get</code>	get DMA interrupt flag

Function name	Function description
dma_interrupt_flag_clear	clear DMA interrupt flag

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

Table 3-327. DMAMUX firmware function

Function name	Function description
dmamux_sync_struct_para_init	initialize the parameters of DMAMUX synchronization mode structure with the default values
dmamux_synchronization_init	initialize DMAMUX request multiplexer channel synchronization mode
dmamux_synchronization_enable	enable synchronization mode
dmamux_synchronization_disable	disable synchronization mode
dmamux_event_generation_enable	enable event generation
dmamux_event_generation_disable	disable event generation
dmamux_gen_struct_para_init	initialize the parameters of DMAMUX request generator structure with the default values
dmamux_request_generator_init	initialize DMAMUX request generator channel
dmamux_request_generator_channel_enable	enable DMAMUX request generator channel
dmamux_request_generator_channel_disable	disable DMAMUX request generator channel
dmamux_synchronization_polarity_config	configure synchronization input polarity
dmamux_request_forward_number_config	configure number of DMA requests to forward
dmamux_sync_id_config	configure synchronization input identification
dmamux_request_id_config	configure multiplexer input identification
dmamux_trigger_polarity_config	configure trigger input polarity
dmamux_request_generate_number_config	configure number of DMA requests to be generated
dmamux_trigger_id_config	configure trigger input identification
dmamux_flag_get	get DMAMUX flag
dmamux_flag_clear	clear DMAMUX flag
dmamux_interrupt_enable	enable DMAMUX interrupt
dmamux_interrupt_disable	disable DMAMUX interrupt
dmamux_interrupt_flag_get	get DMAMUX interrupt flag
dmamux_interrupt_flag_clear	clear DMAMUX interrupt flag

Structure dma_multi_data_parameter_struct

Table 3-328. Structure dma_multi_data_parameter_struct

Member name	Function description
request	channel input identification

Member name	Function description
periph_addr	peripheral base address
periph_width	transfer data size of peripheral
periph_inc	peripheral increasing mode
memory0_addr	memory 0 base address
memory_width	transfer data size of memory
memory_inc	memory increasing mode
memory_burst_width	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 3-329. Structure dma_single_data_parameter_struct

Member name	Function description
request	channel input identification
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 3-330. 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 3-331. 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 3-332. 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 3-333. 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	DMAMUX request multiplexer channel 9

Member name	Function description
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 3-334. Enum dmamux_generator_channel_enum

Member name	Function description
DMAMUX_GENCH0	DMAMUX request generator channel0
DMAMUX_GENCH1	DMAMUX request generator channel1
DMAMUX_GENCH2	DMAMUX request generator channel2
DMAMUX_GENCH3	DMAMUX request generator channel3
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 3-335. Enum dmamux_interrupt_enum

Member name	Function description
DMAMUX_INT_MUX XCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun interrupt
DMAMUX_INT_MUX XCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun interrupt
DMAMUX_INT_MUX XCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun interrupt
DMAMUX_INT_MUX XCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun interrupt
DMAMUX_INT_MUX XCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun interrupt
DMAMUX_INT_MUX	DMAMUX request multiplexer channel 5 synchronization overrun interrupt

XCH5_SO	
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 NCH0_TO	DMAMUX request generator channel 0 trigger overrun interrupt
DMAMUX_INT_GE NCH1_TO	DMAMUX request generator channel 1 trigger overrun interrupt
DMAMUX_INT_GE NCH2_TO	DMAMUX request generator channel 2 trigger overrun interrupt
DMAMUX_INT_GE NCH3_TO	DMAMUX request generator channel 3 trigger overrun interrupt
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 3-336. Enum dmamux_flag_enum

Member name	Function description
DMAMUX_FLAG_M	DMAMUX request multiplexer channel 0 synchronization overrun flag

UXCH0_SO	
DMAMUX_FLAG_M UXCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun flag
DMAMUX_FLAG_M UXCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun flag
DMAMUX_FLAG_M UXCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun flag
DMAMUX_FLAG_M UXCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun flag
DMAMUX_FLAG_M UXCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun flag
DMAMUX_FLAG_M UXCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun flag
DMAMUX_FLAG_M UXCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun flag
DMAMUX_FLAG_M UXCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun flag
DMAMUX_FLAG_M UXCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun flag
DMAMUX_FLAG_M UXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun flag
DMAMUX_FLAG_M UXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun flag
DMAMUX_FLAG_M UXCH12_SO	DMAMUX request multiplexer channel 12 synchronization overrun flag
DMAMUX_FLAG_M UXCH13_SO	DMAMUX request multiplexer channel 13 synchronization overrun flag
DMAMUX_FLAG_M UXCH14_SO	DMAMUX request multiplexer channel 14 synchronization overrun flag
DMAMUX_FLAG_M UXCH15_SO	DMAMUX request multiplexer channel 15 synchronization overrun flag
DMAMUX_FLAG_G ENCH0_TO	DMAMUX request generator channel 0 trigger overrun flag
DMAMUX_FLAG_G ENCH1_TO	DMAMUX request generator channel 1 trigger overrun flag
DMAMUX_FLAG_G ENCH2_TO	DMAMUX request generator channel 2 trigger overrun flag
DMAMUX_FLAG_G ENCH3_TO	DMAMUX request generator channel 3 trigger overrun flag
DMAMUX_FLAG_G ENCH4_TO	DMAMUX request generator channel 4 trigger overrun flag
DMAMUX_FLAG_G ENCH5_TO	DMAMUX request generator channel 5 trigger overrun flag

DMAMUX_FLAG_G ENCH6_TO	DMAMUX request generator channel 6 trigger overrun flag
DMAMUX_FLAG_G ENCH7_TO	DMAMUX request generator channel 7 trigger overrun flag

Enum dmamux_interrupt_flag_enum

Table 3-337. Enum dmamux_interrupt_flag_enum

Member name	Function description
DMAMUX_INT_FL G_MUXCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH12_SO	DMAMUX request multiplexer channel 12 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH13_SO	DMAMUX request multiplexer channel 13 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH14_SO	DMAMUX request multiplexer channel 14 synchronization overrun interrupt flag
DMAMUX_INT_FL G_MUXCH15_SO	DMAMUX request multiplexer channel 15 synchronization overrun interrupt flag
DMAMUX_INT_FL G_GENCH0_TO	DMAMUX request generator channel 0 trigger overrun interrupt flag

Member name	Function description
DMAMUX_INT_FLAG_GENCH1_TO	DMAMUX request generator channel 1 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH2_TO	DMAMUX request generator channel 2 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH3_TO	DMAMUX request generator channel 3 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH4_TO	DMAMUX request generator channel 4 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH5_TO	DMAMUX request generator channel 5 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH6_TO	DMAMUX request generator channel 6 trigger overrun interrupt flag

dma_deinit

The description of dma_deinit is shown as below:

Table 3-338. 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}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Output parameter{out}	
-	-
Return value	
-	-

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 3-339. Function dma_single_data_para_struct_init

Function name	dma_single_data_para_struct_init
----------------------	----------------------------------

Function prototype	void dma_single_data_para_struct_init(dma_single_data_parameter_struct *init_struct);
Function descriptions	initialize the DMA single data mode parameters struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
init_struct	the initialization data needed to initialize DMA channel, refer to Table 3-329. Structure dma_single_data_parameter_struct.
Return value	
-	-

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 3-340. Function dma_multi_data_para_struct_init

Function name	dma_multi_data_para_struct_init
Function prototype	void dma_multi_data_para_struct_init(dma_multi_data_parameter_struct *init_struct);
Function descriptions	initialize the DMA multi data mode parameters struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
init_struct	the initialization data needed to initialize DMA channel, refer to Table 3-328. Structure dma_multi_data_parameter_struct.
Return value	
-	-

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 3-341. Function dma_single_data_mode_init

Function name	dma_single_data_mode_init
Function prototype	void dma_single_data_mode_init(uint32_t dma_periph, dma_channel_enum channelx, dma_single_data_parameter_struct *init_struct);
Function descriptions	initialize DMA single data mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to Table 3-329. Structure dma_single_data_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

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 3-342. Function dma_multi_data_mode_init

Function name	dma_multi_data_mode_init
Function prototype	void dma_multi_data_mode_init(uint32_t dma_periph, dma_channel_enum channelx, dma_multi_data_parameter_struct *init_struct);
Function descriptions	initialize DMA multi data mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to Table 3-328. Structure dma_multi_data_parameter_struct .
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 3-343. Function dma_periph_address_config

Function name	dma_periph_address_config
Function prototype	void dma_periph_address_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t address);
Function descriptions	configure DMA peripheral base address
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
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 3-344. 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
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
memory_flag	DMA memory selection
<i>DMA_MEMORY_0</i>	DMA memory 0
<i>DMA_MEMORY_1</i>	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 3-345. Function dma_transfer_number_config

Function name	dma_transfer_number_config
Function prototype	void dma_transfer_number_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t number);
Function descriptions	configure the number of remaining data to be transferred by the DMA
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
number	the number of remaining data to be transferred by the DMA, 0x00000000-

	0x0000FFFF
Output parameter{out}	
-	-
Return value	
-	-

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 3-346. Function dma_transfer_number_get

Function name	dma_transfer_number_get
Function prototype	uint32_t dma_transfer_number_get(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	get the number of remaining data to be transferred by the DMA
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
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 3-347. 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
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
priority	priority level of this channel
<i>DMA_PRIORITY_LOW</i>	low priority
<i>DMA_PRIORITY_MEDIUM</i>	medium priority
<i>DMA_PRIORITY_HIGH</i>	high priority
<i>DMA_PRIORITY_ULTRA_HIGH</i>	ultra high priority
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure 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 3-348. Function dma_memory_burst_beats_config

Function name	dma_memory_burst_beats_config
Function prototype	void dma_memory_burst_beats_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t mbeat);
Function descriptions	configure transfer burst beats of memory
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .

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

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 3-349. Function dma_periph_burst_beats_config

Function name	dma_periph_burst_beats_config
Function prototype	void dma_periph_burst_beats_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t pbeat);
Function descriptions	configure transfer burst beats of peripheral
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
mbeat	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-350. Function dma_memory_width_config

Function name	dma_memory_width_config
Function prototype	void dma_memory_width_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t msize);
Function descriptions	configure transfer data size of memory
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
msize	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-351. Function dma_periph_width_config

Function name	dma_periph_width_config
Function prototype	void dma_periph_width_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t psize);
Function descriptions	configure transfer data size of peripheral
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
psize	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-352. Function dma_memory_address_generation_config

Function name	dma_memory_address_generation_config
Function prototype	void dma_memory_address_generation_config(uint32_t dma_periph, dma_channel_enum channelx, uint8_t generation_algorithm);
Function descriptions	configure memory address generation algorithm
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-

Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
generation_algorithm	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-353. Function dma_peripheral_address_generation_config

Function name	dma_peripheral_address_generation_config
Function prototype	void dma_peripheral_address_generation_config(uint32_t dma_periph, dma_channel_enum channelx, uint8_t generation_algorithm);
Function descriptions	configure peripheral address generation algorithm
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
generation_algorithm	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-354. Function dma_circulation_enable

Function name	dma_circulation_enable
Function prototype	void dma_circulation_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable DMA circulation mode
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Output parameter{out}	
-	-
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 3-355. 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}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
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 3-356. Function dma_channel_enable

Function name	dma_channel_enable
Function prototype	void dma_channel_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Output parameter{out}	
-	-
Return value	
-	-

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 3-357. Function dma_channel_disable

Function name	dma_channel_disable
Function prototype	void dma_channel_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Output parameter{out}	
-	-
Return value	
-	-

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 3-358. Function dma_transfer_direction_config

Function name	dma_transfer_direction_config
Function prototype	void dma_transfer_direction_config(uint32_t dma_periph, dma_channel_enum channelx, uint8_t direction);
Function descriptions	configure the direction of data transfer on the channel
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
direction	specify the direction of data transfer
<i>DMA_PERIPH_TO_ME</i>	read from peripheral and write to memory

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

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 3-359. Function dma_switch_buffer_mode_config

Function name	dma_switch_buffer_mode_config
Function prototype	void dma_switch_buffer_mode_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t memory1_addr, uint32_t memory_select);
Function descriptions	configure DMA switch buffer mode
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
memory1_addr	memory1 base address, 0x00000000-0xFFFFFFFF
Input parameter{in}	
memory_select	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 3-360. Function dma_using_memory_get

Function name	dma_using_memory_get
Function prototype	uint32_t dma_using_memory_get(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	get DMA using memory
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-361. Function dma_switch_buffer_mode_enable

Function name	dma_switch_buffer_mode_enable
Function prototype	void dma_switch_buffer_mode_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable DMA switch buffer mode
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	

dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Output parameter{out}	
-	-
Return value	
-	-

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 3-362. Function dma_switch_buffer_mode_disable

Function name	dma_switch_buffer_mode_disable
Function prototype	void dma_switch_buffer_mode_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable DMA switch buffer mode
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Output parameter{out}	
-	-
Return value	
-	-

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 3-363. Function dma_fifo_status_get

Function name	dma_fifo_status_get
Function prototype	uint32_t dma_fifo_status_get(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	get DMA FIFO status
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Output parameter{out}	
-	-
Return value	
uint32_t	the number of words stored in FIFO
DMA_FIFO_CNT_NO_DATA	no data
DMA_FIFO_CNT_1_WORD	1 word
DMA_FIFO_CNT_2_WORD	2 words
DMA_FIFO_CNT_3_WORD	3 words
DMA_FIFO_CNT_EMPTY	FIFO empty
DMA_FIFO_CNT_FULL	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 3-364. Function dma_flag_get

Function name	dma_flag_get
Function prototype	FlagStatus dma_flag_get(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	get DMA flag
Precondition	-

The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum.
Input parameter{in}	
flag	specify get which flag
<i>DMA_FLAG_FEE</i>	FIFO error and exception flag
<i>DMA_FLAG_SDE</i>	single data mode exception flag
<i>DMA_FLAG_TAE</i>	transfer access error flag
<i>DMA_FLAG_HTF</i>	half transfer finish flag
<i>DMA_FLAG_FTF</i>	full transfer finish flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* 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 3-365. Function dma_flag_clear

Function name	dma_flag_clear
Function prototype	void dma_flag_clear(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear DMA flag
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum.
Input parameter{in}	
flag	specify get which flag
<i>DMA_FLAG_FEE</i>	FIFO error and exception flag

<i>DMA_FLAG_SDE</i>	single data mode exception flag
<i>DMA_FLAG_TAE</i>	transfer access error flag
<i>DMA_FLAG_HTF</i>	half transfer finish flag
<i>DMA_FLAG_FTF</i>	full transfer finish flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMA0 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 3-366. Function dma_interrupt_enable

Function name	dma_interrupt_enable
Function prototype	void dma_interrupt_enable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t interrupt);
Function descriptions	enable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
interrupt	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 3-367. 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
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum .
Input parameter{in}	
interrupt	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:

```
/* 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 3-368. 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
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum.
Input parameter{in}	
int_flag	specify get which interrupt flag
<i>DMA_INT_FLAG_FEE</i>	FIFO error and exception interrupt flag
<i>DMA_INT_FLAG_SDE</i>	single data mode exception interrupt flag
<i>DMA_INT_FLAG_TAE</i>	transfer access error interrupt flag
<i>DMA_INT_FLAG_HTF</i>	half transfer finish interrupt flag
<i>DMA_INT_FLAG_FTF</i>	full 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 3-369. 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
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to Table 3-332. Enum dma_channel_enum.
Input parameter{in}	
int_flag	specify get which interrupt flag
<i>DMA_INT_FLAG_FEE</i>	FIFO error and exception interrupt flag

<i>DMA_INT_FLAG_SDE</i>	single data mode exception interrupt flag
<i>DMA_INT_FLAG_TAE</i>	transfer access error interrupt flag
<i>DMA_INT_FLAG_HTF</i>	half transfer finish interrupt flag
<i>DMA_INT_FLAG_FTF</i>	full transfer finish interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMA0 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 3-370. Function dmamux_sync_struct_para_init

Function name	dmamux_sync_struct_para_init
Function prototype	void dmamux_sync_struct_para_init(dmamux_sync_parameter_struct *init_struct);
Function descriptions	initialize the parameters of DMAMUX synchronization mode structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
init_struct	the initialization data needed to initialize DMAMUX request multiplexer channel synchronization mode, refer to Table 3-330. Structure dmamux_sync_parameter_struct .
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 3-371. Function dmamux_synchronization_init

Function name	dmamux_synchronization_init
Function prototype	void dmamux_synchronization_init(dmamux_multiplexer_channel_enum channelx, dmamux_sync_parameter_struct *init_struct);
Function descriptions	initialize DMAMUX request multiplexer channel synchronization mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to Table 3-333. Enum dmamux_multiplexer_channel_enum .
Input parameter{in}	
init_struct	the initialization data needed to initialize DMAMUX request multiplexer channel synchronization mode, refer to Table 3-330. Structure dmamux_sync_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize DMAMUX synchronization mode structure */

dmamux_sync_parameter_struct dmamux_sync_init_struct;

dmamux_sync_struct_para_init(&dmamux_sync_init_struct);

/* initialize DMA request multiplexer channel 0 with synchronization mode */

dmamux_sync_init_struct.sync_id          = DMAMUX_SYNC_EXTI0;
dmamux_sync_init_struct.sync_polarity    = DMAMUX_SYNC_RISING;
dmamux_sync_init_struct.request_number = 4;

dmamux_synchronization_init(DMAMUX_MUXCH0, &dmamux_sync_init_struct);

```

dmamux_synchronization_enable

The description of dmamux_synchronization_enable is shown as below:

Table 3-372. Function dmamux_synchronization_enable

Function name	dmamux_synchronization_enable
Function prototype	void dmamux_synchronization_enable(dmamux_multiplexer_channel_enum channelx);
Function descriptions	enable synchronization mode
Precondition	-
The called functions	-

Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to Table 3-333. Enum dmamux_multiplexer_channel_enum .
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

dmamux_synchronization_disable

The description of dmamux_synchronization_disable is shown as below:

Table 3-373. 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 Table 3-333. Enum dmamux_multiplexer_channel_enum .
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

dmamux_event_generation_enable

The description of dmamux_event_generation_enable is shown as below:

Table 3-374. Function dmamux_event_generation_enable

Function name	dmamux_event_generation_enable
Function prototype	void dmamux_event_generation_enable(dmamux_multiplexer_channel_enum

	channelx);
Function descriptions	enable event generation
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to Table 3-333. Enum dmamux multiplexer channel enum.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable event generation */
```

```
dmamux_event_generation_enable(DMAMUX_MUXCH0);
```

dmamux_event_generation_disable

The description of dmamux_event_generation_disable is shown as below:

Table 3-375. Function dmamux_event_generation_disable

Function name	dmamux_event_generation_disable
Function prototype	void dmamux_event_generation_disable(dmamux_multiplexer_channel_enum channelx);
Function descriptions	disable event generation
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to Table 3-333. Enum dmamux multiplexer channel enum.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable event generation */
```

```
dmamux_event_generation_disable(DMAMUX_MUXCH0);
```

dmamux_gen_struct_para_init

The description of dmamux_gen_struct_para_init is shown as below:

Table 3-376. Function `dmamux_gen_struct_para_init`

Function name	<code>dmamux_gen_struct_para_init</code>
Function prototype	<code>void dmamux_gen_struct_para_init(dmamux_gen_parameter_struct *init_struct);</code>
Function descriptions	initialize the parameters of DMAMUX request generator structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
<code>init_struct</code>	the initialization data needed to initialize DMAMUX request generator channel, refer to Table 3-331. Structure <code>dmamux_gen_parameter_struct</code> .
Return value	
-	-

Example:

```
/* initialize DMA request generator structure */

dmamux_gen_parameter_struct    dmamux_gen_init_struct;

dmamux_gen_struct_para_init(&dmamux_gen_init_struct);
```

`dmamux_request_generator_init`

The description of `dmamux_request_generator_init` is shown as below:

Table 3-377. Function `dmamux_request_generator_init`

Function name	<code>dmamux_request_generator_init</code>
Function prototype	<code>void dmamux_request_generator_init(dmamux_generator_channel_enum channelx, dmamux_gen_parameter_struct *init_struct);</code>
Function descriptions	initialize DMAMUX request generator channel
Precondition	-
The called functions	-
Input parameter{in}	
<code>channelx</code>	DMAMUX generation channel, refer to Table 3-334. Enum <code>dmamux_generator_channel_enum</code> .
Input parameter{in}	
<code>init_struct</code>	the initialization data needed to initialize DMAMUX request generator channel, refer to Table 3-331. Structure <code>dmamux_gen_parameter_struct</code> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize DMA request generator channel 0 */

dmamux_gen_parameter_struct    dmamux_gen_init_struct;

dmamux_gen_struct_para_init(&dmamux_gen_init_struct);

dmamux_gen_init_struct.trigger_id      = DMAMUX_TRIGGER_EXTI13;

dmamux_gen_init_struct.trigger_polarity = DMAMUX_GEN_RISING;

dmamux_gen_init_struct.request_number = 1;

dmamux_request_generator_init(DMAMUX_GENCH0, &dmamux_gen_init_struct);
```

dmamux_request_generator_channel_enable

The description of dmamux_request_generator_channel_enable is shown as below:

Table 3-378. 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 Table 3-334. Enum dmamux_generator_channel_enum .
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 3-379. Function dmamux_request_generator_channel_disable

Function name	dmamux_request_generator_channel_disable
Function prototype	void dmamux_request_generator_channel_disable(dmamux_generator_channel

	_enum channelx);
Function descriptions	disable DMAMUX request generator channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX generation channel, refer to Table 3-334. Enum dmamux_generator_channel_enum .
Output parameter{out}	
-	-
Return value	
-	-

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 3-380. Function dmamux_synchronization_polarity_config

Function name	dmamux_synchronization_polarity_config
Function prototype	void dmamux_synchronization_polarity_config(dmamux_multiplexer_channel_enum channelx, uint32_t polarity);
Function descriptions	configure synchronization input polarity
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to Table 3-333. Enum dmamux_multiplexer_channel_enum .
Input parameter{in}	
polarity	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
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* configure synchronization input polarity */
```

```
dmamux_synchronization_polarity_config(DMAMUX_MUXCH0, DMAMUX_SYNC_RISING);
```

dmamux_request_forward_number_config

The description of dmamux_request_forward_number_config is shown as below:

Table 3-381. Function dmamux_request_forward_number_config

Function name	dmamux_request_forward_number_config
Function prototype	void dmamux_request_forward_number_config(dmamux_multiplexer_channel_enum channelx, uint32_t number);
Function descriptions	configure number of DMA requests to forward
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to Table 3-333. Enum dmamux_multiplexer_channel_enum .
Input parameter{in}	
number	DMA requests number to forward (1 - 32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure number of DMA requests to forward */
```

```
dmamux_request_forward_number_config(DMAMUX_MUXCH0, 4);
```

dmamux_sync_id_config

The description of dmamux_sync_id_config is shown as below:

Table 3-382. Function dmamux_sync_id_config

Function name	dmamux_sync_id_config
Function prototype	void dmamux_sync_id_config(dmamux_multiplexer_channel_enum channelx, uint32_t id);
Function descriptions	configure synchronization input identification
Precondition	-
The called functions	-

Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to Table 3-333. Enum dmamux multiplexer channel enum .
Input parameter{in}	
id	synchronization input identification
DMAMUX_SYNC_EXTI0	synchronization input is EXTI0
DMAMUX_SYNC_EXTI1	synchronization input is EXTI1
DMAMUX_SYNC_EXTI2	synchronization input is EXTI2
DMAMUX_SYNC_EXTI3	synchronization input is EXTI3
DMAMUX_SYNC_EXTI4	synchronization input is EXTI4
DMAMUX_SYNC_EXTI5	synchronization input is EXTI5
DMAMUX_SYNC_EXTI6	synchronization input is EXTI6
DMAMUX_SYNC_EXTI7	synchronization input is EXTI7
DMAMUX_SYNC_EXTI8	synchronization input is EXTI8
DMAMUX_SYNC_EXTI9	synchronization input is EXTI9
DMAMUX_SYNC_EXTI10	synchronization input is EXTI10
DMAMUX_SYNC_EXTI11	synchronization input is EXTI11
DMAMUX_SYNC_EXTI12	synchronization input is EXTI12
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

DMAMUX_SYNC_EVT X_OUT3	synchronization input is Evt_out3
DMAMUX_SYNC_EVT X_OUT4	synchronization input is Evt_out4
DMAMUX_SYNC_EVT X_OUT5	synchronization input is Evt_out5
DMAMUX_SYNC_EVT X_OUT6	synchronization input is Evt_out6
DMAMUX_SYNC_RTC _WAKEUP	synchronization input is RTC wakeup
DMAMUX_SYNC_CMP 0_OUTPUT	synchronization input is CMP0 output
DMAMUX_SYNC_I2C0 _WAKEUP	synchronization input is I2C0 wakeup
DMAMUX_SYNC_I2C1 _WAKEUP	synchronization input is I2C1 wakeup
DMAMUX_SYNC_I2C2 _WAKEUP	synchronization input is I2C2 wakeup
DMAMUX_SYNC_I2C3 _WAKEUP	synchronization input is I2C3 wakeup
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure synchronization input identification */
```

```
dmamux_sync_id_config(DMAMUX_MUXCH0, DMAMUX_SYNC_EXTI0);
```

dmamux_request_id_config

The description of dmamux_request_id_config is shown as below:

Table 3-383. Function dmamux_request_id_config

Function name	dmamux_request_id_config
Function prototype	void dmamux_request_id_config(dmamux_multiplexer_channel_enum channelx, uint32_t id);
Function descriptions	configure multiplexer input identification
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to Table 3-333. Enum dmamux_multiplexer_channel_enum .

Input parameter{in}	
id	input DMA request identification
<i>DMA_REQUEST_M2M</i>	memory to memory transfer
<i>DMA_REQUEST_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_USART0_RT0_RX</i>	DMAMUX USART0 RX request
<i>DMA_REQUEST_USART0_RT0_TX</i>	DMAMUX USART0 TX request
<i>DMA_REQUEST_USART0_RT1_RX</i>	DMAMUX USART1 RX request
<i>DMA_REQUEST_USART0_RT1_TX</i>	DMAMUX USART1 TX request
<i>DMA_REQUEST_USART0_RT2_RX</i>	DMAMUX USART2 RX request
<i>DMA_REQUEST_USART0_RT2_TX</i>	DMAMUX USART2 TX request
<i>DMA_REQUEST_TIMER7_R7_CH0</i>	DMAMUX TIMER7 CH0 request
<i>DMA_REQUEST_TIMER7_R7_CH1</i>	DMAMUX TIMER7 CH1 request
<i>DMA_REQUEST_TIMER7_R7_CH2</i>	DMAMUX TIMER7 CH2 request
<i>DMA_REQUEST_TIMER7_R7_CH3</i>	DMAMUX TIMER7 CH3 request
<i>DMA_REQUEST_TIMER7_R7_MCH0</i>	DMAMUX TIMER7 MCH0 request
<i>DMA_REQUEST_TIMER7_R7_MCH1</i>	DMAMUX TIMER7 MCH1 request
<i>DMA_REQUEST_TIMER7_R7_MCH2</i>	DMAMUX TIMER7 MCH2 request
<i>DMA_REQUEST_TIMER7_R7_MCH3</i>	DMAMUX TIMER7 MCH3 request
<i>DMA_REQUEST_TIMER7_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_DCI</i>	DMAMUX DCI request
<i>DMA_REQUEST_CAU</i> <i>_IN</i>	DMAMUX CAU IN request
<i>DMA_REQUEST_CAU</i> <i>_OUT</i>	DMAMUX CAU OUT request
<i>DMA_REQUEST_HAU</i> <i>_IN</i>	DMAMUX HAU IN request
<i>DMA_REQUEST_UAR</i> <i>T6_RX</i>	DMAMUX UART6 RX request
<i>DMA_REQUEST_UAR</i> <i>T6_TX</i>	DMAMUX UART6 TX request
<i>DMA_REQUEST_UAR</i> <i>T7_RX</i>	DMAMUX UART7 RX request
<i>DMA_REQUEST_UAR</i> <i>T7_TX</i>	DMAMUX UART7 TX request
<i>DMA_REQUEST_SPI3</i> <i>_RX</i>	DMAMUX SPI3 RX request
<i>DMA_REQUEST_SPI3</i> <i>_TX</i>	DMAMUX SPI3 TX request
<i>DMA_REQUEST_SPI4</i> <i>_RX</i>	DMAMUX SPI4 RX request
<i>DMA_REQUEST_SPI4</i> <i>_TX</i>	DMAMUX SPI4 TX request
<i>DMA_REQUEST_SAI0</i> <i>_B0</i>	DMAMUX SAI0 B0 request
<i>DMA_REQUEST_SAI0</i> <i>_B1</i>	DMAMUX SAI0 B1 request
<i>DMA_REQUEST_RSP</i> <i>DIF_DATA</i>	DMAMUX RSPDIF DATA request
<i>DMA_REQUEST_RSP</i> <i>DIF_CS</i>	DMAMUX RSPDIF CS request
<i>DMA_REQUEST_HPD</i> <i>F_FLT0</i>	DMAMUX HPDF FLT0 request
<i>DMA_REQUEST_HPD</i> <i>F_FLT1</i>	DMAMUX HPDF FLT1 request
<i>DMA_REQUEST_HPD</i> <i>F_FLT2</i>	DMAMUX HPDF FLT2 request
<i>DMA_REQUEST_HPD</i> <i>F_FLT3</i>	DMAMUX HPDF FLT3 request
<i>DMA_REQUEST_TIME</i> <i>R14_CH0</i>	DMAMUX TIMER14 CH0 request
<i>DMA_REQUEST_TIME</i>	DMAMUX TIMER14 CH1 request

<i>R14_CH1</i>	
<i>DMA_REQUEST_TIME</i> <i>R14_MCH0</i>	DMAMUX TIMER14 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R14_UP</i>	DMAMUX TIMER14 UP request
<i>DMA_REQUEST_TIME</i> <i>R14_TRG</i>	DMAMUX TIMER14 TRG request
<i>DMA_REQUEST_TIME</i> <i>R14_CMT</i>	DMAMUX TIMER14 CMT request
<i>DMA_REQUEST_TIME</i> <i>R15_CH0</i>	DMAMUX TIMER15 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R15_MCH0</i>	DMAMUX TIMER15 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>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>R30_CH0</i>	DMAMUX TIMER30 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R30_CH1</i>	DMAMUX TIMER30 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R30_CH2</i>	DMAMUX TIMER30 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R30_CH3</i>	DMAMUX TIMER30 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R30_UP</i>	DMAMUX TIMER30 UP request
<i>DMA_REQUEST_TIME</i> <i>R30_TRG</i>	DMAMUX TIMER30 TRG request
<i>DMA_REQUEST_TIME</i> <i>R31_CH0</i>	DMAMUX TIMER31 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R31_CH1</i>	DMAMUX TIMER31 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R31_CH2</i>	DMAMUX TIMER31 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R31_CH3</i>	DMAMUX TIMER31 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R31_UP</i>	DMAMUX TIMER31 UP request
<i>DMA_REQUEST_TIME</i> <i>R31_TRG</i>	DMAMUX TIMER31 TRG request
<i>DMA_REQUEST_TIME</i> <i>R40_CH0</i>	DMAMUX TIMER40 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R40_MCH0</i>	DMAMUX TIMER40 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R40_CMT</i>	DMAMUX TIMER40 CMT request
<i>DMA_REQUEST_TIME</i>	DMAMUX TIMER40 UP request

<i>R40_UP</i>	
<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_SAI1</i> <i>_B0</i>	DMAMUX SAI1 B0 request
<i>DMA_REQUEST_SAI1</i> <i>_B1</i>	DMAMUX SAI1 B1 request
<i>DMA_REQUEST_SAI2</i> <i>_B0</i>	DMAMUX SAI2 B0 request

<i>DMA_REQUEST_SAI2</i> <i>_B1</i>	DMAMUX SAI2 B1 request
<i>DMA_REQUEST_SPI5</i> <i>_RX</i>	DMAMUX SPI5 RX request
<i>DMA_REQUEST_SPI5</i> <i>_TX</i>	DMAMUX SPI5 TX request
<i>DMA_REQUEST_I2C3</i> <i>_RX</i>	DMAMUX I2C3 RX request
<i>DMA_REQUEST_I2C3</i> <i>_TX</i>	DMAMUX I2C3 TX request
<i>DMA_REQUEST_CAN</i> <i>_0</i>	DMAMUX CAN0 request
<i>DMA_REQUEST_CAN</i> <i>_1</i>	DMAMUX CAN1 request
<i>DMA_REQUEST_CAN</i> <i>_2</i>	DMAMUX CAN2 request
<i>DMA_REQUEST_TIME</i> <i>R40_CH1</i>	DMAMUX TIMER40 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R40_TRG</i>	DMAMUX TIMER40 TRG request
<i>DMA_REQUEST_TIME</i> <i>R41_CH1</i>	DMAMUX TIMER41 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R41_TRG</i>	DMAMUX TIMER41 TRG request
<i>DMA_REQUEST_TIME</i> <i>R42_CH1</i>	DMAMUX TIMER42 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R42_TRG</i>	DMAMUX TIMER42 TRG request
<i>DMA_REQUEST_TIME</i> <i>R43_CH1</i>	DMAMUX TIMER43 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R43_TRG</i>	DMAMUX TIMER43 TRG request
<i>DMA_REQUEST_TIME</i> <i>R44_CH1</i>	DMAMUX TIMER44 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R44_TRG</i>	DMAMUX TIMER44 TRG request
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure multiplexer input identification */
```



```
dmamux_request_id_config(DMAMUX_MUXCH0, DMA_REQUEST_GENERATOR0);
```

dmamux_trigger_polarity_config

The description of dmamux_trigger_polarity_config is shown as below:

Table 3-384. 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 Table 3-334. Enum dmamux_generator_channel_enum .
Input parameter{in}	
polarity	trigger input polarity
DMAMUX_GEN_NO_EVENT	no event detection
DMAMUX_GEN_RISING	rising edge
DMAMUX_GEN_FALLING	falling edge
DMAMUX_GEN_RISING_FALLING	rising and falling edges
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

dmamux_request_generate_number_config

The description of dmamux_request_generate_number_config is shown as below:

Table 3-385. 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 Table 3-334. Enum dmamux_generator_channel_enum.
Input parameter{in}	
number	DMA requests number to be generated (1 - 32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure number of DMA requests to be generated */
```

```
dmamux_request_generate_number_config(DMAMUX_GENCH0, 1);
```

dmamux_trigger_id_config

The description of dmamux_trigger_id_config is shown as below:

Table 3-386. 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 Table 3-334. Enum dmamux_generator_channel_enum.
Input parameter{in}	
id	trigger input identification
DMAMUX_TRIGGER_EXTI0	trigger input is EXTI0
DMAMUX_TRIGGER_EXTI1	trigger input is EXTI1
DMAMUX_TRIGGER_EXTI2	trigger input is EXTI2
DMAMUX_TRIGGER_EXTI3	trigger input is EXTI3
DMAMUX_TRIGGER_EXTI4	trigger input is EXTI4

<i>DMAMUX_TRIGGER_EXTI5</i>	trigger input is EXTI5
<i>DMAMUX_TRIGGER_EXTI6</i>	trigger input is EXTI6
<i>DMAMUX_TRIGGER_EXTI7</i>	trigger input is EXTI7
<i>DMAMUX_TRIGGER_EXTI8</i>	trigger input is EXTI8
<i>DMAMUX_TRIGGER_EXTI9</i>	trigger input is EXTI9
<i>DMAMUX_TRIGGER_EXTI10</i>	trigger input is EXTI10
<i>DMAMUX_TRIGGER_EXTI11</i>	trigger input is EXTI11
<i>DMAMUX_TRIGGER_EXTI12</i>	trigger input is EXTI12
<i>DMAMUX_TRIGGER_EXTI13</i>	trigger input is EXTI13
<i>DMAMUX_TRIGGER_EXTI14</i>	trigger input is EXTI14
<i>DMAMUX_TRIGGER_EXTI15</i>	trigger input is EXTI15
<i>DMAMUX_TRIGGER_EVT_OUT0</i>	trigger input is Evt_out0
<i>DMAMUX_TRIGGER_EVT_OUT1</i>	trigger input is Evt_out1
<i>DMAMUX_TRIGGER_EVT_OUT2</i>	trigger input is Evt_out2
<i>DMAMUX_TRIGGER_EVT_OUT3</i>	trigger input is Evt_out3
<i>DMAMUX_TRIGGER_EVT_OUT4</i>	trigger input is Evt_out4
<i>DMAMUX_TRIGGER_EVT_OUT5</i>	trigger input is Evt_out5
<i>DMAMUX_TRIGGER_EVT_OUT6</i>	trigger input is Evt_out6
<i>DMAMUX_TRIGGER_EVT_OUT7</i>	trigger input is Evt_out7
<i>DMAMUX_TRIGGER_RTC_WAKEUP</i>	trigger input is wakeup
<i>DMAMUX_TRIGGER_CMP0_OUTPUT</i>	trigger input is CMP0 output
<i>DMAMUX_TRIGGER_</i>	trigger input is CMP1 output

<i>CMP1_OUTPUT</i>	
<i>DMAMUX_TRIGGER_I2C0_WAKEUP</i>	trigger input is I2C0 wakeup
<i>DMAMUX_TRIGGER_I2C1_WAKEUP</i>	trigger input is I2C1 wakeup
<i>DMAMUX_TRIGGER_I2C2_WAKEUP</i>	trigger input is I2C2 wakeup
<i>DMAMUX_TRIGGER_I2C3_WAKEUP</i>	trigger input is I2C3 wakeup
<i>DMAMUX_TRIGGER_I2C0_INT_EVENT</i>	trigger input is I2C0 interrupt event
<i>DMAMUX_TRIGGER_I2C1_INT_EVENT</i>	trigger input is I2C1 interrupt event
<i>DMAMUX_TRIGGER_I2C2_INT_EVENT</i>	trigger input is I2C2 interrupt event
<i>DMAMUX_TRIGGER_I2C3_INT_EVENT</i>	trigger input is I2C3 interrupt event
<i>DMAMUX_TRIGGER_ADC2_INT</i>	ADC2 interrupt
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 3-387. Function dmamux_flag_get

Function name	dmamux_flag_get
Function prototype	FlagStatus dmamux_flag_get(dmamux_flag_enum flag);
Function descriptions	get DMAMUX flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag type, refer to Table 3-336 .
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 3-388. Function dmamux_flag_clear

Function name	dmamux_flag_clear
Function prototype	void dmamux_flag_clear(dmamux_flag_enum flag);
Function descriptions	clear DMAMUX flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag type, refer to Table 3-336 .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMAMUX flag */
```

```
dmamux_flag_clear(DMAMUX_FLAG_GENCH0_TO);
```

dmamux_interrupt_enable

The description of dmamux_interrupt_enable is shown as below:

Table 3-389. Function dmamux_interrupt_enable

Function name	dmamux_interrupt_enable
Function prototype	void dmamux_interrupt_enable(dmamux_interrupt_enum interrupt);
Function descriptions	enable DMAMUX interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify which interrupt to enable, refer to Table 3-335. Enum dmamux_interrupt_enum .
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* enable DMAMUX interrupt */
```

```
dmamux_interrupt_enable(DMAMUX_INT_MUXCH0_SO);
```

dmamux_interrupt_disable

The description of dmamux_interrupt_disable is shown as below:

Table 3-390. Function dmamux_interrupt_disable

Function name	dmamux_interrupt_disable
Function prototype	void dmamux_interrupt_disable(dmamux_interrupt_enum interrupt);
Function descriptions	disable DMAMUX interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify which interrupt to disable, refer to Table 3-335. Enum dmamux_interrupt_enum .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMAMUX interrupt */
```

```
dmamux_interrupt_disable(DMAMUX_INT_MUXCH0_SO);
```

dmamux_interrupt_flag_get

The description of dmamux_interrupt_flag_get is shown as below:

Table 3-391. Function dmamux_interrupt_flag_get

Function name	dmamux_interrupt_flag_get
Function prototype	FlagStatus dmamux_interrupt_flag_get(dmamux_interrupt_flag_enum int_flag);
Function descriptions	get DMAMUX interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	flag type, refer to Table 3-337. Enum dmamux_interrupt_flag_enum .
Output parameter{out}	

-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get DMAMUX interrupt flag */
```

```
FlagStatus flag = RESET;
```

```
flag = dmamux_interrupt_flag_get(DMAMUX_INT_FLAG_MUXCH0_SO);
```

dmamux_interrupt_flag_clear

The description of dmamux_interrupt_flag_clear is shown as below:

Table 3-392. Function dmamux_interrupt_flag_clear

Function name	dmamux_interrupt_flag_clear
Function prototype	FlagStatus dmamux_interrupt_flag_get(dmamux_interrupt_flag_enum int_flag);
Function descriptions	clear DMAMUX interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	flag type, refer to Table 3-337. Enum dmamux_interrupt_flag_enum .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMAMUX interrupt flag */
```

```
dmamux_interrupt_flag_clear(DMAMUX_INT_FLAG_MUXCH0_SO);
```

3.13. 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 [3.13.1](#), the EDOUT firmware functions are introduced in chapter [3.13.2](#).

3.13.1. Descriptions of Peripheral registers

EDOUT registers are listed in the table shown as below:

Table 3-393. 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

3.13.2. Descriptions of Peripheral functions

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

Table 3-394. 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 3-395. 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 3-396. 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 3-397. 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 3-398. Function edout_disable

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

Example:

```
/* disable EDOUT */
edout_disable();
```

edout_polarity_config

The description of edout_polarity_config is shown as below:

Table 3-399. Function edout_polarity_config

Function name	edout_polarity_config
Function prototype	void edout_polarity_config(uint32_t pol);
Function descriptions	set B-phase active polarity
Precondition	-
The called functions	-
Input parameter{in}	

pol	the active polarity of the B-phase output signal selection
<i>EDOUT_POL_POSITIVE</i>	active polarity is positive
<i>EDOUT_POL_NEGATIV</i> <i>E</i>	active polarity is negative
Output parameter{out}	
-	-
Return value	
-	-

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 3-400. Function edout_max_location_value_config

Function name	edout_max_location_value_config
Function prototype	void edout_max_location_value_config(uint32_t max_loc);
Function descriptions	set the maximum location value for one rotation
Precondition	-
The called functions	-
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.
Output parameter{out}	
-	-
Return value	
-	-

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 3-401. Function edout_output_counter_update

Function name	edout_output_counter_update
Function prototype	void edout_output_counter_update(int16_t num_edges, uint16_t phase_diff);
Function descriptions	update the output counter, used to set the phase difference and the number of edges for the next update period

Precondition	-
The called functions	-
Input parameter{in}	
num_edges	edge count, value range is -32768~32767, positive means clockwise rotation, negative means counter-clockwise rotation
Input parameter{in}	
phase_diff	phase difference, value range is 2~65535, in units of PCLK
Output parameter{out}	
-	-
Return value	
-	-

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 3-402. Function edout_current_location_config

Function name	edout_current_location_config
Function prototype	void edout_current_location_config(uint32_t cur_loc);
Function descriptions	set the current location value
Precondition	-
The called functions	-
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:

```
/* 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 3-403. Function edout_current_location_get

Function name	edout_current_location_get
----------------------	----------------------------

Function prototype	uint16_t edout_current_location_get(void)
Function descriptions	get the current location value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	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 3-404. Function edout_z_output_mode_config

Function name	edout_z_output_mode_config
Function prototype	void edout_z_output_mode_config(uint32_t mode);
Function descriptions	configure Z-phase output mode
Precondition	-
The called functions	-
Input parameter{in}	
mode	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-405. Function edout_z_output_start_loc_and_width_config

Function name	edout_z_output_start_loc_and_width_config
Function prototype	void edout_z_output_start_loc_and_width_config(uint32_t start_loc, uint32_t width);
Function descriptions	configure Z-phase output start location and width
Precondition	-
The called functions	-
Input parameter{in}	
start_loc	Z-phase output start location
Input parameter{in}	
width	Z-phase output width
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure Z-phase output start location and width */
edout_z_output_start_loc_and_width_config(92, 2);
```

3.14. 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 [3.14.1](#), the EFUSE firmware functions are introduced in chapter [3.14.2](#).

3.14.1. Descriptions of Peripheral registers

EFUSE registers are listed in the table shown as below:

Table 3-406. 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

Registers	Descriptions
EFUSE_DP x ($x = 0,1$)	EFUSE debug password register x ($x = 0,1$)
EFUSE_AES_KEY x ($x = 0...3$)	EFUSE firmware AES key register x ($x = 0...3$)
EFUSE_USER_DATA x ($x = 0...3$)	user data register x ($x = 0...3$)

3.14.2. Descriptions of Peripheral functions

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

Table 3-407. 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
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 3-408. Enum efuse_system_para_size_enum

Member name	Function description
USER_CTL_SIZE	user control parameter size
MCU_RESERVED_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 3-409. 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 3-410. 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 3-411. Enum efuse_interrupt_flag_enum

Member name	Function description
EFUSE_INT_FLAG_ILL EGAL_ACCESS_ERR	illegal access error interrupt flag
EFUSE_INT_FLAG_PR OGRAM_COMPLETE	programming operation completion interrupt flag
EFUSE_INT_FLAG_RE AD_COMPLETE	read operation completion interrupt flag
EFUSE_INT_FLAG_PR OGRAM_VOLTAGE_ER R	program voltage setting error flag

efuse_read

The description of efuse_read is shown as below:

Table 3-412. 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
<i>USER_CTL_EFADDR</i>	user control parameter start address
<i>MCU_RESERVED_EFADDR</i>	MCU reserved parameter start address
<i>DP_EFADDR</i>	debug password parameter start address
<i>USER_DATA_EFADDR</i>	user data parameter start address
Input parameter{in}	
size	size of the system parameters to be read
<i>USER_CTL_SIZE</i>	user control parameter size
<i>MCU_RESERVED_SIZE</i>	MCU reserved parameter size
<i>DP_SIZE</i>	debug password parameter size
<i>USER_DATA_SIZE</i>	user data parameter size
Input parameter{in}	
buf	the buffer for data read from EFUSE macro
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-413. Function efuse_program

Function name	efuse_write
Function prototype	ErrStatus efuse_write(uint32_t ef_addr, uint32_t size, uint8_t *buf);
Function descriptions	program register values to EFUSE macro system parameters
Precondition	-
The called functions	-
Input parameter{in}	
ef_addr	the EFUSE address to be programmed, pgm_addr cannot exceed 384, and must be an integral multiple of 8
Input parameter{in}	
size	byte count to program
Input parameter{in}	

buf	the buffer for data written to EFUSE macro
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-414. Function efuse_user_control_write

Function name	efuse_user_control_write
Function prototype	ErrStatus efuse_user_control_write(uint8_t *buf);
Function descriptions	program all user control parameters
Precondition	-
The called functions	efuse_program
Input parameter{in}	
buf	the buffer for data written to EFUSE macro
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-415. Function efuse_mcu_reserved_write

Function name	efuse_mcu_reserved_write
Function prototype	ErrStatus efuse_mcu_reserved_write(uint8_t *buf);

Function descriptions	program all MCU reserved parameters
Precondition	-
The called functions	efuse_program
Input parameter{in}	
buf	the buffer for data written to EFUSE macro
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-416. Function efuse_dp_write

Function name	efuse_dp_write
Function prototype	ErrStatus efuse_dp_write(uint8_t *buf);
Function descriptions	program all debug password parameters
Precondition	-
The called functions	efuse_program
Input parameter{in}	
buf	the buffer for data written to EFUSE macro
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-417. Function efuse_aes_key_write

Function name	efuse_aes_key_write
Function prototype	ErrStatus efuse_aes_key_write(uint8_t *buf);
Function descriptions	program all AES key parameters
Precondition	-
The called functions	efuse_program
Input parameter{in}	
buf	the buffer for data written to EFUSE macro
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-418. Function efuse_user_data_write

Function name	efuse_user_data_write
Function prototype	ErrStatus efuse_user_data_write(uint8_t *buf);
Function descriptions	program all user data parameters
Precondition	-
The called functions	efuse_program
Input parameter{in}	
buf	the buffer for data written to EFUSE macro
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-419. Function efuse_aes_key_crc_get

Function name	efuse_aes_key_crc_get
Function prototype	uint8_t efuse_aes_key_crc_get(void);
Function descriptions	get 8-bits CRC calculation result value of AES key
Precondition	-
The called functions	-
Input parameter{in}	
-	-
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 3-420. 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 3-421. 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 3-422. Function efuse_monitor_program_voltage_get

Function name	efuse_monitor_program_voltage_get
Function prototype	void efuse_monitor_program_voltage_get(void);
Function descriptions	get monitor program voltage function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-423. Function efuse_ldo_ready_get

Function name	efuse_ldo_ready_get
Function prototype	FlagStatus efuse_ldo_ready_get(void);
Function descriptions	get ldo ready signal
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-424. Function efuse_flag_get

Function name	efuse_flag_get
Function prototype	FlagStatus efuse_flag_get(uint32_t efuse_flag);
Function descriptions	check EFUSE flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
efuse_flag	specifies to get a flag
EFUSE_FLAG_ILLEGAL_ACCESS_ERR	illegal access error flag
EFUSE_FLAG_PROGRAM_RAM_COMPLETE	programming operation completion flag
EFUSE_FLAG_READ_COMPLETE	read operation completion flag
EFUSE_FLAG_PROGRAM_VOLTAGE_ERR	program voltage setting error flag

Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-425. Function efuse_flag_clear

Function name	efuse_flag_clear
Function prototype	void efuse_flag_clear(uint32_t efuse_cflag);
Function descriptions	clear EFUSE pending flag
Precondition	-
The called functions	-
Input parameter{in}	
efuse_periph	specifies to clear a flag
EFUSE_FLAG_ILLEGAL_ACCESS_ERR_CLR	illegal access error flag
EFUSE_FLAG_PROGRAM_COMPLETE_CLR	programming operation completion flag
EFUSE_FLAG_READ_COMPLETE_CLR	read operation completion flag
EFUSE_FLAG_PROGRAM_VOLTAGE_ERROR_CLR	program voltage setting error flag
Output parameter{out}	
-	-
Return value	
-	-

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 3-426. Function efuse_interrupt_enable

Function name	efuse_interrupt_enable
Function prototype	void efuse_interrupt_enable(uint32_t source);
Function descriptions	enable EFUSE interrupt
Precondition	-
The called functions	-
Input parameter{in}	
source	specifies an interrupt to enable
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
Output parameter{out}	
-	-
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 3-427. Function efuse_interrupt_disable

Function name	efuse_interrupt_disable
Function prototype	void efuse_interrupt_disable(uint32_t source);
Function descriptions	disable EFUSE interrupt
Precondition	-
The called functions	-
Input parameter{in}	
source	specifies an interrupt to disable
EFUSE_INT_ILLEGAL_ACCESS_ERR	illegal access error interrupt
EFUSE_INT_PROGRAM_COMPLETE	programming operation completion interrupt

<i>M_COMPLETE</i>	
<i>EFUSE_INT_READ_COMPLETE</i>	read operation completion interrupt
<i>EFUSE_INT_PROGRAM_VOLTAGE_ERR</i>	program voltage setting error interrupt
Output parameter{out}	
-	-
Return value	
-	-

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 3-428. Function efuse_interrupt_flag_get

Function name	efuse_interrupt_flag_get
Function prototype	FlagStatus efuse_interrupt_flag_get(uint32_t int_flag);
Function descriptions	check EFUSE interrupt flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
efuse_flag	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
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-429. Function efuse_interrupt_flag_clear

Function name	efuse_interrupt_flag_clear
Function prototype	void efuse_interrupt_flag_clear(uint32_t efuse_periph, uint32_t int_flag);
Function descriptions	clear EFUSE pending interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
efuse_flag	specifies to clear a flag
EFUSE_INT_FLAG_ILLEGAL_ACCESS_ERR_CLR	clear illegal access error interrupt flag
EFUSE_INT_FLAG_PROGRAM_COMPLETE_CLR	clear programming operation completion interrupt flag
EFUSE_INT_FLAG_READ_COMPLETE_CLR	clear operation completion interrupt flag
EFUSE_INT_FLAG_PROGRAM_VOLTAGE_ERROR_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);
```

3.15. ENET

There are two media access controllers (MAC) designed in Ethernet module to support 10/100Mbps interface speed. For more efficient data transfer between Ethernet and memory, a DMA controller is designed in this module. The support interface protocol for Ethernet is media independent interface (MII) and reduced media independent interface (RMII). The ENET registers are listed in chapter [3.15.1](#), the ENET firmware functions are introduced in chapter [3.15.2](#).

3.15.1. Descriptions of Peripheral registers

ENET registers are listed in the table shown as below:

Table 3-430. ENET Registers

Registers	Descriptions
ENET_MAC_CFG	MAC configuration register
ENET_MAC_FRMF	MAC frame filter register
ENET_MAC_HLH	MAC hash list high register
ENET_MAC_HLL	MAC hash list low register
ENET_MAC_PHY_CTL	MAC PHY control register
ENET_MAC_PHY_DATA	MAC PHY data register
ENET_MAC_FCTL	MAC flow control register
ENET_MAC_VLT	MAC VLAN tag register
ENET_MAC_RWFF	MAC remote wakeup frame filter register
ENET_MAC_WUM	MAC wakeup management register
ENET_MAC_DBG	MAC debug register
ENET_MAC_INTF	MAC interrupt flag register
ENET_MAC_INTMSK	MAC interrupt mask register
ENET_MAC_ADDR0H	MAC address 0 high register
ENET_MAC_ADDR0L	MAC address 0 low register
ENET_MAC_ADDR1H	MAC address 1 high register
ENET_MAC_ADDR1L	MAC address 1 low register
ENET_MAC_ADDT2H	MAC address 2 high register
ENET_MAC_ADDR2L	MAC address 2 low register
ENET_MAC_ADDR3H	MAC address 3 high register
ENET_MAC_ADDR3L	MAC address 3 low register
ENET_MAC_FCTH	MAC flow control threshold register
ENET_MSC_CTL	MSC control register
ENET_MSC_RINTF	MSC receive interrupt flag register
ENET_MSC_TINTF	MSC transmit interrupt flag register
ENET_MSC_RINT	MSC receive interrupt mask register

Registers	Descriptions
MSK	
ENET_MSC_TINTMSK	MSC transmit interrupt mask register
ENET_MSC_SCCNT	MSC transmitted good frames after a single collision counter register
ENET_MSC_MSCCNT	MSC transmitted good frames after more than a single collision counter register
ENET_MSC_TGFCNT	MSC transmitted good frames counter register
ENET_MSC_RFCECNT	MSC received frames with CRC error counter register
ENET_MSC_RFAECNT	MSC received frames with alignment error counter register
ENET_MSC_RGUFcnt	MSC received good unicast frames counter register
ENET_PTP_TSCTL	PTP time stamp control register
ENET_PTP_SSINC	PTP subsecond increment register
ENET_PTP_TSH	PTP time stamp high register
ENET_PTP_TSL	PTP time stamp low register
ENET_PTP_TSUH	PTP time stamp update high register
ENET_PTP_TSUL	PTP time stamp update low register
ENET_PTP_TSADDEND	PTP time stamp addend register
ENET_PTP_ETH	PTP expected time high register
ENET_PTP_ETL	PTP expected time low register
ENET_PTP_TSF	PTP time stamp flag register
ENET_PTP_PPSCTL	PTP PPS control register
ENET_DMA_BCTL	DMA bus control register
ENET_DMA_TPEN	DMA transmit poll enable register
ENET_DMA_RPEN	DMA receive poll enable register
ENET_DMA_RDТАDDR	DMA receive descriptor table address register
ENET_DMA_TDТАDDR	DMA transmit descriptor table address register
ENET_DMA_STAT	DMA status register
ENET_DMA_CTL	DMA control register
ENET_DMA_INTEN	DMA interrupt enable register
ENET_DMA_MFBOCNT	DMA missed frame and buffer overflow counter register
ENET_DMA_RSWDc	DMA receive state watchdog counter register

Registers	Descriptions
ENET_DMA_CTDA DDR	DMA current transmit descriptor address register
ENET_DMA_CRDA DDR	DMA current receive descriptor address register
ENET_DMA_CTBA DDR	DMA current transmit buffer address register
ENET_DMA_CRBA DDR	DMA current receive buffer address register

3.15.2. Descriptions of Peripheral functions

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

Table 3-431. ENET firmware function

Function name	Function description
main function	
enet_deinit	deinitialize the ENET, and reset structure parameters for ENET initialization
enet_initpara_config	configure the parameters which are usually less cared for initialization
enet_init	initialize ENET peripheral with generally concerned parameters and the less cared parameters
enet_software_reset	reset all core internal registers located in CLK_TX and CLK_RX
enet_rxframe_size_get	check receive frame valid and return frame size
enet_descriptors_chain_init	initialize the dma tx/rx descriptors's parameters in chain mode
enet_descriptors_ring_init	initialize the dma tx/rx descriptors's parameters in ring mode
enet_frame_receive	handle current received frame data to application buffer
enet_frame_transmit	handle application buffer data to transmit it
enet_transmit_checksum_config	configure the transmit IP frame checksum offload calculation and insertion
enet_enable	ENET Tx and Rx function enable (include MAC and DMA module)
enet_disable	ENET Tx and Rx function disable (include MAC and DMA module)
enet_mac_address_set	configure MAC address
enet_mac_address_get	get MAC address
MAC function	
enet_tx_enable	ENET Tx function enable (include MAC and DMA module)
enet_tx_disable	ENET Tx function disable (include MAC and DMA module)
enet_rx_enable	ENET Rx function enable (include MAC and DMA module)
enet_rx_disable	ENET Rx function disable (include MAC and DMA module)

Function name	Function description
enet_registers_get	put registers value into the application buffer
enet_debug_status_get	get the enet debug status from the debug register
enet_address_filter_enable	enable the MAC address filter
enet_address_filter_disable	disable the MAC address filter
enet_address_filter_config	configure the MAC address filter
enet_phy_config	PHY interface configuration (configure SMI clock and reset PHY chip)
enet_phy_write_read	write to/read from a PHY register
enet_phyloopback_enable	enable the loopback function of phy chip
enet_phyloopback_disable	disable the loopback function of phy chip
enet_forward_feature_enable	enable ENET forward feature
enet_forward_feature_disable	disable ENET forward feature
enet_fliter_feature_enable	enable ENET fliter feature
enet_fliter_feature_disable	disable ENET fliter feature
flow control function	
enet_pauseframe_generate	generate the pause frame, ENET will send pause frame after enable transmit flow control
enet_pauseframe_detect_config	configure the pause frame detect type
enet_pauseframe_config	configure the pause frame parameters
enet_flowcontrol_threshold_config	configure the threshold of the flow control(deactive and active threshold)
enet_flowcontrol_feature_enable	enable ENET flow control feature
enet_flowcontrol_feature_disable	disable ENET flow control feature
DMA function	
enet_dmaprocess_state_get	get the dma transmit/receive process state
enet_dmaprocess_resume	poll the dma transmission/reception enable
enet_rxprocess_check_recovery	check and recover the Rx process
enet_txfifo_flush	flush the ENET transmit fifo, and wait until the flush operation completes
enet_current_desc_address_get	get the transmit/receive address of current descriptor, or current buffer, or descriptor table
enet_desc_information_get	get the Tx or Rx descriptor information
enet_missed_frame_counter_get	get the number of missed frames during receiving
descriptor function	
enet_desc_flag_get	get the bit flag of ENET dma descriptor
enet_desc_flag_set	set the bit flag of ENET dma tx descriptor
enet_desc_flag_clear	clear the bit flag of ENET dma tx descriptor
enet_rx_desc_immediate_receive_complete_interrupt	when receiving the completed, set RS bit in ENET_DMA_STAT register will immediately set
enet_rx_desc_delay_receive_complete_interrupt	when receiving the completed, set RS bit in ENET_DMA_STAT register will is set after a configurable

Function name	Function description
	delay time
enet_rxframe_drop	drop current receive frame
enet_dma_feature_enable	enable DMA feature
enet_dma_feature_disable	disable DMA feature
enhanced descriptor	
enet_rx_desc_enhanced_status_get	get the bit of extended status flag in ENET DMA descriptor
enet_desc_select_enhanced_mode	configure descriptor to work in enhanced mode
enet_ptp_enhanced_descriptors_chain_init	initialize the dma Tx/Rx descriptors's parameters in enhanced chain mode with ptp function
enet_ptp_enhanced_descriptors_ring_init	initialize the dma Tx/Rx descriptors's parameters in enhanced ring mode with ptp function
enet_ptpframe_receive_enhanced_mode	receive a packet data with timestamp values to application buffer, when the DMA is in enhanced mode
enet_ptpframe_transmit_enhanced_mode	send data with timestamp values in application buffer as a transmit packet, when the DMA is in enhanced mode
normal descriptor	
enet_desc_select_normal_mode	configure descriptor to work in normal mode
enet_ptp_normal_descriptors_chain_init	initialize the dma Tx/Rx descriptors's parameters in normal chain mode with ptp function
enet_ptp_normal_descriptors_ring_init	initialize the dma Tx/Rx descriptors's parameters in normal ring mode with ptp function
enet_ptpframe_receive_normal_mode	receive a packet data with timestamp values to application buffer, when the DMA is in normal mode
enet_ptpframe_transmit_normal_mode	send data with timestamp values in application buffer as a transmit packet, when the DMA is in normal mode
WUM function	
enet_wum_filter_register_pointer_reset	wakeup frame filter register pointer reset
enet_wum_filter_config	set the remote wakeup frame registers
enet_wum_feature_enable	enable wakeup management features
enet_wum_feature_disable	disable wakeup management features
MSC function	
enet_msc_counters_reset	reset the MAC statistics counters
enet_msc_feature_enable	enable the MAC statistics counter features
enet_msc_feature_disable	disable the MAC statistics counter features
enet_msc_counters_preset_config	configure MAC statistics counters preset mode
enet_msc_counters_get	get MAC statistics counter
PTP function	
enet_ptp_subsecond_2_nanosecond	change subsecond to nanosecond
enet_ptp_nanosecond_2_subsecond	change nanosecond to subsecond
enet_ptp_feature_enable	enable the PTP features

Function name	Function description
enet_ptp_feature_disable	disable the PTP features
enet_ptp_timestamp_function_config	configure the PTP timestamp function
enet_ptp_subsecond_increment_config	configure the PTP system time subsecond increment value
enet_ptp_timestamp_addend_config	adjusting the PTP clock frequency only in fine update mode
enet_ptp_timestamp_update_config	initializing or adding/subtracting to second of the PTP system time
enet_ptp_expected_time_config	configure the PTP expected target time
enet_ptp_system_time_get	get the PTP current system time
enet_ptp_pps_output_frequency_config	configure the PPS output frequency
enet_ptp_start	configure and start PTP timestamp counter
enet_ptp_finecorrection_adjfreq	adjust frequency in fine method by configure addend register
enet_ptp_coarsecorrection_systime_update	update system time in coarse method
enet_ptp_finecorrection_settime	set system time in fine method
enet_ptp_flag_get	get the ptp flag status
internal function	
enet_initpara_reset	reset the ENET initpara struct, call it before using enet_initpara_config()
interrupt and flag function	
enet_flag_get	get the ENET MAC/MSR/PTP/DMA status flag
enet_flag_clear	clear the ENET DMA status flag
enet_interrupt_enable	enable ENET MAC/MSR/DMA interrupt
enet_interrupt_disable	disable ENET MAC/MSR/DMA interrupt
enet_interrupt_flag_get	get ENET MAC/MSR/DMA interrupt flag
enet_interrupt_flag_clear	clear ENET DMA interrupt flag

Structure enet_initpara_struct

Table 3-432. Structure enet_initpara_struct

member name	Function description
option_enable	select which function to configure
forward_frame	frame forward related parameters
dmabus_mode	DMA bus mode related parameters
dma_maxburst	DMA max burst related parameters
dma_arbitration	DMA Tx and Rx arbitration related parameters
store_forward_mode	store forward mode related parameters
dma_function	DMA control related parameters
vlan_config	VLAN tag related parameters
flow_control	flow control related parameters
hashtable_high	hash list high 32-bit related parameters

hashtable_low	hash list low 32-bit related parameters
framesfilter_mode	frame filter control related parameters
halfduplex_param	halfduplex related parameters
timer_config	frame timer related parameters
interframegap	inter frame gap related parameters

Structure enet_descriptors_struct

Table 3-433. Structure enet_descriptors_struct

member name	Function description
status	status
control_buffer_size	control and buffer1, buffer2 lengths
buffer1_addr	buffer1 address pointer/timestamp low
buffer2_next_desc_addr	buffer2 or next descriptor address pointer/timestamp high
extended_status	extended status
reserved	reserved
timestamp_low	timestamp low
timestamp_high	timestamp high

Structure enet_ptp_systime_struct

Table 3-434. Structure enet_ptp_systime_struct

member name	Function description
second	second of system time
nanosecond	nanosecond of system time
sign	sign of system time

Enum enet_flag_enum

Table 3-435. Enum enet_flag_enum

member name	Function description
ENET_MAC_FLAG_MPKR	magic packet received flag
ENET_MAC_FLAG_WUFR	wakeup frame received flag
ENET_MAC_FLAG_FLOWCONTROL	flow control status flag
ENET_MAC_FLAG_WUM	WUM status flag
ENET_MAC_FLAG_MSC	MSC status flag
ENET_MAC_FLAG_MSCR	MSC receive status flag

ENET_MAC_FLAG_ MSCT	MSC transmit status flag
ENET_MAC_FLAG_ TMST	timestamp trigger status flag
ENET_PTP_FLAG_ TSSCO	timestamp second counter overflow flag
ENET_PTP_FLAG_ TTM	target time match flag
ENET_MSC_FLAG_ RFCE	received frames CRC error flag
ENET_MSC_FLAG_ RFAE	received frames alignment error flag
ENET_MSC_FLAG_ RGUF	received good unicast frames flag
ENET_MSC_FLAG_ TGFSC	transmitted good frames single collision flag
ENET_MSC_FLAG_ TGFMSC	transmitted good frames more single collision flag
ENET_MSC_FLAG_ TGF	transmitted good frames flag
ENET_DMA_FLAG_ TS	transmit status flag
ENET_DMA_FLAG_ TPS	transmit process stopped status flag
ENET_DMA_FLAG_ TBU	transmit buffer unavailable status flag
ENET_DMA_FLAG_ TJT	transmit jabber timeout status flag
ENET_DMA_FLAG_ RO	receive overflow status flag
ENET_DMA_FLAG_ TU	transmit underflow status flag
ENET_DMA_FLAG_ RS	receive status flag
ENET_DMA_FLAG_ RBU	receive buffer unavailable status flag
ENET_DMA_FLAG_ RPS	receive process stopped status flag
ENET_DMA_FLAG_ RWT	receive watchdog timeout status flag
ENET_DMA_FLAG_ ET	early transmit status flag
ENET_DMA_FLAG_ 	fatal bus error status flag

FBE	
ENET_DMA_FLAG_ER	early receive status flag
ENET_DMA_FLAG_AI	abnormal interrupt summary flag
ENET_DMA_FLAG_NI	normal interrupt summary flag
ENET_DMA_FLAG_EB_DMA_ERROR	error during data transfer by RxDMA/TxDMA flag
ENET_DMA_FLAG_EB_TRANSFER_ERROR	error during write/read transfer flag
ENET_DMA_FLAG_EB_ACCESS_ERROR	error during data buffer/descriptor access flag
ENET_DMA_FLAG_MSC	MSC status flag
ENET_DMA_FLAG_WUM	WUM status flag
ENET_DMA_FLAG_TST	timestamp trigger status flag

Enum enet_flag_clear_enum

Table 3-436. Enum enet_flag_clear_enum

member name	Function description
ENET_DMA_FLAG_TS_CLR	transmit status flag clear
ENET_DMA_FLAG_TPS_CLR	transmit process stopped status flag clear
ENET_DMA_FLAG_TBU_CLR	transmit buffer unavailable status flag clear
ENET_DMA_FLAG_TJT_CLR	transmit jabber timeout status flag clear
ENET_DMA_FLAG_RO_CLR	receive overflow status flag clear
ENET_DMA_FLAG_TU_CLR	transmit underflow status flag clear
ENET_DMA_FLAG_RS_CLR	receive status flag clear
ENET_DMA_FLAG_RBU_CLR	receive buffer unavailable status flag clear
ENET_DMA_FLAG_	receive process stopped status flag clear

RPS_CLR	
ENET_DMA_FLAG_ RWT_CLR	receive watchdog timeout status flag clear
ENET_DMA_FLAG_ ET_CLR	early transmit status flag clear
ENET_DMA_FLAG_ FBE_CLR	fatal bus error status flag clear
ENET_DMA_FLAG_ ER_CLR	early receive status flag clear
ENET_DMA_FLAG_ AI_CLR	abnormal interrupt summary flag clear
ENET_DMA_FLAG_ NI_CLR	normal interrupt summary flag clear

Enum enet_int_enum

Table 3-437. Enum enet_int_enum

member name	Function description
<i>ENET_MAC_INT_WUMIM</i>	WUM interrupt mask
<i>ENET_MAC_INT_TSTMIM</i>	timestamp trigger interrupt mask
<i>ENET_MSC_INT_RFCEIM</i>	received frame CRC error interrupt mask
<i>ENET_MSC_INT_RFAEIM</i>	received frames alignment error interrupt mask
<i>ENET_MSC_INT_RGUFIM</i>	received good unicast frames interrupt mask
<i>ENET_MSC_INT_TGFSCIM</i>	transmitted good frames single collision interrupt mask
<i>ENET_MSC_INT_TGFMSCIM</i>	transmitted good frames more single collision interrupt mask
<i>ENET_MSC_INT_TGFIM</i>	transmitted good frames interrupt mask
<i>ENET_DMA_INT_TIE</i>	transmit interrupt enable
<i>ENET_DMA_INT_TPSIE</i>	transmit process stopped interrupt enable
<i>ENET_DMA_INT_TBUIE</i>	transmit buffer unavailable interrupt enable
<i>ENET_DMA_INT_TJTIE</i>	transmit jabber timeout interrupt enable
<i>ENET_DMA_INT_R</i>	receive overflow interrupt enable

<i>OIE</i>	
<i>ENET_DMA_INT_TUIE</i>	transmit underflow interrupt enable
<i>ENET_DMA_INT_RIE</i>	receive interrupt enable
<i>ENET_DMA_INT_RBUIE</i>	receive buffer unavailable interrupt enable
<i>ENET_DMA_INT_RPSIE</i>	receive process stopped interrupt enable
<i>ENET_DMA_INT_RWTIE</i>	receive watchdog timeout interrupt enable
<i>ENET_DMA_INT_ETIE</i>	early transmit interrupt enable
<i>ENET_DMA_INT_FBEIE</i>	fatal bus error interrupt enable
<i>ENET_DMA_INT_ERIE</i>	early receive interrupt enable
<i>ENET_DMA_INT_AIE</i>	abnormal interrupt summary enable
<i>ENET_DMA_INT_NIE</i>	normal interrupt summary enable

Enum enet_int_flag_enum

Table 3-438. Enum enet_int_flag_enum

member name	Function description
<i>ENET_MAC_INT_FLAG_WUM</i>	WUM status flag
<i>ENET_MAC_INT_FLAG_MSC</i>	MSC status flag
<i>ENET_MAC_INT_FLAG_MSCR</i>	MSC receive status flag
<i>ENET_MAC_INT_FLAG_MSCT</i>	MSC transmit status flag
<i>ENET_MAC_INT_FLAG_TMST</i>	time stamp trigger status flag
<i>ENET_MSC_INT_FLAG_RFCE</i>	received frames CRC error flag
<i>ENET_MSC_INT_FLAG_RFAE</i>	received frames alignment error flag
<i>ENET_MSC_INT_FLAG_RGUF</i>	received good unicast frames flag
<i>ENET_MSC_INT_FLAG</i>	transmitted good frames single collision flag

<i>LAG_TGFSC</i>	
<i>ENET_MSC_INT_F</i> <i>LAG_TGFMSC</i>	transmitted good frames more single collision flag
<i>ENET_MSC_INT_F</i> <i>LAG_TGF</i>	transmitted good frames flag
<i>ENET_DMA_INT_F</i> <i>LAG_TS</i>	transmit status flag
<i>ENET_DMA_INT_F</i> <i>LAG_TPS</i>	transmit process stopped status flag
<i>ENET_DMA_INT_F</i> <i>LAG_TBU</i>	transmit buffer unavailable status flag
<i>ENET_DMA_INT_F</i> <i>LAG_TJT</i>	transmit jabber timeout status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RO</i>	receive overflow status flag
<i>ENET_DMA_INT_F</i> <i>LAG_TU</i>	transmit underflow status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RS</i>	receive status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RBU</i>	receive buffer unavailable status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RPS</i>	receive process stopped status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RWT</i>	receive watchdog timeout status flag
<i>ENET_DMA_INT_F</i> <i>LAG_ET</i>	early transmit status flag
<i>ENET_DMA_INT_F</i> <i>LAG_FBE</i>	fatal bus error status flag
<i>ENET_DMA_INT_F</i> <i>LAG_ER</i>	early receive status flag
<i>ENET_DMA_INT_F</i> <i>LAG_AI</i>	abnormal interrupt summary flag
<i>ENET_DMA_INT_F</i> <i>LAG_NI</i>	normal interrupt summary flag
<i>ENET_DMA_INT_F</i> <i>LAG_MSC</i>	MSC status flag
<i>ENET_DMA_INT_F</i> <i>LAG_WUM</i>	WUM status flag
<i>ENET_DMA_INT_F</i> <i>LAG_TST</i>	timestamp trigger status flag

Enum enet_int_flag_clear_enum

Table 3-439. Enum enet_int_flag_clear_enum

member name	Function description
<i>ENET_DMA_INT_F</i> <i>LAG_TS_CLR</i>	transmit status flag
<i>ENET_DMA_INT_F</i> <i>LAG_TPS_CLR</i>	transmit process stopped status flag
<i>ENET_DMA_INT_F</i> <i>LAG_TBU_CLR</i>	transmit buffer unavailable status flag
<i>ENET_DMA_INT_F</i> <i>LAG_TJT_CLR</i>	transmit jabber timeout status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RO_CLR</i>	receive overflow status flag
<i>ENET_DMA_INT_F</i> <i>LAG_TU_CLR</i>	transmit underflow status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RS_CLR</i>	receive status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RBU_CLR</i>	receive buffer unavailable status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RPS_CLR</i>	receive process stopped status flag
<i>ENET_DMA_INT_F</i> <i>LAG_RWT_CLR</i>	receive watchdog timeout status flag
<i>ENET_DMA_INT_F</i> <i>LAG_ET_CLR</i>	early transmit status flag
<i>ENET_DMA_INT_F</i> <i>LAG_FBE_CLR</i>	fatal bus error status flag
<i>ENET_DMA_INT_F</i> <i>LAG_ER_CLR</i>	early receive status flag
<i>ENET_DMA_INT_F</i> <i>LAG_AI_CLR</i>	abnormal interrupt summary flag
<i>ENET_DMA_INT_F</i> <i>LAG_NI_CLR</i>	normal interrupt summary flag

Enum enet_desc_reg_enum

Table 3-440. Enum enet_desc_reg_enum

member name	Function description
<i>ENET_RX_DESC_T</i> <i>ABLE</i>	the start address of the receive descriptor table
<i>ENET_RX_CURRE</i> <i>NT_DESC</i>	the start descriptor address of the current receive descriptor read by the RxDMA controller

<i>ENET_RX_CURRENT_BUFFER</i>	the current receive buffer address being read by the RxDMA controller
<i>ENET_TX_DESCRIPTOR_TABLE</i>	the start address of the transmit descriptor table
<i>ENET_TX_CURRENT_DESCRIPTOR</i>	the start descriptor address of the current transmit descriptor read by the TxDMA controller
<i>ENET_TX_CURRENT_BUFFER</i>	the current transmit buffer address being read by the TxDMA controller

Enum enet_msc_counter_enum

Table 3-441. Enum enet_msc_counter_enum

member name	Function description
<i>ENET_MSC_TX_SINGLE_COLL_CNT</i>	MSC transmitted good frames after a single collision counter
<i>ENET_MSC_TX_MULT_COLL_CNT</i>	MSC transmitted good frames after more than a single collision counter
<i>ENET_MSC_TX_GOOD_FRAMES_CNT</i>	MSC transmitted good frames counter
<i>ENET_MSC_RX_CRC_ERR_CNT</i>	MSC received frames with CRC error counter
<i>ENET_MSC_RX_ALIGN_ERR_CNT</i>	MSC received frames with alignment error counter
<i>ENET_MSC_RX_GOOD_UNICAST_FRAMES_CNT</i>	MSC received good unicast frames counter

Enum enet_option_enum

Table 3-442. Enum enet_option_enum

member name	Function description
<i>FORWARD_OPTION</i>	choose to configure the frame forward related parameters
<i>DMABUS_OPTION</i>	choose to configure the DMA bus mode related parameters
<i>DMA_MAXBURST_OPTION</i>	choose to configure the DMA max burst related parameters
<i>DMA_ARBITRATION_OPTION</i>	choose to configure the DMA arbitration related parameters
<i>STORE_OPTION</i>	choose to configure the store forward mode related parameters
<i>DMA_OPTION</i>	choose to configure the DMA related parameters
<i>VLAN_OPTION</i>	choose to configure vlan related parameters
<i>FLOWCTL_OPTION</i>	choose to configure flow control related parameters
<i>HASHH_OPTION</i>	choose to configure hash high
<i>HASHL_OPTION</i>	choose to configure hash low

<i>FILTER_OPTION</i>	choose to configure frame filter related parameters
<i>HALFDUPLEX_OPTION</i>	choose to configure halfduplex mode related parameters
<i>TIMER_OPTION</i>	choose to configure time counter related parameters
<i>INTERFRAMEGAP_OPTION</i>	choose to configure the inter frame gap related parameters

Enum enet_mediamode_enum

Table 3-443. Enum enet_mediamode_enum

member name	Function description
<i>ENET_AUTO_NEGOTIATION</i>	PHY auto negotiation
<i>ENET_100M_FULLDUPLEX</i>	100Mbit/s, full-duplex
<i>ENET_100M_HALFDUPLEX</i>	100Mbit/s, half-duplex
<i>ENET_10M_FULLDUPLEX</i>	10Mbit/s, full-duplex
<i>ENET_10M_HALFDUPLEX</i>	10Mbit/s, half-duplex
<i>ENET_LOOPBACK_MODE</i>	MAC in loopback mode at the MII

Enum enet_chksumconf_enum

Table 3-444. Enum enet_chksumconf_enum

member name	Function description
<i>ENET_NO_AUTOCHECKSUM</i>	disable IP frame checksum function
<i>ENET_AUTOCHECKSUM_DROP_FAILFRAMES</i>	enable IP frame checksum function
<i>ENET_AUTOCHECKSUM_ACCEPT_FAILFRAMES</i>	enable IP frame checksum function, and the received frame with only payload error but no other errors will not be dropped

Enum enet_frmrecept_enum

Table 3-445. Enum enet_frmrecept_enum

member name	Function description
<i>ENET_PROMISCUOUS_MODE</i>	promiscuous mode enabled
<i>ENET_RECEIVEALL</i>	all received frame are forwarded to application

<i>L</i>	
<i>ENET_BROADCAST_FILTER_PASS</i>	the address filters pass all received broadcast frames
<i>ENET_BROADCAST_FILTER_DROP</i>	the address filters filter all incoming broadcast frames

Enum `enet_registers_type_enum`

Table 3-446. Enum `enet_registers_type_enum`

member name	Function description
<i>ALL_MAC_REG</i>	get the registers within the offset scope range <code>ENET_MAC_CFG</code> to <code>ENET_MAC_FCTH</code>
<i>ALL_MSC_REG</i>	get the registers within the offset scope range <code>ENET_MSC_CTL</code> to <code>ENET_MSC_RGUFCNT</code>
<i>ALL_PTP_REG</i>	get the registers within the offset scope range <code>ENET_PTP_TSCTL</code> to <code>ENET_PTP_PPSCTL</code>
<i>ALL_DMA_REG</i>	get the registers within the offset scope range <code>ENET_DMA_BCTL</code> to <code>ENET_DMA_CRBADDR</code>

Enum `enet_dmadirection_enum`

Table 3-447. Enum `enet_dmadirection_enum`

member name	Function description
<i>ENET_DMA_TX</i>	DMA Tx descriptors
<i>ENET_DMA_RX</i>	DMA Rx descriptors

Enum `enet_phydirection_enum`

Table 3-448. Enum `enet_phydirection_enum`

member name	Function description
<i>ENET_PHY_READ</i>	read data from phy register
<i>ENET_PHY_WRITE</i>	write data to phy register

Enum `enet_regdirection_enum`

Table 3-449. Enum `enet_regdirection_enum`

member name	Function description
<i>ENET_REG_READ</i>	read register
<i>ENET_REG_WRITE</i>	write register

Enum `enet_macaddress_enum`

Table 3-450. Enum `enet_macaddress_enum`

member name	Function description
<i>ENET_MAC_ADDR</i>	set MAC address 0 filter

<i>ESS0</i>	
<i>ENET_MAC_ADDR</i> <i>ESS1</i>	set MAC address 1 filter
<i>ENET_MAC_ADDR</i> <i>ESS2</i>	set MAC address 2 filter
<i>ENET_MAC_ADDR</i> <i>ESS3</i>	set MAC address 3 filter

Enum `enet_descstate_enum`

Table 3-451. Enum `enet_descstate_enum`

member name	Function description
<i>TXDESC_COLLISION_COUNT</i>	the number of collisions occurred before the frame was transmitted
<i>TXDESC_BUFFER_1_ADDR</i>	the buffer1 address of the Tx frame
<i>RXDESC_FRAME_LENGTH</i>	the byte length of the received frame that was transferred to the buffer
<i>RXDESC_BUFFER_1_SIZE</i>	receive buffer 1 size
<i>RXDESC_BUFFER_2_SIZE</i>	receive buffer 2 size
<i>RXDESC_BUFFER_1_ADDR</i>	the buffer1 address of the Rx frame

Enum `enet_msc_preset_enum`

Table 3-452. Enum `enet_msc_preset_enum`

member name	Function description
<i>ENET_MSC_PRESET_NONE</i>	do not preset MSC counter
<i>ENET_MSC_PRESET_HALF</i>	preset all MSC counters to almost-half(0x7FFF FFF0) value
<i>ENET_MSC_PRESET_FULL</i>	preset all MSC counters to almost-full(0xFFFF FFF0) value

`enet_deinit`

The description of `enet_deinit` is shown as below:

Table 3-453. Function `enet_deinit`

Function name	<code>enet_deinit</code>
Function prototype	<code>void enet_deinit(uint32_t enet_periph);</code>
Function descriptions	deinitialize the ENET, and reset structure parameters for ENET initialization

Precondition	-
The called functions	rcu_periph_reset_enable() /rcu_periph_reset_disable()/enet_initpara_reset()
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the ENET */
```

```
enet_deinit(ENET0);
```

enet_initpara_config

The description of enet_initpara_config is shown as below:

Table 3-454. Function enet_initpara_config

Function name	enet_initpara_config
Function prototype	void enet_initpara_config(enet_option_enum option, uint32_t para);
Function descriptions	configure the parameters which are usually less cared for initialization, this function must be called before enet_init(), otherwise configuration will be no effect
Precondition	enet_initpara_reset(void)
The called functions	-
Input parameter{in}	
option	different function option, which is related to several parameters, refer to Table 3-442. Enum enet_option_enum only one parameter can be selected which is shown as below
<i>FORWARD_OPTION</i>	choose to configure the frame forward related parameters
<i>DMABUS_OPTION</i>	choose to configure the DMA bus mode related parameters
<i>DMA_MAXBURST_OPTION</i>	choose to configure the DMA max burst related parameters
<i>DMA_ARBITRATION_OPTION</i>	choose to configure the DMA arbitration related parameters
<i>STORE_OPTION</i>	choose to configure the store forward mode related parameters
<i>DMA_OPTION</i>	choose to configure the DMA related parameters
<i>VLAN_OPTION</i>	choose to configure vlan related parameters
<i>FLOWCTL_OPTION</i>	choose to configure flow control related parameters
<i>HASHH_OPTION</i>	choose to configure hash high
<i>HASHL_OPTION</i>	choose to configure hash low

<i>FILTER_OPTION</i>	choose to configure frame filter related parameters
<i>HALFDUPLEX_OPTION</i>	choose to configure halfduplex mode related parameters
<i>TIMER_OPTION</i>	choose to configure time counter related parameters
<i>INTERFRAMEGAP_OPTION</i>	choose to configure the inter frame gap related parameters
Input parameter{in}	
para (the value according to the parameter option)	all the related values should be configured which are shown as below example: para = (value1 value2 value3...)
When value of parameter option is <i>FORWARD_OPTION</i>	
value1	<i>ENET_AUTO_PADCRC_DROP_ENABLE</i> / <i>ENET_AUTO_PADCRC_DROP_DISABLE</i>
value2	<i>ENET_FORWARD_ERRFRAMES_ENABLE</i> / <i>ENET_FORWARD_ERRFRAMES_DISABLE</i>
value3	<i>ENET_FORWARD_UNDEFSZ_GOODFRAMES_ENABLE</i> / <i>ENET_FORWARD_UNDEFSZ_GOODFRAMES_DISABLE</i>
value4	<i>ENET_FORWARD_UNDEFSZ_GOODFRAMES_ENABLE</i> / <i>ENET_FORWARD_UNDEFSZ_GOODFRAMES_DISABLE</i>
When value of parameter option is <i>DMABUS_OPTION</i>	
value1	<i>ENET_ADDRESS_ALIGN_ENABLE</i> / <i>ENET_ADDRESS_ALIGN_DISABLE</i>
value2	<i>ENET_FIXED_BURST_ENABLE</i> / <i>ENET_FIXED_BURST_DISABLE</i>
value3	<i>ENET_MIXED_BURST_ENABLE</i> / <i>ENET_MIXED_BURST_DISABLE</i>
When value of parameter option is <i>DMA_MAXBURST_OPTION</i>	
value1	<i>ENET_RXDP_1BEAT</i> / <i>ENET_RXDP_2BEAT</i> / <i>ENET_RXDP_4BEAT</i> / <i>ENET_RXDP_8BEAT</i> / <i>ENET_RXDP_16BEAT</i> / <i>ENET_RXDP_32BEAT</i> / <i>ENET_RXDP_4xPGBL_4BEAT</i> / <i>ENET_RXDP_4xPGBL_8BEAT</i> / <i>ENET_RXDP_4xPGBL_16BEAT</i> / <i>ENET_RXDP_4xPGBL_32BEAT</i> / <i>ENET_RXDP_4xPGBL_64BEAT</i> / <i>ENET_RXDP_4xPGBL_128BEAT</i>
value2	<i>ENET_PGBL_1BEAT</i> / <i>ENET_PGBL_2BEAT</i> / <i>ENET_PGBL_4BEAT</i> / <i>ENET_PGBL_8BEAT</i> / <i>ENET_PGBL_16BEAT</i> / <i>ENET_PGBL_32BEAT</i> / <i>ENET_PGBL_4xPGBL_4BEAT</i> / <i>ENET_PGBL_4xPGBL_8BEAT</i> / <i>ENET_PGBL_4xPGBL_16BEAT</i> / <i>ENET_PGBL_4xPGBL_32BEAT</i> / <i>ENET_PGBL_4xPGBL_64BEAT</i> / <i>ENET_PGBL_4xPGBL_128BEAT</i>
value3	<i>ENET_RXTX_DIFFERENT_PGBL</i> / <i>ENET_RXTX_SAME_PGBL</i>
When value of parameter option is <i>DMA_ARBITRATION_OPTION</i>	
value1	<i>ENET_ARBITRATION_RXPRIORTX</i>
value2	<i>ENET_ARBITRATION_RXTX_1_1</i> / <i>ENET_ARBITRATION_RXTX_2_1</i> / <i>ENET_ARBITRATION_RXTX_3_1</i> / <i>ENET_ARBITRATION_RXTX_4_1</i>
When value of parameter option is <i>STORE_OPTION</i>	
value1	<i>ENET_RX_MODE_STOREFORWARD</i> / <i>ENET_RX_MODE_CUTTHROUGH</i>
value2	<i>ENET_TX_MODE_STOREFORWARD</i> / <i>ENET_TX_MODE_CUTTHROUGH</i>

value3	ENET_RX_THRESHOLD_64BYTES / ENET_RX_THRESHOLD_32BYTES / ENET_RX_THRESHOLD_96BYTES / ENET_RX_THRESHOLD_128BYTES
value4	ENET_TX_THRESHOLD_64BYTES / ENET_TX_THRESHOLD_128BYTES / ENET_TX_THRESHOLD_192BYTES / ENET_TX_THRESHOLD_256BYTES / ENET_TX_THRESHOLD_40BYTES / ENET_TX_THRESHOLD_32BYTES / ENET_TX_THRESHOLD_24BYTES / ENET_TX_THRESHOLD_16BYTES
When value of parameter option is DMA_OPTION	
value1	ENET_FLUSH_RXFRAME_ENABLE / ENET_FLUSH_RXFRAME_DISABLE
value2	ENET_SECONDFRAME_OPT_ENABLE / ENET_SECONDFRAME_OPT_DISABLE
value3	ENET_ENHANCED_DESCRIPTOR/ ENET_NORMAL_DESCRIPTOR
When value of parameter option is VLAN_OPTION	
value1	ENET_VLANTAGCOMPARISON_12BIT/ ENET_VLANTAGCOMPARISON_16BIT
value2	MAC_VLT_VLTI(regval)
When value of parameter option is FLOWCTL_OPTION	
value1	MAC_FCTL_PTM(regval)
value2	ENET_ZERO_QUANTA_PAUSE_ENABLE / ENET_ZERO_QUANTA_PAUSE_DISABLE
value3	ENET_PAUSETIME_MINUS4 / ENET_PAUSETIME_MINUS28 / ENET_PAUSETIME_MINUS144/ENET_PAUSETIME_MINUS256
value4	ENET_MAC0_AND_UNIQUE_ADDRESS_PAUSEDetect / ENET_UNIQUE_PAUSEDetect
value5	ENET_RX_FLOWCONTROL_ENABLE / ENET_RX_FLOWCONTROL_DISABLE
value6	ENET_TX_FLOWCONTROL_ENABLE / ENET_TX_FLOWCONTROL_DISABLE
When value of parameter option is HASHH_OPTION	
value1	0x0~0xFFFF FFFFU
When value of parameter option is HASHL_OPTION	
value1	0x0~0xFFFF FFFFU
When value of parameter option is FILTER_OPTION	
value1	ENET_SRC_FILTER_NORMAL_ENABLE / ENET_SRC_FILTER_INVERSE_ENABLE / ENET_SRC_FILTER_DISABLE
value2	ENET_DEST_FILTER_INVERSE_ENABLE / ENET_DEST_FILTER_INVERSE_DISABLE
value3	ENET_MULTICAST_FILTER_HASH_OR_PERFECT / ENET_MULTICAST_FILTER_HASH / ENET_MULTICAST_FILTER_PERFECT / ENET_MULTICAST_FILTER_NONE
value4	ENET_UNICAST_FILTER_EITHER / ENET_UNICAST_FILTER_HASH /

	<i>ENET_UNICAST_FILTER_PERFECT</i>
value5	<i>ENET_PCFRM_PREVENT_ALL /</i> <i>ENET_PCFRM_PREVENT_PAUSEFRAME /</i> <i>ENET_PCFRM_FORWARD_ALL / ENET_PCFRM_FORWARD_FILTERED</i>
When value of parameter option is <i>HALFDUPLEX_OPTION</i>	
value1	<i>ENET_CARRIERSENSE_ENABLE / ENET_CARRIERSENSE_DISABLE</i>
value2	<i>ENET_RECEIVEOWN_ENABLE / ENET_RECEIVEOWN_DISABLE</i>
value3	<i>ENET_RETRYTRANSMISSION_ENABLE /</i> <i>ENET_RETRYTRANSMISSION_DISABLE</i>
value4	<i>ENET_BACKOFFLIMIT_10 / ENET_BACKOFFLIMIT_8 /</i> <i>ENET_BACKOFFLIMIT_4 / ENET_BACKOFFLIMIT_1</i>
value5	<i>ENET_DEFERRALCHECK_ENABLE /</i> <i>ENET_DEFERRALCHECK_DISABLE</i>
When value of parameter option is <i>TIMER_OPTION</i>	
value1	<i>ENET_WATCHDOG_ENABLE / ENET_WATCHDOG_DISABLE</i>
value2	<i>ENET_JABBER_ENABLE / ENET_JABBER_DISABLE</i>
When value of parameter option is <i>INTERFRAMEGAP_OPTION</i>	
value1	<i>ENET_INTERFRAMEGAP_96BIT / ENET_INTERFRAMEGAP_88BIT /</i> <i>ENET_INTERFRAMEGAP_80BIT / ENET_INTERFRAMEGAP_72BIT /</i> <i>ENET_INTERFRAMEGAP_64BIT / ENET_INTERFRAMEGAP_56BIT /</i> <i>ENET_INTERFRAMEGAP_48BIT / ENET_INTERFRAMEGAP_40BIT</i>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config DMA option of the ENET */
```

```
enet_initpara_reset();
```

```
enet_initpara_config(DMA_OPTION, ENET_FLUSH_RXFRAME_ENABLE | ENET_SECONDFRAME_OPT_ENABLE | ENET_NORMAL_DESCRIPTOR);
```

enet_init

The description of enet_init is shown as below:

Table 3-455. Function enet_init

Function name	enet_init
Function prototype	ErrStatus enet_init(uint32_t enet_periph, enet_mediamode_enum mediamode, enet_chksumconf_enum checksum, enet_frmrecept_enum recept);
Function descriptions	initialize ENET peripheral with generally concerned parameters and the less cared parameters

Precondition	enet_deinit ()
The called functions	enet_phy_config /enet_phy_write_read
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
mediamode	PHY mode and mac loopback configurations, refer to Table 3-443. Enum enet_mediamode_enum , only one parameter can be selected
ENET_AUTO_NEGOTIATION	PHY auto negotiation
ENET_100M_FULLDUPLEX	100Mbit/s, full-duplex
ENET_100M_HALFDUPLEX	100Mbit/s, half-duplex
ENET_10M_FULLDUPLEX	10Mbit/s, full-duplex
ENET_10M_HALFDUPLEX	10Mbit/s, half-duplex
ENET_LOOPBACKMODE	MAC in loopback mode at the MII
Input parameter{in}	
checksum	IP frame checksum offload function, refer to Table 3-444. Enum enet_chksumconf_enum , only one parameter can be selected
ENET_NO_AUTOCHECKSUM	disable IP frame checksum function
ENET_AUTOCHECKSUM_DROP_FAILFRAMES	enable IP frame checksum function
ENET_AUTOCHECKSUM_ACCEPT_FAILFRAMES	enable IP frame checksum function, and the received frame with only payload error but no other errors will not be dropped
Input parameter{in}	
recept	frame filter function, refer to Table 3-445. Enum enet_frmrecept_enum , only one parameter can be selected
ENET_PROMISCUOUS_MODE	promiscuous mode enabled
ENET_RECEIVEALL	all received frame are forwarded to application
ENET_BROADCAST_FILTERS_PASS	the address filters pass all received broadcast frames
ENET_BROADCAST_FILTERS_DROP	the address filters filter all incoming broadcast frames
Output parameter{out}	

-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* initialize ENET peripheral */
```

```
ErrStatus enet_init_status;
```

```
enet_init_status = enet_init(ENET0, ENET_AUTO_NEGOTIATION, ENET_AUTOCHECK  
SUM_DROP_FAILFRAMES, ENET_BROADCAST_FRAMES_PASS);
```

enet_software_reset

The description of enet_software_reset is shown as below:

Table 3-456. Function enet_software_reset

Function name	enet_software_reset
Function prototype	ErrStatus enet_software_reset(uint32_t enet_periph);
Function descriptions	reset all core internal registers located in CLK_TX and CLK_RX
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* reset all core internal registers located in CLK_TX and CLK_RX */
```

```
ErrStatus reval_state = ERROR;
```

```
reval_state = enet_software_reset(ENET0);
```

enet_rxframe_size_get

The description of enet_rxframe_size_get is shown as below:

Table 3-457. Function enet_rxframe_size_get

Function name	enet_rxframe_size_get
Function prototype	uint32_t enet_rxframe_size_get(uint32_t enet_periph);

Function descriptions	check receive frame valid and return frame size
Precondition	-
The called functions	enet_rxframe_drop()
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
uint32_t	size of received frame 0x0 - 0x3FFF

Example:

```
/* check receive frame valid */
uint32_t reval;
reval = enet_rxframe_size_get(ENET0);
```

enet_descriptors_chain_init

The description of enet_descriptors_chain_init is shown as below:

Table 3-458. Function enet_descriptors_chain_init

Function name	enet_descriptors_chain_init
Function prototype	void enet_descriptors_chain_init(uint32_t enet_periph, enet_dmadirection_enum direction);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in chain mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
direction	the descriptors which users want to init, refer to Table 3-447. Enum enet_dmadirection_enum only one parameter can be selected which is shown as below
<i>ENET_DMA_TX</i>	DMA Tx descriptors

<i>ENET_DMA_RX</i>	DMA Rx descriptors
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the DMA Tx/Rx descriptors's parameters in chain mode */
enet_descriptors_chain_init(ENET0, ENET_DMA_TX);
```

enet_descriptors_ring_init

The description of enet_descriptors_ring_init is shown as below:

Table 3-459. Function enet_descriptors_ring_init

Function name	enet_descriptors_ring_init
Function prototype	void enet_descriptors_ring_init(uint32_t enet_periph, enet_dmadirection_enum direction);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in ring mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
direction	the descriptors which users want to init, refer to Table 3-447. Enum enet_dmadirection_enum only one parameter can be selected which is shown as below
<i>ENET_DMA_TX</i>	DMA Tx descriptors
<i>ENET_DMA_RX</i>	DMA Rx descriptors
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the DMA Tx/Rx descriptors's parameters in ring mode */
enet_descriptors_ring_init(ENET0, ENET_DMA_TX);
```

enet_frame_receive

The description of enet_frame_receive is shown as below:

Table 3-460. Function `enet_frame_receive`

Function name	<code>enet_frame_receive</code>
Function prototype	<code>ErrStatus enet_frame_receive(uint32_t enet_periph, uint8_t buffer[], uint32_t bufsize);</code>
Function descriptions	handle current received frame data to application buffer
Precondition	-
The called functions	-
Input parameter{in}	
<code>enet_periph</code>	ENET peripheral
<code>ENET0</code>	ENET peripheral 0
<code>ENET1</code>	ENET peripheral 1
Input parameter{in}	
<code>bufsize</code>	the size of buffer which is the parameter in function, (0 -- 1524)
Output parameter{out}	
<code>buffer</code>	pointer to the received frame data if the input is NULL, user should copy data in application by himself
Return value	
<code>ErrStatus</code>	ERROR or SUCCESS

Example:

```
/* transfer received frame data to application buffer */

uint8_t data_buffer[1500];

uint32_t data_size;

enet_frame_receive(ENET0, data_buffer, &data_size);
```

`enet_frame_transmit`

The description of `enet_frame_transmit` is shown as below:

Table 3-461. Function `enet_frame_transmit`

Function name	<code>enet_frame_transmit</code>
Function prototype	<code>ErrStatus enet_frame_transmit(uint32_t enet_periph, uint8_t buffer[], uint32_t length);</code>
Function descriptions	handle application buffer data to transmit it
Precondition	-
The called functions	-
Input parameter{in}	
<code>enet_periph</code>	ENET peripheral
<code>ENET0</code>	ENET peripheral 0
<code>ENET1</code>	ENET peripheral 1
Input parameter{in}	
<code>buffer</code>	pointer to the frame data to be transmitted

	if the input is NULL, user should handle the data in application by himself
Input parameter{in}	
length	the length of frame data to be transmitted, (0 -- 1524)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```

/* transfer buffer data of application */

uint8_t data_buffer[1500];

uint32_t data_size = 800;

enet_frame_transmit (ENET0, data_buffer, data_size);

```

enet_transmit_checksum_config

The description of enet_transmit_checksum_config is shown as below:

Table 3-462. Function enet_transmit_checksum_config

Function name	enet_transmit_checksum_config
Function prototype	ErrStatus enet_transmit_checksum_config(enet_descriptors_struct *desc, uint32_t checksum);
Function descriptions	configure the transmit IP frame checksum offload calculation and insertion
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to configure, the structure members can refer to Table 3-433. Structure enet_descriptors_struct
Input parameter{in}	
checksum	IP frame checksum configuration only one parameter can be selected which is shown as below
ENET_CHECKSUM_DISABLE	checksum insertion disabled
ENET_CHECKSUM_IPV4HEADER	only IP header checksum calculation and insertion are enabled
ENET_CHECKSUM_TCPUDPICMP_SEGMENT	TCP/UDP/ICMP checksum insertion calculated but pseudo-header
ENET_CHECKSUM_TCPUDPICMP_FULL	TCP/UDP/ICMP checksum insertion fully calculated
Output parameter{out}	

Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* configure the transmit IP frame checksum offload calculation and insertion */
enet_descriptors_struct rx_desc;
enet_transmit_checksum_config(rx_desc, ENET_CHECKSUM_TCPUDPICMP_FULL);
```

enet_enable

The description of enet_enable is shown as below:

Table 3-463. Function enet_enable

Function name	enet_enable
Function prototype	void enet_enable(uint32_t enet_periph);
Function descriptions	ENET Tx and Rx function enable (include MAC and DMA module)
Precondition	-
The called functions	enet_tx_enable() /enet_rx_enable()
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the ENET */
enet_enable(ENET0);
```

enet_disable

The description of enet_disable is shown as below:

Table 3-464. Function enet_disable

Function name	enet_disable
Function prototype	void enet_disable(uint32_t enet_periph);
Function descriptions	ENET Tx and Rx function disable (include MAC and DMA module)
Precondition	-
The called functions	enet_tx_disable() /enet_rx_disable()
Input parameter{in}	
enet_periph	ENET peripheral

<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the ENET */
```

```
enet_disable(ENET0);
```

enet_mac_address_set

The description of enet_mac_address_set is shown as below:

Table 3-465. Function enet_mac_address_set

Function name	enet_mac_address_set
Function prototype	void enet_mac_address_set(uint32_t enet_periph, enet_macaddress_enum mac_addr, uint8_t paddr[]);
Function descriptions	configure MAC address
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
mac_addr	select which MAC address will be set, refer to Table 3-450. Enum enet_macaddress_enum only one parameter can be selected which is shown as below
<i>ENET_MAC_ADDRES S0</i>	set MAC address 0 filter
<i>ENET_MAC_ADDRES S1</i>	set MAC address 1 filter
<i>ENET_MAC_ADDRES S2</i>	set MAC address 2 filter
<i>ENET_MAC_ADDRES S3</i>	set MAC address 3 filter
Input parameter{in}	
paddr	the buffer pointer which stores the MAC address little-ending store, such as MAC address is aa:bb:cc:dd:ee:22, the buffer is {22, ee, dd, cc, bb, aa}
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* config mac address */

netif->hwaddr[0] = 0x02;

netif->hwaddr[1] = 0xaa;

netif->hwaddr[2] = 0xbb;

netif->hwaddr[3] = 0xcc;

netif->hwaddr[4] = 0xdd;

netif->hwaddr[5] = 0xee;

enet_mac_address_set(ENET0, ENET_MAC_ADDRESS0, netif->hwaddr);
```

enet_mac_address_get

The description of enet_mac_address_get is shown as below:

Table 3-466. Function enet_mac_address_get

Function name	enet_mac_address_get
Function prototype	ErrStatus enet_mac_address_get(uint32_t enet_periph, enet_macaddress_enum mac_addr, uint8_t paddr[], uint8_t bufsize);
Function descriptions	get MAC address
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
mac_addr	select which MAC address will be set, refer to Table 3-450. Enum enet_macaddress_enum only one parameter can be selected which is shown as below
ENET_MAC_ADDRES S0	set MAC address 0 filter
ENET_MAC_ADDRES S1	set MAC address 1 filter
ENET_MAC_ADDRES S2	set MAC address 2 filter
ENET_MAC_ADDRES	set MAC address 3 filter

S3	
Output parameter{out}	
paddr	the buffer pointer which stores the MAC address little-ending store, such as MAC address is aa:bb:cc:dd:ee:22, the buffer is {22, ee, dd, cc, bb, aa}
Input parameter{in}	
bufsize	refer to the size of the buffer which stores the MAC address
Return value	
-	-

Example:

```
/* get mac address */
```

```
enet_mac_address_get (ENET0, ENET_MAC_ADDRESS0, netif->hwaddr, 0x100);
```

enet_flag_get

The description of enet_flag_get is shown as below:

Table 3-467. Function enet_flag_get

Function name	enet_flag_get
Function prototype	FlagStatus enet_flag_get(uint32_t enet_periph, enet_flag_enum enet_flag);
Function descriptions	get the ENET MAC/MSR/PTP/DMA status flag
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
enet_flag	ENET status flag, refer to Table 3-435. Enum enet_flag_enum only one parameter can be selected which is shown as below
<i>ENET_MAC_FLAG_MPKR</i>	magic packet received flag
<i>ENET_MAC_FLAG_WUFR</i>	wakeup frame received flag
<i>ENET_MAC_FLAG_FLOWCONTROL</i>	flow control status flag
<i>ENET_MAC_FLAG_WUM</i>	WUM status flag
<i>ENET_MAC_FLAG_MSC</i>	MSC status flag
<i>ENET_MAC_FLAG_MSCR</i>	MSC receive status flag

<i>ENET_MAC_FLAG_MSC CT</i>	MSC transmit status flag
<i>ENET_MAC_FLAG_TM ST</i>	time stamp trigger status flag
<i>ENET_PTP_FLAG_TS SCO</i>	timestamp second counter overflow flag
<i>ENET_PTP_FLAG_TT M</i>	target time match flag
<i>ENET_MSC_FLAG_RF CE</i>	received frames CRC error flag
<i>ENET_MSC_FLAG_RF AE</i>	received frames alignment error flag
<i>ENET_MSC_FLAG_RG UF</i>	received good unicast frames flag
<i>ENET_MSC_FLAG_TG FSC</i>	transmitted good frames single collision flag
<i>ENET_MSC_FLAG_TG FMSC</i>	transmitted good frames more single collision flag
<i>ENET_MSC_FLAG_TG F</i>	transmitted good frames flag
<i>ENET_DMA_FLAG_TS</i>	transmit status flag
<i>ENET_DMA_FLAG_TP S</i>	transmit process stopped status flag
<i>ENET_DMA_FLAG_TB U</i>	transmit buffer unavailable status flag
<i>ENET_DMA_FLAG_TJ T</i>	transmit jabber timeout status flag
<i>ENET_DMA_FLAG_RO</i>	receive overflow status flag
<i>ENET_DMA_FLAG_TU</i>	transmit underflow status flag
<i>ENET_DMA_FLAG_RS</i>	receive status flag
<i>ENET_DMA_FLAG_RB U</i>	receive buffer unavailable status flag
<i>ENET_DMA_FLAG_RP S</i>	receive process stopped status flag
<i>ENET_DMA_FLAG_R WT</i>	receive watchdog timeout status flag
<i>ENET_DMA_FLAG_ET</i>	early transmit status flag
<i>ENET_DMA_FLAG_FB E</i>	fatal bus error status flag
<i>ENET_DMA_FLAG_ER</i>	early receive status flag
<i>ENET_DMA_FLAG_AI</i>	abnormal interrupt summary flag
<i>ENET_DMA_FLAG_NI</i>	normal interrupt summary flag
<i>ENET_DMA_FLAG_EB</i>	DMA error flag

<code>_DMA_ERROR</code>	
<code>ENET_DMA_FLAG_EB</code> <code>_TRANSFER_ERROR</code>	transfer error flag
<code>ENET_DMA_FLAG_EB</code> <code>_ACCESS_ERROR</code>	access error flag
<code>ENET_DMA_FLAG_MS</code> <code>C</code>	MSC status flag
<code>ENET_DMA_FLAG_W</code> <code>UM</code>	WUM status flag
<code>ENET_DMA_FLAG_TS</code> <code>T</code>	timestamp trigger status flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check whether the specified flag bit is set */
```

```
enet_flag_get (ENET0, ENET_DMA_FLAG_RS);
```

enet_flag_clear

The description of enet_flag_clear is shown as below:

Table 3-468. Function enet_flag_clear

Function name	enet_flag_clear
Function prototype	void enet_flag_clear(enet_flag_clear_enum enet_flag);
Function descriptions	clear the ENET DMA status flag
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<code>ENET0</code>	ENET peripheral 0
<code>ENET1</code>	ENET peripheral 1
Input parameter{in}	
enet_flag	ENET DMA flag clear, refer to Table 3-436. Enum enet_flag_clear_enum only one parameter can be selected which is shown as below
<code>ENET_DMA_FLAG_TS</code> <code>_CLR</code>	transmit status flag clear
<code>ENET_DMA_FLAG_TP</code> <code>S_CLR</code>	transmit process stopped status flag clear
<code>ENET_DMA_FLAG_TB</code> <code>U_CLR</code>	transmit buffer unavailable status flag clear

<i>ENET_DMA_FLAG_TJ_T_CLR</i>	transmit jabber timeout status flag clear
<i>ENET_DMA_FLAG_RO_CLR</i>	receive overflow status flag clear
<i>ENET_DMA_FLAG_TU_CLR</i>	transmit underflow status flag clear
<i>ENET_DMA_FLAG_RS_CLR</i>	receive status flag clear
<i>ENET_DMA_FLAG_RB_U_CLR</i>	receive buffer unavailable status flag clear
<i>ENET_DMA_FLAG_RP_S_CLR</i>	receive process stopped status flag clear
<i>ENET_DMA_FLAG_R_WT_CLR</i>	receive watchdog timeout status flag clear
<i>ENET_DMA_FLAG_ET_CLR</i>	early transmit status flag clear
<i>ENET_DMA_FLAG_FB_E_CLR</i>	fatal bus error status flag clear
<i>ENET_DMA_FLAG_ER_CLR</i>	early receive status flag clear
<i>ENET_DMA_FLAG_AI_CLR</i>	abnormal interrupt summary flag clear
<i>ENET_DMA_FLAG_NI_CLR</i>	normal interrupt summary flag clear
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the specified flag bit */
```

```
enet_flag_clear(ENET0, ENET_DMA_FLAG_RS_CLR);
```

enet_interrupt_enable

The description of enet_interrupt_enable is shown as below:

Table 3-469. Function enet_interrupt_enable

Function name	enet_interrupt_enable
Function prototype	void enet_interrupt_enable(uint32_t enet_periph, enet_int_enum enet_int);
Function descriptions	enable ENET MAC/MSD/DMA interrupt
Precondition	-
The called functions	-

Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
enet_int	ENET interrupt, refer to Table 3-437. Enum enet_int_enum only one parameter can be selected which is shown as below
<i>ENET_MAC_INT_WUMIM</i>	WUM interrupt mask
<i>ENET_MAC_INT_TMS TIM</i>	timestamp trigger interrupt mask
<i>ENET_MSC_INT_RFC EIM</i>	received frame CRC error interrupt mask
<i>ENET_MSC_INT_RFA EIM</i>	received frames alignment error interrupt mask
<i>ENET_MSC_INT_RGU FIM</i>	received good unicast frames interrupt mask
<i>ENET_MSC_INT_TGF SCIM</i>	transmitted good frames single collision interrupt mask
<i>ENET_MSC_INT_TGF MSCIM</i>	transmitted good frames more single collision interrupt mask
<i>ENET_MSC_INT_TGFI M</i>	transmitted good frames interrupt mask
<i>ENET_DMA_INT_TIE</i>	transmit interrupt enable
<i>ENET_DMA_INT_TPSI E</i>	transmit process stopped interrupt enable
<i>ENET_DMA_INT_TBUI E</i>	transmit buffer unavailable interrupt enable
<i>ENET_DMA_INT_TJTI E</i>	transmit jabber timeout interrupt enable
<i>ENET_DMA_INT_ROIE</i>	receive overflow interrupt enable
<i>ENET_DMA_INT_TUIE</i>	transmit underflow interrupt enable
<i>ENET_DMA_INT_RIE</i>	receive interrupt enable
<i>ENET_DMA_INT_RBUI E</i>	receive buffer unavailable interrupt enable
<i>ENET_DMA_INT_RPSI E</i>	receive process stopped interrupt enable
<i>ENET_DMA_INT_RWT IE</i>	receive watchdog timeout interrupt enable
<i>ENET_DMA_INT_ETIE</i>	early transmit interrupt enable
<i>ENET_DMA_INT_FBEI E</i>	fatal bus error interrupt enable
<i>ENET_DMA_INT_ERIE</i>	early receive interrupt enable

<i>ENET_DMA_INT_AIE</i>	abnormal interrupt summary enable
<i>ENET_DMA_INT_NIE</i>	normal interrupt summary enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable normal interrupt summary */
```

```
enet_interrupt_enable(ENET0, ENET_DMA_INT_NIE);
```

enet_interrupt_disable

The description of enet_interrupt_disable is shown as below:

Table 3-470. Function enet_interrupt_disable

Function name	enet_interrupt_disable
Function prototype	void enet_interrupt_disable(uint32_t enet_periph, enet_int_enum enet_int);
Function descriptions	disable ENET MAC/MSC/DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
enet_int	ENET interrupt, refer to Table 3-437. Enum enet_int_enum only one parameter can be selected which is shown as below
<i>ENET_MAC_INT_WUM</i> <i>IM</i>	WUM interrupt mask
<i>ENET_MAC_INT_TMS</i> <i>TIM</i>	timestamp trigger interrupt mask
<i>ENET_MSC_INT_RFC</i> <i>EIM</i>	received frame CRC error interrupt mask
<i>ENET_MSC_INT_RFA</i> <i>EIM</i>	received frames alignment error interrupt mask
<i>ENET_MSC_INT_RGU</i> <i>FIM</i>	received good unicast frames interrupt mask
<i>ENET_MSC_INT_TGF</i> <i>SCIM</i>	transmitted good frames single collision interrupt mask
<i>ENET_MSC_INT_TGF</i> <i>MSCIM</i>	transmitted good frames more single collision interrupt mask
<i>ENET_MSC_INT_TGFI</i>	transmitted good frames interrupt mask

<i>M</i>	
<i>ENET_DMA_INT_TIE</i>	transmit interrupt enable
<i>ENET_DMA_INT_TPSIE</i>	transmit process stopped interrupt enable
<i>ENET_DMA_INT_TBUIE</i>	transmit buffer unavailable interrupt enable
<i>ENET_DMA_INT_TJTI</i>	transmit jabber timeout interrupt enable
<i>ENET_DMA_INT_ROIE</i>	receive overflow interrupt enable
<i>ENET_DMA_INT_TUIE</i>	transmit underflow interrupt enable
<i>ENET_DMA_INT_RIE</i>	receive interrupt enable
<i>ENET_DMA_INT_RBUIE</i>	receive buffer unavailable interrupt enable
<i>ENET_DMA_INT_RPSIE</i>	receive process stopped interrupt enable
<i>ENET_DMA_INT_RWTIE</i>	receive watchdog timeout interrupt enable
<i>ENET_DMA_INT_ETIE</i>	early transmit interrupt enable
<i>ENET_DMA_INT_FBEIE</i>	fatal bus error interrupt enable
<i>ENET_DMA_INT_ERIE</i>	early receive interrupt enable
<i>ENET_DMA_INT_AIE</i>	abnormal interrupt summary enable
<i>ENET_DMA_INT_NIE</i>	normal interrupt summary enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable normal interrupt summary */
```

```
enet_interrupt_disable(ENET0, ENET_DMA_INT_NIE);
```

enet_interrupt_flag_get

The description of enet_interrupt_flag_get is shown as below:

Table 3-471. Function enet_interrupt_flag_get

Function name	enet_interrupt_flag_get
Function prototype	FlagStatus enet_interrupt_flag_get(uint32_t enet_periph, enet_int_flag_enum int_flag);
Function descriptions	get ENET MAC/MSD/DMA interrupt flag
Precondition	-
The called functions	-

Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
int_flag	ENET interrupt flag, refer to Table 3-438. Enum enet_int_flag_enum only one parameter can be selected which is shown as below
<i>ENET_MAC_INT_FLG_WUM</i>	WUM status flag
<i>ENET_MAC_INT_FLG_MSC</i>	MSC status flag
<i>ENET_MAC_INT_FLG_MSCR</i>	MSC receive status flag
<i>ENET_MAC_INT_FLG_MSCT</i>	MSC transmit status flag
<i>ENET_MAC_INT_FLG_TMST</i>	time stamp trigger status flag
<i>ENET_MSC_INT_FLG_RFCE</i>	received frames CRC error flag
<i>ENET_MSC_INT_FLG_RFAE</i>	received frames alignment error flag
<i>ENET_MSC_INT_FLG_RGUF</i>	received good unicast frames flag
<i>ENET_MSC_INT_FLG_TGFSC</i>	transmitted good frames single collision flag
<i>ENET_MSC_INT_FLG_TGFMSC</i>	transmitted good frames more single collision flag
<i>ENET_MSC_INT_FLG_TGF</i>	transmitted good frames flag
<i>ENET_DMA_INT_FLG_TS</i>	transmit status flag
<i>ENET_DMA_INT_FLG_TPS</i>	transmit process stopped status flag
<i>ENET_DMA_INT_FLG_TBU</i>	transmit buffer unavailable status flag
<i>ENET_DMA_INT_FLG_TJT</i>	transmit jabber timeout status flag
<i>ENET_DMA_INT_FLG_RO</i>	receive overflow status flag
<i>ENET_DMA_INT_FLG_TU</i>	transmit underflow status flag
<i>ENET_DMA_INT_FLG_RS</i>	receive status flag

ENET_DMA_INT_FLG G_RBU	receive buffer unavailable status flag
ENET_DMA_INT_FLG G_RPS	receive process stopped status flag
ENET_DMA_INT_FLG G_RWT	receive watchdog timeout status flag
ENET_DMA_INT_FLG G_ET	early transmit status flag
ENET_DMA_INT_FLG G_FBE	fatal bus error status flag
ENET_DMA_INT_FLG G_ER	early receive status flag
ENET_DMA_INT_FLG G_AI	abnormal interrupt summary flag
ENET_DMA_INT_FLG G_NI	normal interrupt summary flag
ENET_DMA_INT_FLG G_MSC	MSC status flag
ENET_DMA_INT_FLG G_WUM	WUM status flag
ENET_DMA_INT_FLG G_TST	timestamp trigger status flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check whether the specified flag bit is set or not */
```

```
enet_interrupt_flag_get(ENET0, ENET_DMA_INT_FLAG_RS);
```

enet_interrupt_flag_clear

The description of enet_interrupt_flag_clear is shown as below:

Table 3-472. Function enet_interrupt_flag_clear

Function name	enet_interrupt_flag_clear
Function prototype	void enet_interrupt_flag_clear(uint32_t enet_periph, enet_int_flag_clear_enum int_flag_clear);
Function descriptions	clear ENET DMA interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	

enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
int_flag_clear	clear ENET interrupt flag, refer to Table 3-439. Enum enet_int_flag_clear_enum . only one parameter can be selected which is shown as below
<i>ENET_DMA_INT_FLG_TS_CLR</i>	transmit status flag
<i>ENET_DMA_INT_FLG_TPS_CLR</i>	transmit process stopped status flag
<i>ENET_DMA_INT_FLG_TBU_CLR</i>	transmit buffer unavailable status flag
<i>ENET_DMA_INT_FLG_TJT_CLR</i>	transmit jabber timeout status flag
<i>ENET_DMA_INT_FLG_RO_CLR</i>	receive overflow status flag
<i>ENET_DMA_INT_FLG_TU_CLR</i>	transmit underflow status flag
<i>ENET_DMA_INT_FLG_RS_CLR</i>	receive status flag
<i>ENET_DMA_INT_FLG_RBU_CLR</i>	receive buffer unavailable status flag
<i>ENET_DMA_INT_FLG_RPS_CLR</i>	receive process stopped status flag
<i>ENET_DMA_INT_FLG_RWT_CLR</i>	receive watchdog timeout status flag
<i>ENET_DMA_INT_FLG_ET_CLR</i>	early transmit status flag
<i>ENET_DMA_INT_FLG_FBE_CLR</i>	fatal bus error status flag
<i>ENET_DMA_INT_FLG_ER_CLR</i>	early receive status flag
<i>ENET_DMA_INT_FLG_AI_CLR</i>	abnormal interrupt summary flag
<i>ENET_DMA_INT_FLG_NI_CLR</i>	normal interrupt summary flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear receive status flag bit */
```

```
enet_interrupt_flag_clear(ENET0, ENET_DMA_INT_FLAG_RS);
```

enet_tx_enable

The description of enet_tx_enable is shown as below:

Table 3-473. Function enet_tx_enable

Function name	enet_tx_enable
Function prototype	void enet_tx_enable(uint32_t enet_periph);
Function descriptions	ENET Tx function enable (include MAC and DMA module)
Precondition	-
The called functions	enet_txfifo_flush()
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable transport function of MAC and DMA */
```

```
enet_tx_enable(ENET0);
```

enet_tx_disable

The description of enet_tx_disable is shown as below:

Table 3-474. Function enet_tx_disable

Function name	enet_tx_disable
Function prototype	void enet_tx_disable(uint32_t enet_periph);
Function descriptions	ENET Tx function disable (include MAC and DMA module)
Precondition	-
The called functions	enet_txfifo_flush()
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* disable transport function of MAC and DMA */
```

```
enet_tx_disable(ENET0);
```

enet_rx_enable

The description of enet_rx_enable is shown as below:

Table 3-475. Function enet_rx_enable

Function name	enet_rx_enable
Function prototype	void enet_rx_enable(uint32_t enet_periph);
Function descriptions	ENET Rx function enable (include MAC and DMA module)
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable reception function of MAC and DMA */
```

```
enet_rx_enable(ENET0);
```

enet_rx_disable

The description of enet_rx_disable is shown as below:

Table 3-476. Function enet_rx_disable

Function name	enet_rx_disable
Function prototype	void enet_rx_disable(uint32_t enet_periph);
Function descriptions	ENET Rx function disable (include MAC and DMA module)
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable reception function of MAC and DMA */
```

```
enet_rx_disable(ENET0);
```

enet_registers_get

The description of enet_registers_get is shown as below:

Table 3-477. Function enet_registers_get

Function name	enet_registers_get
Function prototype	void enet_registers_get(uint32_t enet_periph, enet_registers_type_enum type, uint32_t *preg, uint32_t num);
Function descriptions	put registers value into the application buffer
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
type	register type which will be get, refer to Table 3-446. Enum enet_registers_type_enum only one parameter can be selected which is shown as below
<i>ALL_MAC_REG</i>	get the registers within the offset scope range ENET_MAC_CFG to ENET_MAC_FCTH
<i>ALL_MSC_REG</i>	get the registers within the offset scope range ENET_MSC_CTL to ENET_MSC_RGUFCNT
<i>ALL_PTP_REG</i>	get the registers within the offset scope range ENET_PTP_TSCTL to ENET_PTP_PPSCTL
<i>ALL_DMA_REG</i>	get the registers within the offset scope range ENET_DMA_BCTL to ENET_DMA_CRBADDR
Input parameter{in}	
num	the number of registers that the user want to get, (0 -- 54)
Output parameter{out}	
preg	the application buffer pointer for storing the register value
Return value	
-	-

Example:

```
/* get all mac registers value */
```

```
uint32_t register_buffer[5];
```

```
enet_registers_get(ENET0, ALL_MAC_REG, 5, register_buffer);
```

enet_debug_status_get

The description of enet_debug_status_get is shown as below:

Table 3-478. Function enet_debug_status_get

Function name	enet_debug_status_get
Function prototype	uint32_t enet_debug_status_get(uint32_t enet_periph, uint32_t mac_debug);
Function descriptions	get the enet debug status from the debug register
Precondition	-
The called functions	--
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
mac_debug	enet debug status only one parameter can be selected which is shown as below
ENET_MAC_RECEIVE R_NOT_IDLE	MAC receiver is not in idle state
ENET_RX_ASYNC HROUS_FIFO_STATE	Rx asynchronous FIFO status
ENET_RXFIFO_WRITE NG	RxFIFO is doing write operation
ENET_RXFIFO_READ _STATUS	RxFIFO read operation status
ENET_RXFIFO_STATE	RxFIFO state
ENET_MAC_TRANSMI TTER_NOT_IDLE	MAC transmitter is not in idle state
ENET_MAC_TRANSMI TTER_STATUS	status of MAC transmitter
ENET_PAUSE_CONDI TION_STATUS	pause condition status
ENET_TXFIFO_READ _STATUS	TxFIFO read operation status
ENET_TXFIFO_WRITE NG	TxFIFO is doing write operation
ENET_TXFIFO_NOT_E MPTY	TxFIFO is not empty

<i>ENET_TXFIFO_FULL</i>	TxFIFO is full
Output parameter{out}	
-	-
Return value	
uint32_t	value of the status users want to get

Example:

```
/* get debug message of RxFIFO state */

uint32_t debug_value;

debug_value = enet_debug_status_get (ENET0, ENET_RXFIFO_STATE);
```

enet_address_filter_enable

The description of enet_address_filter_enable is shown as below:

Table 3-479. Function enet_address_filter_enable

Function name	enet_address_filter_enable
Function prototype	void enet_address_filter_enable(uint32_t enet_periph, enet_macaddress_enum mac_addr);
Function descriptions	enable the MAC address filter
Precondition	-
The called functions	--
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
mac_addr	select which MAC address will be enable refer to Table 3-450. Enum enet_macaddress_enum . only one parameter can be selected which is shown as below
<i>ENET_MAC_ADDRES S1</i>	enable MAC address 1 filter
<i>ENET_MAC_ADDRES S2</i>	enable MAC address 2 filter
<i>ENET_MAC_ADDRES S3</i>	enable MAC address 3 filter
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the MAC address 1 filter */
```



```
enet_address_filter_enable(ENET0, ENET_MAC_ADDRESS1);
```

enet_address_filter_disable

The description of enet_address_filter_disable is shown as below:

Table 3-480. Function enet_address_filter_disable

Function name	enet_address_filter_disable
Function prototype	void enet_address_filter_disable(uint32_t enet_periph, enet_macaddress_enum mac_addr);
Function descriptions	disable the MAC address filter
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
mac_addr	select which MAC address will be enable, refer to Table 3-450. Enum enet_macaddress_enum . only one parameter can be selected which is shown as below
ENET_MAC_ADDRES S1	enable MAC address 1 filter
ENET_MAC_ADDRES S2	enable MAC address 2 filter
ENET_MAC_ADDRES S3	enable MAC address 3 filter
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the MAC address 1 filter */
```

```
enet_address_filter_disable(ENET0, ENET_MAC_ADDRESS1);
```

enet_address_filter_config

The description of enet_address_filter_config is shown as below:

Table 3-481. Function enet_address_filter_config

Function name	enet_address_filter_config
Function prototype	void enet_address_filter_config(uint32_t enet_periph, enet_macaddress_enum mac_addr, uint32_t addr_mask, uint32_t

	filter_type);
Function descriptions	configure the MAC address filter
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
mac_addr	select which MAC address will be enable, refer to Table 3-450. Enum enet_macaddress_enum . only one parameter can be selected which is shown as below
ENET_MAC_ADDRES S1	enable MAC address 1 filter
ENET_MAC_ADDRES S2	enable MAC address 2 filter
ENET_MAC_ADDRES S3	enable MAC address 3 filter
Input parameter{in}	
addr_mask	select which MAC address bytes will be mask one or more parameters can be selected which are shown as below
ENET_ADDRESS_MA SK_BYTE0	mask ENET_MAC_ADDR1L[7:0] bits
ENET_ADDRESS_MA SK_BYTE1	mask ENET_MAC_ADDR1L[15:8] bits
ENET_ADDRESS_MA SK_BYTE2	mask ENET_MAC_ADDR1L[23:16] bits
ENET_ADDRESS_MA SK_BYTE3	mask ENET_MAC_ADDR1L [31:24] bits
ENET_ADDRESS_MA SK_BYTE4	mask ENET_MAC_ADDR1H [7:0] bits
ENET_ADDRESS_MA SK_BYTE5	mask ENET_MAC_ADDR1H [15:8] bits
Input parameter{in}	
filter_type	select which MAC address filter type will be selected only one parameter can be selected which is shown as below
ENET_ADDRESS_FILT ER_SA	The MAC address is used to compared with the SA field of the received frame
ENET_ADDRESS_FILT ER_DA	The MAC address is used to compared with the DA field of the received frame
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* config the MAC address 1 filter */
```

```
enet_address_filter_config(ENET0, ENET_MAC_ADDRESS1, ENET_ADDRESS_MASK_BYTE0 | ENET_ADDRESS_MASK_BYTE1 | ENET_ADDRESS_MASK_BYTE2, ENET_ADDRESS_FILTER_DA);
```

enet_phy_config

The description of enet_phy_config is shown as below:

Table 3-482. Function enet_phy_config

Function name	enet_phy_config
Function prototype	ErrStatus enet_phy_config(uint32_t enet_periph);
Function descriptions	PHY interface configuration (configure SMI clock and reset PHY chip)
Precondition	-
The called functions	rcu_clock_freq_get()/enet_phy_write_read()
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* config PHY interface */
```

```
enet_phy_config(ENET0);
```

enet_phy_write_read

The description of enet_phy_write_read is shown as below:

Table 3-483. Function enet_phy_write_read

Function name	enet_phy_write_read
Function prototype	ErrStatus enet_phy_write_read(uint32_t enet_periph, enet_phydirection_enum direction, uint16_t phy_address, uint16_t phy_reg, uint16_t *pvalue);
Function descriptions	write to / read from a PHY register
Precondition	-
The called functions	-
Input parameter{in}	

enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
direction	refer to Table 3-448. Enum enet_phydirection_enum , only one parameter can be selected which is shown as below
<i>ENET_PHY_WRITE</i>	write data to phy register
<i>ENET_PHY_READ</i>	read data from phy register
Input parameter{in}	
phy_address	0x0 - 0x1F
Input parameter{in}	
phy_reg	0x0 - 0x1F
Input parameter{in}	
pvalue	the value will be written to the PHY register in ENET_PHY_WRITE direction
Output parameter{out}	
pvalue	the value will be read from the PHY register in ENET_PHY_READ direction
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write 0 to PHY BCR register */
```

```
uint16_t temp_phy = 0U;
```

```
phy_state = enet_phy_write_read(ENET0, ENET_PHY_WRITE, PHY_ADDRESS,
PHY_REG_BCR, &temp_phy);
```

enet_phyloopback_enable

The description of enet_phyloopback_enable is shown as below:

Table 3-484. Function enet_phyloopback_enable

Function name	enet_phyloopback_enable
Function prototype	ErrStatus enet_phyloopback_enable(uint32_t enet_periph);
Function descriptions	enable the loopback function of PHY chip
Precondition	-
The called functions	enet_phy_write_read()
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	

ErrStatus	ERROR or SUCCESS
------------------	------------------

Example:

```
/* enable the loopback function of PHY chip */

ErrStatus phy_state = ERROR;

phy_state = enet_phyloopback_enable(ENET0);
```

enet_phyloopback_disable

The description of enet_phyloopback_disable is shown as below:

Table 3-485. Function enet_phyloopback_disable

Function name	enet_phyloopback_disable
Function prototype	ErrStatus enet_phyloopback_disable(uint32_t enet_periph);
Function descriptions	disable the loopback function of PHY chip
Precondition	-
The called functions	enet_phy_write_read
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* disable the loopback function of PHY chip */

ErrStatus phy_state = ERROR;

phy_state = enet_phyloopback_disable(ENET0);
```

enet_forward_feature_enable

The description of enet_forward_feature_enable is shown as below:

Table 3-486. Function enet_forward_feature_enable

Function name	enet_forward_feature_enable
Function prototype	void enet_forward_feature_enable(uint32_t enet_periph, uint32_t feature);
Function descriptions	enable ENET forward feature
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral

<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	the feature of ENET forward mode one or more parameters can be selected which are shown as below
<i>ENET_AUTO_PADCRC_DROP</i>	the function of the MAC strips the Pad/FCS field on received frames
<i>ENET_TYPEFRAME_CRC_DROP</i>	the function that FCS field(last 4 bytes) of frame will be dropped before forwarding
<i>ENET_FORWARD_ERROR_FRAMES</i>	the function that all frame received with error except runt error are forwarded to memory
<i>ENET_FORWARD_UNDERSZ_GOODFRAMES</i>	the function that forwarding undersized good frames
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the function that forwarding undersized good frames */
```

```
enet_forward_feature_enable(ENET0, ENET_FORWARD_UNDERSZ_GOODFRAMES);
```

enet_forward_feature_disable

The description of enet_forward_feature_disable is shown as below:

Table 3-487. Function enet_forward_feature_disable

Function name	enet_forward_feature_disable
Function prototype	void enet_forward_feature_disable(uint32_t enet_periph, uint32_t feature);
Function descriptions	disable ENET forward feature
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	the feature of ENET forward mode one or more parameters can be selected which are shown as below
<i>ENET_AUTO_PADCRC_DROP</i>	the function of the MAC strips the Pad/FCS field on received frames
<i>ENET_TYPEFRAME_CRC_DROP</i>	the function that FCS field(last 4 bytes) of frame will be dropped before

<i>RC_DROP</i>	forwarding
<i>ENET_FORWARD_ERRORFRAMES</i>	the function that all frame received with error except runt error are forwarded to memory
<i>ENET_FORWARD_UNDERSZ_GOODFRAMES</i>	the function that forwarding undersized good frames
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the function that forwarding undersized good frames */
```

```
enet_forward_feature_enable(ENET0, ENET_FORWARD_UNDERSZ_GOODFRAMES);
```

enet_fliter_feature_enable

The description of enet_fliter_feature_enable is shown as below:

Table 3-488. Function enet_fliter_feature_enable

Function name	enet_fliter_feature_enable
Function prototype	void enet_fliter_feature_enable(uint32_t enet_periph, uint32_t feature);
Function descriptions	enable ENET fliter feature
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	the feature of ENET fliter mode one or more parameters can be selected which are shown as below
<i>ENET_SRC_FILTER</i>	filter source address function
<i>ENET_SRC_FILTER_INVERSE</i>	inverse source address filtering result function
<i>ENET_DEST_FILTER_INVERSE</i>	inverse DA filtering result function
<i>ENET_MULTICAST_FILTER_PASS</i>	pass all multicast frames function
<i>ENET_MULTICAST_FILTER_HASH_MODE</i>	HASH multicast filter function
<i>ENET_UNICAST_FILTER_HASH_MODE</i>	HASH unicast filter function

<i>ENET_FILTER_MODE</i> <i>_EITHER</i>	HASH or perfect filter function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable filter source address function */
```

```
enet_fliter_feature_enable(ENET0, ENET_SRC_FILTER);
```

enet_fliter_feature_disable

The description of enet_fliter_feature_disable is shown as below:

Table 3-489. Function enet_fliter_feature_disable

Function name	enet_fliter_feature_disable
Function prototype	void enet_fliter_feature_disable(uint32_t enet_periph, uint32_t feature);
Function descriptions	disable ENET fliter feature
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	the feature of ENET fliter mode one or more parameters can be selected which are shown as below
<i>ENET_SRC_FILTER</i>	filter source address function
<i>ENET_SRC_FILTER_I</i> <i>INVERSE</i>	inverse source address filtering result function
<i>ENET_DEST_FILTER_</i> <i>INVERSE</i>	inverse DA filtering result function
<i>ENET_MULTICAST_FI</i> <i>LTER_PASS</i>	pass all multicast frames function
<i>ENET_MULTICAST_FI</i> <i>LTER_HASH_MODE</i>	HASH multicast filter function
<i>ENET_UNICAST_FILT</i> <i>ER_HASH_MODE</i>	HASH unicast filter function
<i>ENET_FILTER_MODE</i> <i>_EITHER</i>	HASH or perfect filter function
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* enable filter source address function */
```

```
enet_fliter_feature_enable(ENET0, ENET_SRC_FILTER);
```

enet_pauseframe_generate

The description of enet_pauseframe_generate is shown as below:

Table 3-490. Function enet_pauseframe_generate

Function name	enet_pauseframe_generate
Function prototype	ErrStatus enet_pauseframe_generate(uint32_t enet_periph);
Function descriptions	generate the pause frame, ENET will send pause frame after enable transmit flow control this function only use in full-duplex mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* generate the pause frame */
```

```
ErrStatus reval;
```

```
reval = enet_pauseframe_generate(ENET0);
```

enet_pauseframe_detect_config

The description of enet_pauseframe_detect_config is shown as below:

Table 3-491. Function enet_pauseframe_detect_config

Function name	enet_pauseframe_detect_config
Function prototype	void enet_pauseframe_detect_config(uint32_t enet_periph, uint32_t detect);
Function descriptions	configure the pause frame detect type
Precondition	-
The called functions	-
Input parameter{in}	

enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
detect	pause frame detect type only one parameter can be selected which is shown as below
<i>ENET_MAC0_AND_UNIQUE_ADDRESS_PAUSEDETECT</i>	besides the unique multicast address, MAC can also use the MAC0 address to detecting pause frame
<i>ENET_UNIQUE_PAUSEDETECT</i>	only the unique multicast address for pause frame which is specified in IEEE802.3 can be detected
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config the pause frame detect type as ENET_UNIQUE_PAUSEDETECT */
enet_pauseframe_detect_config(ENET0, ENET_UNIQUE_PAUSEDETECT);
```

enet_pauseframe_config

The description of enet_pauseframe_config is shown as below:

Table 3-492. Function enet_pauseframe_config

Function name	enet_pauseframe_config
Function prototype	void enet_pauseframe_config(uint32_t enet_periph, uint32_t pausetime, uint32_t pause_threshold);
Function descriptions	configure the pause frame parameters
Precondition	-
The called functions	-
Input parameter{in}	
pausetime	pause time in transmit pause control frame, (0 – 0xFFFF)
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
pause_threshold	the threshold of the pause timer for retransmitting frames automatically, this value must make sure to be less than configured pause time, only one parameter can be selected which is shown as below
<i>ENET_PAUSETIME_MINUS_4_SLOT_TIMES</i>	pause time minus 4 slot times

<i>ENET_PAUSETIME_MINUS28</i>	pause time minus 28 slot times
<i>ENET_PAUSETIME_MINUS144</i>	pause time minus 144 slot times
<i>ENET_PAUSETIME_MINUS256</i>	pause time minus 256 slot times
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config pause time minus 4 slot times */
```

```
enet_pauseframe_config(ENET0, 30, ENET_PAUSETIME_MINUS4);
```

enet_flowcontrol_threshold_config

The description of enet_flowcontrol_threshold_config is shown as below:

Table 3-493. Function enet_flowcontrol_threshold_config

Function name	enet_flowcontrol_threshold_config
Function prototype	void enet_flowcontrol_threshold_config(uint32_t enet_periph, uint32_t deactive, uint32_t active);
Function descriptions	configure the threshold of the flow control(deactive and active threshold)
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
deactive	the threshold of the deactive flow control, this value should always be less than active flow control value, only one parameter can be selected which is shown as below
<i>ENET_DEACTIVE_THRESHOLD_256BYTES</i>	threshold level is 256 bytes
<i>ENET_DEACTIVE_THRESHOLD_512BYTES</i>	threshold level is 512 bytes
<i>ENET_DEACTIVE_THRESHOLD_768BYTES</i>	threshold level is 768 bytes
<i>ENET_DEACTIVE_THRESHOLD_1024BYTE</i> S	threshold level is 1024 bytes

ENET_DEACTIVE_THRESHOLD_1280BYTES	threshold level is 1280 bytes
ENET_DEACTIVE_THRESHOLD_1536BYTES	threshold level is 1536 bytes
ENET_DEACTIVE_THRESHOLD_1792BYTES	threshold level is 1792 bytes
Input parameter{in}	
active	the threshold of the active flow control, only one parameter can be selected which is shown as below
ENET_ACTIVE_THRESHOLD_256BYTES	threshold level is 256 bytes
ENET_ACTIVE_THRESHOLD_512BYTES	threshold level is 512 bytes
ENET_ACTIVE_THRESHOLD_768BYTES	threshold level is 768 bytes
ENET_ACTIVE_THRESHOLD_1024BYTES	threshold level is 1024 bytes
ENET_ACTIVE_THRESHOLD_1280BYTES	threshold level is 1280 bytes
ENET_ACTIVE_THRESHOLD_1536BYTES	threshold level is 1536 bytes
ENET_ACTIVE_THRESHOLD_1792BYTES	threshold level is 1792 bytes
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the threshold of the flow control */
```

```
enet_flowcontrol_threshold_config(ENET0, ENET_DEACTIVE_THRESHOLD_256BYTES,
ENET_ACTIVE_THRESHOLD_256BYTES);
```

enet_flowcontrol_feature_enable

The description of enet_flowcontrol_feature_enable is shown as below:

Table 3-494. Function enet_flowcontrol_feature_enable

Function name	enet_flowcontrol_feature_enable
Function prototype	void enet_flowcontrol_feature_enable(uint32_t enet_periph, uint32_t

	feature);
Function descriptions	enable ENET flow control feature
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	the feature of ENET flow control mode one or more parameters can be selected which are shown as below
<i>ENET_ZERO_QUANTA_PAUSE</i>	the automatic zero-quanta generation function
<i>ENET_TX_FLOWCONTROL</i>	the flow control operation in the MAC
<i>ENET_RX_FLOWCONTROL</i>	decoding function for the received pause frame and process it
<i>ENET_BACK_PRESSURE</i>	back pressure operation in the MAC(only use in half-duplex mode)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the flow control operation in the MAC */
```

```
enet_flowcontrol_feature_enable(ENET0, ENET_ZERO_QUANTA_PAUSE);
```

enet_flowcontrol_feature_disable

The description of enet_flowcontrol_feature_disable is shown as below:

Table 3-495. Function enet_flowcontrol_feature_disable

Function name	enet_flowcontrol_feature_disable
Function prototype	void enet_flowcontrol_feature_disable(uint32_t enet_periph, uint32_t feature);
Function descriptions	disable ENET flow control feature
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1

Input parameter{in}	
feature	the feature of ENET flow control mode one or more parameters can be selected which are shown as below
<i>ENET_ZERO_QUANTA_PAUSE</i>	the automatic zero-quanta generation function
<i>ENET_TX_FLOWCONTROL</i>	the flow control operation in the MAC
<i>ENET_RX_FLOWCONTROL</i>	decoding function for the received pause frame and process it
<i>ENET_BACK_PRESSURE</i>	back pressure operation in the MAC(only use in half-duplex mode)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the automatic zero-quanta generation function */
```

```
enet_flowcontrol_feature_disable(ENET0, ENET_ZERO_QUANTA_PAUSE);
```

enet_dmaprocess_state_get

The description of enet_dmaprocess_state_get is shown as below:

Table 3-496. Function enet_dmaprocess_state_get

Function name	enet_dmaprocess_state_get
Function prototype	uint32_t enet_dmaprocess_state_get(uint32_t enet_periph, enet_dmadirection_enum direction);
Function descriptions	get the dma transmit/receive process state
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
direction	choose the direction of DMA process which users want to resume only one parameter can be selected which is shown as below
<i>ENET_DMA_TX</i>	DMA transmit process
<i>ENET_DMA_RX</i>	DMA receive process
Output parameter{out}	
-	-
Return value	

uint32_t	state of dma process, the value range shows below: ENET_RX_STATE_STOPPED / ENET_RX_STATE_FETCHING / ENET_RX_STATE_WAITING / ENET_RX_STATE_SUSPENDED / ENET_RX_STATE_CLOSING / ENET_RX_STATE_QUEUEING / ENET_TX_STATE_STOPPED / ENET_TX_STATE_FETCHING / ENET_TX_STATE_WAITING / ENET_TX_STATE_READING / ENET_TX_STATE_SUSPENDED / ENET_TX_STATE_CLOSING
-----------------	---

Example:

```

/* get the dma receive process state */

uint32_t reval;

reval = enet_dmaprocess_state_get(ENET0, ENET_DMA_RX);

if(ENET_RX_STATE_SUSPENDED == reval){

    do...

}

```

enet_dmaprocess_resume

The description of enet_dmaprocess_resume is shown as below:

Table 3-497. Function enet_dmaprocess_resume

Function name	enet_dmaprocess_resume
Function prototype	void enet_dmaprocess_resume(uint32_t enet_periph, enet_dmadirection_enum direction);
Function descriptions	poll the DMA transmission/reception enable by writing any value to the ENET_DMA_TPEN/ENET_DMA_RPEN register, this will make the DMA to resume transmission/reception
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
direction	choose the direction of DMA process which users want to resume only one parameter can be selected which is shown as below
ENET_DMA_TX	DMA transmit process
ENET_DMA_RX	DMA receive process
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable DMA receive process */
```

```
enet_dmaprocess_resume(ENET0, ENET_DMA_RX);
```

enet_rxprocess_check_recovery

The description of enet_rxprocess_check_recovery is shown as below:

Table 3-498. Function enet_rxprocess_check_recovery

Function name	enet_rxprocess_check_recovery
Function prototype	void enet_rxprocess_check_recovery(uint32_t enet_periph);
Function descriptions	check and recover the Rx process
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* check and recover the Rx process */
```

```
enet_rxprocess_check_recovery(ENET0);
```

enet_txfifo_flush

The description of enet_txfifo_flush is shown as below:

Table 3-499. Function enet_txfifo_flush

Function name	enet_txfifo_flush
Function prototype	ErrStatus enet_txfifo_flush(uint32_t enet_periph);
Function descriptions	flush the ENET transmit FIFO, and wait until the flush operation completes
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1

Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* flush the ENET transmit FIFO */
```

```
ErrStatus reval = ERROR;
```

```
reval = enet_txfifo_flush(ENET0);
```

enet_current_desc_address_get

The description of enet_current_desc_address_get is shown as below:

Table 3-500. Function enet_current_desc_address_get

Function name	enet_current_desc_address_get
Function prototype	uint32_t enet_current_desc_address_get(uint32_t enet_periph, enet_desc_reg_enum addr_get);
Function descriptions	get the transmit/receive address of current descriptor, or current buffer, or descriptor table
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
addr_get	choose the address which users want to get, refer to Table 3-440. Enum enet_desc_reg_enum only one parameter can be selected which is shown as below
ENET_RX_DESC_TABLE	the start address of the receive descriptor table
ENET_RX_CURRENT_DESC	the start descriptor address of the current receive descriptor read by the RxDMA controller
ENET_RX_CURRENT_BUFFER	the current receive buffer address being read by the RxDMA controller
ENET_TX_DESC_TABLE	the start address of the transmit descriptor table
ENET_TX_CURRENT_DESC	the start descriptor address of the current transmit descriptor read by the TxDMA controller
ENET_TX_CURRENT_BUFFER	the current transmit buffer address being read by the TxDMA controller
Output parameter{out}	

-	-
Return value	
uint32_t	0- 0xFFFFFFFF

Example:

```
/* get the start address of the receive descriptor table */
```

```
uint32_t reval;
```

```
reval = enet_current_desc_address_get(ENET0, ENET_RX_DESC_TABLE);
```

enet_desc_information_get

The description of enet_desc_information_get is shown as below:

Table 3-501. Function enet_desc_information_get

Function name	enet_desc_information_get
Function prototype	uint32_t enet_desc_information_get(uint32_t enet_periph, enet_descriptors_struct *desc, enet_descstate_enum info_get);
Function descriptions	get the Tx or Rx descriptor information
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
desc	the descriptor pointer which users want to get information, the structure members can refer to Table 3-433. Structure enet_descriptors_struct
Input parameter{in}	
info_get	the descriptor information type which is selected, refer to Table 3-451. Enum enet_descstate_enum only one parameter can be selected which is shown as below
TXDESC_COLLISION_COUNT	the number of collisions occurred before the frame was transmitted
TXDESC_BUFFER_1_ADDR	the buffer1 address of the Tx frame
RXDESC_FRAME_LENGTH	the byte length of the received frame that was transferred to the buffer
RXDESC_BUFFER_1_SIZE	receive buffer 1 size
RXDESC_BUFFER_2_SIZE	receive buffer 2 size
RXDESC_BUFFER_1_ADDR	the buffer1 address of the Rx frame

Output parameter{out}	
-	-
Return value	
uint32_t	descriptor information if value is 0xFFFFFFFFU, means the false input parameter

Example:

```
/* get the reception buffer 1 size */

uint32_t reval;

reval = enet_desc_information_get(ENET0, rx_desc, RXDESC_BUFFER_1_SIZE);
```

enet_missed_frame_counter_get

The description of enet_missed_frame_counter_get is shown as below:

Table 3-502. Function enet_missed_frame_counter_get

Function name	enet_missed_frame_counter_get
Function prototype	void enet_missed_frame_counter_get(uint32_t enet_periph, uint32_t *rxfifo_drop, uint32_t *rxdma_drop);
Function descriptions	get the number of missed frames during receiving
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Output parameter{out}	
rxfifo_drop	pointer to the number of frames dropped by RxFIFO
Output parameter{out}	
rxdma_drop	pointer to the number of frames missed by the RxDMA controller
Return value	
-	-

Example:

```
/* get the number of missed frames during receiving */

uint32_t rxcnt, txcnt;

enet_missed_frame_counter_get(ENET0, &rxcnt, &txcnt);
```

enet_desc_flag_get

The description of enet_desc_flag_get is shown as below:

Table 3-503. Function `enet_desc_flag_get`

Function name	<code>enet_desc_flag_get</code>
Function prototype	<code>FlagStatus enet_desc_flag_get(enet_descriptors_struct *desc, uint32_t desc_flag);</code>
Function descriptions	get the bit flag of ENET DMA descriptor
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to get flag, the structure members can refer to Table 3-433. Structure <code>enet_descriptors_struct</code>
Input parameter{in}	
desc_flag (the value according to the parameter desc)	the bit flag of ENET DMA descriptor only one parameter can be selected which is shown as below
When type of parameter desc is TX	
<code>ENET_TDES0_DB</code>	deferred
<code>ENET_TDES0_UFE</code>	underflow error
<code>ENET_TDES0_EXD</code>	excessive deferral
<code>ENET_TDES0_VFRM</code>	VLAN frame
<code>ENET_TDES0_ECO</code>	excessive collision
<code>ENET_TDES0_LCO</code>	late collision
<code>ENET_TDES0_NCA</code>	no carrier
<code>ENET_TDES0_LCA</code>	loss of carrier
<code>ENET_TDES0_IPPE</code>	IP payload error
<code>ENET_TDES0_FRMF</code>	frame flushed
<code>ENET_TDES0_JT</code>	jabber timeout
<code>ENET_TDES0_ES</code>	error summary
<code>ENET_TDES0_IPHE</code>	IP header error
<code>ENET_TDES0_TTMSS</code>	transmit timestamp status
<code>ENET_TDES0_TCHM</code>	the second address chained mode
<code>ENET_TDES0_TERM</code>	transmit end of ring mode
<code>ENET_TDES0_TTSSEN</code>	transmit timestamp function enable
<code>ENET_TDES0_DPAD</code>	disable adding pad
<code>ENET_TDES0_DCRC</code>	disable CRC
<code>ENET_TDES0_FSG</code>	first segment
<code>ENET_TDES0_LSG</code>	last segment
<code>ENET_TDES0_INTC</code>	interrupt on completion
<code>ENET_TDES0_DAV</code>	DAV bit
When type of parameter desc is RX	
<code>ENET_RDES0_PCERR</code>	payload checksum error
<code>ENET_RDES0_CERR</code>	CRC error
<code>ENET_RDES0_DBERR</code>	dribble bit error

ENET_RDES0_RERR	receive error
ENET_RDES0_RWDT	receive watchdog timeout
ENET_RDES0_FRMT	frame type
ENET_RDES0_LCO	late collision
ENET_RDES0_IPHER R	IP frame header error
ENET_RDES0_LDES	last descriptor
ENET_RDES0_FDES	first descriptor
ENET_RDES0_VTAG	VLAN tag
ENET_RDES0_OERR	overflow error
ENET_RDES0_LERR	length error
ENET_RDES0_SAFF	SA filter fail
ENET_RDES0_DERR	descriptor error
ENET_RDES0_ERRS	error summary
ENET_RDES0_DAFF	destination address filter fail
ENET_RDES0_DAV	descriptor available
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the bit flag of ENET DMA descriptor */
```

```
FlagStatus reval;
```

```
reval = enet_desc_flag_get(p_txdesc, ENET_TDES0_TCHM);
```

enet_desc_flag_set

The description of enet_desc_flag_set is shown as below:

Table 3-504. Function enet_desc_flag_set

Function name	enet_desc_flag_set
Function prototype	void enet_desc_flag_set(enet_descriptors_struct *desc, uint32_t desc_flag);
Function descriptions	set the bit flag of ENET DMA descriptor
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to set flag, the structure members can refer to Table 3-433. Structure enet_descriptors_struct
Input parameter{in}	
desc_flag (the value according to	the bit flag of ENET DMA descriptor only one parameter can be selected which is shown as below

the parameter desc)	
When type of parameter desc is TX	
<i>ENET_TDES0_VFRM</i>	VLAN frame
<i>ENET_TDES0_FRMF</i>	frame flushed
<i>ENET_TDES0_TCHM</i>	the second address chained mode
<i>ENET_TDES0_TERM</i>	transmit end of ring mode
<i>ENET_TDES0_TTSEN</i>	transmit timestamp function enable
<i>ENET_TDES0_DPAD</i>	disable adding pad
<i>ENET_TDES0_DCRC</i>	disable CRC
<i>ENET_TDES0_FSG</i>	first segment
<i>ENET_TDES0_LSG</i>	last segment
<i>ENET_TDES0_INTC</i>	interrupt on completion
<i>ENET_TDES0_DAV</i>	DAV bit
When type of parameter desc is RX	
<i>ENET_RDES0_DAV</i>	descriptor available
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set VLAN frame bit flag of ENET DMA descriptor */
enet_desc_flag_set(p_txdesc, ENET_TDES0_VFRM);
```

enet_desc_flag_clear

The description of enet_desc_flag_clear is shown as below:

Table 3-505. Function enet_desc_flag_clear

Function name	enet_desc_flag_clear
Function prototype	void enet_desc_flag_clear(enet_descriptors_struct *desc, uint32_t desc_flag);
Function descriptions	clear the bit flag of ENET DMA descriptor
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to set flag, the structure members can refer to Table 3-433. Structure enet_descriptors_struct
Input parameter{in}	
desc_flag (the value according to the parameter desc)	the bit flag of ENET DMA descriptor only one parameter can be selected which is shown as below
When type of parameter desc is TX	

<i>ENET_TDES0_VFRM</i>	VLAN frame
<i>ENET_TDES0_FRMF</i>	frame flushed
<i>ENET_TDES0_TCHM</i>	the second address chained mode
<i>ENET_TDES0_TERM</i>	transmit end of ring mode
<i>ENET_TDES0_TTSEN</i>	transmit timestamp function enable
<i>ENET_TDES0_DPAD</i>	disable adding pad
<i>ENET_TDES0_DCRC</i>	disable CRC
<i>ENET_TDES0_FSG</i>	first segment
<i>ENET_TDES0_LSG</i>	last segment
<i>ENET_TDES0_INTC</i>	interrupt on completion
<i>ENET_TDES0_DAV</i>	DAV bit
When type of parameter desc is RX	
<i>ENET_RDES0_DAV</i>	descriptor available
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear VLAN frame bit flag of ENET DMA descriptor */
```

```
enet_desc_flag_clear(p_txdesc, ENET_TDES0_VFRM);
```

enet_rx_desc_immediate_receive_complete_interrupt

The description of enet_rx_desc_immediate_receive_complete_interrupt is shown as below:

Table 3-506. Function enet_rx_desc_immediate_receive_complete_interrupt

Function name	enet_rx_desc_immediate_receive_complete_interrupt
Function prototype	void enet_rx_desc_immediate_receive_complete_interrupt(enet_descriptors_struct *desc);
Function descriptions	when receiving completed, set RS bit in ENET_DMA_STAT register will immediately set
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to get flag, the structure members can refer to Table 3-433. Structure enet_descriptors_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set RS bit in ENET_DMA_STAT register immediately when receiving completed */
```

```
enet_rx_desc_immediate_receive_complete_interrupt(p_rxdesc);
```

enet_rx_desc_delay_receive_complete_interrupt

The description of enet_rx_desc_delay_receive_complete_interrupt is shown as below:

Table 3-507. Function enet_rx_desc_delay_receive_complete_interrupt

Function name	enet_rx_desc_delay_receive_complete_interrupt
Function prototype	void enet_rx_desc_delay_receive_complete_interrupt(uint32_t enet_periph, enet_descriptors_struct *desc, uint32_t delay_time);
Function descriptions	when receiving completed, set RS bit in ENET_DMA_STAT register will is set after a configurable delay time
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
desc	the descriptor pointer which users want to get flag, the structure members can refer to Table 3-433. Structure enet_descriptors_struct
Input parameter{in}	
delay_time	delay a time of 256*delay_time HCLK(0x00000000 - 0x000000FF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* when receiving completed, RS bit in ENET_DMA_STAT register will be set after 256*16 HCLK */
```

```
enet_rx_desc_delay_receive_complete_interrupt(ENET0, p_rxdesc, 0x00000010);
```

enet_rxframe_drop

The description of enet_rxframe_drop is shown as below:

Table 3-508. Function enet_rxframe_drop

Function name	enet_rxframe_drop
Function prototype	void enet_rxframe_drop(uint32_t enet_periph);
Function descriptions	drop current receive frame
Precondition	-
The called functions	-

Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* drop current receive frame */
enet_rxfame_drop(ENET0);
```

enet_dma_feature_enable

The description of enet_dma_feature_enable is shown as below:

Table 3-509. Function enet_dma_feature_enable

Function name	enet_dma_feature_enable
Function prototype	void enet_dma_feature_enable(uint32_t enet_periph, uint32_t feature);
Function descriptions	enable DMA feature
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	the feature of DMA mode one or more parameters can be selected which are shown as below
<i>ENET_NO_FLUSH_RXFRAME</i>	RxDMA does not flushes frames function
<i>ENET_SECONDFRAME_E_OPT</i>	TxDMA controller operate on second frame function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RxDMA does not flushes frames function */
enet_dma_feature_enable(ENET0, ENET_NO_FLUSH_RXFRAME);
```

enet_dma_feature_disable

The description of enet_dma_feature_disable is shown as below:

Table 3-510. Function enet_dma_feature_disable

Function name	enet_dma_feature_disable
Function prototype	void enet_dma_feature_disable(uint32_t enet_periph, uint32_t feature);
Function descriptions	disable DMA feature
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	the feature of DMA mode one or more parameters can be selected which are shown as below
<i>ENET_NO_FLUSH_RXFRAME</i>	RxDMA does not flushes frames function
<i>ENET_SECONDFRAME_E_OPT</i>	TxDMA controller operate on second frame function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RxDMA does not flushes frames function */
enet_dma_feature_disable(ENET0, ENET_NO_FLUSH_RXFRAME);
```

enet_rx_desc_enhanced_status_get

The description of enet_rx_desc_enhanced_status_get is shown as below:

Table 3-511. Function enet_rx_desc_enhanced_status_get

Function name	enet_rx_desc_enhanced_status_get
Function prototype	uint32_t enet_rx_desc_enhanced_status_get(enet_descriptors_struct *desc, uint32_t desc_status);
Function descriptions	get the bit of extended status flag in ENET DMA descriptor
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to get the extended status flag, the structure members can refer to Table 3-433. Structure

	<u>enet descriptors struct</u>
Input parameter{in}	
desc_status	desc_status: the extended status want to get only one parameter can be selected which is shown as below
<i>ENET_RDES4_IPPLDT</i>	IP frame payload type
<i>ENET_RDES4_IPHER</i> <i>R</i>	IP frame header error
<i>ENET_RDES4_IPPLDE</i> <i>RR</i>	IP frame payload error
<i>ENET_RDES4_IPCKS</i> <i>B</i>	IP frame checksum bypassed
<i>ENET_RDES4_IPF4</i>	IP frame in version 4
<i>ENET_RDES4_IPF6</i>	IP frame in version 6
<i>ENET_RDES4_PTPMT</i>	PTP message type
<i>ENET_RDES4_PTPOE</i> <i>F</i>	PTP on ethernet frame
<i>ENET_RDES4_PTPVF</i>	PTP version format
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* get the IP frame payload type in ENET DMA descriptor */
```

```
uint32_t status;
```

```
status = enet_rx_desc_enhanced_status_get(p_rxdesc, ENET_RDES4_IPPLDT);
```

enet_desc_select_enhanced_mode

The description of enet_desc_select_enhanced_mode is shown as below:

Table 3-512. Function enet_desc_select_enhanced_mode

Function name	enet_desc_select_enhanced_mode
Function prototype	void enet_desc_select_enhanced_mode(uint32_t enet_periph);
Function descriptions	configure descriptor to work in enhanced mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* configure descriptor to work in enhanced mode */
```

```
enet_desc_select_enhanced_mode(ENET0);
```

enet_ptp_enhanced_descriptors_chain_init

The description of enet_ptp_enhanced_descriptors_chain_init is shown as below:

Table 3-513. Function enet_ptp_enhanced_descriptors_chain_init

Function name	enet_ptp_enhanced_descriptors_chain_init
Function prototype	void enet_ptp_enhanced_descriptors_chain_init(uint32_t enet_periph, enet_dmadirection_enum direction);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in enhanced chain mode with ptp function
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
direction	the descriptors which users want to init, only one parameter can be selected which is shown as below
ENET_DMA_TX	DMA Tx descriptors
ENET_DMA_RX	DMA Rx descriptors
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the DMA Tx descriptors's parameters in enhanced chain mode with ptp function */
```

```
enet_ptp_enhanced_descriptors_chain_init(ENET0, ENET_DMA_TX);
```

enet_ptp_enhanced_descriptors_ring_init

The description of enet_ptp_enhanced_descriptors_ring_init is shown as below:

Table 3-514. Function enet_ptp_enhanced_descriptors_ring_init

Function name	enet_ptp_enhanced_descriptors_ring_init
----------------------	---

Function prototype	void enet_ptp_enhanced_descriptors_ring_init(uint32_t enet_periph, enet_dmadirection_enum direction);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in enhanced ring mode with ptp function
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
direction	the descriptors which users want to init, only one parameter can be selected which is shown as below
<i>ENET_DMA_TX</i>	DMA Tx descriptors
<i>ENET_DMA_RX</i>	DMA Rx descriptors
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the DMA Rx descriptors's parameters in enhanced ring mode with ptp function */
enet_ptp_enhanced_descriptors_ring_init(ENET0, ENET_DMA_RX);
```

enet_ptpframe_receive_enhanced_mode

The description of enet_ptpframe_receive_enhanced_mode is shown as below:

Table 3-515. Function enet_ptpframe_receive_enhanced_mode

Function name	enet_ptpframe_receive_enhanced_mode
Function prototype	ErrStatus enet_ptpframe_receive_enhanced_mode(uint32_t enet_periph, uint8_t buffer[], uint32_t bufsize, uint32_t timestamp[]);
Function descriptions	receive a packet data with timestamp values to application buffer, when the DMA is in enhanced mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
bufsize	the size of buffer which is the parameter in function
Output parameter{out}	

buffer	pointer to the application buffer
Output parameter{out}	
timestamp	pointer to the table which stores the timestamp high and low
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* receive a packet data with timestamp values to application buffer in DMA enhanced mode
*/
```

```
uint32_t rx_buffer[500];
```

```
uint32_t time_stamp[2];
```

```
ErrStatus status;
```

```
status = enet_ptpframe_receive_enhanced_mode (ENET0, rx_buffer, 500, time_stamp);
```

enet_ptpframe_transmit_enhanced_mode

The description of enet_ptpframe_transmit_enhanced_mode is shown as below:

Table 3-516. Function enet_ptpframe_transmit_enhanced_mode

Function name	enet_ptpframe_transmit_enhanced_mode
Function prototype	ErrStatus enet_ptpframe_transmit_enhanced_mode(uint32_t enet_periph, uint8_t buffer[], uint32_t length, uint32_t timestamp[]);
Function descriptions	send data with timestamp values in application buffer as a transmit packet, when the DMA is in enhanced mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
buffer	pointer on the application buffer note -- if the input is NULL, user should copy data in application by himself
Input parameter{in}	
length	the length of frame data to be transmitted
Output parameter{out}	
timestamp	pointer to the table which stores the timestamp high and low note -- if the input is NULL, timestamp is ignored
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* send data and timestamp values in application buffer as a transmit packet with DMA
enhanced mode */
```

```
uint32_t tx_buffer[500];
```

```
uint32_t time_stamp[2];
```

```
ErrStatus status;
```

```
status = enet_ptpframe_transmit_enhanced_mode(ENET0, tx_buffer, 500, time_stamp);
```

enet_desc_select_normal_mode

The description of enet_desc_select_normal_mode is shown as below:

Table 3-517. Function enet_desc_select_normal_mode

Function name	enet_desc_select_normal_mode
Function prototype	void enet_desc_select_normal_mode(uint32_t enet_periph);
Function descriptions	configure descriptor to work in normal mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure descriptor to work in normal mode */
```

```
enet_desc_select_normal_mode(ENET0);
```

enet_ptp_normal_descriptors_chain_init

The description of enet_ptp_normal_descriptors_chain_init is shown as below:

Table 3-518. Function enet_ptp_normal_descriptors_chain_init

Function name	enet_ptp_normal_descriptors_chain_init
Function prototype	void enet_ptp_normal_descriptors_chain_init(uint32_t enet_periph, enet_dmadirection_enum direction, enet_descriptors_struct *desc_ptptab);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in normal chain mode with PTP function
Precondition	-
The called functions	-

Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
direction	the descriptors which users want to init only one parameter can be selected which is shown as below
<i>ENET_DMA_TX</i>	DMA Tx descriptors
<i>ENET_DMA_RX</i>	DMA Rx descriptors
Input parameter{in}	
desc_ptptab	pointer to the first descriptor address of PTP Rx descriptor table, the structure members can refer to Table 3-433. Structure <i>enet_descriptors_struct</i>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the DMA Rx descriptors's parameters in normal chain mode with PTP function */
enet_descriptors_struct ptp_rxdesc_tab[ENET_RXBUF_NUM];
enet_ptp_normal_descriptors_chain_init(ENET0, ENET_DMA_RX, ptp_rxdesc_tab);
```

enet_ptp_normal_descriptors_ring_init

The description of enet_ptp_normal_descriptors_ring_init is shown as below:

Table 3-519. Function enet_ptp_normal_descriptors_ring_init

Function name	enet_ptp_normal_descriptors_ring_init
Function prototype	void enet_ptp_normal_descriptors_ring_init(uint32_t enet_periph, enet_dmadirection_enum direction, enet_descriptors_struct *desc_ptptab);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in normal ring mode with PTP function
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
direction	the descriptors which users want to init only one parameter can be selected which is shown as below
<i>ENET_DMA_TX</i>	DMA Tx descriptors

<i>ENET_DMA_RX</i>	DMA Rx descriptors
Input parameter{in}	
desc_ptptab	pointer to the first descriptor address of PTP Rx descriptor table, the structure members can refer to Table 3-433. Structure <i>enet_descriptors_struct</i>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the DMA Rx descriptors's parameters in normal ring mode with PTP function */
enet_descriptors_struct ptp_rxdesc_tab[ENET_RXBUF_NUM];

enet_ptp_normal_descriptors_ring_init(ENET0, ENET_DMA_RX, ptp_rxdesc_tab);

```

enet_ptpframe_receive_normal_mode

The description of enet_ptpframe_receive_normal_mode is shown as below:

Table 3-520. Function enet_ptpframe_receive_normal_mode

Function name	enet_ptpframe_receive_normal_mode
Function prototype	ErrStatus enet_ptpframe_receive_normal_mode(uint32_t enet_periph, uint8_t buffer[], uint32_t bufsize, uint32_t timestamp[]);
Function descriptions	receive a packet data with timestamp values to application buffer, when the DMA is in normal mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
bufsize	the size of buffer which is the parameter in function
Output parameter{out}	
timestamp	pointer to the table which stores the timestamp high and low
Output parameter{out}	
buffer	pointer to the application buffer if the input is NULL, user should copy data in application by himself
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```

/* receive a packet data with timestamp values to application buffer in DMA normal mode */

```

```
uint32_t rx_buffer[500];

uint32_t time_stamp[2];

ErrStatus status;

status = enet_ptpframe_receive_normal_mode(ENET0, rx_buffer, 500, time_stamp);
```

enet_ptpframe_transmit_normal_mode

The description of enet_ptpframe_transmit_normal_mode is shown as below:

Table 3-521. Function enet_ptpframe_transmit_normal_mode

Function name	enet_ptpframe_transmit_normal_mode
Function prototype	ErrStatus enet_ptpframe_transmit_normal_mode(uint32_t enet_periph, uint8_t buffer[], uint32_t length, uint32_t timestamp[]);
Function descriptions	send data with timestamp values in application buffer as a transmit packet, when the DMA is in normal mode
Precondition	-
The called functions	--
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
buffer	pointer on the application buffer if the input is NULL, user should copy data in application by himself
Input parameter{in}	
length	the length of frame data to be transmitted
Output parameter{out}	
timestamp	pointer to the table which stores the timestamp high and low if the input is NULL, timestamp is ignored
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* send data and timestamp values in application buffer as a transmit packet with DMA normal mode */

uint32_t tx_buffer[500];

uint32_t time_stamp[2];

ErrStatus status;

status = enet_ptpframe_transmit_normal_mode(ENET0, tx_buffer, 500, time_stamp);
```

enet_wum_filter_register_pointer_reset

The description of enet_wum_filter_register_pointer_reset is shown as below:

Table 3-522. Function enet_wum_filter_register_pointer_reset

Function name	enet_wum_filter_register_pointer_reset
Function prototype	void enet_wum_filter_register_pointer_reset(uint32_t enet_periph);
Function descriptions	wakeup frame filter register pointer reset
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset wakeup frame filter register pointer */
```

```
enet_wum_filter_register_pointer_reset (ENET0);
```

enet_wum_filter_config

The description of enet_wum_filter_config is shown as below:

Table 3-523. Function enet_wum_filter_config

Function name	enet_wum_filter_config
Function prototype	void enet_wum_filter_config(uint32_t enet_periph, uint32_t pdata[]);
Function descriptions	set the remote wakeup frame registers
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
pdata	pointer to buffer data which is written to remote wakeup frame registers (8 words total)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the remote wakeup frame registers */

uint32_t wum_data[8];

enet_wum_filter_config (ENET0, wum_data);
```

enet_wum_feature_enable

The description of enet_wum_feature_enable is shown as below:

Table 3-524. Function enet_wum_feature_enable

Function name	enet_wum_feature_enable
Function prototype	void enet_wum_feature_enable(uint32_t enet_periph, uint32_t feature);
Function descriptions	enable wakeup management features
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
feature	one or more parameters can be selected which are shown as below
ENET_WUM_POWER_DOWN	power down mode
ENET_WUM_MAGIC_PACKET_FRAME	enable a wakeup event due to magic packet reception
ENET_WUM_WAKEUP_FRAME	enable a wakeup event due to wakeup frame reception
ENET_WUM_GLOBAL_UNICAST	any received unicast frame passed filter is considered to be a wakeup frame
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable power down mode */

enet_wum_feature_enable(ENET0, ENET_WUM_POWER_DOWN);
```

enet_wum_feature_disable

The description of enet_wum_feature_disable is shown as below:

Table 3-525. Function `enet_wum_feature_disable`

Function name	<code>enet_wum_feature_disable</code>
Function prototype	<code>void enet_wum_feature_disable(uint32_t enet_periph, uint32_t feature);</code>
Function descriptions	disable wakeup management features
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	one or more parameters can be selected which are shown as below
<i>ENET_WUM_MAGIC_PACKET_FRAME</i>	enable a wakeup event due to magic packet reception
<i>ENET_WUM_WAKEUP_FRAME</i>	enable a wakeup event due to wakeup frame reception
<i>ENET_WUM_GLOBAL_UNICAST</i>	any received unicast frame passed filter is considered to be a wakeup frame
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable power down mode */
```

```
enet_wum_feature_disable(ENET0, ENET_WUM_POWER_DOWN);
```

enet_msc_counters_reset

The description of `enet_msc_counters_reset` is shown as below:

Table 3-526. Function `enet_msc_counters_reset`

Function name	<code>enet_msc_counters_reset</code>
Function prototype	<code>void enet_msc_counters_reset(uint32_t enet_periph);</code>
Function descriptions	reset the MAC statistics counters
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* reset the MAC statistics counters */
```

```
enet_msc_counters_reset(ENET0);
```

enet_msc_feature_enable

The description of enet_msc_feature_enable is shown as below:

Table 3-527. Function enet_msc_feature_enable

Function name	enet_msc_feature_enable
Function prototype	void enet_msc_feature_enable(uint32_t enet_periph, uint32_t feature);
Function descriptions	enable the MAC statistics counter features
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
feature	one or more parameters can be selected which are shown as below
ENET_MSC_COUNTER_STOP_ROLLOVER	counter stop rollover
ENET_MSC_COUNTER_RESET_ON_READ	reset on read
ENET_MSC_COUNTER_FREEZE	MSC counter freeze
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable counter stop rollover function */
```

```
enet_msc_feature_enable(ENET0, ENET_MSC_COUNTER_STOP_ROLLOVER);
```

enet_msc_feature_disable

The description of enet_msc_feature_disable is shown as below:

Table 3-528. Function enet_msc_feature_disable

Function name	enet_msc_feature_disable
---------------	--------------------------

Function prototype	void enet_msc_feature_disable(uint32_t enet_periph, uint32_t feature);
Function descriptions	disable the MAC statistics counter features
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	one or more parameters can be selected which are shown as below
<i>ENET_MSC_COUNTER_STOP_ROLLOVER</i>	counter stop rollover
<i>ENET_MSC_COUNTER_RESET_ON_READ</i>	reset on read
<i>ENET_MSC_COUNTER_FREEZE</i>	MSC counter freeze
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable counter stop rollover function */
```

```
enet_msc_feature_disable(ENET0, ENET_MSC_COUNTER_STOP_ROLLOVER);
```

enet_msc_counters_preset_config

The description of enet_msc_counters_preset_config is shown as below:

Table 3-529. Function enet_msc_counters_preset_config

Function name	enet_msc_counters_preset_config
Function prototype	void enet_msc_counters_preset_config(uint32_t enet_periph, enet_msc_preset_enum mode);
Function descriptions	configure MAC statistics counters preset mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
mode	MSC counters preset mode, refer to Table 3-452. Enum enet_msc_preset_enum

	only one parameter can be selected which is shown as below
<i>ENET_MSC_PRESET_NONE</i>	do not preset MSC counter
<i>ENET_MSC_PRESET_HALF</i>	preset all MSC counters to almost-half(0x7FFF FFF0) value
<i>ENET_MSC_PRESET_FULL</i>	preset all MSC counters to almost-full(0xFFFF FFF0) value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* preset all MSC counters to almost-half */
```

```
enet_msc_counters_preset_config (ENET0, ENET_MSC_PRESET_HALF);
```

enet_msc_counters_get

The description of enet_msc_counters_get is shown as below:

Table 3-530. Function enet_msc_counters_get

Function name	enet_msc_counters_get
Function prototype	uint32_t enet_msc_counters_get(uint32_t enet_periph, enet_msc_counter_enum counter);
Function descriptions	get MAC statistics counter
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
counter	MSC counters which is selected, refer to Table 3-441. Enum enet_msc_counter_enum only one parameter can be selected which is shown as below
<i>ENET_MSC_TX_SCCNT</i>	MSC transmitted good frames after a single collision counter
<i>ENET_MSC_TX_MSCCNT</i>	MSC transmitted good frames after more than a single collision counter
<i>ENET_MSC_TX_TGFCNT</i>	MSC transmitted good frames counter
<i>ENET_MSC_RX_RFCECNT</i>	MSC received frames with CRC error counter

<i>ENET_MSC_RX_RFAE</i> <i>CNT</i>	MSC received frames with alignment error counter
<i>ENET_MSC_RX_RGU</i> <i>FCNT</i>	MSC received good unicast frames counter
Output parameter{out}	
-	-
Return value	
uint32_t	the MSC counter value

Example:

```
/* get MSC transmitted good frames after a single collision counter value*/
```

```
uint32_t reval;
```

```
reval = enet_msc_counters_get(ENET0, ENET_MSC_TX_SCCNT);
```

enet_ptp_subsecond_2_nanosecond

The description of enet_ptp_subsecond_2_nanosecond is shown as below:

Table 3-531. Function enet_ptp_subsecond_2_nanosecond

Function name	enet_ptp_subsecond_2_nanosecond
Function prototype	uint32_t enet_ptp_subsecond_2_nanosecond(uint32_t subsecond);
Function descriptions	change subsecond to nanosecond
Precondition	-
The called functions	-
Input parameter{in}	
subsecond	subsecond value
Output parameter{out}	
-	-
Return value	
uint32_t	the nanosecond value

Example:

```
/* change subsecond to nanosecond */
```

```
uint32_t reval;
```

```
reval = enet_ptp_subsecond_2_nanosecond (2);
```

enet_ptp_nanosecond_2_subsecond

The description of enet_ptp_nanosecond_2_subsecond is shown as below:

Table 3-532. Function enet_ptp_nanosecond_2_subsecond

Function name	enet_ptp_nanosecond_2_subsecond
Function prototype	uint32_t enet_ptp_nanosecond_2_subsecond(uint32_t nanosecond);

Function descriptions	change nanosecond to subsecond
Precondition	-
The called functions	-
Input parameter{in}	
nanosecond	Nanosecond value
Output parameter{out}	
-	-
Return value	
uint32_t	the subsecond value

Example:

```
/* change nanosecond to subsecond */
```

```
uint32_t reval;
```

```
reval = enet_ptp_nanosecond_2_subsecond (2);
```

enet_ptp_feature_enable

The description of enet_ptp_feature_enable is shown as below:

Table 3-533. Function enet_ptp_feature_enable

Function name	enet_ptp_feature_enable
Function prototype	void enet_ptp_feature_enable(uint32_t enet_periph, uint32_t feature);
Function descriptions	enable the PTP features
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	the feature of ENET PTP mode one or more parameters can be selected which are shown as below
<i>ENET_RXTX_TIMESTAMP</i>	timestamp function for transmit and receive frames
<i>ENET_PTP_TIMESTAMP_INTERRUPT</i>	timestamp interrupt trigger
<i>ENET_ALL_RX_TIMESTAMP</i>	all received frames are taken snapshot
<i>ENET_NONTYPE_FRAME_SNAPSHOT</i>	take snapshot when received non type frame
<i>ENET_IPV6_FRAME_SNAPSHOT</i>	take snapshot for IPv6 frame
<i>ENET_IPV4_FRAME_SNAPSHOT</i>	take snapshot for IPv4 frame

<i>NAPSHOT</i>	
<i>ENET_PTP_FRAME_USE_MACADDRESS_FILTER</i>	use MAC address1-3 to filter the PTP frame
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable PTP function for all received frames */
```

```
enet_ptp_feature_enable(ENET0, ENET_ALL_RX_TIMESTAMP);
```

enet_ptp_feature_disable

The description of enet_ptp_feature_disable is shown as below:

Table 3-534. Function enet_ptp_feature_disable

Function name	enet_ptp_feature_disable
Function prototype	void enet_ptp_feature_disable(uint32_t enet_periph, uint32_t feature);
Function descriptions	disable the PTP features
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
feature	the feature of ENET PTP mode one or more parameters can be selected which are shown as below
<i>ENET_RXTX_TIMESTAMP</i>	timestamp function for transmit and receive frames
<i>ENET_PTP_TIMESTAMP_INT</i>	timestamp interrupt trigger
<i>ENET_ALL_RX_TIMESTAMP</i>	all received frames are taken snapshot
<i>ENET_NONTYPE_FRAME_SNAPSHOT</i>	take snapshot when received non type frame
<i>ENET_IPV6_FRAME_SNAPSHOT</i>	take snapshot for IPv6 frame
<i>ENET_IPV4_FRAME_SNAPSHOT</i>	take snapshot for IPv4 frame
<i>ENET_PTP_FRAME_USE_MACADDRESS_FILTER</i>	use MAC address1-3 to filter the PTP frame

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

Example:

```
/* disable PTP function for all received frames */
```

```
enet_ptp_feature_disable(ENET0, ENET_ALL_RX_TIMESTAMP);
```

enet_ptp_timestamp_function_config

The description of enet_ptp_timestamp_function_config is shown as below:

Table 3-535. Function enet_ptp_timestamp_function_config

Function name	enet_ptp_timestamp_function_config
Function prototype	ErrStatus enet_ptp_timestamp_function_config(uint32_t enet_periph, enet_ptp_function_enum func);
Function descriptions	configure the PTP timestamp function
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
func	only one parameter can be selected which is shown as below
ENET_CKNT_ORDINARY	type of ordinary clock node type for timestamp
ENET_CKNT_BOUNDARY	type of boundary clock node type for timestamp
ENET_CKNT_END_TO_END	type of end-to-end transparent clock node type for timestamp
ENET_CKNT_PEER_TO_PEER	type of peer-to-peer transparent clock node type for timestamp
ENET_PTP_ADDEND_UPDATE	addend register update
ENET_PTP_SYSTIME_UPDATE	timestamp update
ENET_PTP_SYSTIME_INIT	timestamp initialize
ENET_PTP_FINEMOD	the system timestamp uses the fine method for updating

<i>E</i>	
<i>ENET_PTP_COARSE MODE</i>	the system timestamp uses the coarse method for updating
<i>ENET_SUBSECOND_ DIGITAL_ROLLOVER</i>	digital rollover mode
<i>ENET_SUBSECOND_ BINARY_ROLLOVER</i>	digital rollover mode
<i>ENET_SNOOPING_PT P_VERSION_2</i>	version 2
<i>ENET_SNOOPING_PT P_VERSION_1</i>	version 1
<i>ENET_EVENT_TYPE_ MESSAGES_SNAPSH OT</i>	only event type messages are taken snapshot
<i>ENET_ALL_TYPE_ME SSAGES_SNAPSHOT</i>	all type messages are taken snapshot except announce, management and signaling message
<i>ENET_MASTER_NOD E_MESSAGE_SNAPS HOT</i>	snapshot is only take for master node message
<i>ENET_SLAVE_NODE_ MESSAGE_SNAPSHO T</i>	snapshot is only taken for slave node message
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* config addend register update function */
```

```
ErrStatus reval = ERROR;
```

```
reval = enet_ptp_timestamp_function_config(ENET_PTP_ADDEND_UPDATE);
```

enet_ptp_subsecond_increment_config

The description of enet_ptp_subsecond_increment_config is shown as below:

Table 3-536. Function enet_ptp_subsecond_increment_config

Function name	enet_ptp_subsecond_increment_config
Function prototype	void enet_ptp_subsecond_increment_config(uint32_t enet_periph, uint32_t subsecond);
Function descriptions	configure system time subsecond increment value
Precondition	-
The called functions	-

Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
subsecond	the value will be added to the subsecond value of system time, this value must be between 0 and 0xFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure 0x1F as system time subsecond increment value */
```

```
enet_ptp_subsecond_increment_config(ENET0, 0x1F);
```

enet_ptp_timestamp_addend_config

The description of enet_ptp_timestamp_addend_config is shown as below:

Table 3-537. Function enet_ptp_timestamp_addend_config

Function name	enet_ptp_timestamp_addend_config
Function prototype	void enet_ptp_timestamp_addend_config(uint32_t enet_periph, uint32_t add);
Function descriptions	adjusting the clock frequency only in fine update mode
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
add	the value will be added to the accumulator register to achieve time synchronization (0 – 0xFFFF FFFF)
Output parameter{out}	
-	-
Return value	
-	-

```
/* added 0x1FFF to the accumulator register */
```

```
enet_ptp_timestamp_addend_config(ENET0, 0x1FFF);
```

enet_ptp_timestamp_update_config

The description of enet_ptp_timestamp_update_config is shown as below:

Table 3-538. Function enet_ptp_timestamp_update_config

Function name	enet_ptp_timestamp_update_config
Function prototype	void enet_ptp_timestamp_update_config(uint32_t enet_periph, uint32_t sign, uint32_t second, uint32_t subsecond);
Function descriptions	initialize or add/subtract to second of the system time
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
sign	timestamp update positive or negative sign, only one parameter can be selected which is shown as below
<i>ENET_PTP_ADD_TO_TIME</i>	update value is added to system time
<i>ENET_PTP_SUBSTRACT_FROM_TIME</i>	timestamp update value is subtracted from system time
Input parameter{in}	
second	initializing or adding/subtracting to second of the system time (0 – 0xFFFF FFFF)
Input parameter{in}	
subsecond	the current subsecond of the system time with 0.46 ns accuracy (0 – 0x7FFF FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize system time with timestamp update value */
enet_ptp_timestamp_update_config(ENET0, ENET_PTP_ADD_TO_TIME, 0, 0);
```

enet_ptp_expected_time_config

The description of enet_ptp_expected_time_config is shown as below:

Table 3-539. Function enet_ptp_expected_time_config

Function name	enet_ptp_expected_time_config
Function prototype	void enet_ptp_expected_time_config(uint32_t enet_periph, uint32_t second,

	uint32_t nanosecond);
Function descriptions	configure the expected target time
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
second	the expected target second time (0 – 0xFFFF FFFF)
Input parameter{in}	
nanosecond	the expected target nanosecond time (signed) (0 – 0xFFFF FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the expected target time */
enet_ptp_expected_time_config(ENET0, 2000, 0);
```

enet_ptp_system_time_get

The description of enet_ptp_system_time_get is shown as below:

Table 3-540. Function enet_ptp_system_time_get

Function name	enet_ptp_system_time_get
Function prototype	void enet_ptp_system_time_get(uint32_t enet_periph, enet_ptp_systime_struct *systime_struct);
Function descriptions	get the current system time
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Output parameter{out}	
systime_struct	pointer to a enet_ptp_systime_struct structure which contains parameters of PTP system time, the structure members can refer to Table 3-434. Structure enet_ptp_systime_struct
Return value	
-	-

Example:


```
/* get the current system time */
```

```
enet_ptp_systime_struct systime;
```

```
enet_ptp_system_time_get(ENET0, &systime);
```

enet_ptp_pps_output_frequency_config

The description of enet_ptp_pps_output_frequency_config is shown as below:

Table 3-541. Function enet_ptp_pps_output_frequency_config

Function name	enet_ptp_pps_output_frequency_config
Function prototype	void enet_ptp_pps_output_frequency_config(uint32_t enet_periph, uint32_t freq);
Function descriptions	configure the PPS output frequency
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
freq	frequency
ENET_PPSOFC_1HZ	PPS output 1Hz frequency
ENET_PPSOFC_2HZ	PPS output 2Hz frequency
ENET_PPSOFC_4HZ	PPS output 4Hz frequency
ENET_PPSOFC_8HZ	PPS output 8Hz frequency
ENET_PPSOFC_16HZ	PPS output 16Hz frequency
ENET_PPSOFC_32HZ	PPS output 32Hz frequency
ENET_PPSOFC_64HZ	PPS output 64Hz frequency
ENET_PPSOFC_128HZ	PPS output 128Hz frequency
ENET_PPSOFC_256HZ	PPS output 256Hz frequency
ENET_PPSOFC_512HZ	PPS output 512Hz frequency
ENET_PPSOFC_1024HZ	PPS output 1024Hz frequency
ENET_PPSOFC_2048HZ	PPS output 2048Hz frequency
ENET_PPSOFC_4096HZ	PPS output 4096Hz frequency
ENET_PPSOFC_8192HZ	PPS output 8192Hz frequency
ENET_PPSOFC_16384HZ	PPS output 16384Hz frequency

<i>HZ</i>	
<i>ENET_PPISOFC_32768</i>	PPS output 32768Hz frequency
<i>HZ</i>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PPS output frequency as 1Hz */
```

```
enet_ptp_pps_output_frequency_config(ENET0, ENET_PPISOFC_1HZ);
```

enet_ptp_start

The enet_ptp_start is shown as below:

Table 3-542. Function enet_ptp_start

Function name	enet_ptp_start
Function prototype	void enet_ptp_start(uint32_t enet_periph, int32_t updatemethod, uint32_t init_sec, uint32_t init_subsec, uint32_t carry_cfg, uint32_t accuracy_cfg);
Function descriptions	configure and start PTP timestamp counter
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
<i>ENET0</i>	ENET peripheral 0
<i>ENET1</i>	ENET peripheral 1
Input parameter{in}	
updatemethod	method for updating
<i>ENET_PTP_FINEMODE</i>	fine correction method
<i>ENET_PTP_COARSEMODE</i>	coarse correction method
Input parameter{in}	
init_sec	second value for initializing system time
Input parameter{in}	
init_subsec	subsecond value for initializing system time
Input parameter{in}	
carry_cfg	the value to be added to the accumulator register (in fine method is used)
Input parameter{in}	
accuracy_cfg	the value to be added to the subsecond value of system time
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* gconfigure and start PTP timestamp counter*/
```

```
enet_ptp_start(ENET0, ENET_PTP_FINEMODE, 10, 10, 10, 10);
```

enet_ptp_finecorrection_adjfreq

The enet_ptp_finecorrection_adjfreq is shown as below:

Table 3-543. Function enet_ptp_finecorrection_adjfreq

Function name	enet_ptp_finecorrection_adjfreq
Function prototype	void enet_ptp_finecorrection_adjfreq(uint32_t enet_periph, int32_t carry_cfg);
Function descriptions	adjust frequency in fine method by configure addend register
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
carry_cfg	the value to be added to the accumulator register
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* adjust frequency in fine method by configure addend register */
```

```
enet_ptp_finecorrection_adjfreq(ENET0, 10);
```

enet_ptp_coarsecorrection_systime_update

The enet_ptp_coarsecorrection_systime_update is shown as below:

Table 3-544. Function enet_ptp_coarsecorrection_systime_update

Function name	enet_ptp_coarsecorrection_systime_update
Function prototype	void enet_ptp_coarsecorrection_systime_update(uint32_t enet_periph, enet_ptp_systime_struct *systime_struct);
Function descriptions	update system time in coarse method
Precondition	-

The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
systime_struct	the descriptor pointer which users want to configure, the structure members can refer to Table 3-434. Structure enet_ptp_systime_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* update system time in coarse method */
enet_ptp_systime_struct systime_struct;
enet_ptp_coarsecorrection_systime_update (ENET0, &systime_struct);
```

enet_ptp_finecorrection_settime

The enet_ptp_finecorrection_settime is shown as below:

Table 3-545. Function enet_ptp_finecorrection_settime

Function name	enet_ptp_finecorrection_settime
Function prototype	void enet_ptp_finecorrection_settime(uint32_t enet_periph, enet_ptp_systime_struct * systime_struct);
Function descriptions	set system time in fine method
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
systime_struct	the descriptor pointer which users want to configure, the structure members can refer to Table 3-434. Structure enet_ptp_systime_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set system time in fine method */
```

```
enet_ptp_systime_struct systime_struct;
```

```
enet_ptp_finecorrection_settime (ENET0, &systime_struct);
```

enet_ptp_flag_get

The enet_ptp_flag_get is shown as below:

Table 3-546. Function enet_ptp_flag_get

Function name	enet_ptp_flag_get
Function prototype	FlagStatus enet_ptp_flag_get(uint32_t enet_periph, uint32_t flag);
Function descriptions	get the ptp flag status
Precondition	-
The called functions	-
Input parameter{in}	
enet_periph	ENET peripheral
ENET0	ENET peripheral 0
ENET1	ENET peripheral 1
Input parameter{in}	
flag	ptp flag status to be checked
ENET_PTP_ADDEND_UPDATE	addend register update
ENET_PTP_SYSTIME_UPDATE	timestamp update
ENET_PTP_SYSTIME_INIT	timestamp initialize
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the ptp flag status */
```

```
FlagStatus status = enet_ptp_flag_get(ENET0, ENET_PTP_ADDEND_UPDATE);
```

enet_initpara_reset

The description of enet_initpara_reset is shown as below:

Table 3-547. Function enet_initpara_reset

Function name	enet_initpara_reset
Function prototype	void enet_initpara_reset(void);
Function descriptions	reset the ENET initpara struct, call it before using enet_initpara_config()
Precondition	-

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

Example:

```
/* reset the ENET initpara struct */
enet_initpara_reset();
```

3.16. 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 [3.16.1](#), the EXMC firmware functions are introduced in chapter [3.16.2](#).

3.16.1. Descriptions of Peripheral registers

EXMC registers are listed in the table shown as below:

Table 3-548. 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

3.16.2. Descriptions of Peripheral functions

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

Table 3-549. 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 3-550. 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 3-551. 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 3-552. 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

Member name	Function description
setuptime	configure the address setup time

Structure exmc_nand_parameter_struct

Table 3-553. 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 3-554. 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 3-555. 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
sdclock_config	the SDCLK memory clock for both SDRAM banks

Member name	Function description
write_protection	enable or disable SDRAM bank write protection function
cas_latency	configure the SDRAM CAS latency
internal_bank_number	the number of internal bank
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 3-556. 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 3-557. Function `exmc_norsram_deinit`

Function name	<code>exmc_norsram_deinit</code>
Function prototype	<code>void exmc_norsram_deinit(uint32_t exmc_norsram_region);</code>
Function descriptions	deinitialize EXMC NOR/SRAM region
Precondition	-
The called functions	-
Input parameter{in}	
<code>exmc_norsram_region</code>	EXMC NOR/SRAM region
<code>EXMC_BANK0_NORSRAM_REGIONx</code>	x=0,1,2,3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the EXMC NOR/SRAM region1 of bank0 */
```

```
exmc_norsram_deinit(EXMC_BANK0_NORSRAM_REGION1);
```

exmc_norsram_struct_para_init

The description of exmc_norsram_struct_para_init is shown as below:

Table 3-558. Function exmc_norsram_struct_para_init

Function name	exmc_norsram_struct_para_init
Function prototype	void exmc_norsram_struct_para_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);
Function descriptions	initialize the struct exmc_norsram_parameter_struct
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
exmc_norsram_init_struct	Structure for initialization, the structure members can refer to Table 3-551. Structure exmc_norsram_parameter_struct
Return value	
-	-

Example:

```
/* initialize the struct nor_init_struct */
exmc_norsram_parameter_struct nor_init_struct;
exmc_norsram_struct_para_init(&nor_init_struct);
```

exmc_norsram_init

The description of exmc_norsram_init is shown as below:

Table 3-559. Function exmc_norsram_init

Function name	exmc_norsram_init
Function prototype	void exmc_norsram_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);
Function descriptions	initialize EXMC NOR/SRAM region
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
exmc_norsram_init_struct	Structure for initialization, the structure members can refer to Table 3-551. Structure exmc_norsram_parameter_struct
Return value	
-	-

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_ASYNC_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 3-560. Function exmc_norsram_enable

Function name	exmc_norsram_enable
Function prototype	void exmc_norsram_enable(uint32_t exmc_norsram_region);
Function descriptions	enable EXMC NOR/PSRAM region
Precondition	-
The called functions	-
Input parameter{in}	
exmc_norsram_region	EXMC NOR/SRAM region
<i>EXMC_BANK0_NORSRAM_REGIONx</i>	x=0,1,2,3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the EXMC NOR/SRAM region1 of bank0 */
exmc_norsram_enable(EXMC_BANK0_NORSRAM_REGION1);
```

exmc_norsram_disable

The description of exmc_norsram_disable is shown as below:

Table 3-561. Function exmc_norsram_disable

Function name	exmc_norsram_disable
Function prototype	void exmc_norsram_disable(uint32_t exmc_norsram_region);
Function descriptions	disable EXMC NOR/PSRAM region
Precondition	-
The called functions	-
Input parameter{in}	
exmc_norsram_region	EXMC NOR/SRAM region
<i>EXMC_BANK0_NORSRAM_REGIONx</i>	x=0,1,2,3
Output parameter{out}	
-	-
Return value	

-	-
---	---

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 3-562. Function exmc_nand_deinit

Function name	exmc_nand_deinit
Function prototype	void exmc_nand_deinit(void);
Function descriptions	deinitialize EXMC NAND bank
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-563. Function exmc_nand_struct_para_init

Function name	exmc_nand_struct_para_init
Function prototype	void exmc_nand_struct_para_init(exmc_nand_parameter_struct* exmc_nand_init_struct);
Function descriptions	initialize the struct exmc_nand_parameter_struct
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
exmc_nand_init_struct	Structure for initialization, the structure members can refer to Table 3-553. Structure exmc_nand_parameter_struct

Return value	
-	-

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 3-564. Function exmc_nand_init

Function name	exmc_nand_init
Function prototype	void exmc_nand_init(exmc_nand_parameter_struct* exmc_nand_init_struct);
Function descriptions	initialize EXMC NAND bank
Precondition	-
The called functions	-
Input parameter{in}	
exmc_nand_init_struct t	Structure for initialization, the structure members can refer to Table 3-553. Structure exmc_nand_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

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 3-565. Function exmc_nand_enable

Function name	exmc_nand_enable
Function prototype	void exmc_nand_enable(void);
Function descriptions	enable EXMC NAND bank
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable EXMC NAND bank2 */

exmc_nand_enable();

```

exmc_nand_disable

The description of exmc_nand_disable is shown as below:

Table 3-566. Function exmc_nand_disable

Function name	exmc_nand_disable
Function prototype	exmc_nand_disable(void);
Function descriptions	disable EXMC NAND bank
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable EXMC NAND bank2 */
exmc_nand_disable();
```

exmc_sdram_deinit

The description of exmc_sdram_deinit is shown as below:

Table 3-567. Function exmc_sdram_deinit

Function name	exmc_sdram_deinit
Function prototype	void exmc_sdram_deinit(uint32_t exmc_sdram_device);
Function descriptions	deinitialize EXMC SDRAM device
Precondition	-
The called functions	-
Input parameter{in}	
exmc_sdram_device	EXMC SDRAM device
EXMC_SDRAM_DEVICE Ex	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

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 3-568. Function exmc_sdram_struct_para_init

Function name	exmc_sdram_struct_para_init
Function prototype	exmc_sdram_struct_para_init(exmc_sdram_parameter_struct* exmc_sdram_init_struct);
Function descriptions	initialize the struct exmc_sdram_parameter_struct
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
exmc_sdram_init_struct	Structure for initialization, the structure members can refer to Table 3-555 . Structure exmc_sdram_parameter_struct
Return value	
-	-

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 3-569. 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 Table 3-555 . Structure exmc_sdram_parameter_struct
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 3-570. 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
<i>EXMC_BANK_REMAP_DEFAULT</i>	default mapping
<i>EXMC_BANK_NORPSRAM_SDRAM_SWAP</i>	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 3-571. Function exmc_norsram_sdram_remap_get

Function name	exmc_norsram_sdram_remap_get
Function prototype	uint32_t exmc_norsram_sdram_remap_get(void);
Function descriptions	get NOR/PSRAM and SDRAM remap configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
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 3-572. Function exmc_norsram_consecutive_clock_config

Function name	exmc_norsram_consecutive_clock_config
----------------------	---------------------------------------

Function prototype	void exmc_norsram_consecutive_clock_config(uint32_t clock_mode);
Function descriptions	configure consecutive clock mode (consecutive clock is only supported in EXMC BANK0 REGION0)
Precondition	-
The called functions	-
Input parameter{in}	
clock_mode	the mode of consecutive clock
EXMC_CLOCK_SYN_MODE	the clock is generated only during synchronous access
EXMC_CLOCK_UNCONDITIONALLY	the clock is generated unconditionally
Output parameter{out}	
-	-
Return value	
-	-

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 3-573. Function exmc_norsram_page_size_config

Function name	exmc_norsram_page_size_config
Function prototype	void exmc_norsram_page_size_config(uint32_t page_size);
Function descriptions	configure CRAM page size
Precondition	-
The called functions	-
Input parameter{in}	
page_size	CRAM page size
EXMC_CRAM_AUTO_SPLIT	the clock is generated only during synchronous access
EXMC_CRAM_PAGE_SIZE_128_BYTES	page size is 128 bytes
EXMC_CRAM_PAGE_SIZE_256_BYTES	page size is 256 bytes
EXMC_CRAM_PAGE_SIZE_512_BYTES	page size is 512 bytes
EXMC_CRAM_PAGE_SIZE_1024_BYTES	page size is 1024 bytes
Output parameter{out}	

-	-
Return value	
-	-

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 3-574. Function exmc_nand_ecc_config

Function name	exmc_nand_ecc_config
Function prototype	void exmc_nand_ecc_config(ControlStatus newvalue);
Function descriptions	enable or disable the EXMC NAND ECC function
Precondition	-
The called functions	-
Input parameter{in}	
newvalue	ENABLE or DISABLE
Output parameter{out}	
-	-
Return value	
-	-

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 3-575. Function exmc_ecc_get

Function name	exmc_ecc_get
Function prototype	uint32_t exmc_ecc_get(void);
Function descriptions	get the EXMC ECC value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
uint32_t	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 3-576. Function exmc_sdram_readsample_enable

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

Example:

```
/* enable read sample */
```

```
exmc_sdram_readsample_enable();
```

exmc_sdram_readsample_disable

The description of exmc_sdram_readsample_disable is shown as below:

Table 3-577. Function exmc_sdram_readsample_disable

Function name	exmc_sdram_readsample_disable
Function prototype	void exmc_sdram_readsample_disable(void);
Function descriptions	disable read sample function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* disable read sample */
```

```
exmc_sdram_readsample_disable();
```

exmc_sdram_readsample_config

The description of exmc_sdram_readsample_config is shown as below:

Table 3-578. Function exmc_sdram_readsample_config

Function name	exmc_sdram_readsample_config
Function prototype	void exmc_sdram_readsample_config(uint32_t delay_cell, uint32_t extra_clk);
Function descriptions	configure the delayed sample clock of read data
Precondition	-
The called functions	-
Input parameter{in}	
delay_cell	SDRAM the delayed sample clock of read data
EXMC_SDRAM_x_DEL AY_CELL	x=0...15
Input parameter{in}	
extra_clk	sample cycle of read data
EXMC_SDRAM_READ SAMPLE_x_EXTRACK	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 3-579. Function exmc_sdram_command_config

Function name	exmc_sdram_command_config
Function prototype	void exmc_sdram_command_config(exmc_sdram_command_parameter_struct*)

	exmc_sdram_command_init_struct);
Function descriptions	configure the SDRAM memory command
Precondition	-
The called functions	-
Input parameter{in}	
exmc_sdram_command_init_struct	Structure for initialization, the structure members can refer to Table 3-556. Structure exmc_sdram_command_parameter_struct
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 3-580. Function exmc_sdram_refresh_count_set

Function name	exmc_sdram_refresh_count_set
Function prototype	void exmc_sdram_refresh_count_set(uint32_t exmc_count);
Function descriptions	set auto-refresh interval
Precondition	-
The called functions	-
Input parameter{in}	
exmc_count	the number SDRAM clock cycles unit between two successive auto-refresh commands, 0x0000~0x1FFF
Output parameter{out}	
-	-
Return value	
-	-

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 3-581. Function exmc_sdram_autorefresh_number_set

Function name	exmc_sdram_autorefresh_number_set
Function prototype	void exmc_sdram_autorefresh_number_set(uint32_t exmc_number);
Function descriptions	set the number of successive auto-refresh command
Precondition	-
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 3-582. 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
<i>EXMC_SDRAM_DEVICE</i> 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 3-583. Function exmc_sdram_bankstatus_get

Function name	exmc_sdram_bankstatus_get
Function prototype	uint32_t exmc_sdram_bankstatus_get(uint32_t exmc_sdram_device);
Function descriptions	get the status of SDRAM device0 or device1
Precondition	-
The called functions	-
Input parameter{in}	
exmc_sdram_device	specifie the SDRAM device
<i>EXMC_SDRAM_DEVIC</i> <i>Ex</i>	x=0,1
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-584. Function exmc_flag_get

Function name	exmc_flag_get
Function prototype	FlagStatus exmc_flag_get(uint32_t exmc_bank,uint32_t flag);
Function descriptions	get EXMC flag status
Precondition	-
The called functions	-
Input parameter{in}	
exmc_bank	specifies the NAND bank or SDRAM device

<i>EXMC_BANK2_NAND</i>	the NAND bank2
<i>EXMC_SDRAM_DEVIC E0</i>	the SDRAM device0
<i>EXMC_SDRAM_DEVIC E1</i>	the SDRAM device1
Input parameter{in}	
flag	specify get which flag
<i>EXMC_NAND_FLAG_L EVEL</i>	interrupt high-level status
<i>EXMC_NAND_FLAG_ RISE</i>	interrupt rising edge status
<i>EXMC_NAND_FLAG_F ALL</i>	interrupt falling edge status
<i>EXMC_SDRAM_FLAG _REFRESH</i>	refresh error interrupt flag
<i>EXMC_SDRAM_FLAG _NREADY</i>	not ready status
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-585. Function exmc_flag_clear

Function name	exmc_flag_clear
Function prototype	FlagStatus exmc_flag_clear (uint32_t exmc_bank,uint32_t flag);
Function descriptions	clear EXMC flag status
Precondition	-
The called functions	-
Input parameter{in}	
exmc_bank	specifies the NAND bank or SDRAM device
<i>EXMC_BANK2_NAND</i>	the NAND bank2
<i>EXMC_SDRAM_DEVIC E0</i>	the SDRAM device0
<i>EXMC_SDRAM_DEVIC E1</i>	the SDRAM device1

Input parameter{in}	
flag	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_FALLING</i>	interrupt falling edge status
<i>EXMC_SDRAM_FLAG_REFRESH</i>	refresh error interrupt flag
<i>EXMC_SDRAM_FLAG_NREADY</i>	not ready status
Output parameter{out}	
-	-
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 3-586. 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}	
exmc_bank	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}	
interrupt	interrupt
<i>EXMC_NAND_INTERRUPT_FLAG_LEVEL</i>	high-level interrupt
<i>EXMC_NAND_INTERRUPT_FLAG_RISE</i>	rising edge interrupt

AG_RISE	
EXMC_NAND_INT_FL	falling edge interrupt
AG_FALL	
EXMC_SDRAM_INT_F	refresh error interrupt
LAG_REFRESH	
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 3-587. Function exmc_interrupt_disable

Function name	exmc_interrupt_disable
Function prototype	void exmc_interrupt_disable(uint32_t exmc_bank, uint32_t interrupt);
Function descriptions	disable EXMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
exmc_bank	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
Input parameter{in}	
interrupt	interrupt
EXMC_NAND_INT_FL	high-level interrupt
AG_LEVEL	
EXMC_NAND_INT_FL	rising edge interrupt
AG_RISE	
EXMC_NAND_INT_FL	falling edge interrupt
AG_FALL	
EXMC_SDRAM_INT_F	refresh error interrupt
LAG_REFRESH	
Output parameter{out}	
-	-

Return value	
-	-

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 3-588. Function exmc_interrupt_flag_get

Function name	exmc_interrupt_flag_get
Function prototype	FlagStatus exmc_interrupt_flag_get(uint32_t exmc_bank, uint32_t interrupt);
Function descriptions	get EXMC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
exmc_bank	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}	
interrupt	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
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-589. Function exmc_interrupt_flag_clear

Function name	exmc_interrupt_flag_clear
Function prototype	void exmc_interrupt_flag_clear(uint32_t exmc_bank, uint32_t interrupt);
Function descriptions	clear EXMC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
exmc_bank	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
Input parameter{in}	
interrupt	specify get which interrupt flag
EXMC_NAND_INT_FLAG_LEVEL	high-level interrupt and flag
EXMC_NAND_INT_FLAG_RISE	rising edge interrupt and flag
EXMC_NAND_INT_FLAG_FALL	falling edge interrupt and flag
EXMC_SDRAM_INT_FLAG_REFRESH	refresh error interrupt and flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear EXMC interrupt flag */
exmc_interrupt_flag_clear(EXMC_BANK2_NAND, EXMC_NAND_INT_FLAG_RISE);
```

3.17. 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 [3.17.1](#), the EXTI firmware functions are introduced in chapter [3.17.2](#).

3.17.1. Descriptions of Peripheral registers

EXTI registers are listed in the table shown as below:

Table 3-590. EXTI Registers

Registers	Descriptions
EXTI_INTEN0	interrupt enable register 0
EXTI_EVEN0	event enable register 0
EXTI_RTEN0	rising edge trigger enable register 0
EXTI_FTEN0	falling edge trigger enable register 0
EXTI_SWIEV0	software interrupt event register 0
EXTI_PD0	pending register 0
EXTI_INTEN1	interrupt enable register 1
EXTI_EVEN1	event enable register 1
EXTI_RTEN1	rising edge trigger enable register 1
EXTI_FTEN1	falling edge trigger enable register 1
EXTI_SWIEV1	software interrupt event register 1
EXTI_PD1	pending register 1

3.17.2. Descriptions of Peripheral functions

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

Table 3-591. 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 3-592. Enum exti_line_enum

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

enum name	Function description
EXTI_2	EXTI line 2
EXTI_3	EXTI line 3
EXTI_4	EXTI line 4
EXTI_5	EXTI line 5
EXTI_6	EXTI line 6
EXTI_7	EXTI line 7
EXTI_8	EXTI line 8
EXTI_9	EXTI line 9
EXTI_10	EXTI line 10
EXTI_11	EXTI line 11
EXTI_12	EXTI line 12
EXTI_13	EXTI line 13
EXTI_14	EXTI line 14
EXTI_15	EXTI line 15
EXTI_16	EXTI line 16
EXTI_17	EXTI line 17
EXTI_18	EXTI line 18
EXTI_19	EXTI line 19
EXTI_20	EXTI line 20
EXTI_21	EXTI line 21
EXTI_22	EXTI line 22
EXTI_23	EXTI line 23
EXTI_24	EXTI line 24
EXTI_25	EXTI line 25
EXTI_26	EXTI line 26
EXTI_27	EXTI line 27
EXTI_28	EXTI line 28
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 3-593. Enum exti_mode_enum

enum name	Function description
EXTI_INTERRUPT	EXTI interrupt mode

enum name	Function description
EXTI_EVENT	EXTI event mode

Enum exti_trig_type_enum

Table 3-594. 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 3-595. 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 3-596. Function exti_init

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

linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Input parameter{in}	
mode	EXTI mode, refer to Table 3-593. Enum exti_mode_enum
Input parameter{in}	
trig_type	trigger type, refer to Table 3-594. Enum exti_trig_type_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

exti_interrupt_enable

The description of exti_interrupt_enable is shown as below:

Table 3-597. Function exti_interrupt_enable

Function name	exti_interrupt_enable
Function prototype	void exti_interrupt_enable(exti_line_enum linex);
Function descriptions	enable the interrupts from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_interrupt_enable(EXTI_0);
```

exti_interrupt_disable

The description of exti_interrupt_disable is shown as below:

Table 3-598. Function exti_interrupt_disable

Function name	exti_interrupt_disable
Function prototype	void exti_interrupt_disable(exti_line_enum linex);
Function descriptions	disable the interrupts from EXTI line x

Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_interrupt_disable(EXTI_0);
```

exti_event_enable

The description of exti_event_enable is shown as below:

Table 3-599. Function exti_event_enable

Function name	exti_event_enable
Function prototype	void exti_event_enable(exti_line_enum linex);
Function descriptions	enable the events from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_event_enable(EXTI_0);
```

exti_event_disable

The description of exti_event_disable is shown as below:

Table 3-600. Function exti_event_disable

Function name	exti_event_disable
Function prototype	void exti_event_disable(exti_line_enum linex);
Function descriptions	disable the events from EXTI line x
Precondition	-

The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the events from EXTI line 0 */
exti_event_disable(EXTI_0);
```

exti_software_interrupt_enable

The description of exti_software_interrupt_enable is shown as below:

Table 3-601. Function exti_software_interrupt_enable

Function name	exti_software_interrupt_enable
Function prototype	void exti_software_interrupt_enable(exti_line_enum linex);
Function descriptions	enable the software interrupt event from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable EXTI line 0 software interrupt */
exti_software_interrupt_enable(EXTI_0);
```

exti_software_interrupt_disable

The description of exti_software_interrupt_disable is shown as below:

Table 3-602. Function exti_software_interrupt_disable

Function name	exti_software_interrupt_disable
Function prototype	void exti_software_interrupt_disable(exti_line_enum linex);
Function descriptions	disable the software interrupt event from EXTI line x
Precondition	-
The called functions	-

Input parameter{in}	
linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_software_interrupt_disable(EXTI_0);
```

exti_flag_get

The description of exti_flag_get is shown as below:

Table 3-603. Function exti_flag_get

Function name	exti_flag_get
Function prototype	FlagStatus exti_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

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

exti_flag_clear

The description of exti_flag_clear is shown as below:

Table 3-604. Function exti_flag_clear

Function name	exti_flag_clear
Function prototype	void exti_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	

linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_flag_clear(EXTI_0);
```

exti_interrupt_flag_get

The description of exti_interrupt_flag_get is shown as below:

Table 3-605. Function exti_interrupt_flag_get

Function name	exti_interrupt_flag_get
Function prototype	FlagStatus exti_interrupt_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

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

exti_interrupt_flag_clear

The description of exti_interrupt_flag_clear is shown as below:

Table 3-606. Function exti_interrupt_flag_clear

Function name	exti_interrupt_flag_clear
Function prototype	void exti_interrupt_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-592. Enum exti_line_enum

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear EXTI line 0 interrupt flag status */
exti_interrupt_flag_clear(EXTI_0);
```

3.18. 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 [3.18.1](#), the FAC firmware functions are introduced in chapter [3.18.2](#).

3.18.1. Descriptions of Peripheral registers

FAC registers are listed in the table shown as below:

Table 3-607. FAC Registers

Registers	Descriptions
FAC_X0BCFG	FAC X0 buffer configure register
FAC_X1BCFG	FAC X1 buffer configure register
FAC_YBCFG	FAC Y buffer configure register
FAC_PARACFG	FAC Parameter configure register
FAC_CTL	FAC control register
FAC_STAT	FAC status register
FAC_WDATA	FAC write data register
FAC_RDATA	FAC read data register

3.18.2. Descriptions of Peripheral functions

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

Table 3-608. 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

Function name	Function description
fac_fixed_buffer_preload	FAC preload X0 X1 Y fixed buffer
fac_float_buffer_preload	FAC preload X0 X1 Y float buffer
fac_fixed_data_preload	FAC preload fixed data
fac_float_data_preload	FAC preload float data
fac_reset	FAC reset write and read pointers
fac_clip_config	config the FAC clip feature
fac_float_enable	enable FAC float point format
fac_float_disable	disable FAC float point format
fac_dma_enable	enable the FAC dma
fac_dma_disable	disable the FAC dma
fac_x0_config	FAC config input buffer
fac_x1_config	FAC config coefficient buffer
fac_y_config	FAC config output buffer
fac_function_config	FAC config function execute
fac_start	start the FAC
fac_stop	stop the FAC
fac_finish_calculate	finish the filter calculate
fac_interrupt_enable	enable the FAC interrupt
fac_interrupt_disable	disable the FAC interrupt
fac_interrupt_flag_get	get the FAC interrupt flag status
fac_flag_get	get the FAC flag status
fac_fixed_data_write	FAC write data with fixed point format
fac_fixed_data_read	FAC read data with fixed point format
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 3-609. 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)

Member name	Function description
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 3-610. 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 3-611. 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 3-612. 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 3-613. Function fac_struct_para_init

Function name	fac_struct_para_init
Function prototype	void fac_struct_para_init(fac_parameter_struct* fac_parameter);
Function descriptions	initialize the FAC filter parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
fac_parameter	FAC init parameter struct, the structure members can refer to Structure fac_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize FAC init parameter struct with a default value */
```

```
fac_parameter_struct facconfig;
```

```
fac_struct_para_init(&facconfig);
```

fac_fixed_data_preload_init

The description of fac_fixed_data_preload_init is shown as below:

Table 3-614. Function fac_fixed_data_preload_init

Function name	fac_fixed_data_preload_init
Function prototype	void fac_fixed_data_preload_init(fac_fixed_data_preload_struct *init_struct);

Function descriptions	initialize the FAC fixed data preload parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	FAC fixed data preload init parameter struct, the structure members can refer to Structure fac_fixed_data_preload_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize FAC init parameter struct with a default value */
```

```
fac_fixed_data_preload_struct init_struct;
```

```
fac_fixed_data_preload_init (&init_struct);
```

fac_float_data_preload_init

The description of fac_float_data_preload_init is shown as below:

Table 3-615. Function fac_float_data_preload_init

Function name	fac_float_data_preload_init
Function prototype	void fac_float_data_preload_init(fac_float_data_preload_struct *init_struct);
Function descriptions	initialize the FAC float data preload parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	FAC float data preload init parameter struct, the structure members can refer to Structure fac_float_data_preload_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize FAC init parameter struct with a default value */
```

```
fac_float_data_preload_struct init_struct;
```

```
fac_float_data_preload_init (&init_struct);
```

fac_init

The description of fac_init is shown as below:

Table 3-616. Function fac_init

Function name	fac_init
Function prototype	void fac_init(fac_parameter_struct* fac_parameter);
Function descriptions	Initialize the FAC peripheral
Precondition	-
The called functions	-
Input parameter{in}	
fac_parameter	FAC init parameter struct, the structure members can refer to Structure fac_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

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 3-617. 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 Structure fac_fixed_data_preload_struct .
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 3-618. 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 Structure fac_float_data_preload_struct .
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 3-619. Function fac_float_preload

Function name	fac_fixed_data_preload
Function prototype	void fac_fixed_data_preload(uint8_t size, int16_t *data);
Function descriptions	FAC preload fixed data pointer
Precondition	-
The called functions	-
Input parameter{in}	
size	size of data
Input parameter{in}	
*data	16-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* FAC preload context of the coefficient vector B */
```

```
fac_fixed_data_preload_struct init_struct;
```

```
fac_fixed_data_preload(init_struct->coeffB_size,(init_struct->coeffB_ctx));
```

fac_float_data_preload

The description of fac_float_data_preload is shown as below:

Table 3-620. Function fac_float_data_preload

Function name	fac_float_data_preload
Function prototype	void fac_float_data_preload(uint8_t size, float *data);
Function descriptions	FAC preload float data pointer
Precondition	-
The called functions	-
Input parameter{in}	
size	size of data
Input parameter{in}	
*data	32-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* FAC preload context of the coefficient vector B */
```

```
fac_float_data_preload_struct init_struct;
```

```
fac_float_data_preload (init_struct->coeffB_size,(init_struct->coeffB_ctx));
```

fac_reset

The description of fac_reset is shown as below:

Table 3-621. Function fac_init

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

-	-
---	---

Example:

```
/* FAC reset write and read pointers */
```

```
fac_reset();
```

fac_clip_config

The description of fac_clip_config is shown as below:

Table 3-622. Function fac_clip_config

Function name	fac_clip_config
Function prototype	void fac_clip_config(uint8_t cpmode);
Function descriptions	config the FAC clip feature
Precondition	-
The called functions	-
Input parameter{in}	
cpmode	the state of clip
<i>FAC_CP_ENABLED</i>	enable clip
<i>FAC_CP_DISABLE</i>	disable clip
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config the FAC clip enable */
```

```
fac_clip_config(FAC_CP_ENABLE);
```

fac_float_enable

The description of fac_float_enable is shown as below:

Table 3-623. Function fac_init

Function name	fac_float_enable
Function prototype	void fac_float_enable(void);
Function descriptions	enable FAC float point format
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* enable FAC float point format */
```

```
fac_float_enable ();
```

fac_float_disable

The description of fac_float_disable is shown as below:

Table 3-624. Function fac_float_disable

Function name	fac_float_disable
Function prototype	void fac_float_disable(void);
Function descriptions	disable FAC float point format
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable FAC float point format */
```

```
fac_float_disable ();
```

fac_dma_enable

The description of fac_dma_enable is shown as below:

Table 3-625. Function fac_dma_enable

Function name	fac_dma_enable
Function prototype	void fac_dma_enable(uint32_t dma_req);
Function descriptions	enable the FAC DMA
Precondition	-
The called functions	-
Input parameter{in}	
dma_req	transfer type
FAC_DMA_READ	enable DMA read buffer
FAC_DMA_WRITE	enable DMA write buffer
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* enable the FAC read buffer by DMA */
```

```
fac_dma_enable(FAC_DMA_READ);
```

fac_dma_disable

The description of fac_dma_disable is shown as below:

Table 3-626. Function fac_dma_enable

Function name	fac_dma_disable
Function prototype	void fac_dma_disable(uint32_t dma_req);
Function descriptions	disable the FAC DMA
Precondition	-
The called functions	-
Input parameter{in}	
dma_req	transfer type
FAC_DMA_READ	disable DMA read buffer
FAC_DMA_WRITE	disable DMA write buffer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the FAC read buffer by DMA */
```

```
fac_dma_disable(FAC_DMA_READ);
```

fac_x0_config

The description of fac_x0_config is shown as below:

Table 3-627. Function fac_dma_enable

Function name	fac_x0_config
Function prototype	void fac_x0_config(uint32_t watermark, uint8_t baseaddr, uint8_t bufsize);
Function descriptions	FAC config input buffer
Precondition	-
The called functions	-
Input parameter{in}	
watermark	threshold of input buffer

<i>FAC_THRESHOLD_x</i>	X0 buffer full threshold(x =1,2,4,8)
Input parameter{in}	
baseaddr	base address of input buffer, 0..255
Input parameter{in}	
bufsize	buffer size of input buffer, 0..255
Output parameter{out}	
-	-
Return value	
-	-

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 3-628. Function fac_dma_enable

Function name	fac_x1_config
Function prototype	void fac_x1_config(uint8_t baseaddr, uint8_t bufsize);
Function descriptions	FAC config coefficient buffer
Precondition	-
The called functions	-
Input parameter{in}	
baseaddr	base address of coefficientbuffer, 0..255
Input parameter{in}	
bufsize	buffer size of coefficient buffer, 0..255
Output parameter{out}	
-	-
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 3-629. Function fac_dma_enable

Function name	fac_y_config
----------------------	--------------

Function prototype	void fac_y_config(uint32_t watermark, uint8_t baseaddr, uint8_t bufsize);
Function descriptions	FAC config output buffer
Precondition	-
The called functions	-
Input parameter{in}	
watermark	threshold of output buffer
<i>FAC_THRESHOLD_x</i>	Y buffer empty threshold(x =1,2,4,8)
Input parameter{in}	
baseaddr	base address of outputbuffer, 0..255
Input parameter{in}	
bufsize	buffer size of output buffer, 0..255
Output parameter{out}	
-	-
Return value	
-	-

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 3-630. Function fac_dma_enable

Function name	fac_function_config
Function prototype	void fac_function_config(fac_parameter_struct* fac_parameter);
Function descriptions	FAC config function execute
Precondition	-
The called functions	-
Input parameter{in}	
fac_parameter	FAC init parameter struct, the structure members can refer to Structure fac_parameter_struct .
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 3-631. 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 3-632. 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 3-633. 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 3-634. Function fac_fixed_data_write

Function name	fac_fixed_data_write
Function prototype	void fac_fixed_data_write(int16_t data);
Function descriptions	FAC write data with fixed ponit format
Precondition	-
The called functions	-
Input parameter{in}	
data	16-bit data
Output parameter{out}	
-	-
Return value	
-	-

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 3-635. Function fac_fixed_data_write

Function name	fac_fixed_data_read
Function prototype	int16_t fac_fixed_data_read(void);
Function descriptions	FAC read data with fixed point format
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
int16_t	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 3-636. Function fac_float_data_write

Function name	fac_float_data_write
Function prototype	void fac_float_data_write(float data);
Function descriptions	FAC write data with float point format
Precondition	-
The called functions	-
Input parameter{in}	
data	32-bit data
Output parameter{out}	
-	-
Return value	
-	-

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 3-637. Function fac_float_data_read

Function name	fac_float_data_read
Function prototype	int16_t fac_float_data_read(void);
Function descriptions	FAC read data with float point format
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
float	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 3-638. Function fac_interrupt_enable

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

Output parameter{out}	
-	-
Return value	
-	-

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 3-639. Function fac_interrupt_disable

Function name	fac_interrupt_disable
Function prototype	void fac_interrupt_disable(uint32_t interrupt);
Function descriptions	disable the FAC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-640. Function fac_interrupt_flag_get

Function name	fac_interrupt_flag_get
Function prototype	FlagStatus fac_interrupt_flag_get(uint8_t interrupt);

Function descriptions	get FAC interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	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
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-641. Function fac_flag_get

Function name	fac_flag_get
Function prototype	FlagStatus fac_flag_get(uint32_t flag);
Function descriptions	get FAC flag status
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	FAC interrupt flag status
<i>FAC_FLAG_YBEF</i>	Y buffer empty flag
<i>FAC_FLAG_X0BFF</i>	X0 buffer full flag
<i>FAC_FLAG_OFEF</i>	overflow error flag
<i>FAC_FLAG_UFEF</i>	underflow error flag
<i>FAC_FLAG_STEF</i>	saturation error flag
<i>FAC_FLAG_GSTEF</i>	gain saturation error flag
Output parameter{out}	
-	-
Return value	

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

Example:

```
/* get Y buffer empty flag status */
fac_flag_get(FAC_FLAG_YBEF);
```

3.19. FMC

There is flash controller and option byte. The FMC registers are listed in chapter [3.19.1](#) the FMC firmware functions are introduced in chapter [3.19.2](#).

3.19.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

Table 3-642. 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
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

Registers	Descriptions
T	
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)

3.19.2. Descriptions of Peripheral functions

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

Table 3-643. 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
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

Function name	Function description
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_tcm_shared_ram_size_get	get the option byte TCM 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 3-644. Enum fmc_state_enum

enum name	enum description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_WPERR	erase/program protection error
FMC_PGSEERR	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

enum name	enum description
FMC_OBMERR	option byte modify error
FMC_TOERR	timeout error

Enum fmc_flag_enum

Table 3-645. 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 3-646. 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
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

enum name	enum description
FMC_INT_FLAG_OBMERR	option byte modify error flag

Enum fmc_interrupt_enum

Table 3-647. 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_ECCCOR	FMC one bit correct error flag
FMC_INT_ECCDET	FMC two bits detect error flag
FMC_INT_OBMER	FMC option byte modify error flag

fmc_unlock

The description of fmc_unlock is shown as below:

Table 3-648. Function fmc_unlock

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

Example:

```
/* unlock FMC_CTL register */
fmc_unlock();
```

fmc_lock

The description of fmc_lock is shown as below:

Table 3-649. Function `fmc_lock`

Function name	<code>fmc_lock</code>
Function prototype	<code>void fmc_lock(void);</code>
Function descriptions	lock FMC_CTL register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock FMC_CTL register */
fmc_lock();
```

`fmc_sector_erase`

The description of `fmc_sector_erase` is shown as below:

Table 3-650. Function `fmc_page_erase`

Function name	<code>fmc_sector_erase</code>
Function prototype	<code>fmc_state_enum fmc_sector_erase(uint32_t address);</code>
Function descriptions	FMC erase sector
Precondition	<code>fmc_unlock</code>
The called functions	-
Input parameter{in}	
address	address to erase
Output parameter{out}	
-	-
Return value	
<code>fmc_state_enum</code>	state of FMC, refer to Table 3-644. Enum <code>fmc_state_enum</code>

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 3-651. Function `fmc_mass_erase`

Function name	<code>fmc_typical_mass_erase</code>
Function prototype	<code>fmc_state_enum fmc_typical_mass_erase(void);</code>
Function descriptions	FMC typical mass erase
Precondition	<code>fmc_unlock</code>
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<code>fmc_state_enum</code>	state of FMC, refer to Table 3-644. Enum <code>fmc_state_enum</code>

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 3-652. Function `fmc_protection_removed_mass_erase`

Function name	<code>fmc_protection_removed_mass_erase</code>
Function prototype	<code>fmc_state_enum fmc_protection_removed_mass_erase(void);</code>
Function descriptions	FMC protection-removed mass erase
Precondition	<code>fmc_unlock</code>
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<code>fmc_state_enum</code>	state of FMC, refer to Table 3-644. Enum <code>fmc_state_enum</code>

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 3-653. Function fmc_word_program

Function name	fmc_word_program
Function prototype	fmc_state_enum fmc_word_program(uint32_t address, uint32_t data);
Function descriptions	FMC program a word at the corresponding address
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
address	address to program
Input parameter{in}	
data	word to program
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-654. Function fmc_doubleword_program

Function name	fmc_doubleword_program
Function prototype	fmc_state_enum fmc_doubleword_program(uint32_t address, uint64_t data);
Function descriptions	FMC program a double-word at the corresponding address
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
address	address to program
Input parameter{in}	
data	double word to program
Output parameter{out}	
-	-
Return value	

fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum
-----------------------	---

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 3-655. Function fmc_check_programming_area_enable

Function name	fmc_check_programming_area_enable
Function prototype	fmc_state_enum fmc_check_programming_area_enable(void);
Function descriptions	enable check programming area
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-656. Function fmc_check_programming_area_disable

Function name	fmc_check_programming_area_disable
Function prototype	fmc_state_enum fmc_check_programming_area_disable(void);
Function descriptions	disable check programming area
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-657. 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 3-658. 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 3-659. Function ob_start

Function name	ob_start
Function prototype	fmc_state_enum ob_start(void);
Function descriptions	send option bytes modification start command
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-660. Function ob_factory_value_config

Function name	ob_factory_value_config
---------------	-------------------------

Function prototype	fmc_state_enum ob_factory_value_config(void);
Function descriptions	modify option byte to factory value
Precondition	ob_unlock
The called functions	ob_start
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-661. Function ob_secure_access_mode_enable

Function name	ob_secure_access_mode_enable
Function prototype	fmc_state_enum ob_secure_access_mode_enable(void);
Function descriptions	enable secure access mode
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-662. Function ob_secure_access_mode_disable

Function name	ob_secure_access_mode_disable
Function prototype	fmc_state_enum ob_secure_access_mode_disable(void);
Function descriptions	disable secure access mode
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-663. Function ob_security_protection_config

Function name	ob_security_protection_config
Function prototype	fmc_state_enum ob_security_protection_config(uint8_t ob_spc)
Function descriptions	configure the option byte security protection level
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_spc	specify security protection level
<i>FMC_NSPC</i>	no protection
<i>FMC_LSPC</i>	protection level low
<i>FMC_HSPC</i>	protection level high
Output parameter{out}	
-	-

Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-664. Function ob_bor_threshold_config

Function name	ob_bor_threshold_config
Function prototype	fmc_state_enum ob_bor_threshold_config(uint32_t ob_bor_th);
Function descriptions	configure the option byte BOR threshold value
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_bor_th	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
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-665. Function ob_low_power_config

Function name	ob_low_power_config
Function prototype	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);
Function descriptions	configure low power related option byte
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_fwdgt	option byte watchdog value
<i>OB_FWDGT_SW</i>	software free watchdog
<i>OB_FWDGT_HW</i>	hardware free watchdog
Input parameter{in}	
ob_deepsleep	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
Input parameter{in}	
ob_stdby	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
Input parameter{in}	
ob_fwdg_suspend_deepsleep	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
Input parameter{in}	
ob_fwdg_suspend_standby	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
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-666. Function ob_tcm_ecc_config

Function name	ob_tcm_ecc_config
Function prototype	fmc_state_enum ob_tcm_ecc_config(uint32_t ob_itcmecc, uint32_t ob_dtcmm0ecc, uint32_t ob_dtcmm1ecc);
Function descriptions	configure TCM ECC option byte
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_itcmecc	ITCM ECC function enable bit
OB_ITCMECCEN_DISABLE	disabled ITCM ECC function
OB_ITCMECCEN_ENABLE	enabled ITCM ECC function
Input parameter{in}	
ob_dtcmm0ecc	DTCM0 ECC function enable bit
OB_DTCM0ECCEN_DISABLE	disabled DTCM0 ECC function
OB_DTCM0ECCEN_ENABLE	enabled DTCM0 ECC function
Input parameter{in}	
ob_dtcmm1ecc	DTCM1 ECC function enable bit
OB_DTCM1ECCEN_DISABLE	disabled DTCM1 ECC function
OB_DTCM1ECCEN_ENABLE	enabled DTCM1 ECC function
Output parameter{out}	
-	-
Return value	

fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum
-----------------------	---

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_DTCM0ECCEN_DISABLE, OB_DTCM1ECCEN_DISABLE);
```

ob_iospeed_optimize_config

The description of ob_iospeed_optimize_config is shown as below:

Table 3-667. Function ob_iospeed_optimize_config

Function name	ob_iospeed_optimize_config
Function prototype	fmc_state_enum ob_iospeed_optimize_config(uint32_t ob_iospeed_op);
Function descriptions	configure I/O speed optimization option byte
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_iospeed_op	configure I/O speed optimization, high-speed at low-voltage enable bit
OB_IOSPDOPEN_DISABLE	disabled I/O speed optimization
OB_IOSPDOPEN_ENABLE	enabled I/O speed optimization
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-668. Function ob_tcm_shared_ram_config

Function name	ob_tcm_shared_ram_config
Function prototype	fmc_state_enum ob_tcm_shared_ram_config(uint32_t itcm_shared_ram_size, uint32_t dtcm_shared_ram_size);
Function descriptions	configure option byte TCM shared RAM size
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
itcm_shared_ram_size	ITCM shared RAM size
OB_ITCM_SHARED_RAM_0KB	ITCM shared RAM size is 0KB
OB_ITCM_SHARED_RAM_64KB	ITCM shared RAM size is 64KB
OB_ITCM_SHARED_RAM_128KB	ITCM shared RAM size is 128KB
OB_ITCM_SHARED_RAM_256KB	ITCM shared RAM size is 256KB
OB_ITCM_SHARED_RAM_512KB	ITCM shared RAM size is 512KB
Input parameter{in}	
dtcm_shared_ram_size	DTCM shared RAM size
OB_DTCM_SHARED_RAM_0KB	DTCM shared RAM size is .KB
OB_DTCM_SHARED_RAM_64KB	DTCM shared RAM size is 64KB
OB_DTCM_SHARED_RAM_128KB	DTCM shared RAM size is 128KB
OB_DTCM_SHARED_RAM_256KB	DTCM shared RAM size is 256KB
OB_DTCM_SHARED_RAM_512KB	DTCM shared RAM size is 512KB
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-669. 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 Table 3-644. Enum fmc_state_enum

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 3-670. 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
<i>BOOT_PIN_0</i>	boot pin value is 0
<i>BOOT_PIN_1</i>	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 Table 3-644. Enum fmc_state_enum

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 3-671. Function ob_dcrp_area_config

Function name	ob_dcrp_area_config
Function prototype	fmc_state_enum ob_dcrp_area_config(uint32_t dcrp_eren, uint32_t dcrp_start, uint32_t dcrp_end);
Function descriptions	configure DCRP area
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
dcrp_eren	DCRP area erase enable bit
<i>OB_DCRP_AREA_ERASE_DISABLE</i>	DCRP area erase disable
<i>OB_DCRP_AREA_ERASE_ENABLE</i>	DCRP area erase enable
Input parameter{in}	
dcrp_start	DCRP area start address
Input parameter{in}	
dcrp_end	DCRP area end address
Output parameter{out}	
-	-
Return value	

fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum
-----------------------	---

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 3-672. Function ob_secure_area_config

Function name	ob_secure_area_config
Function prototype	fmc_state_enum ob_secure_area_config(uint32_t scr_eren, uint32_t scr_start, uint32_t scr_end);
Function descriptions	configure secure-access area
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
scr_eren	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
Input parameter{in}	
scr_start	secure-access area start address
Input parameter{in}	
dcrp_end	secure-access area end address
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-673. Function ob_write_protection_enable

Function name	ob_write_protection_enable
Function prototype	fmc_state_enum ob_write_protection_enable(uint32_t ob_wp);
Function descriptions	enable option bytes sector erase/program protection
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_wp	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
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-674. Function ob_write_protection_disable

Function name	ob_write_protection_disable
Function prototype	fmc_state_enum ob_write_protection_disable(uint32_t ob_wp);
Function descriptions	disable option bytes sector erase/program protection
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_wp	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

Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-675. Function ob_secure_mode_get

Function name	ob_secure_mode_get
Function prototype	FlagStatus ob_secure_mode_get(void);
Function descriptions	get the option byte secure access mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-676. Function ob_secure_mode_get

Function name	ob_security_protection_flag_get
Function prototype	FlagStatus ob_security_protection_flag_get(void);
Function descriptions	get the option byte security protection level
Precondition	-
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-677. Function ob_bor_threshold_get

Function name	ob_bor_threshold_get
Function prototype	uint32_t ob_bor_threshold_get(void);
Function descriptions	get the option byte BOR threshold value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	the 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

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 3-678. Function ob_low_power_get

Function name	ob_low_power_get
Function prototype	void ob_low_power_get(uint32_t *fwdgt, uint32_t *deepsleep, uint32_t *standby, uint32_t *fwdg_suspend_deepsleep, uint32_t *fwdg_suspend_standby);

Function descriptions	get low power related option byte
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
fwdgt	watchdog option
<i>OB_FWDGT_SW</i>	software free watchdog
<i>OB_FWDGT_HW</i>	hardware free watchdog
Output parameter{out}	
deepsleep	deepsleep reset option
<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
Output parameter{out}	
standby	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}	
fwdg_suspend_deepsleep	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}	
fwdg_suspend_standby	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, fwgd_suspend_deepsleep_value,
fwgd_suspend_standby_value;
```

```
ob_low_power_get(&fwdgt_value, &deepsleep_value, &standby_value, &fwgd_suspend_d
eepsleep_value, &fwgd_suspend_standby_value);
```

ob_tcm_ecc_get

The description of ob_tcm_ecc_get is shown as below:

Table 3-679. Function ob_tcm_ecc_get

Function name	ob_tcm_ecc_get
Function prototype	void ob_tcm_ecc_get(uint32_t *itcmecc_option, uint32_t *dtcm0ecc_option, uint32_t *dtcm1ecc_option);
Function descriptions	get TCM ECC configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
itcmecc_option	ITCM ECC function enable option
OB_ITCMECCEN_DISABLE	disabled ITCM ECC function
OB_ITCMECCEN_ENABLE	enabled ITCM ECC function
Output parameter{out}	
dtcm0ecc_option	DTCM0 ECC function enable bit
OB_DTCM0ECCEN_DISABLE	disabled DTCM0 ECC function
OB_DTCM0ECCEN_ENABLE	enabled DTCM0 ECC function
Output parameter{out}	
ob_dtcm1ecc	DTCM1 ECC function enable bit
OB_DTCM1ECCEN_DISABLE	disabled DTCM1 ECC function
OB_DTCM1ECCEN_ENABLE	enabled DTCM1 ECC function
Return value	
-	-

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 3-680. Function `ob_iospeed_option_get`

Function name	<code>ob_iospeed_option_get</code>
Function prototype	<code>FlagStatus ob_iospeed_option_get(void);</code>
Function descriptions	get IO speed optimize configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get IO speed optimize configuration */
```

```
FlagStatus flag = ob_iospeed_option_get();
```

`ob_tcm_shared_ram_size_get`

The description of `ob_tcm_shared_ram_size_get` is shown as below:

Table 3-681. Function `ob_tcm_shared_ram_size_get`

Function name	<code>ob_tcm_shared_ram_size_get</code>
Function prototype	<code>void ob_tcm_shared_ram_size_get(uint32_t *itcm_shared_ram_kb_size, uint32_t *dcm_shared_ram_kb_size);</code>
Function descriptions	get the option byte TCM shared RAM size
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
<code>itcm_shared_ram_size</code>	ITCM shared RAM size
<code>OB_ITCM_SHARED_RAM_0KB</code>	ITCM shared RAM size is 0KB
<code>OB_ITCM_SHARED_RAM_64KB</code>	ITCM shared RAM size is 64KB
<code>OB_ITCM_SHARED_RAM_128KB</code>	ITCM shared RAM size is 128KB
<code>OB_ITCM_SHARED_RAM_256KB</code>	ITCM shared RAM size is 256KB
<code>OB_ITCM_SHARED_RAM_512KB</code>	ITCM shared RAM size is 512KB

Output parameter{out}	
dtcm_shared_ram_size	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
Return value	
-	-

Example:

```

/* get TCM shared RAM size */
uint32_t itcm_size, dtcm_size;

ob_tcm_shared_ram_size_get(&itcm_size, &dtcm_size);

```

ob_data_get

The description of ob_data_get is shown as below:

Table 3-682. Function ob_data_get

Function name	ob_data_get
Function prototype	uint16_t ob_data_get(void);
Function descriptions	get user data value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
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 3-683. Function ob_boot_address_get

Function name	ob_boot_address_get
Function prototype	uint32_t ob_boot_address_get(uint8_t boot_pin);
Function descriptions	get boot address
Precondition	-
The called functions	-
Input parameter{in}	
boot_pin	boot pin configuration
<i>BOOT_PIN_0</i>	boot pin value is 0
<i>BOOT_PIN_1</i>	boot pin value is 1
Output parameter{out}	
-	-
Return value	
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 3-684. Function ob_dcrp_area_get

Function name	ob_dcrp_area_get
Function prototype	uint8_t ob_dcrp_area_get(uint32_t *dcrp_erase_option, uint32_t *dcrp_area_start_addr, uint32_t *dcrp_area_end_addr);
Function descriptions	get DCRP area configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
dcrp_erase_option	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
Output parameter{out}	
dcrp_start_addr	DCRP area start address

Output parameter{out}	
dcrp_end_addr	DCRP area end address
Return value	
uint8_t	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 3-685. Function ob_secure_area_get

Function name	ob_secure_area_get
Function prototype	uint8_t ob_secure_area_get(uint32_t *secure_area_option, uint32_t *scr_area_start_addr, uint32_t *scr_area_end_addr);
Function descriptions	get secure-access area configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
secure_erase_option	secure-access area erase option
<i>OB_SCR_AREA_ERAS_E_DISABLE</i>	secure-access area erase disable
<i>OB_SCR_AREA_ERAS_E_ENABLE</i>	secure-access area erase enable
Output parameter{out}	
scr_area_start_addr	secure-access area start address
Output parameter{out}	
scr_area_end_addr	secure-access area end address
Return value	
uint8_t	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 3-686. Function ob_write_protection_get

Function name	ob_write_protection_get
Function prototype	uint32_t ob_write_protection_get(void);
Function descriptions	get the option byte erase/program protection
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-687. Function fmc_no_rtdec_config

Function name	fmc_no_rtdec_config
Function prototype	fmc_state_enum fmc_no_rtdec_config(uint32_t nodec_area_start, uint32_t nodec_area_end);
Function descriptions	configure NO-RTDEC area
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
nodec_area_start	no rtdec area start address
nodec_area_end	no rtdec area end address
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-688. Function fmc_aes_iv_config

Function name	fmc_aes_iv_config
Function prototype	fmc_state_enum fmc_aes_iv_config(uint32_t *aes_iv);
Function descriptions	configure aes initialization vector
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
aes_iv	high 96 bits of AES initialization vector
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-644. Enum fmc_state_enum

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 3-689. Function fmc_flash_ecc_get

Function name	fmc_flash_ecc_get
Function prototype	FlagStatus fmc_flash_ecc_get(void);
Function descriptions	get Flash ECC function enable flag
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-690. Function fmc_no_rtdec_get

Function name	fmc_no_rtdec_get
Function prototype	void fmc_no_rtdec_get(uint32_t *nodec_area_start, uint32_t *nodec_area_end);
Function descriptions	get NO-RTDEC area
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
nodec_area_start	no rtdec area start address
Output parameter{out}	
nodec_area_end	no rtdec area end address
Return value	
-	-

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 3-691. Function fmc_aes_iv_get

Function name	fmc_aes_iv_get
Function prototype	void fmc_aes_iv_get(uint32_t *aes_iv);
Function descriptions	get AES initialization vector
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
aes_iv	high 96 bits of AES initialization vector
Return value	
-	-

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 3-692. Function fmc_pid_get

Function name	fmc_pid_get
Function prototype	void fmc_pid_get(uint32_t *pid);
Function descriptions	get product ID
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
pid	product ID
Return value	
-	-

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 3-693. Function fmc_flag_get

Function name	fmc_flag_get
Function prototype	FlagStatus fmc_flag_get(fmc_flag_enum flag);
Function descriptions	get FMC flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	FMC flag, refer to Table 3-645. Enum fmc_flag_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-694. Function fmc_flag_clear

Function name	fmc_flag_clear
Function prototype	void fmc_flag_clear(fmc_flag_enum flag)
Function descriptions	clear FMC flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	FMC flag, refer to Table 3-645. Enum fmc_flag_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear RPERR flag */
```

```
fmc_flag_clear(FMC_FLAG_RPERR);
```

fmc_interrupt_enable

The description of fmc_interrupt_enable is shown as below:

Table 3-695. Function fmc_interrupt_enable

Function name	fmc_interrupt_enable
Function prototype	void fmc_interrupt_enable(fmc_interrupt_enum interrupt);
Function descriptions	enable FMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	FMC interrupt, refer to Table 3-647. Enum fmc_interrupt_enum
Output parameter{out}	
-	-
Return value	
-	-

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 3-696. Function fmc_interrupt_disable

Function name	fmc_interrupt_disable
Function prototype	void fmc_interrupt_disable(fmc_interrupt_enum interrupt);
Function descriptions	disable FMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	FMC interrupt, refer to Table 3-647. Enum fmc_interrupt_enum
Output parameter{out}	
-	-
Return value	
-	-

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 3-697. Function fmc_interrupt_flag_get

Function name	fmc_interrupt_flag_get
Function prototype	FlagStatus fmc_interrupt_flag_get(fmc_interrupt_flag_enum int_flag);
Function descriptions	get FMC interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
fmc_int_flag	FMC interrupt flag, refer to Table 3-646. Enum fmc_interrupt_flag_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-698. Function fmc_interrupt_flag_clear

Function name	fmc_interrupt_flag_clear
Function prototype	void fmc_interrupt_flag_clear(fmc_interrupt_flag_enum int_flag);
Function descriptions	clear FMC interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	FMC interrupt flag, refer to Table 3-646. Enum fmc_interrupt_flag_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear FMC program sequence error flag */
```

```
fmc_interrupt_flag_get(FMC_INT_FLAG_PGSERR);
```

3.20. 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 [3.20.1](#). The FWDGT firmware functions are introduced in chapter [3.20.2](#).

3.20.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

Table 3-699. FWDGT Registers

Registers	Descriptions
FWDGT_CTL	control register
FWDGT_PSC	prescaler register
FWDGT_RLD	reload register
FWDGT_STAT	status register
FWDGT_WND	window register

3.20.2. Descriptions of Peripheral functions

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

Table 3-700. 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 3-701. 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 3-702. Function fwdgt_write_disable

Function name	fwdgt_write_disable
Function prototype	void fwdgt_write_disable(void);
Function descriptions	disable write access to FWDGT_PSC、FWDGT_RLD and FWDGT_WND
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable write access to FWDGT_PSC、FWDGT_RLD and FWDGT_WND */
fwdgt_write_disable();
```

fwdgt_enable

The description of fwdgt_enable is shown as below:

Table 3-703. Function fwdgt_enable

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

Example:

```
/* start the free watchdog timer counter */
fwdgt_enable();
```

fwdgt_prescaler_value_config

The description of fwdgt_prescaler_value_config is shown as below:

Table 3-704. Function fwdgt_prescaler_value_config

Function name	fwdgt_prescaler_value_config
Function prototype	ErrStatus fwdgt_prescaler_value_config(uint16_t prescaler_value);
Function descriptions	configure the FWDGT counter clock prescaler value
Precondition	-
The called functions	-
Input parameter{in}	
prescaler_value	specify prescaler value
<i>FWDGT_PSC_DIVx</i>	FWDGT prescaler set to x(x=4,8,16,32,64,128,256)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

```
/* set FWDGT prescaler to 4 */
```

```
ErrStatus flag;
```

```
flag = fwdgt_prescaler_value_config(FWDGT_PSC_DIV4);
```

fwdgt_reload_value_config

The description of fwdgt_reload_value_config is shown as below:

Table 3-705. Function fwdgt_reload_value_config

Function name	fwdgt_reload_value_config
Function prototype	ErrStatus fwdgt_reload_value_config(uint16_t reload_value);
Function descriptions	configure the FWDGT counter reload value
Precondition	-
The called functions	-
Input parameter{in}	
reload_value	reload_value, specify reload value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

```
/* set FWDGT reload value to 0xFFF */
```

```
ErrStatus flag;
```

```
flag = fwdgt_reload_value_config(0xFFF);
```

fwdgt_window_value_config

The description of fwdgt_window_value_config is shown as below:

Table 3-706. Function fwdgt_window_value_config

Function name	fwdgt_window_value_config
Function prototype	ErrStatus fwdgt_window_value_config(uint16_t window_value);
Function descriptions	configure the FWDGT counter window value
Precondition	-
The called functions	-
Input parameter{in}	
window_value	window_value, specify window value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

```

/* set FWDGT window value to 0xFFFF */

ErrStatus flag;

flag = fwdgt_window_value_config(0xFFFF);

```

fwdgt_counter_reload

The description of fwdgt_counter_reload is shown as below:

Table 3-707. Function fwdgt_counter_reload

Function name	fwdgt_counter_reload
Function prototype	void fwdgt_counter_reload(void);
Function descriptions	reload the counter of FWDGT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* reload FWDGT counter */

fwdgt_counter_reload();

```

fwdgt_config

The description of fwdgt_config is shown as below:

Table 3-708. Function fwdgt_config

Function name	fwdgt_config
Function prototype	ErrStatus fwdgt_config(uint16_t reload_value, uint8_t prescaler_div);
Function descriptions	configure counter reload value, and prescaler divider value
Precondition	-
The called functions	-
Input parameter{in}	
reload_value	specify reload value(0x0000 - 0x0FFF)
Input parameter{in}	
prescaler_div	FWDGT prescaler value
<i>FWDGT_PSC_DIV4</i>	FWDGT prescaler set to 4
<i>FWDGT_PSC_DIV8</i>	FWDGT prescaler set to 8
<i>FWDGT_PSC_DIV16</i>	FWDGT prescaler set to 16
<i>FWDGT_PSC_DIV32</i>	FWDGT prescaler set to 32
<i>FWDGT_PSC_DIV64</i>	FWDGT prescaler set to 64
<i>FWDGT_PSC_DIV128</i>	FWDGT prescaler set to 128
<i>FWDGT_PSC_DIV256</i>	FWDGT prescaler set to 256
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* 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 3-709. Function fwdgt_flag_get

Function name	fwdgt_flag_get
Function prototype	FlagStatus fwdgt_flag_get(uint16_t flag);
Function descriptions	get flag state of FWDGT
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag to get
<i>FWDGT_FLAG_PUD</i>	a write operation to FWDGT_PSC register is on going
<i>FWDGT_FLAG_RUD</i>	a write operation to FWDGT_RLD register is on going

<i>FWDGT_FLAG_WUD</i>	a write operation to FWDGT_WND register is on going
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* test if a prescaler value update is on going */
```

```
FlagStatus status;
```

```
status = fwdgt_flag_get(FWDGT_FLAG_PUD);
```

3.21. GPIO

GPIO is used to implement logic input/output functions for the devices. The GPIO registers are listed in chapter [3.21.1](#), the GPIO firmware functions are introduced in chapter [3.21.2](#).

3.21.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

Table 3-710. GPIO Registers

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

3.21.2. Descriptions of Peripheral functions

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

Table 3-711. 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 3-712. Function gpio_deinit

Function name	gpio_deinit
Function prototype	void gpio_deinit(uint32_t gpio_periph);
Function descriptions	reset GPIO port
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Output parameter{out}	
-	-
Return value	
-	-

Example:

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


gpio_mode_set

The description of gpio_mode_set is shown as below:

Table 3-713. Function gpio_mode_set

Function name	gpio_mode_set
Function prototype	void gpio_mode_set(uint32_t gpio_periph, uint32_t mode, uint32_t pull_up_down, uint32_t pin);
Function descriptions	set GPIO mode
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Input parameter{in}	
mode	gpio pin mode
<i>GPIO_MODE_INPUT</i>	input mode
<i>GPIO_MODE_OUTPUT</i>	output mode
<i>GPIO_MODE_AF</i>	alternate function mode
<i>GPIO_MODE_ANALOG</i>	analog mode
Input parameter{in}	
pull_up_down	gpio pin with pull-up or pull-down resistor
<i>GPIO_PUPD_NONE</i>	floating mode, no pull-up and pull-down resistors
<i>GPIO_PUPD_PULLUP</i>	with pull-up resistor
<i>GPIO_PUPD_PULLDOWN</i>	with pull-down resistor
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

gpio_output_options_set

The description of gpio_output_options_set is shown as below:

Table 3-714. Function gpio_output_options_set

Function name	gpio_output_options_set
Function prototype	void gpio_output_options_set(uint32_t gpio_periph, uint8_t otype, uint32_t speed, uint32_t pin);
Function descriptions	set GPIO output type and speed
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Input parameter{in}	
otype	gpio pin output mode
<i>GPIO_OTYPE_PP</i>	push pull mode
<i>GPIO_OTYPE_OD</i>	open drain mode
Input parameter{in}	
speed	gpio pin output max speed
<i>GPIO_OSPEED_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_220MHZ</i>	output max speed 100/220MHz
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_12MHZ,
GPIO_PIN_0);
```

gpio_bit_set

The description of gpio_bit_set is shown as below:

Table 3-715. Function gpio_bit_set

Function name	gpio_bit_set
Function prototype	void gpio_bit_set(uint32_t gpio_periph,uint32_t pin);
Function descriptions	set GPIO pin bit
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
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	
-	-

Example:

```
/* set PA0*/
```

```
gpio_bit_set(GPIOA, GPIO_PIN_0);
```

gpio_bit_reset

The description of gpio_bit_reset is shown as below:

Table 3-716. Function gpio_bit_reset

Function name	gpio_bit_reset
Function prototype	void gpio_bit_reset(uint32_t gpio_periph,uint32_t pin);
Function descriptions	reset GPIO pin
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
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	
-	-

Example:

```
/* reset PA0*/

gpio_bit_set(GPIOA, GPIO_PIN_0);
```

gpio_bit_write

The description of gpio_bit_write is shown as below:

Table 3-717. Function gpio_bit_write

Function name	gpio_bit_write
Function prototype	void gpio_bit_write(uint32_t gpio_periph,uint32_t pin,bit_status bit_value);
Function descriptions	write data to the specified GPIO pin
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Input parameter{in}	
bit_value	SET or RESET
<i>RESET</i>	clear the port pin
<i>SET</i>	set the port pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write 1 to PA0 */

gpio_bit_write(GPIOA, GPIO_PIN_0, SET);
```

gpio_port_write

The description of gpio_port_write is shown as below:

Table 3-718. Function gpio_port_write

Function name	gpio_port_write
Function prototype	void gpio_port_write(uint32_t gpio_periph,uint16_t data);
Function descriptions	write data to the specified GPIO port
Precondition	-

The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Input parameter{in}	
data	specify the value to be written to the port output data register
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*write 1010 0101 1010 0101 to Port A */
```

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

gpio_input_filter_set

The description of gpio_input_filter_set is shown as below:

Table 3-719. Function gpio_input_filter_set

Function name	gpio_input_filter_set
Function prototype	void gpio_input_filter_set(uint32_t gpio_periph, uint8_t speriod, uint32_t iftype, uint32_t pin);
Function descriptions	set GPIO input filter
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Input parameter{in}	
speriod	gpio pin input sample period
<i>GPIO_ISPERIOD(x)</i>	period (x = 0 ~ 255)
Input parameter{in}	
iftype	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
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)

<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set 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 3-720. Function gpio_input_bit_get

Function name	gpio_input_bit_get
Function prototype	FlagStatus gpio_input_bit_get(uint32_t gpio_periph, uint32_t pin);
Function descriptions	get GPIO pin input status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

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

```
FlagStatus bit_state;
```

```
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 3-721. Function gpio_input_port_get

Function name	gpio_input_port_get
----------------------	---------------------

Function prototype	uint16_t gpio_input_port_get(uint32_t gpio_periph);
Function descriptions	get GPIO all pins input status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,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 3-722. Function gpio_output_bit_get

Function name	gpio_output_bit_get
Function prototype	FlagStatus gpio_output_bit_get(uint32_t gpio_periph,uint32_t pin);
Function descriptions	get GPIO pin output status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

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

FlagStatus bit_state;
```

```
bit_state = gpio_output_bit_get(GPIOA, GPIO_PIN_0);
```

gpio_output_port_get

The description of gpio_output_port_get is shown as below:

Table 3-723. Function gpio_output_port_get

Function name	gpio_output_port_get
Function prototype	uint16_t gpio_output_port_get(uint32_t gpio_periph);
Function descriptions	get GPIO all pins output status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Output parameter{out}	
-	-
Return value	
Uint16_t	0x0000-0xFFFF

Example:

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

```
uint16_t port_state;
```

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

gpio_af_set

The description of gpio_af_set is shown as below:

Table 3-724. Function gpio_af_set

Function name	gpio_af_set
Function prototype	void gpio_af_set(uint32_t gpio_periph, uint32_t alt_func_num, uint32_t pin);
Function descriptions	set GPIO alternate function
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Input parameter{in}	
alt_func_num	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, SAI1, SAI2
<i>GPIO_AF_2</i>	TIMER2, TIMER3, TIMER4, TIMER7, TIMER14, TLI, CAN2, SAI0, EXMC

<i>GPIO_AF_3</i>	<i>TIMER7, TIMER9, EDOUT, EXMC, TLI, HPDF, OSPIM</i>
<i>GPIO_AF_4</i>	<i>TIMER14, TIMER30, TIMER31, I2C0, I2C1, I2C2, I2C3, USART0, HPDF, OSPIM, TLI</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, SAI0, ETH1, EDOUT, OSPIM</i>
<i>GPIO_AF_7</i>	<i>USART0, USART1, USART2, USART5, UART6, TIMER40, TIMER41, TIMER42, TIMER43, SPI1, SPI2, SPI5, SDIO0, USBHS1</i>
<i>GPIO_AF_8</i>	<i>UART3, UART4, UART7, SPI5, SDIO0, RSPDIF, TIMER44, USBHS1, SAI1, SAI2</i>
<i>GPIO_AF_9</i>	<i>SDIO1, TRGSEL, CAN0, CAN1, TLI, OPSIM, EXMC, RSPDIF, SAI2</i>
<i>GPIO_AF_10</i>	<i>SAI1, SAI2, OTG0, SDIO1, CMP, USBHS0, OPSIM, EXMC</i>
<i>GPIO_AF_11</i>	<i>ETH0, MDIO, CMP, UART6, EXMC, HPDF, I2C3, TLI, SDIO1, OPSIM</i>
<i>GPIO_AF_12</i>	<i>TIMER0, MDIOS, SDIO0, EXMC, OPSIM, CMP, TLI, USBHS1</i>
<i>GPIO_AF_13</i>	<i>TRGSEL, DCI, COMP0, CMP, TIMER22</i>
<i>GPIO_AF_14</i>	<i>TLI, UART4, TIMER23</i>
<i>GPIO_AF_15</i>	<i>EVENTOUT</i>
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set PA0 alternate function 0 */
```

```
gpio_af_set(GPIOA, GPIO_AF_0, GPIO_PIN_0);
```

gpio_pin_lock

The description of gpio_pin_lock is shown as below:

Table 3-725. Function gpio_pin_lock

Function name	gpio_pin_lock
Function prototype	void gpio_pin_lock(uint32_t gpio_periph, uint32_t pin);
Function descriptions	lock GPIO pin bit
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Input parameter{in}	

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

Example:

```
/* lock PA0*/
```

```
gpio_pin_lock(GPIOA, GPIO_PIN_0);
```

gpio_bit_toggle

The description of gpio_bit_toggle is shown as below:

Table 3-726. Function gpio_bit_toggle

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

Example:

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

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

gpio_port_toggle

The description of gpio_port_toggle is shown as below:

Table 3-727. 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);
```

3.22. HAU

The HASH Acceleration Unit supports acceleration of SHA-1, SHA-224, SHA-256, MD5 algorithm and the HMAC (keyed-hash message authentication code) algorithm. The HAU registers are listed in chapter [3.22.1](#). the HAU firmware functions are introduced in chapter [3.22.2](#).

3.22.1. Descriptions of Peripheral registers

HAU registers are listed in the table shown as below:

Table 3-728. HAU Registers

Registers	Descriptions
HAU_CTL	control register
HAU_DI	data input register
HAU_CFG	configuration register
HAU_DO0	data output register 0
HAU_DO1	data output register 1
HAU_DO2	data output register 2
HAU_DO3	data output register 3
HAU_DO4	data output register 4
HAU_DO5	data output register 5
HAU_DO6	data output register 6
HAU_DO7	data output register 7

Registers	Descriptions
HAU_INTEN	interrupt enable register
HAU_STAT	status and interrupt flag register
HAU_CTXSx (x = 0..53)	context switch register

3.22.2. Descriptions of Peripheral functions

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

Table 3-729. HAU firmware function

Function name	Function description
hau_deinit	reset the HAU peripheral
hau_init	initialize the HAU peripheral parameters
hau_init_struct_para_init	initialize the struct hau_initpara
hau_reset	reset the HAU processor core
hau_last_word_validbits_num_config	configure the number of valid bits in last word of the message
hau_data_write	write data to the IN FIFO
hau_infifo_words_num_get	return the number of words already written into the IN FIFO
hau_digest_read	read the message digest result
hau_digest_calculation_enable	enable digest calculation
hau_multiple_single_dma_config	configure multiple or single DMA is used, and digest calculation at the end of a DMA transfer or not
hau_dma_enable	enable the HAU DMA interface
hau_dma_disable	disable the HAU DMA interface
hau_context_struct_para_init	initialize the struct context
hau_context_save	save the HAU peripheral context
hau_context_restore	restore the HAU peripheral context
hau_hash_sha_1	calculate digest using SHA1 in HASH mode
hau_hmac_sha_1	calculate digest using SHA1 in HMAC mode
hau_hash_sha_224	calculate digest using SHA224 in HASH mode
hau_hmac_sha_224	calculate digest using SHA224 in HMAC mode
hau_hash_sha_256	calculate digest using SHA256 in HASH mode
hau_hmac_sha_256	calculate digest using SHA256 in HMAC mode
hau_hash_md5	calculate digest using MD5 in HASH mode
hau_hmac_md5	calculate digest using MD5 in HMAC mode
hau_flag_get	get the HAU flag status
hau_flag_clear	clear the HAU flag status
hau_interrupt_enable	enable the HAU interrupts
hau_interrupt_disable	disable the HAU interrupts
hau_interrupt_flag_get	get the HAU interrupt flag status
hau_interrupt_flag_clear	clear the HAU interrupt flag status

Structure hau_init_parameter_struct

Table 3-730. Structure hau_init_parameter_struct

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

Structure hau_digest_parameter_struct

Table 3-731. Structure hau_digest_parameter_struct

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

Structure hau_context_parameter_struct

Table 3-732. Structure hau_context_parameter_struct

Member name	Function description
hau_ctl_bak	backup of HAU_CTL register
hau_cfg_bak	backup of HAU_CFG register
hau_inten_bak	backup of HAU_INTEN register
hau_ctxs_bak[54]	backup of HAU_CTXSx registers

hau_deinit

The description of hau_deinit is shown as below:

Table 3-733. Function hau_deinit

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

Example:

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

```
hau_deinit( );
```

hau_init

The description of hau_init is shown as below:

Table 3-734. Function hau_init

Function name	hau_init
Function prototype	void hau_init(hau_init_parameter_struct* initpara);
Function descriptions	initialize the HAU peripheral parameters
Precondition	-
The called functions	-
Input parameter{in}	
initpara	refer to structure Table 3-730. Structure hau_init_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the HAU peripheral parameters */
```

```
hau_init_parameter_struct hau_initpara;
```

```
...
```

```
hau_initpara.algo = algo;
```

```
hau_initpara.mode = HAU_MODE_HMAC;
```

```
hau_initpara.datatype = HAU_SWAPPING_8BIT;
```

```
if(key_len > 64U){
```

```
    hau_initpara.keytype = HAU_KEY_LONGGER_64;
```

```
}else{
```

```
    hau_initpara.keytype = HAU_KEY_SHORTER_64;
```

```
}
```

```
hau_init(&hau_initpara);
```

hau_init_struct_para_init

The description of hau_init_struct_para_init is shown as below:

Table 3-735. Function hau_init_struct_para_init

Function name	hau_init_struct_para_init
Function prototype	void hau_init_struct_para_init(hau_init_parameter_struct* initpara)
Function descriptions	initialize the struct hau_initpara
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
initpara	refer to structure Table 3-730. Structure hau_init_parameter_struct
Return value	
-	-

Example:

```
/* initialize the HAU peripheral parameters */
hau_init_parameter_struct hau_initpara;
hau_init_struct_para_init(&hau_initpara);
```

hau_reset

The description of hau_reset is shown as below:

Table 3-736. Function hau_reset

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

Example:

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

hau_last_word_validbits_num_config

The description of hau_last_word_validbits_num_config is shown as below:

Table 3-737. Function hau_last_word_validbits_num_config

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

Example:

```
/* configure the number of valid bits in last word of the message */
hau_last_word_validbits_num_config(0x00000010);
```

hau_data_write

The description of hau_data_write is shown as below:

Table 3-738. Function hau_data_write

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

Example:

```
/* write data to the IN FIFO */
hau_data_write(0x00000010);
```

hau_infifo_words_num_get

The description of hau_infifo_words_num_get is shown as below:

Table 3-739. Function hau_infifo_words_num_get

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

Example:

```
/* return the number of words already written into the IN FIFO */
uint32_t num;
num = hau_infifo_words_num_get();
```

hau_digest_read

The description of hau_digest_read is shown as below:

Table 3-740. Function hau_digest_read

Function name	hau_digest_read
Function prototype	void hau_digest_read(hau_digest_parameter_struct* digestpara);
Function descriptions	read the message digest result
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
digestpara	refer to structure Table 3-731. Structure hau_digest_parameter_struct
Return value	
-	-

Example:

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

hau_digest_calculation_enable

The description of hau_digest_calculation_enable is shown as below:

Table 3-741. Function hau_digest_calculation_enable

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

Example:

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

hau_multiple_single_dma_config

The description of hau_multiple_single_dma_config is shown as below:

Table 3-742. Function hau_multiple_single_dma_config

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

Example:

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

```
hau_multiple_single_dma_config(SINGLE_DMA_AUTO_DIGEST);
```

hau_dma_enable

The description of hau_dma_enable is shown as below:

Table 3-743. Function hau_dma_enable

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

Example:

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

```
hau_dma_enable();
```

hau_dma_disable

The description of hau_dma_disable is shown as below:

Table 3-744. Function hau_dma_disable

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

Example:

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

```
hau_dma_disable();
```

hau_context_struct_para_init

The description of hau_context_struct_para_init is shown as below:

Table 3-745. Function hau_context_struct_para_init

Function name	hau_context_struct_para_init
Function prototype	void hau_context_struct_para_init(hau_context_parameter_struct* context)
Function descriptions	initialize the struct context
Precondition	-
The called functions	-
Input parameter{in}	
context	refer to structure Table 3-732. Structure hau_context_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the struct context */

hau_context_parameter_struct context_para;

hau_context_struct_para_init(&context_para);
```

hau_context_save

The description of hau_context_save is shown as below:

Table 3-746. Function hau_context_save

Function name	hau_context_save
Function prototype	void hau_context_save(hau_context_parameter_struct* context_save)
Function descriptions	save the HAU peripheral context
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
context_save	refer to structure Table 3-732. Structure hau_context_parameter_struct
Return value	
-	-

Example:

```
hau_context_parameter_struct context_para;
```

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

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

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

hau_context_restore

The description of hau_context_restore is shown as below:

Table 3-747. Function hau_context_restore

Function name	hau_context_restore
Function prototype	void hau_context_restore(hau_context_parameter_struct* context_restore)
Function descriptions	restore the HAU peripheral context
Precondition	-
The called functions	-
Input parameter{in}	
context_restore	refer to structure Table 3-732. Structure hau context parameter struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hau_context_parameter_struct context_para;
```

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

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

```
.....
```

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

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

hau_hash_sha_1

The description of hau_hash_sha_1 is shown as below:

Table 3-748. Function hau_hash_sha_1

Function name	hau_hash_sha_1
Function prototype	ErrStatus hau_hash_sha_1(uint8_t *input, uint32_t in_length, uint8_t output[]);
Function descriptions	calculate digest using SHA1 in HASH mode
Precondition	-
The called functions	-
Input parameter{in}	

input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

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

```
ErrStatus status = ERROR;
```

```
uint8_t output[20];
```

```
uint8_t input[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                  0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
status = hau_hash_sha_1(input, 0x10, output);
```

hau_hmac_sha_1

The description of hau_hmac_sha_1 is shown as below:

Table 3-749. Function hau_hmac_sha_1

Function name	hau_hmac_sha_1
Function prototype	ErrStatus hau_hmac_sha_1(uint8_t *key, uint32_t keysize, uint8_t *input, uint32_t in_length, uint8_t output[]);
Function descriptions	calculate digest using SHA1 in HMAC mode
Precondition	-
The called functions	-
Input parameter{in}	
key	pointer to the key used for HMAC
Input parameter{in}	
keysize	length of the key used for HMAC
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

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

```
ErrStatus status = ERROR;
```

```
uint8_t output[20];
```

```
uint8_t key[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
uint8_t input[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                  0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
status = hau_hmac_sha_1(key, 0x10, input, 0x10, output);
```

hau_hash_sha_224

The description of hau_hash_sha_224 is shown as below:

Table 3-750. Function hau_hash_sha_224

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

Example:

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

```
ErrStatus status = ERROR;
```

```
uint8_t output[28];
```

```
uint8_t input[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                  0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
status = hau_hash_sha_224 (input, 0x10, output);
```

hau_hmac_sha_224

The description of hau_hmac_sha_224 is shown as below:

Table 3-751. Function hau_hmac_sha_224

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

Example:

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

```
ErrStatus status = ERROR;
```

```
uint8_t output[28];
```

```
uint8_t key[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
uint8_t input[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                  0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
status = hau_hmac_sha_224 (key, 0x10, input, 0x10, output);
```

hau_hash_sha_256

The description of hau_hash_sha_256 is shown as below:

Table 3-752. Function hau_hash_sha_256

Function name	hau_hash_sha_256
Function prototype	ErrStatus hau_hash_sha_256(uint8_t *input, uint32_t in_length, uint8_t

	output[]);
Function descriptions	calculate digest using SHA256 in HASH mode
Precondition	-
The called functions	-
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

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

```
ErrStatus status = ERROR;
```

```
uint8_t output[32];
```

```
uint8_t input[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,  
                  0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
status = hau_hash_sha_256 (input, 0x10, output);
```

hau_hmac_sha_256

The description of hau_hmac_sha_256 is shown as below:

Table 3-753. Function hau_hmac_sha_256

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

output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

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

```
ErrStatus status = ERROR;
```

```
uint8_t output[32];
```

```
uint8_t key[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
uint8_t input[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                  0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
status = hau_hmac_sha_256 (key, 0x10, input, 0x10, output);
```

hau_hash_md5

The description of hau_hash_md5 is shown as below:

Table 3-754. Function hau_hash_md5

Function name	hau_hash_md5
Function prototype	ErrStatus hau_hash_md5(uint8_t *input, uint32_t in_length, uint8_t output[]);
Function descriptions	calculate digest using MD5 in HASH mode
Precondition	-
The called functions	-
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

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

```
ErrStatus status = ERROR;
```

```
uint8_t output[16];
```

```
uint8_t input[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
```

```
0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
status = hau_hash_md5 (input, 0x10, output);
```

hau_hmac_md5

The description of hau_hmac_md5 is shown as below:

Table 3-755. Function hau_hmac_md5

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

Example:

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

```
ErrStatus status = ERROR;
```

```
uint8_t output[16];
```

```
uint8_t key[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
uint8_t input[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
                  0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
```

```
status = hau_hmac_md5 (key, 0x10, input, 0x10, output);
```

hau_flag_get

The description of hau_flag_get is shown as below:

Table 3-756. Function hau_flag_get

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

Example:

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

```
FlagStatus status;
```

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

hau_flag_clear

The description of hau_flag_clear is shown as below:

Table 3-757. Function hau_flag_clear

Function name	hau_flag_clear
Function prototype	void hau_flag_clear(uint32_t flag);
Function descriptions	clear the HAU flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	HAU flag status
<i>HAU_FLAG_DATA_INPUT</i>	there is enough space (16 bytes) in the input FIFO
<i>HAU_FLAG_CALCULATION_COMPLETE</i>	digest calculation is completed
Output parameter{out}	

-	-
Return value	
-	-

Example:

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

```
hau_flag_clear (HAU_FLAG_DATA_INPUT);
```

hau_interrupt_enable

The description of hau_interrupt_enable is shown as below:

Table 3-758. Function hau_interrupt_enable

Function name	hau_interrupt_enable
Function prototype	void hau_interrupt_enable(uint32_t interrupt);
Function descriptions	enable the HAU interrupts
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify the HAU interrupt source to be enabled
HAU_INT_DATA_INPU T	a new block can be entered into the IN buffer
HAU_INT_CALCULATI ON_COMPLETE	calculation complete
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
hau_interrupt_enable (HAU_INT_DATA_INPUT);
```

hau_interrupt_disable

The description of hau_interrupt_disable is shown as below:

Table 3-759. Function hau_interrupt_disable

Function name	hau_interrupt_disable
Function prototype	void hau_interrupt_disable(uint32_t interrupt);
Function descriptions	disable the HAU interrupts
Precondition	-
The called functions	-
Input parameter{in}	

interrupt	specify the HAU interrupt source to be disabled
<i>HAU_INT_DATA_INPUT</i> <i>T</i>	a new block can be entered into the IN buffer
<i>HAU_INT_CALCULATION_COMPLETE</i>	calculation complete
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable hau interrupt */
```

```
hau_interrupt_disable(HAU_INT_DATA_INPUT);
```

hau_interrupt_flag_get

The description of hau_interrupt_flag_get is shown as below:

Table 3-760. Function hau_interrupt_flag_get

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

Example:

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

```
FlagStatus status = hau_interrupt_flag_get(HAU_INT_FLAG_DATA_INPUT);
```

hau_interrupt_flag_clear

The description of hau_interrupt_flag_clear is shown as below:

Table 3-761. Function hau_interrupt_flag_clear

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

Example:

```

/* clear the HAU interrupt flag status */
hau_interrupt_flag_clear(HAU_INT_FLAG_DATA_INPUT);

```

3.23. HPDF

A high performance digital filter module (HPDF) for external sigma delta (Σ - Δ) modulator is integrated in GD32W51x. The HPDF registers are listed in chapter [3.23.1](#), the HPDF firmware functions are introduced in chapter [3.23.2](#).

3.23.1. Descriptions of Peripheral registers

HPDF registers are listed in the table shown as below:

Table 3-762. HPDF Registers

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

3.23.2. Descriptions of Peripheral functions

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

Table 3-763. 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

Function name	Function description
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
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

Function name	Function description
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 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

Function name	Function description
	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 3-764. 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 3-765. 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 3-766. Structure hpdf_rc_parameter_struct

Member name	Function description
fast_mode	fast conversion mode enable for regular conversions
rsc_channel	regular conversion channel
rscdmaen	DMA channel enabled to read data for the regular conversion
rscsyn	regular conversion synchronously
continuous_mode	regular conversions continuous mode

Structure hpdf_ic_parameter_struct

Table 3-767. 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 3-768. Enum hpdf_channel_enum

enum name	enum description
CHANNEL0	HPDF channel0
CHANNEL1	HPDF channel1
CHANNEL2	HPDF channel2
CHANNEL3	HPDF channel3
CHANNEL4	HPDF channel4
CHANNEL5	HPDF channel5
CHANNEL6	HPDF channel6
CHANNEL7	HPDF channel7

Enum hpdf_filter_enum

Table 3-769. Enum hpdf_filter_enum

enum name	enum description
FLT0	HPDF filter0
FLT1	HPDF filter1

enum name	enum description
FLT2	HPDF filter2
FLT3	HPDF filter3

Enum hpdf_flag_enum

Table 3-770. Enum hpdf_flag_enum

enum name	enum description
HPDF_FLAG_FLTy _ICEF	inserted conversion end flag
HPDF_FLAG_FLTy _RCEF	regular conversion end flag
HPDF_FLAG_FLTy _ICDOF	inserted conversion overflow flag
HPDF_FLAG_FLTy _RCDOF	regular conversion overflow flag
HPDF_FLAG_FLTy _TMEOF	threshold monitor event occurred flag
HPDF_FLAG_FLTy _ICPF	inserted conversion in progress flag
HPDF_FLAG_FLTy _RCPF	regular conversion in progress flag
HPDF_FLAG_FLT0 _CKLF0	clock loss on channel 0 flag
HPDF_FLAG_FLT0 _CKLF1	clock loss on channel 1 flag
HPDF_FLAG_FLT0 _CKLF2	clock loss on channel 2 flag
HPDF_FLAG_FLT0 _CKLF3	clock loss on channel 3 flag
HPDF_FLAG_FLT0 _CKLF4	clock loss on channel 4 flag
HPDF_FLAG_FLT0 _CKLF5	clock loss on channel 5 flag
HPDF_FLAG_FLT0 _CKLF6	clock loss on channel 6 flag
HPDF_FLAG_FLT0 _CKLF7	clock loss on channel 7 flag
HPDF_FLAG_FLT0 _MMF0	malfunction event occurred on channel 0 flag
HPDF_FLAG_FLT0 _MMF1	malfunction event occurred on channel 1 flag
HPDF_FLAG_FLT0	malfunction event occurred on channel 2 flag

enum name	enum description
_MMF2	
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
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	threshold monitor high threshold on channel 6 flag

enum name	enum description
_HTF6	
HPDF_FLAG_FLTy_ _HTF7	threshold monitor high threshold on channel 7 flag

Enum hpdf_interrput_flag_enum

Table 3-771. Enum hpdf_interrput_flag_enum

enum name	enum description
HPDF_INT_FLAG_ FLTy_ICEF	inserted conversion end interrupt flag
HPDF_INT_FLAG_ FLTy_RCEF	regular conversion end interruptflag
HPDF_INT_FLAG_ FLTy_ICDOF	inserted conversion overflow interrupt flag
HPDF_INT_FLAG_ FLTy_RCDOF	regular conversion overflow interrupt flag
HPDF_INT_FLAG_ FLTy_TMEOF	threshold monitor event occurred interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF0	clock loss on channel 0 interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF1	clock loss on channel 1 interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF2	clock loss on channel 2 interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF3	clock loss on channel 3 interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF4	clock loss on channel 4 interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF5	clock loss on channel 5 interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF6	clock loss on channel 6 interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF7	clock loss on channel 7 interrupt flag
HPDF_INT_FLAG_ FLT0_MMF0	malfunction monitor detection on channel 0 interrupt flag
HPDF_INT_FLAG_ FLT0_MMF1	malfunction monitor detection on channel 1 interrupt flag
HPDF_INT_FLAG_ FLT0_MMF2	malfunction monitor detection on channel 2 interrupt flag
HPDF_INT_FLAG_ FLT0_MMF3	malfunction monitor detection on channel 3 interrupt flag

enum name	enum description
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 3-772. 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 3-773. Function hpdf_deinit

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

-	-
---	---

Example:

```
/* reset HPDF */
hpdf_deinit();
```

hpdf_channel_struct_para_init

The description of hpdf_channel_struct_para_init is shown as below:

Table 3-774. Function hpdf_channel_struct_para_init

Function name	hpdf_channel_struct_para_init
Function prototype	void hpdf_channel_struct_para_init(hpdf_channel_parameter_struct* init_struct);
Function descriptions	initialize the parameters of HPDF channel struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF channel, refer to Table 3-764. Structure hpdf_channel_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of HPDF channel */
hpdf_channel_parameter_struct hpdf_channel_init_struct;
hpdf_channel_struct_para_init(&hpdf_channel_init_struct);
```

hpdf_filter_struct_para_init

The description of hpdf_filter_struct_para_init is shown as below:

Table 3-775. Function hpdf_filter_struct_para_init

Function name	hpdf_filter_struct_para_init
Function prototype	void hpdf_filter_struct_para_init(hpdf_filter_parameter_struct* init_struct);
Function descriptions	initialize the parameters of HPDF filter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF filter, refer to Table 3-765. Structure hpdf_filter_parameter_struct

Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the parameters of HPDF filter */

hpdf_filter_parameter_struct hpdf_filter_init_struct;

hpdf_filter_struct_para_init(&hpdf_filter_init_struct);

```

hpdf_rc_struct_para_init

The description of hpdf_rc_struct_para_init is shown as below:

Table 3-776. Function hpdf_rc_struct_para_init

Function name	hpdf_rc_struct_para_init
Function prototype	void hpdf_rc_struct_para_init(hpdf_rc_parameter_struct* init_struct);
Function descriptions	initialize the parameters of regular conversion struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF regular conversion, refer to Table 3-766. Structure hpdf_rc_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the parameters of regular conversion */

hpdf_rc_parameter_struct hpdf_rc_init_struct;

hpdf_rc_struct_para_init(&hpdf_rc_init_struct);

```

hpdf_ic_struct_para_init

The description of hpdf_ic_struct_para_init is shown as below:

Table 3-777. Function hpdf_ic_struct_para_init

Function name	hpdf_ic_struct_para_init
Function prototype	void hpdf_ic_struct_para_init(hpdf_ic_parameter_struct* init_struct);
Function descriptions	initialize the parameters of inserted conversion struct with the default values
Precondition	-
The called functions	-

Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF inserted conversion, refer to Table 3-767. Structure hpdf_ic_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of inserted conversion */
hpdf_ic_parameter_struct hpdf_ic_init_struct;
hpdf_ic_struct_para_init(&hpdf_ic_init_struct);
```

hpdf_enable

The description of hpdf_enable is shown as below:

Table 3-778. Function hpdf_enable

Function name	hpdf_enable
Function prototype	void hpdf_enable(void);
Function descriptions	enable the HPDF module globally
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the HPDF module */
hpdf_enable();
```

hpdf_disable

The description of hpdf_disable is shown as below:

Table 3-779. Function hpdf_disable

Function name	hpdf_disable
Function prototype	void hpdf_disable(void);
Function descriptions	disable the HPDF module globally
Precondition	-

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

Example:

```
/* disable the HPDF module */
hpdf_disable();
```

hpdf_channel_init

The description of hpdf_channel_init is shown as below:

Table 3-780. Function hpdf_channel_init

Function name	hpdf_channel_init
Function prototype	void hpdf_channel_init(hpdf_channel_enum channelx, hpdf_channel_parameter_struct* init_struct);
Function descriptions	initialize the HPDF channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF channel, refer to Table 3-764. Structure hpdf_channel_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the HPDF channel0 */
hpdf_channel_parameter_struct hpdf_channel_init_struct;

/* initialize HPDF channel0 */
hpdf_channel_init_struct.data_packing_mode = DPM_STANDARD_MODE;
hpdf_channel_init_struct.malfunction_monitor = MM_ENABLE;
hpdf_channel_init_struct.spi_ck_source = EXTERNAL_CKIN;
```

```

hpdf_channel_init_struct.channel_muxlexer = SERIAL_INPUT;

hpdf_channel_init_struct.serial_interface = SPI_RISING_EDGE;

hpdf_channel_init_struct.calibration_offset = 0;

hpdf_channel_init_struct.right_bit_shift = 0;

hpdf_channel_init_struct.mm_counter_threshold = 110;

hpdf_channel_init_struct.plsk_value = 0;

hpdf_channel_init(CHANNEL0, &hpdf_channel_init_struct);

```

hpdf_filter_init

The description of hpdf_filter_init is shown as below:

Table 3-781. Function hpdf_filter_init

Function name	hpdf_filter_init
Function prototype	void hpdf_filter_init(hpdf_filter_enum filtery, hpdf_filter_parameter_struct* init_struct);
Function descriptions	initialize the HPDF filter
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..7)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF filter, refer to Table 3-765. Structure hpdf_filter_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

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 3-782. Function hpdf_rc_init

Function name	hpdf_rc_init
Function prototype	void hpdf_rc_init(hpdf_filter_enum filtery, hpdf_rc_parameter_struct* init_struct);
Function descriptions	initialize the regular conversion
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF regular conversion, refer to Table 3-766. Structure hpdf_rc_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize regular conversion of the HPDF filter0 */

hpdf_rc_parameter_struct hpdf_rc_init_struct;

hpdf_rc_init_struct.fast_mode = FAST_DISABLE;

hpdf_rc_init_struct.rcs_channel = RCS_CHANNEL0;

hpdf_rc_init_struct.rcdmaen = RCDMAEN_ENABLE;

hpdf_rc_init_struct.continuous_mode = RCCM_ENABLE;

hpdf_rc_init(FLT0, &hpdf_rc_init_struct);

```

hpdf_ic_init

The description of hpdf_ic_init is shown as below:

Table 3-783. Function `hpdf_ic_init`

Function name	<code>hpdf_ic_init</code>
Function prototype	<code>void hpdf_ic_init(hpdf_filter_enum filtery, hpdf_ic_parameter_struct* init_struct);</code>
Function descriptions	initialize the inserted conversion
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy</i> (y=0..3)	select HPDF filter, refer to Table 3-769. Enum <code>hpdf_filter_enum</code>
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF inserted conversion, refer to Table 3-767. Structure <code>hpdf_ic_parameter_struct</code>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize inserted conversion of the HPDF filter0 */

hpdf_ic_parameter_struct hpdf_ic_init_struct;

hpdf_ic_init_struct.ic_channel_group = IGCSEL_CHANNEL0;

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 3-784. Function `hpdf_clock_output_config`

Function name	<code>hpdf_clock_output_config</code>
Function prototype	<code>void hpdf_clock_output_config(uint32_t source, uint8_t divider, uint32_t mode);</code>
Function descriptions	configure serial output clock
Precondition	disable HPDF
The called functions	-
Input parameter{in}	
source	the HPDF serial clock output source
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from system clock

<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from audio clock
Input parameter{in}	
divider	serial clock output divider(0-255)
Input parameter{in}	
mode	serial clock output duty mode
<i>CKOUTDM_DISABLE</i>	disable serial clock output duty mode
<i>CKOUTDM_ENABLE</i>	enable serial clock output duty mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure serial clock output */
hpdf_clock_output_config(SERIAL_SYSTEM_CLK, 175, CKOUTDM_ENABLE);
```

hpdf_clock_output_source_config

The description of hpdf_clock_output_source_config is shown as below:

Table 3-785. Function hpdf_clock_output_source_config

Function name	hpdf_clock_output_source_config
Function prototype	void hpdf_clock_output_source_config(uint32_t source);
Function descriptions	configure serial clock output source
Precondition	disable HPDF
The called functions	-
Input parameter{in}	
source	the HPDF serial clock output source
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from system clock
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from audio clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure serial clock output source */
hpdf_clock_output_config(SERIAL_SYSTEM_CLK);
```

hpdf_clock_output_duty_mode_disable

The description of hpdf_clock_output_duty_mode_disable is shown as below:

Table 3-786. Function `hpdf_clock_output_duty_mode_disable`

Function name	<code>hpdf_clock_output_duty_mode_disable</code>
Function prototype	<code>void hpdf_clock_output_duty_mode_disable(void);</code>
Function descriptions	disable serial clock output duty mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable serial clock output duty mode*/
```

```
hpdf_clock_output_duty_mode_disable();
```

`hpdf_clock_output_duty_mode_enable`

The description of `hpdf_clock_output_duty_mode_enable` is shown as below:

Table 3-787. Function `hpdf_clock_output_duty_mode_enable`

Function name	<code>hpdf_clock_output_duty_mode_enable</code>
Function prototype	<code>void hpdf_clock_output_duty_mode_enable(void);</code>
Function descriptions	enable serial clock output duty mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable serial clock output duty mode*/
```

```
hpdf_clock_output_duty_mode_enable();
```

`hpdf_clock_output_divider_config`

The description of `hpdf_clock_output_divider_config` is shown as below:

Table 3-788. Function `hpdf_clock_output_divider_config`

Function name	<code>hpdf_clock_output_divider_config</code>
Function prototype	<code>void hpdf_clock_output_divider_config(uint8_t divider);</code>
Function descriptions	configure serial clock output divider
Precondition	disable HPDF
The called functions	-
Input parameter{in}	
divider	serial clock output divider(0-255)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure serial clock output divider */
hpdf_clock_output_divider_config (255);
```

`hpdf_channel_enable`

The description of `hpdf_channel_enable` is shown as below:

Table 3-789. Function `hpdf_channel_enable`

Function name	<code>hpdf_channel_enable</code>
Function prototype	<code>void hpdf_channel_enable(hpdf_channel_enum channelx);</code>
Function descriptions	enable channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<code>CHANNELx(x=0..7)</code>	select HPDF channel, refer to Table 3-768. Enum <code>hpdf_channel_enum</code>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable channel0 */
hpdf_channel_enable(CHANNEL0);
```

`hpdf_channel_disable`

The description of `hpdf_channel_disable` is shown as below:

Table 3-790. 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}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum <code>hpdf_channel_enum</code>
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 3-791. 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}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum <code>hpdf_channel_enum</code>
Input parameter{in}	
clock_source	SPI clock source
<i>EXTERNAL_CKIN</i>	external input clock
<i>INTERNAL_CKOUT</i>	internal CKOUT clock
<i>HALF_CKOUT_FALLING_EDGE</i>	internal CKOUT clock, sampling point on each second CKOUT falling edge
<i>HALF_CKOUT_RISING_EDGE</i>	internal CKOUT clock, sampling point on each second CKOUT rising edge
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* configure external input clock as SPI clock source */
hpdf_spi_clock_source_config(CHANNEL0, EXTERNAL_CKIN);
```

hpdf_serial_interface_type_config

The description of hpdf_serial_interface_type_config is shown as below:

Table 3-792. Function hpdf_serial_interface_type_config

Function name	hpdf_serial_interface_type_config
Function prototype	void hpdf_serial_interface_type_config(hpdf_channel_enum channelx, uint32_t type);
Function descriptions	configure serial interface type
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	
type	serial interface type
SPI_RISING_EDGE	SPI interface, sample data on rising edge
SPI_FALLING_EDGE	SPI interface, sample data on 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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure external input clock as SPI clock source */
hpdf_spi_clock_source_config(CHANNEL0, EXTERNAL_CKIN);
```

hpdf_malfunction_monitor_disable

The description of hpdf_malfunction_monitor_disable is shown as below:

Table 3-793. Function hpdf_malfunction_monitor_disable

Function name	hpdf_malfunction_monitor_disable
Function prototype	void hpdf_malfunction_monitor_disable(hpdf_channel_enum channelx);

Function descriptions	disable malfunction monitor
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable malfunction monitor */
hpdf_malfunction_monitor_disable(CHANNEL0);
```

hpdf_malfunction_monitor_enable

The description of hpdf_malfunction_monitor_enable is shown as below:

Table 3-794. Function hpdf_malfunction_monitor_enable

Function name	hpdf_malfunction_monitor_enable
Function prototype	void hpdf_malfunction_monitor_enable(hpdf_channel_enum channelx);
Function descriptions	enable malfunction monitor
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable malfunction monitor */
hpdf_malfunction_monitor_enable(CHANNEL1);
```

hpdf_clock_loss_disable

The description of hpdf_clock_loss_disable is shown as below:

Table 3-795. 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 Table 3-768. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable clock loss detector */
```

```
hpdf_clock_loss_disable(CHANNEL0);
```

hpdf_clock_loss_enable

The description of hpdf_clock_loss_enable is shown as below:

Table 3-796. 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 Table 3-768. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable clock loss detector */
```

```
hpdf_clock_loss_enable(CHANNEL0);
```

hpdf_channel_pin_redirection_disable

The description of hpdf_channel_pin_redirection_disable is shown as below:

Table 3-797. 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 Table 3-768. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable channel0 inputs pins redirection */
```

```
hpdf_channel_pin_redirection_disable(CHANNEL0);
```

hpdf_channel_pin_redirection_enable

The description of hpdf_channel_pin_redirection_enable is shown as below:

Table 3-798. 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 Table 3-768. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable channel0 inputs pins redirection */
```

```
hpdf_channel_pin_redirection_enable(CHANNEL0);
```

hpdf_channel_multiplexer_config

The description of hpdf_channel_multiplexer_config is shown as below:

Table 3-799. Function `hpdf_channel_mux_config`

Function name	<code>hpdf_channel_mux_config</code>
Function prototype	<code>void hpdf_channel_mux_config(hpdf_channel_enum channelx, uint32_t data_source);</code>
Function descriptions	configure channel multiplexer select input data source
Precondition	disable <code>CHANNELx(x=0..7)</code>
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<code>CHANNELx(x=0..7)</code>	select HPDF channel, refer to Table 3-768. Enum <code>hpdf_channel_enum</code>
Input parameter{in}	
data_source	input data source
<code>SERIAL_INPUT</code>	input data source is taken from serial inputs
<code>INTERNAL_INPUT</code>	input data source is taken from internal HPDF_CHxPDI register
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure channel multiplexer select input data source */
```

```
hpdf_channel_mux_config(CHANNEL0, SERIAL_INPUT);
```

`hpdf_data_pack_mode_config`

The description of `hpdf_data_pack_mode_config` is shown as below:

Table 3-800. Function `hpdf_data_pack_mode_config`

Function name	<code>hpdf_data_pack_mode_config</code>
Function prototype	<code>void hpdf_data_pack_mode_config(hpdf_channel_enum channelx, uint32_t mode);</code>
Function descriptions	configure data packing mode
Precondition	disable <code>CHANNELx(x=0..7)</code>
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<code>CHANNELx(x=0..7)</code>	select HPDF channel, refer to Table 3-768. Enum <code>hpdf_channel_enum</code>
Input parameter{in}	
mode	parallel data packing mode
<code>DPM_STANDARD_MODE</code>	standard mode
<code>DPM_INTERLEAVED_MODE</code>	interleaved mode

<i>DPM_DUAL_MODE</i>	dual mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure data packing mode */
```

```
hpdf_data_pack_mode_config(CHANNEL0, DPM_STANDARD_MODE);
```

hpdf_data_right_bit_shift_config

The description of hpdf_data_right_bit_shift_config is shown as below:

Table 3-801. Function hpdf_data_right_bit_shift_config

Function name	hpdf_data_right_bit_shift_config
Function prototype	void hpdf_data_right_bit_shift_config(hpdf_channel_enum channelx, uint8_t right_shift);
Function descriptions	configure data right bit-shift
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	
right_shift	the number of bits that determine the right shift(0-31)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure data right bit-shift */
```

```
hpdf_data_right_bit_shift_config(CHANNEL0, 5);
```

hpdf_calibration_offset_config

The description of hpdf_calibration_offset_config is shown as below:

Table 3-802. Function hpdf_calibration_offset_config

Function name	hpdf_calibration_offset_config
Function prototype	void hpdf_calibration_offset_config(hpdf_channel_enum channelx, int32_t offset);
Function descriptions	configure calibration offset

Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	
offset	24-bit calibration offset, must be in (-8388608~8388607)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure calibration offset */
```

```
hpdf_calibration_offset_config (CHANNEL0, -255);
```

hpdf_malfunction_break_signal_config

The description of hpdf_malfunction_break_signal_config is shown as below:

Table 3-803. Function hpdf_malfunction_break_signal_config

Function name	hpdf_malfunction_break_signal_config
Function prototype	void hpdf_malfunction_break_signal_config(hpdf_channel_enum channelx, uint32_t break_signal);
Function descriptions	configure malfunction monitor break signal
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	
break_signal	malfunction monitor break signal distribution
<i>NO_MM_BREAK</i>	break signal is not distributed to malfunction monitor on channel
<i>MM_BREAK0</i>	break signal 0 is distributed to malfunction monitor on channel
<i>MM_BREAK1</i>	break signal 1 is distributed to malfunction monitor on channel
<i>MM_BREAK2</i>	break signal 2 is distributed to malfunction monitor on channel
<i>MM_BREAK3</i>	break signal 3 is distributed to malfunction monitor on channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure break signal is distributed to malfunction monitor on channel0 */
```

```
hpdf_data_pack_mode_config(CHANNEL0, MM_BREAK0);
```

hpdf_malfunction_counter_config

The description of hpdf_malfunction_counter_config is shown as below:

Table 3-804. Function hpdf_malfunction_counter_config

Function name	hpdf_malfunction_counter_config
Function prototype	void hpdf_malfunction_counter_config(hpdf_channel_enum channelx, uint8_t threshold);
Function descriptions	configure malfunction monitor counter threshold
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	
threshold	malfunction monitor counter threshold(0-255)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure malfunction monitor counter threshold */
```

```
hpdf_malfunction_counter_config(CHANNEL0, 255);
```

hpdf_write_parallel_data_standard_mode

The description of hpdf_write_parallel_data_standard_mode is shown as below:

Table 3-805. Function hpdf_write_parallel_data_standard_mode

Function name	hpdf_write_parallel_data_standard_mode
Function prototype	void hpdf_write_parallel_data_standard_mode(hpdf_channel_enum channelx, int16_t data);
Function descriptions	write the parallel data on standard mode of data packing
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	

data	the parallel data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write the parallel data on standard mode of data packing */
```

```
hpdf_write_parallel_data_standard_mode(CHANNEL0, 0xEFFF);
```

hpdf_write_parallel_data_interleaved_mode

The description of hpdf_write_parallel_data_interleaved_mode is shown as below:

Table 3-806. Function hpdf_write_parallel_data_interleaved_mode

Function name	hpdf_write_parallel_data_interleaved_mode
Function prototype	void hpdf_write_parallel_data_interleaved_mode(hpdf_channel_enum channelx, int32_t data);
Function descriptions	write the parallel data on interleaved mode of data packing
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	
data	the parallel data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write the parallel data on interleaved mode of data packing */
```

```
hpdf_write_parallel_data_interleaved_mode(CHANNEL0, 0xEFFFFFFF);
```

hpdf_write_parallel_data_dual_mode

The description of hpdf_write_parallel_data_dual_mode is shown as below:

Table 3-807. Function hpdf_write_parallel_data_dual_mode

Function name	hpdf_write_parallel_data_dual_mode
Function prototype	void hpdf_write_parallel_data_dual_mode(hpdf_channel_enum channelx, int32_t data);
Function descriptions	write the parallel data on dual mode of data packing

Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	
data	the parallel data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write the parallel data on dual mode of data packing */
```

```
hpdf_write_parallel_data_dual_mode(CHANNEL0, 0xEFFFFFFF);
```

hpdf_pulse_skip_update

The description of hpdf_pulse_skip_update is shown as below:

Table 3-808. Function hpdf_pulse_skip_update

Function name	hpdf_pulse_skip_update
Function prototype	void hpdf_pulse_skip_update(hpdf_channel_enum channelx, uint8_t number);
Function descriptions	update the number of pulses to skip
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Input parameter{in}	
number	the number of serial input samples that will be skipped
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* update the number of pulses to skip */
```

```
pdf_pulse_skip_update(CHANNEL0, 63);
```

hpdf_pulse_skip_read

The description of hpdf_pulse_skip_read is shown as below:

Table 3-809. 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 Table 3-768. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
uint8_t	the number of pulses to skip

Example:

```
/* read the number of pulses to skip */

uint8_t value;

value = hpdf_pulse_skip_read(CHANNEL0);
```

hpdf_filter_enable

The description of hpdf_filter_enable is shown as below:

Table 3-810. Function hpdf_filter_enable

Function name	hpdf_filter_enable
Function prototype	void hpdf_filter_enable(hpdf_filter_enum filtery);
Function descriptions	enable filter
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable filter0 */
```

hpdf_filter_enable(FLT0);

hpdf_filter_disable

The description of hpdf_filter_disable is shown as below:

Table 3-811. Function hpdf_filter_disable

Function name	hpdf_filter_disable
Function prototype	void hpdf_filter_disable(hpdf_filter_enum filtery);
Function descriptions	disable filter
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable filter0 */
```

```
hpdf_filter_disable(FLT0);
```

hpdf_filter_config

The description of hpdf_filter_config is shown as below:

Table 3-812. Function hpdf_filter_config

Function name	hpdf_filter_config
Function prototype	void hpdf_filter_config(hpdf_filter_enum filtery, uint32_t order, uint16_t oversample);
Function descriptions	configure sinc filter order and oversample
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
order	sinc filter order
<i>FLT_FASTSINC</i>	FastSinc filter type
<i>FLT_SINC1</i>	Sinc1 filter type
<i>FLT_SINC2</i>	Sinc2 filter type
<i>FLT_SINC3</i>	Sinc3 filter type

<i>FLT_SINC4</i>	Sinc4 filter type
<i>FLT_SINC5</i>	Sinc5 filter type
Input parameter{in}	
oversample	Sinc filter oversampling rate(1-1024)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure sinc filter order and oversample */
```

```
hpdf_filter_config(FLT0, FLT_SINC2, 64);
```

hpdf_integrator_oversample

The description of hpdf_integrator_oversample is shown as below:

Table 3-813. Function hpdf_integrator_oversample

Function name	hpdf_integrator_oversample
Function prototype	void hpdf_integrator_oversample(hpdf_filter_enum filtery, uint8_t oversample);
Function descriptions	configure integrator oversampling rate
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
oversample	integrator oversampling rate(1-256)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure integrator oversampling rate */
```

```
hpdf_integrator_oversample(FLT0, 256);
```

hpdf_threshold_monitor_filter_config

The description of hpdf_threshold_monitor_filter_config is shown as below:

Table 3-814. Function `hpdf_threshold_monitor_filter_config`

Function name	<code>hpdf_threshold_monitor_filter_config</code>
Function prototype	<code>void hpdf_threshold_monitor_filter_config(hpdf_channel_enum channelx, uint32_t order, uint8_t oversample);</code>
Function descriptions	configure threshold monitor filter order and oversample
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum <code>hpdf_channel_enum</code>
Input parameter{in}	
order	threshold monitor Sinc filter order
<i>TM_FASTSINC</i>	FastSinc filter type
<i>TM_SINC1</i>	Sinc1 filter type
<i>TM_SINC2</i>	Sinc2 filter type
<i>TM_SINC3</i>	Sinc3 filter type
Input parameter{in}	
oversample	Sinc filter oversampling rate(1-32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor filter order and oversample */
```

```
hpdf_threshold_monitor_filter_config (FLT0, TM_SINC3);
```

hpdf_threshold_monitor_filter_read_data

The description of `hpdf_threshold_monitor_filter_read_data` is shown as below:

Table 3-815. Function `hpdf_threshold_monitor_filter_read_data`

Function name	<code>hpdf_threshold_monitor_filter_read_data</code>
Function prototype	<code>int16_t hpdf_threshold_monitor_filter_read_data(hpdf_channel_enum channelx);</code>
Function descriptions	read the threshold monitor filter data
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum <code>hpdf_channel_enum</code>
Output parameter{out}	
-	-

Return value	
int16_t	the threshold monitor filter data

Example:

```
/* read the threshold monitor filter data */
```

```
int16_t data;
```

```
data = hpdf_threshold_monitor_filter_read_data(CHANNEL0);
```

hpdf_threshold_monitor_fast_mode_disable

The description of hpdf_threshold_monitor_fast_mode_disable is shown as below:

Table 3-816. Function hpdf_threshold_monitor_fast_mode_disable

Function name	hpdf_threshold_monitor_fast_mode_disable
Function prototype	void hpdf_threshold_monitor_fast_mode_disable(hpdf_filter_enum filtery);
Function descriptions	disable threshold monitor fast mode
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable threshold monitor fast mode */
```

```
hpdf_threshold_monitor_fast_mode_disable (FLT0);
```

hpdf_threshold_monitor_fast_mode_enable

The description of hpdf_threshold_monitor_fast_mode_enable is shown as below:

Table 3-817. Function hpdf_threshold_monitor_fast_mode_enable

Function name	hpdf_threshold_monitor_fast_mode_enable
Function prototype	void hpdf_threshold_monitor_fast_mode_enable(hpdf_filter_enum filtery);
Function descriptions	enable threshold monitor fast mode
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable threshold monitor fast mode */
hpdf_threshold_monitor_fast_mode_enable(FLT0);
```

hpdf_threshold_monitor_channel

The description of hpdf_threshold_monitor_channel is shown as below:

Table 3-818. Function hpdf_threshold_monitor_channel

Function name	hpdf_threshold_monitor_channel
Function prototype	void hpdf_threshold_monitor_channel(hpdf_filter_enum filtery, uint32_t channel);
Function descriptions	configure threshold monitor channel
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
channel	which channel use threshold monitor x
TMCHEN_DISABLE	extremes monitor x does not accept data from any channel
TMCHEN_CHANNELx	extremes monitor accepts data from channel x
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor channel */
hpdf_threshold_monitor_channel(FLT0, TMCHEN_CHANNEL0_1);
```

hpdf_threshold_monitor_high_threshold

The description of hpdf_threshold_monitor_high_threshold is shown as below:

Table 3-819. Function hpdf_threshold_monitor_high_threshold

Function name	hpdf_threshold_monitor_high_threshold
Function prototype	void hpdf_threshold_monitor_channel(hpdf_filter_enum filtery, uint32_t channel);

Function descriptions	configure threshold monitor high threshold value
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
value	high threshold value(-8388608~8388607)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor high threshold value */
hpdf_threshold_monitor_high_threshold(FLT0, 32767);
```

hpdf_threshold_monitor_low_threshold

The description of hpdf_threshold_monitor_low_threshold is shown as below:

Table 3-820. Function hpdf_threshold_monitor_low_threshold

Function name	hpdf_threshold_monitor_low_threshold
Function prototype	void hpdf_threshold_monitor_low_threshold(hpdf_filter_enum filtery, int32_t value);
Function descriptions	configure threshold monitor low threshold value
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
value	low threshold value(-8388608~8388607)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor low threshold value */
hpdf_threshold_monitor_low_threshold(FLT0, -32768);
```

hpdf_high_threshold_break_signal

The description of hpdf_high_threshold_break_signal is shown as below:

Table 3-821. Function hpdf_high_threshold_break_signal

Function name	hpdf_high_threshold_break_signal
Function prototype	void hpdf_high_threshold_break_signal(hpdf_filter_enum filtery, uint32_t break_signal);
Function descriptions	configure threshold monitor high threshold event break signal
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
break_signal	HPDF break signal
<i>NO_TM_HT_BREAK</i>	break signal is not distributed to an threshold monitor high threshold event
<i>TM_HT_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 3-822. Function hpdf_low_threshold_break_signal

Function name	hpdf_low_threshold_break_signal
Function prototype	void hpdf_low_threshold_break_signal(hpdf_filter_enum filtery, uint32_t break_signal);
Function descriptions	configure threshold monitor low threshold event break signal
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
break_signal	HPDF break signal

<i>NO_TM_LT_BREAK</i>	break signal is not distributed to an threshold monitor low threshold event
<i>TM_LT_BREAKx</i> (<i>x=0..3</i>)	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 3-823. Function hpdf_extremes_monitor_channel

Function name	hpdf_extremes_monitor_channel
Function prototype	void hpdf_extremes_monitor_channel(hpdf_filter_enum filtery, uint32_t channel);
Function descriptions	configure extremes monitor channel
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy</i> (<i>y=0..3</i>)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
channel	which channel use extremes monitor y (<i>y=0..3</i>)
<i>EM_CHANNEL_DISABLE</i>	extremes monitor y does not accept data from any channel
<i>EM_CHANNELx</i> (<i>x=0..7</i>)	extremes monitor accepts data from channel x
Output parameter{out}	
-	-
Return value	
-	-

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 3-824. Function hpdf_extremes_monitor_maximum_get

Function name	hpdf_extremes_monitor_maximum_get
Function prototype	int32_t hpdf_extremes_monitor_maximum_get(hpdf_filter_enum filtery);
Function descriptions	get the extremes monitor maximum value
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
int32_t	the maximum value

Example:

```
int32_t vlaue;
```

```
vlaue = hpdf_extremes_monitor_maximum_get(FTL0);
```

hpdf_extremes_monitor_minimum_get

The description of hpdf_extremes_monitor_minimum_get is shown as below:

Table 3-825. Function hpdf_extremes_monitor_minimum_get

Function name	hpdf_extremes_monitor_minimum_get
Function prototype	int32_t hpdf_extremes_monitor_minimum_get(hpdf_filter_enum filtery);
Function descriptions	get the extremes monitor minimum value
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
int32_t	the minimum value

Example:

```
/* get the extremes monitor minimum value */
```

```
int32_t vlaue;
```

```
vlaue = hpdf_extremes_monitor_minimum_get(FTL0,);
```

hpdf_conversion_time_get

The description of hpdf_conversion_time_get is shown as below:

Table 3-826. Function hpdf_conversion_time_get

Function name	hpdf_conversion_time_get
Function prototype	uint32_t hpdf_conversion_time_get(hpdf_filter_enum filtery);
Function descriptions	get the conversion timer value
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
int32_t	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 3-827. Function hpdf_rc_continuous_disable

Function name	hpdf_rc_continuous_disable
Function prototype	void hpdf_rc_continuous_disable(hpdf_filter_enum filtery);
Function descriptions	disable regular conversions continuous mode
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:


```
/* disable regular conversions continuous mode */
```

```
hpdf_rc_continuous_disable(FTL0);
```

hpdf_rc_continuous_enable

The description of hpdf_rc_continuous_enable is shown as below:

Table 3-828. Function hpdf_rc_continuous_enable

Function name	hpdf_rc_continuous_enable
Function prototype	void hpdf_rc_continuous_enable(hpdf_filter_enum filtery);
Function descriptions	enable regular conversions continuous mode
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable regular conversions continuous mode */
```

```
hpdf_rc_continuous_enable(FTL0);
```

hpdf_rc_start_by_software

The description of hpdf_rc_start_by_software is shown as below:

Table 3-829. Function hpdf_rc_start_by_software

Function name	hpdf_rc_start_by_software
Function prototype	void hpdf_rc_start_by_software(hpdf_filter_enum filtery);
Function descriptions	start regular channel conversion by software
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start regular channel conversion by software */
```

```
hpdf_rc_start_by_software(FTL0);
```

hpdf_rc_syn_disable

The description of hpdf_rc_syn_disable is shown as below:

Table 3-830. Function hpdf_rc_syn_disable

Function name	hpdf_rc_syn_disable
Function prototype	void hpdf_rc_syn_disable(hpdf_filter_enum filtery);
Function descriptions	disable regular conversion synchronously
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable regular conversion synchronously */
```

```
hpdf_rc_syn_disable(FTL0);
```

hpdf_rc_syn_enable

The description of hpdf_rc_syn_enable is shown as below:

Table 3-831. Function hpdf_rc_syn_enable

Function name	hpdf_rc_syn_enable
Function prototype	void hpdf_rc_syn_enable(hpdf_filter_enum filtery);
Function descriptions	enable regular conversion synchronously
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable regular conversion synchronously */
```

```
hpdf_rc_syn_enable(FTL0);
```

hpdf_rc_dma_disable

The description of hpdf_rc_dma_disable is shown as below:

Table 3-832. Function hpdf_rc_dma_disable

Function name	hpdf_rc_dma_disable
Function prototype	void hpdf_rc_dma_disable(hpdf_filter_enum filtery);
Function descriptions	disable regular conversion DMA channel
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable regular conversion DMA channel */
```

```
hpdf_rc_dma_disable(FTL0);
```

hpdf_rc_dma_enable

The description of hpdf_rc_dma_enable is shown as below:

Table 3-833. Function hpdf_rc_dma_enable

Function name	hpdf_rc_dma_enable
Function prototype	void hpdf_rc_dma_enable(hpdf_filter_enum filtery);
Function descriptions	enable regular conversion DMA channel
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable regular conversion DMA channel */
```

```
hpdf_rc_dma_enable(FTL0);
```

hpdf_rc_channel_config

The description of hpdf_rc_channel_config is shown as below:

Table 3-834. Function hpdf_rc_channel_config

Function name	hpdf_rc_channel_config
Function prototype	void hpdf_rc_channel_config(hpdf_filter_enum filtery, hpdf_channel_enum channelx);
Function descriptions	configure regular conversion channel
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to Table 3-768. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure regular conversion channel */
```

```
hpdf_rc_channel_config(FTL0, CHANNEL1);
```

hpdf_rc_fast_mode_disable

The description of hpdf_rc_fast_mode_disable is shown as below:

Table 3-835. Function hpdf_rc_fast_mode_disable

Function name	hpdf_rc_fast_mode_disable
Function prototype	void hpdf_rc_fast_mode_disable(hpdf_filter_enum filtery);
Function descriptions	disable regular conversion fast conversion mode
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable regular conversion fast conversion mode */
```

```
hpdf_rc_fast_mode_disable(FTL0);
```

hpdf_rc_fast_mode_enable

The description of hpdf_rc_fast_mode_enable is shown as below:

Table 3-836. Function hpdf_rc_fast_mode_enable

Function name	hpdf_rc_fast_mode_enable
Function prototype	void hpdf_rc_fast_mode_enable(hpdf_filter_enum filtery);
Function descriptions	enable regular conversion fast conversion mode
Precondition	disable FLT _y (y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLT _y (y=0..3)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable regular conversion fast conversion mode */
```

```
hpdf_rc_fast_mode_enable(FTL0);
```

hpdf_rc_data_get

The description of hpdf_rc_data_get is shown as below:

Table 3-837. Function hpdf_rc_data_get

Function name	hpdf_rc_data_get
Function prototype	int32_t hpdf_rc_data_get(hpdf_filter_enum filtery);
Function descriptions	get the regular conversion data
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLT _y (y=0..3)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum

Output parameter{out}	
-	-
Return value	
int32_t	regular conversions data

Example:

```
/* get the regular conversion data */
```

```
int32_t vlaue;
```

```
vlaue = hpdf_rc_data_get(FTL0);
```

hpdf_rc_channel_get

The description of hpdf_rc_channel_get is shown as below:

Table 3-838. Function hpdf_rc_channel_get

Function name	hpdf_rc_channel_get
Function prototype	uint8_t hpdf_rc_channel_get(hpdf_filter_enum filtery);
Function descriptions	get the channel of regular channel most recently converted
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
uint8_t	the channel

Example:

```
/* get the channel of regular channel most recently converted */
```

```
uint8_t channel;
```

```
channel = hpdf_rc_channel_get(FTL0);
```

hpdf_ic_start_by_software

The description of hpdf_ic_start_by_software is shown as below:

Table 3-839. Function hpdf_ic_start_by_software

Function name	hpdf_ic_start_by_software
Function prototype	void hpdf_ic_start_by_software(hpdf_filter_enum filtery);
Function descriptions	start inserted channel conversion by software
Precondition	-
The called functions	-

Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start inserted channel conversion by software */
```

```
hpdf_ic_start_by_software(FTL0);
```

hpdf_ic_syn_disable

The description of hpdf_ic_syn_disable is shown as below:

Table 3-840. Function hpdf_ic_syn_disable

Function name	hpdf_ic_syn_disable
Function prototype	void hpdf_ic_syn_disable(hpdf_filter_enum filtery);
Function descriptions	disable inserted conversion synchronously
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable inserted conversion synchronously */
```

```
hpdf_ic_syn_disable(FTL0);
```

hpdf_ic_syn_enable

The description of hpdf_ic_syn_enable is shown as below:

Table 3-841. 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}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable inserted conversion synchronously */
```

```
hpdf_ic_syn_enable(FTL0);
```

hpdf_ic_dma_disable

The description of hpdf_ic_dma_disable is shown as below:

Table 3-842. 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}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable inserted conversion DMA channel */
```

```
hpdf_ic_dma_disable(FTL0);
```

hpdf_ic_dma_enable

The description of hpdf_ic_dma_enable is shown as below:

Table 3-843. Function hpdf_ic_dma_enable

Function name	hpdf_ic_dma_enable
Function prototype	void hpdf_ic_dma_enable(hpdf_filter_enum filtery);
Function descriptions	enable inserted conversion DMA channel

Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable inserted conversion DMA channel */
```

```
hpdf_ic_dma_enable(FTL0);
```

hpdf_ic_scan_mode_disable

The description of hpdf_ic_scan_mode_disable is shown as below:

Table 3-844. Function hpdf_ic_scan_mode_disable

Function name	hpdf_ic_scan_mode_disable
Function prototype	void hpdf_ic_scan_mode_disable(hpdf_filter_enum filtery);
Function descriptions	disable scan conversion mode
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable scan conversion mode */
```

```
hpdf_ic_scan_mode_disable(FTL0);
```

hpdf_ic_scan_mode_enable

The description of hpdf_ic_scan_mode_enable is shown as below:

Table 3-845. 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}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
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 3-846. 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}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
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 3-847. Function hpdf_ic_trigger_signal_config

Function name	hpdf_ic_trigger_signal_config
----------------------	-------------------------------

Function prototype	void hpdf_ic_trigger_signal_config(hpdf_filter_enum filtery, uint32_t trigger, uint32_t trigger_edge);
Function descriptions	configure inserted conversions trigger signal and trigger edge
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
trigger	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}	
trigger_edge	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 3-848. Function `hpdf_ic_channel_config`

Function name	<code>hpdf_ic_channel_config</code>
Function prototype	<code>void hpdf_ic_channel_config(hpdf_filter_enum filtery, uint32_t channel);</code>
Function descriptions	configure inserted group conversions channel
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum <code>hpdf_filter_enum</code>
Input parameter{in}	
channel	the HPDF channel belongs to inserted group
<i>IGCSEL_CHANNELx</i> (<i>x=0..7</i>)	Channel x belongs to the inserted group
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure inserted group conversions channel */
hpdf_ic_channel_config(FTL0, IGCSEL_CHANNEL1);
```

`hpdf_ic_data_get`

The description of `hpdf_ic_data_get` is shown as below:

Table 3-849. Function `hpdf_ic_data_get`

Function name	<code>hpdf_ic_data_get</code>
Function prototype	<code>int32_t hpdf_ic_data_get(hpdf_filter_enum filtery);</code>
Function descriptions	get the inserted conversions data
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum <code>hpdf_filter_enum</code>
Output parameter{out}	
-	-
Return value	
int32_t	inserted conversions data

Example:

```
/* get the inserted conversions data */
int32_t vlaue;
```

```
vlaue = hpdf_ic_data_get(FTL0);
```

hpdf_ic_channel_get

The description of hpdf_ic_channel_get is shown as below:

Table 3-850. Function hpdf_ic_channel_get

Function name	hpdf_ic_channel_get
Function prototype	uint8_t hpdf_ic_channel_get(hpdf_filter_enum filtery);
Function descriptions	get the channel of inserted group channel most recently converted
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
uint8_t	the channel

Example:

```
/* get the channel of inserted group channel most recently converted */
```

```
uint8_t channel;
```

```
channel = hpdf_ic_channel_get(FTL0);
```

hpdf_flag_get

The description of hpdf_flag_get is shown as below:

Table 3-851. Function hpdf_flag_get

Function name	hpdf_flag_get
Function prototype	FlagStatus hpdf_flag_get(hpdf_filter_enum filtery, hpdf_flag_enum flag);
Function descriptions	get the HPDF flags
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
flag	HPDF flags, refer to Table 3-770. Enum hpdf_flag_enum
<i>HPDF_FLAG_FLTy_IC</i> <i>EF</i>	FLTy inserted conversion end flag
<i>HPDF_FLAG_FLTy_RC</i>	FLTy regular conversion end flag

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

Example:

```
/* get the FLT0 threshold monitor event occurred flag */
```

```
hpdf_flag_get(FTL0, HPDF_FLAG_FLTy_TMEOF);
```

hpdf_flag_clear

The description of hpdf_flag_clear is shown as below:

Table 3-852. Function hpdf_flag_clear

Function name	hpdf_flag_clear
Function prototype	void hpdf_flag_clear(hpdf_filter_enum filtery, hpdf_flag_enum flag);
Function descriptions	clear the HPDF flags
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum

Input parameter{in}	
flag	HPDF flags, refer to Table 3-770. Enum hpdf_flag_enum
<i>HPDF_FLAG_FLTy_IC EF</i>	FLTy inserted conversion end flag
<i>HPDF_FLAG_FLTy_RC EF</i>	FLTy regular conversion end flag
<i>HPDF_FLAG_FLTy_IC DOF</i>	FLTy inserted conversion data overflow flag
<i>HPDF_FLAG_FLTy_RC DOF</i>	FLTy threshold monitor event occurred flag
<i>HPDF_FLAG_FLTy_TM EOF</i>	FLTy inserted conversion in progress flag
<i>HPDF_FLAG_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_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
Output parameter{out}	
-	-
Return value	
-	-

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 3-853. Function hpdf_interrupt_enable

Function name	hpdf_interrupt_enable
Function prototype	void hpdf_interrupt_enable(hpdf_filter_enum filtery, hpdf_interrput_enum interrupt);
Function descriptions	enable HPDF interrupt
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum

Input parameter{in}	
interrupt	HPDF interrupts, refer to Table 3-772. Enum hpdf_interrupt_enum
HPDF_INT_FLTy_ICEI E	FLTy inserted conversion end interrupt enable
HPDF_INT_FLTy_RCEI E	FLTy regular conversion end interrupt enable
HPDF_INT_FLTy_ICD OIE	FLTy inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_RCD OIE	FLTy regular conversion data overflow interrupt enable
HPDF_INT_FLTy_TMIE	FLTy threshold monitor interrupt enable
HPDF_INT_FLT0_MMI E	malfunction monitor interrupt enable
HPDF_INT_FLT0_CKLI E	clock loss interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable FLT0 threshold monitor interrupt */
```

```
hpdf_interrupt_enable(FTL0, HPDF_INT_FLTy_TMEIOIE);
```

hpdf_interrupt_disable

The description of hpdf_interrupt_disable is shown as below:

Table 3-854. Function hpdf_interrupt_disable

Function name	hpdf_interrupt_disable
Function prototype	void hpdf_interrupt_disable(hpdf_filter_enum filtery, hpdf_interrupt_enum interrupt);
Function descriptions	disable HPDF interrupt
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
interrupt	HPDF interrupts, refer to Table 3-772. Enum hpdf_interrupt_enum
HPDF_INT_FLTy_ICEI E	FLTy inserted conversion end interrupt enable
HPDF_INT_FLTy_RCEI	FLTy regular conversion end interrupt enable

E	
HPDF_INT_FLTy_ICD OIE	FLTy inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_RCD OIE	FLTy regular conversion data overflow interrupt enable
HPDF_INT_FLTy_TMIE	FLTy threshold monitor interrupt enable
HPDF_INT_FLT0_MMI E	malfunction monitor interrupt enable
HPDF_INT_FLT0_CKLI E	clock loss interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable FLT0 threshold monitor interrupt */
```

```
hpdf_interrupt_disable(FLT0, HPDF_INT_FLT0_CKLI);
```

hpdf_interrupt_flag_get

The description of hpdf_interrupt_flag_get is shown as below:

Table 3-855. Function hpdf_interrupt_flag_get

Function name	hpdf_interrupt_flag_get
Function prototype	FlagStatus hpdf_interrupt_flag_get(hpdf_filter_enum filtery, hpdf_interrput_flag_enum int_flag);
Function descriptions	get the HPDF interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
int_flag	HPDF flags, refer to Table 3-771. Enum hpdf_interrput_flag_enum
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
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the the clock loss interrupt flag */
hpdf_interrupt_flag_get (FTL0, HPDF_INT_FLAG_FLT0_CKLF0);
```

hpdf_interrupt_flag_clear

The description of hpdf_interrupt_flag_clear is shown as below:

Table 3-856. Function hpdf_interrupt_flag_clear

Function name	hpdf_interrupt_flag_clear
Function prototype	void hpdf_interrupt_flag_clear(hpdf_filter_enum filtery, hpdf_interrput_flag_enum int_flag);
Function descriptions	clear the HPDF interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to Table 3-769. Enum hpdf_filter_enum
Input parameter{in}	
int_flag	HPDF flags, refer to Table 3-771. Enum hpdf_interrput_flag_enum
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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the the clock loss interrupt flag */
```

```
hpdf_interrupt_flag_get(FTL0, HPDF_INT_FLAG_FLT0_CKLF0);
```

3.24. HWSEM

Hardware semaphore (HWSEM) provides a non-blocking mechanism to ensure the synchronous of processes. The HWSEM registers are listed in chapter [3.24.1](#), the HWSEM firmware functions are introduced in chapter [3.24.2](#)

3.24.1. Descriptions of Peripheral registers

HWSEM registers are listed in the table shown as below:

Table 3-857. HWSEM Registers

Registers	Descriptions
HWSEM_CTLx (x = 0..31)	HWSEM control register x
HWSEM_RLKx (x = 0..31)	HWSEM read lock register x
HWSEM_INTEN	HWSEM interrupt enable register
HWSEM_INTC	HWSEM interrupt flag clear register
HWSEM_STAT	HWSEM status register
HWSEM_INTF	HWSEM interrupt flag register
HWSEM_UNLK	HWSEM unlock register
HWSEM_KEY	HWSEM key register

3.24.2. Descriptions of Peripheral functions

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

Table 3-858. HWSEM firmware function

Function name	Function description
hwsem_lock_set	try to lock the specific semaphore by writing process ID
hwsem_lock_release	try to release the lock of the semaphore by writing process ID
hwsem_lock_by_reading	try to lock the semaphore by reading

Function name	Function description
hwsem_unlock_all	unlock all semaphores of the master ID
hwsem_process_id_get	get process ID of the specific semaphore
hwsem_master_id_get	get master ID of the specific semaphore
hwsem_lock_status_get	get the lock status of the semaphore
hwsem_key_set	set the key
hwsem_key_get	get the key
hwsem_flag_get	get the HWSEM flag status
hwsem_flag_clear	clear HWSEM flag status
hwsem_interrupt_flag_get	get HWSEM interrupt flag status
hwsem_interrupt_flag_clear	clear HWSEM interrupt flag
hwsem_interrupt_enable	enable HWSEM interrupt
hwsem_interrupt_disable	disable HWSEM interrupt

Enum hwsem_semaphore_enum

Table 3-859. Enum hwsem_semaphore_enum

Member name	Function description
SEM0	semaphore 0
SEM1	semaphore 1
SEM2	semaphore 2
SEM3	semaphore 3
SEM4	semaphore 4
SEM5	semaphore 5
SEM6	semaphore 6
SEM7	semaphore 7
SEM8	semaphore 8
SEM9	semaphore 9
SEM10	semaphore 10
SEM11	semaphore 11
SEM12	semaphore 12
SEM13	semaphore 13
SEM14	semaphore 14
SEM15	semaphore 15
SEM16	semaphore 16
SEM17	semaphore 17
SEM18	semaphore 18
SEM19	semaphore 19
SEM20	semaphore 20
SEM21	semaphore 21
SEM22	semaphore 22
SEM23	semaphore 23
SEM24	semaphore 24

Member name	Function description
SEM25	semaphore 25
SEM26	semaphore 26
SEM27	semaphore 27
SEM28	semaphore 28
SEM29	semaphore 29
SEM30	semaphore 30
SEM31	semaphore 31

hwsem_lock_set

The description of hwsem_lock_set is shown as below:

Table 3-860. Function hwsem_lock_set

Function name	hwsem_lock_set
Function prototype	ErrStatus hwsem_lock_set(hwsem_semaphore_enum semaphore, uint8_t process);
Function descriptions	try to lock the specific semaphore by writing process ID
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
SEMx	x=0..31
Input parameter{in}	
process	the process to lock the semaphore
0..0xFF	process ID
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

ErrStatus state;

```
/* lock semaphore 0 by process 0 */
```

```
state = hwsem_lock_set(SEM0, 0);
```

hwsem_lock_release

The description of hwsem_lock_release is shown as below:

Table 3-861. Function hwsem_lock_release

Function name	hwsem_lock_release
----------------------	--------------------

Function prototype	ErrStatus hwsem_lock_release(hwsem_semaphore_enum semaphore, uint8_t process);
Function descriptions	try to release the lock of the semaphore by writing process ID
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
<i>SEMx</i>	x=0..31
Input parameter{in}	
process	the process to unlock the semaphore
<i>0..0xFF</i>	process ID
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

ErrStatus state;

```
/* unlock semaphore 0 by process 0 */
```

```
state = hwsem_lock_release(SEM0, 0);
```

hwsem_lock_by_reading

The description of hwsem_lock_by_reading is shown as below:

Table 3-862. Function hwsem_lock_by_reading

Function name	hwsem_lock_by_reading
Function prototype	ErrStatus hwsem_lock_by_reading(hwsem_semaphore_enum semaphore);
Function descriptions	try to lock the semaphore by reading
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
<i>SEMx</i>	x=0..31
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

ErrStatus state;

```
/* unlock semaphore 0 by reading */
```

```
state = hwsem_lock_by_reading(SEM0);
```

hwsem_unlock_all

The description of hwsem_unlock_all is shown as below:

Table 3-863. Function hwsem_unlock_all

Function name	hwsem_unlock_all
Function prototype	ErrStatus hwsem_unlock_all(uint16_t key);
Function descriptions	unlock all semaphores of the master ID
Precondition	-
The called functions	-
Input parameter{in}	
key	key value
0 - 0xFFFF	key value to unlock
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
ErrStatus state;
```

```
/* unlock all semaphores */
```

```
state = hwsem_unlock_all(0);
```

hwsem_process_id_get

The description of hwsem_process_id_get is shown as below:

Table 3-864. Function hwsem_process_id_get

Function name	hwsem_process_id_get
Function prototype	uint32_t hwsem_process_id_get(hwsem_semaphore_enum semaphore);
Function descriptions	get process ID of the specific semaphore
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
SEMx	x=0..31
Output parameter{out}	
-	-
Return value	

uint32_t	0 - 0xFF
----------	----------

Example:

```
uint32_t pid = 0;

/* get process ID of semaphore 0 */

pid = hwsem_process_id_get(SEM0);
```

hwsem_master_id_get

The description of hwsem_master_id_get is shown as below:

Table 3-865. Function hwsem_master_id_get

Function name	hwsem_master_id_get
Function prototype	uint32_t hwsem_master_id_get(hwsem_semaphore_enum semaphore);
Function descriptions	get master ID of the specific semaphore
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
SEMx	x=0..31
Output parameter{out}	
-	-
Return value	
uint32_t	0 - 0xF

Example:

```
uint32_t mid = 0;

/* get master ID of semaphore 0 */

mid = hwsem_master_id_get(SEM0);
```

hwsem_lock_status_get

The description of hwsem_lock_status_get is shown as below:

Table 3-866. Function hwsem_lock_status_get

Function name	hwsem_lock_status_get
Function prototype	FlagStatus hwsem_lock_status_get(hwsem_semaphore_enum semaphore);
Function descriptions	get the lock status of the semaphore
Precondition	-
The called functions	-
Input parameter{in}	

semaphore	semaphore index, refer to enum Table 3-859. Enum <i>hwsem_semaphore</i> enum
<i>SEMx</i>	x=0..31
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

FlagStatus flag;

/ get the lock status */*

flag = hwsem_lock_status_get(SEM0);

hwsem_key_set

The description of hwsem_key_set is shown as below:

Table 3-867. Function hwsem_key_set

Function name	hwsem_key_set
Function prototype	void hwsem_key_set(uint16_t key);
Function descriptions	set the key
Precondition	-
The called functions	-
Input parameter{in}	
key	key value
<i>0 - 0xFFFF</i>	key value to set
Output parameter{out}	
-	-
Return value	
-	-

Example:

/ set unlock key */*

hwsem_key_set(0);

hwsem_key_get

The description of hwsem_key_get is shown as below:

Table 3-868. Function hwsem_key_get

Function name	hwsem_key_get
Function prototype	uint16_t hwsem_key_get(void);
Function descriptions	get the key

Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	0 – 0xFFFF

Example:

```
uint16_t key;
```

```
/* get the unlock key */
```

```
key = hwsem_key_get();
```

hwsem_flag_get

The description of hwsem_flag_get is shown as below:

Table 3-869. Function hwsem_flag_get

Function name	hwsem_flag_get
Function prototype	FlagStatus hwsem_flag_get(hwsem_semaphore_enum semaphore);
Function descriptions	get the HWSEM flag status
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
<i>SEMx</i>	x=0..31
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus flag;
```

```
/* get the HWSEM unlock flag status */
```

```
flag = hwsem_flag_get(SEM0);
```

hwsem_flag_clear

The description of hwsem_flag_clear is shown as below:

Table 3-870. Function hwsem_flag_clear

Function name	hwsem_flag_clear
Function prototype	void hwsem_flag_clear(hwsem_semaphore_enum semaphore);
Function descriptions	clear HWSEM flag status
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
SEMx	x=0..31
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear HWSEM unlock flag status */
```

```
hwsem_flag_clear(SEM0);
```

hwsem_interrupt_flag_get

The description of hwsem_interrupt_flag_get is shown as below:

Table 3-871. Function hwsem_interrupt_flag_get

Function name	hwsem_interrupt_flag_get
Function prototype	FlagStatus hwsem_interrupt_flag_get(hwsem_semaphore_enum semaphore);
Function descriptions	get HWSEM interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
SEMx	x=0..31
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus flag;
```

```
/* get the HWSEM unlock interrupt flag status */
```

```
flag = hwsem_interrupt_flag_get(SEM0);
```

hwsem_interrupt_flag_clear

The description of hwsem_interrupt_flag_clear is shown as below:

Table 3-872. Function hwsem_interrupt_flag_clear

Function name	hwsem_interrupt_flag_clear
Function prototype	void hwsem_interrupt_flag_clear(hwsem_semaphore_enum semaphore);
Function descriptions	clear HWSEM interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
<i>SEM_x</i>	x=0..31
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear HWSEM unlock interrupt flag status */
```

```
hwsem_interrupt_flag_clear(SEM0);
```

hwsem_interrupt_enable

The description of hwsem_interrupt_enable is shown as below:

Table 3-873. Function hwsem_interrupt_enable

Function name	hwsem_interrupt_enable
Function prototype	void hwsem_interrupt_enable(hwsem_semaphore_enum semaphore);
Function descriptions	enable HWSEM interrupt
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
<i>SEM_x</i>	x=0..31
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable HWSEM unlock interrupt */
```

```
hwsem_interrupt_enable(SEM0);
```

hwsem_interrupt_disable

The description of hwsem_interrupt_disable is shown as below:

Table 3-874. Function hwsem_interrupt_disable

Function name	hwsem_interrupt_disable
Function prototype	void hwsem_interrupt_disable(hwsem_semaphore_enum semaphore);
Function descriptions	disable HWSEM interrupt
Precondition	-
The called functions	-
Input parameter{in}	
semaphore	semaphore index, refer to enum Table 3-859. Enum hwsem_semaphore_enum
<i>SEM_x</i>	x=0..31
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable HWSEM unlock interrupt */
```

```
hwsem_interrupt_disable(SEM0);
```

3.25. 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 [3.25.1](#), the I2C firmware functions are introduced in chapter [3.25.2](#).

3.25.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

Table 3-875. 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

Registers	Descriptions
I2C_TIMING	I2C timing register
I2C_TIMEOUT	I2C timeout register
I2C_STAT	I2C status register
I2C_STATC	I2C status clear register
I2C_PEC	I2C PEC register
I2C_RDATA	I2C receive data register
I2C_TDATA	I2C transmit data register
I2C_CTL2	I2C control register 2

3.25.2. Descriptions of Peripheral functions

I2C firmware functions are listed in the table shown as below:

Table 3-876. 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

Function name	Function description
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
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

Function name	Function description
i2c_interrupt_flag_get	get I2C interrupt flag status
i2c_interrupt_flag_clear	clear I2C interrupt flag status

Enum i2c_interrupt_flag_enum

Table 3-877. 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 3-878. Function i2c_deinit

Function name	i2c_deinit
Function prototype	void i2c_deinit(uint32_t i2c_periph);
Function descriptions	reset I2C
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset I2C0 */
i2c_deinit(I2C0);
```


i2c_timing_config

The description of i2c_timing_config is shown as below:

Table 3-879. Function i2c_timing_config

Function name	i2c_timing_config
Function prototype	void i2c_timing_config(uint32_t i2c_periph, uint32_t psc, uint32_t scl_dely, uint32_t sda_dely);
Function descriptions	configure the timing parameters
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-880. Function i2c_digital_noise_filter_config

Function name	i2c_digital_noise_filter_config
Function prototype	void i2c_digital_noise_filter_config(uint32_t i2c_periph, uint32_t filter_length);
Function descriptions	configure digital noise filter
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	

filter_length	filter_length
<i>FILTER_DISABLE</i>	digital filter is disabled
<i>FILTER_LENGTH_1</i>	digital filter is enabled and filter spikes with a length of up to 1 t_{I2CCLK}
<i>FILTER_LENGTH_2</i>	digital filter is enabled and filter spikes with a length of up to 2 t_{I2CCLK}
<i>FILTER_LENGTH_3</i>	digital filter is enabled and filter spikes with a length of up to 3 t_{I2CCLK}
<i>FILTER_LENGTH_4</i>	digital filter is enabled and filter spikes with a length of up to 4 t_{I2CCLK}
<i>FILTER_LENGTH_5</i>	digital filter is enabled and filter spikes with a length of up to 5 t_{I2CCLK}
<i>FILTER_LENGTH_6</i>	digital filter is enabled and filter spikes with a length of up to 6 t_{I2CCLK}
<i>FILTER_LENGTH_7</i>	digital filter is enabled and filter spikes with a length of up to 7 t_{I2CCLK}
<i>FILTER_LENGTH_8</i>	digital filter is enabled and filter spikes with a length of up to 8 t_{I2CCLK}
<i>FILTER_LENGTH_9</i>	digital filter is enabled and filter spikes with a length of up to 9 t_{I2CCLK}
<i>FILTER_LENGTH_10</i>	digital filter is enabled and filter spikes with a length of up to 10 t_{I2CCLK}
<i>FILTER_LENGTH_11</i>	digital filter is enabled and filter spikes with a length of up to 11 t_{I2CCLK}
<i>FILTER_LENGTH_12</i>	digital filter is enabled and filter spikes with a length of up to 12 t_{I2CCLK}
<i>FILTER_LENGTH_13</i>	digital filter is enabled and filter spikes with a length of up to 13 t_{I2CCLK}
<i>FILTER_LENGTH_14</i>	digital filter is enabled and filter spikes with a length of up to 14 t_{I2CCLK}
<i>FILTER_LENGTH_15</i>	digital filter is enabled and filter spikes with a length of up to 15 t_{I2CCLK}
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 digital filter filters spikes with a length of up to 1  $t_{I2CCLK}$  */
```

```
i2c_digital_noise_filter_config(I2C0, FILTER_LENGTH_1);
```

i2c_analog_noise_filter_enable

The description of i2c_analog_noise_filter_enable is shown as below:

Table 3-881. Function i2c_analog_noise_filter_enable

Function name	i2c_analog_noise_filter_enable
Function prototype	void i2c_analog_noise_filter_enable(uint32_t i2c_periph);
Function descriptions	enable analog noise filter
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable analog noise filter */

i2c_analog_noise_filter_enable(I2C0);
```

i2c_analog_noise_filter_disable

The description of i2c_analog_noise_filter_disable is shown as below:

Table 3-882. Function i2c_analog_noise_filter_disable

Function name	i2c_analog_noise_filter_disable
Function prototype	void i2c_analog_noise_filter_disable(uint32_t i2c_periph);
Function descriptions	disable analog noise filter
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable analog noise filter */

i2c_analog_noise_filter_disable(I2C0);
```

i2c_master_clock_config

The description of i2c_master_clock_config is shown as below:

Table 3-883. Function i2c_master_clock_config

Function name	i2c_master_clock_config
Function prototype	void i2c_master_clock_config(uint32_t i2c_periph, uint32_t sclh, uint32_t scll);
Function descriptions	configure the SCL high and low period of clock in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
sclh	0-0xff, SCL high period
Input parameter{in}	

scll	0-0xff, SCL low period
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the SCL and SDA period of clock in master mode */
```

```
i2c_master_clock_config(I2C0, 0x0f, 0x0f);
```

i2c_master_addressing

The description of i2c_master_addressing is shown as below:

Table 3-884. Function i2c_master_addressing

Function name	i2c_master_addressing
Function prototype	void i2c_master_addressing(uint32_t i2c_periph, uint32_t address, uint32_t trans_direction);
Function descriptions	configure i2c slave addresss and transfer direction in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
address	0-0x3FF except reserved address, I2C slave address to be sent
Input parameter{in}	
trans_direction	I2C transfer direction in master mode
<i>I2C_MASTER_TRANSMIT</i>	master transmit
<i>I2C_MASTER_RECEIVE</i>	master receive
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* send slave address to I2C bus */
```

```
i2c_master_addressing(I2C0, 0x82, I2C_MASTER_TRANSMIT);
```

i2c_address10_header_enable

The description of i2c_address10_header_enable is shown as below:

Table 3-885. Function i2c_address10_header_enable

Function name	i2c_address10_header_enable
Function prototype	void i2c_address10_header_enable(uint32_t i2c_periph);
Function descriptions	10-bit address header executes read direction only in master receive mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-886. Function i2c_address10_header_disable

Function name	i2c_address10_header_disable
Function prototype	void i2c_address10_header_disable(uint32_t i2c_periph);
Function descriptions	10-bit address header executes complete sequence in master receive mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-887. Function i2c_address10_enable

Function name	i2c_address10_enable
Function prototype	void i2c_address10_enable(uint32_t i2c_periph);
Function descriptions	enable 10-bit addressing mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-888. Function i2c_address10_disable

Function name	i2c_address10_disable
Function prototype	void i2c_address10_disable(uint32_t i2c_periph);
Function descriptions	disable 10-bit addressing mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-889. Function i2c_automatic_end_enable

Function name	i2c_automatic_end_enable
Function prototype	void i2c_automatic_end_enable(uint32_t i2c_periph);
Function descriptions	enable I2C automatic end mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-890. Function i2c_automatic_end_disable

Function name	i2c_automatic_end_disable
Function prototype	void i2c_automatic_end_disable(uint32_t i2c_periph);
Function descriptions	disable I2C automatic end mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-891. Function i2c_slave_response_to_gcall_enable

Function name	i2c_slave_response_to_gcall_enable
Function prototype	void i2c_slave_response_to_gcall_enable(uint32_t i2c_periph);
Function descriptions	enable the response to a general call
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-892. Function i2c_slave_response_to_gcall_disable

Function name	i2c_slave_response_to_gcall_disable
Function prototype	void i2c_slave_response_to_gcall_disable(uint32_t i2c_periph);
Function descriptions	disable the response to a general call
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-893. Function i2c_stretch_scl_low_enable

Function name	i2c_stretch_scl_low_enable
Function prototype	void i2c_stretch_scl_low_enable(uint32_t i2c_periph);
Function descriptions	enable to stretch SCL low when data is not ready in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-894. Function i2c_stretch_scl_low_disable

Function name	i2c_stretch_scl_low_disable
Function prototype	void i2c_stretch_scl_low_disable(uint32_t i2c_periph);
Function descriptions	disable to stretch SCL low when data is not ready in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-895. Function i2c_address_config

Function name	i2c_address_config
Function prototype	void i2c_address_config(uint32_t i2c_periph, uint32_t address, uint32_t addr_format);
Function descriptions	configure i2c slave address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
address	I2C address
Input parameter{in}	
addr_format	7 bits or 10 bits
<i>I2C_ADDFORMAT_7BITS</i>	7bits
<i>I2C_ADDFORMAT_10BITS</i>	10bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure i2c slave address */
```

```
i2c_address_config(I2C0, 0x82, I2C_ADDFORMAT_7BITS);
```

i2c_address_bit_compare_config

The description of i2c_address_bit_compare_config is shown as below:

Table 3-896. Function i2c_address_bit_compare_config

Function name	i2c_address_bit_compare_config
Function prototype	void i2c_address_bit_compare_config(uint32_t i2c_periph, uint32_t compare_bits);
Function descriptions	define which bits of ADDRESS[7:1] need to compare with the incoming address byte
Precondition	-
The called functions	-
Input parameter{in}	

i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
compare_bits	the bits need to compare
<i>ADDRESS_BIT1_COMPARE</i>	address bit1 needs compare
<i>ADDRESS_BIT2_COMPARE</i>	address bit2 needs compare
<i>ADDRESS_BIT3_COMPARE</i>	address bit3 needs compare
<i>ADDRESS_BIT4_COMPARE</i>	address bit4 needs compare
<i>ADDRESS_BIT5_COMPARE</i>	address bit5 needs compare
<i>ADDRESS_BIT6_COMPARE</i>	address bit6 needs compare
<i>ADDRESS_BIT7_COMPARE</i>	address bit7 needs compare
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* bit 1 of ADDRESS[7:1] need to compare with the incoming address byte */
```

```
i2c_address_bit_compare_config(I2C0, ADDRESS_BIT1_COMPARE);
```

i2c_address_disable

The description of i2c_address_disable is shown as below:

Table 3-897. Function i2c_address_disable

Function name	i2c_address_disable
Function prototype	void i2c_address_disable(uint32_t i2c_periph);
Function descriptions	disable i2c address in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* disable i2c address in slave mode */
```

```
i2c_address_disable(I2C0);
```

i2c_second_address_config

The description of i2c_second_address_config is shown as below:

Table 3-898. Function i2c_second_address_config

Function name	i2c_second_address_config
Function prototype	void i2c_second_address_config(uint32_t i2c_periph, uint32_t address, uint32_t addr_mask);
Function descriptions	configure i2c second slave address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
address	I2C address
Input parameter{in}	
addr_mask	the bits not need to compare
<i>ADDRESS2_NO_MASK</i>	no mask, all the bits must be compared
<i>ADDRESS2_MASK_BIT1</i>	ADDRESS2[1] is masked, only ADDRESS2[7:2] are compared
<i>ADDRESS2_MASK_BIT1_2</i>	ADDRESS2[2:1] is masked, only ADDRESS2[7:3] are compared
<i>ADDRESS2_MASK_BIT1_3</i>	ADDRESS2[3:1] is masked, only ADDRESS2[7:4] are compared
<i>ADDRESS2_MASK_BIT1_4</i>	ADDRESS2[4:1] is masked, only ADDRESS2[7:5] are compared
<i>ADDRESS2_MASK_BIT1_5</i>	ADDRESS2[5:1] is masked, only ADDRESS2[7:6] are compared
<i>ADDRESS2_MASK_BIT1_6</i>	ADDRESS2[6:1] is masked, only ADDRESS2[7] are compared
<i>ADDRESS2_MASK_ALL</i>	all the ADDRESS2[7:1] bits are masked
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 3-899. Function i2c_second_address_disable

Function name	i2c_second_address_disable
Function prototype	void i2c_second_address_disable(uint32_t i2c_periph);
Function descriptions	disable i2c second address in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-900. Function i2c_receved_address_get

Function name	i2c_receved_address_get
Function prototype	uint32_t i2c_receved_address_get(uint32_t i2c_periph);
Function descriptions	get received match address in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-

Return value	
uint32_t	0x00..0x7F

Example:

```
/* get received match address in slave mode */
```

```
uint32_t address;
```

```
address = i2c_receivied_address_get(I2C0);
```

i2c_slave_byte_control_enable

The description of i2c_slave_byte_control_enable is shown as below:

Table 3-901. Function i2c_slave_byte_control_enable

Function name	i2c_slave_byte_control_enable
Function prototype	void i2c_slave_byte_control_enable(uint32_t i2c_periph);
Function descriptions	enable slave byte control
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable slave byte control */
```

```
i2c_slave_byte_control_enable(I2C0);
```

i2c_slave_byte_control_disable

The description of i2c_slave_byte_control_disable is shown as below:

Table 3-902. Function i2c_slave_byte_control_disable

Function name	i2c_slave_byte_control_disable
Function prototype	void i2c_slave_byte_control_disable(uint32_t i2c_periph);
Function descriptions	disable slave byte control
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable slave byte control */
```

```
i2c_slave_byte_control_disable(I2C0);
```

i2c_nack_enable

The description of i2c_nack_enable is shown as below:

Table 3-903. Function i2c_nack_enable

Function name	i2c_nack_enable
Function prototype	void i2c_nack_enable(uint32_t i2c_periph);
Function descriptions	generate a NACK in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* generate a NACK in slave mode */
```

```
i2c_nack_enable(I2C0);
```

i2c_nack_disable

The description of i2c_nack_disable is shown as below:

Table 3-904. Function i2c_nack_disable

Function name	i2c_nack_disable
Function prototype	void i2c_nack_disable(uint32_t i2c_periph);
Function descriptions	generate a ACK in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral

<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* generate a ACK in slave mode */
```

```
i2c_nack_disable(I2C0);
```

i2c_wakeup_from_deepsleep_enable

The description of i2c_wakeup_from_deepsleep_enable is shown as below:

Table 3-905. Function i2c_wakeup_from_deepsleep_enable

Function name	i2c_wakeup_from_deepsleep_enable
Function prototype	void i2c_wakeup_from_deepsleep_enable(uint32_t i2c_periph);
Function descriptions	enable wakeup from Deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable wakeup from Deep-sleep mode */
```

```
i2c_wakeup_from_deepsleep_enable(I2C0);
```

i2c_wakeup_from_deepsleep_disable

The description of i2c_wakeup_from_deepsleep_disable is shown as below:

Table 3-906. Function i2c_wakeup_from_deepsleep_disable

Function name	i2c_wakeup_from_deepsleep_disable
Function prototype	void i2c_wakeup_from_deepsleep_disable(uint32_t i2c_periph);
Function descriptions	disable wakeup from Deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	

i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable wakeup from Deep-sleep mode */
```

```
i2c_wakeup_from_deepsleep_disable(I2C0);
```

i2c_enable

The description of i2c_enable is shown as below:

Table 3-907. Function i2c_enable

Function name	i2c_enable
Function prototype	void i2c_enable(uint32_t i2c_periph);
Function descriptions	enable I2C
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C0 */
```

```
i2c_enable(I2C0);
```

i2c_disable

The description of i2c_disable is shown as below:

Table 3-908. Function i2c_disable

Function name	i2c_disable
Function prototype	void i2c_disable(uint32_t i2c_periph);
Function descriptions	disable I2C
Precondition	-
The called functions	-

Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C0 */
i2c_disable(I2C0);
```

i2c_start_on_bus

The description of i2c_start_on_bus is shown as below:

Table 3-909. Function i2c_start_on_bus

Function name	i2c_start_on_bus
Function prototype	void i2c_start_on_bus(uint32_t i2c_periph);
Function descriptions	generate a START condition on I2C bus
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 send a start condition to I2C bus */
i2c_start_on_bus(I2C0);
```

i2c_stop_on_bus

The description of i2c_stop_on_bus is shown as below:

Table 3-910. Function i2c_stop_on_bus

Function name	i2c_stop_on_bus
Function prototype	void i2c_stop_on_bus(uint32_t i2c_periph);
Function descriptions	generate a STOP condition on I2C bus
Precondition	-

The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 generate a STOP condition to I2C bus */
```

```
i2c_stop_on_bus(I2C0);
```

i2c_data_transmit

The description of i2c_data_transmit is shown as below:

Table 3-911. Function i2c_data_transmit

Function name	i2c_data_transmit
Function prototype	void i2c_data_transmit(uint32_t i2c_periph, uint32_t data);
Function descriptions	I2C transmit data
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Input parameter{in}	
data	transmit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 transmit data */
```

```
i2c_data_transmit(I2C0, 0x80);
```

i2c_data_receive

The description of i2c_data_receive is shown as below:

Table 3-912. Function i2c_data_receive

Function name	i2c_data_receive
---------------	------------------

Function prototype	uint32_t i2c_data_receive(uint32_t i2c_periph);
Function descriptions	I2C receive data
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
uint32_t	0x0000..0x00FF

Example:

```
/* I2C0 receive data */
```

```
uint32_t i2c_receiver;
```

```
i2c_receiver = i2c_data_receive(I2C0);
```

i2c_reload_enable

The description of i2c_reload_enable is shown as below:

Table 3-913. Function i2c_reload_enable

Function name	i2c_reload_enable
Function prototype	void i2c_reload_enable(uint32_t i2c_periph);
Function descriptions	enable I2C reload mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C reload mode */
```

```
i2c_reload_enable(I2C0);
```

i2c_reload_disable

The description of i2c_reload_disable is shown as below:

Table 3-914. Function i2c_reload_disable

Function name	i2c_reload_disable
Function prototype	void i2c_reload_disable(uint32_t i2c_periph);
Function descriptions	disable I2C reload mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C reload mode */
i2c_reload_disable(I2C0);
```

i2c_transfer_byte_number_config

The description of i2c_transfer_byte_number_config is shown as below:

Table 3-915. Function i2c_transfer_byte_number_config

Function name	i2c_transfer_byte_number_config
Function prototype	void i2c_transfer_byte_number_config(uint32_t i2c_periph, uint32_t byte_number);
Function descriptions	configure number of bytes to be transferred
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
byte_number	0x0-0xFF, number of bytes to be transferred
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure number of bytes to be transferred */
i2c_transfer_byte_number_config(I2C0, 0xFF);
```

i2c_dma_enable

The description of i2c_dma_enable is shown as below:

Table 3-916. Function i2c_dma_enable

Function name	i2c_dma_enable
Function prototype	void i2c_dma_enable(uint32_t i2c_periph, uint8_t dma);
Function descriptions	enable I2C DMA for transmission or reception
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
dma	I2C DMA
<i>I2C_DMA_TRANSMIT</i>	transmit data using DMA
<i>I2C_DMA_RECEIVE</i>	receive data using DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C DMA for transmission or reception */
```

```
i2c_dma_enable(I2C0, I2C_DMA_RECEIVE);
```

i2c_dma_disable

The description of i2c_dma_disable is shown as below:

Table 3-917. 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)
Input parameter{in}	
dma	I2C DMA
<i>I2C_DMA_TRANSMIT</i>	transmit data using DMA
<i>I2C_DMA_RECEIVE</i>	receive data using DMA
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable I2C DMA for transmission or reception */
```

```
i2c_dma_disable(I2C0, I2C_DMA_RECEIVE);
```

i2c_pec_transfer

The description of i2c_pec_transfer is shown as below:

Table 3-918. Function i2c_pec_transfer

Function name	i2c_pec_transfer
Function prototype	void i2c_pec_transfer(uint32_t i2c_periph);
Function descriptions	I2C transfers PEC value
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C transfers PEC value */
```

```
i2c_pec_transfer(I2C0);
```

i2c_pec_enable

The description of i2c_pec_enable is shown as below:

Table 3-919. Function i2c_pec_enable

Function name	i2c_pec_enable
Function prototype	void i2c_pec_enable(uint32_t i2c_periph);
Function descriptions	enable I2C PEC calculation
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C PEC calculation */
i2c_pec_enable(I2C0);
```

i2c_pec_disable

The description of i2c_pec_disable is shown as below:

Table 3-920. Function i2c_pec_disable

Function name	i2c_pec_disable
Function prototype	void i2c_pec_disable(uint32_t i2c_periph);
Function descriptions	disable I2C PEC calculation
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C PEC calculation */
i2c_pec_disable(I2C0);
```

i2c_pec_value_get

The description of i2c_pec_value_get is shown as below:

Table 3-921. Function i2c_pec_value_get

Function name	i2c_pec_value_get
Function prototype	uint32_t i2c_pec_value_get(uint32_t i2c_periph);
Function descriptions	get packet error checking value
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral

<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
uint32_t	PEC value

Example:

```
/* I2C0 get packet error checking value */
```

```
uint32_t pec_value;
```

```
pec_value = i2c_pec_value_get(I2C0);
```

i2c_smbus_alert_enable

The description of i2c_smbus_alert_enable is shown as below:

Table 3-922. Function i2c_smbus_alert_enable

Function name	i2c_smbus_alert_enable
Function prototype	void i2c_smbus_alert_enable(uint32_t i2c_periph);
Function descriptions	enable SMBus Alert
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SMBus Alert */
```

```
i2c_smbus_alert_enable(I2C0);
```

i2c_smbus_alert_disable

The description of i2c_smbus_alert_disable is shown as below:

Table 3-923. Function i2c_smbus_alert_disable

Function name	i2c_smbus_alert_disable
Function prototype	void i2c_smbus_alert_disable(uint32_t i2c_periph);
Function descriptions	disable SMBus Alert
Precondition	-
The called functions	-

Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SMBus Alert */
```

```
i2c_smbus_alert_disable(I2C0);
```

i2c_smbus_default_addr_enable

The description of i2c_smbus_default_addr_enable is shown as below:

Table 3-924. Function i2c_smbus_default_addr_enable

Function name	i2c_smbus_default_addr_enable
Function prototype	void i2c_smbus_default_addr_enable(uint32_t i2c_periph);
Function descriptions	enable SMBus device default address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SMBus device default address */
```

```
i2c_smbus_default_addr_enable(I2C0);
```

i2c_smbus_default_addr_disable

The description of i2c_smbus_default_addr_disable is shown as below:

Table 3-925. Function i2c_smbus_default_addr_disable

Function name	i2c_smbus_default_addr_disable
Function prototype	void i2c_smbus_default_addr_disable(uint32_t i2c_periph);
Function descriptions	disable SMBus device default address
Precondition	-

The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SMBus device default address */
```

```
i2c_smbus_default_addr_disable(I2C0);
```

i2c_smbus_host_addr_enable

The description of i2c_smbus_host_addr_enable is shown as below:

Table 3-926. Function i2c_smbus_host_addr_enable

Function name	i2c_smbus_host_addr_enable
Function prototype	void i2c_smbus_host_addr_enable(uint32_t i2c_periph);
Function descriptions	enable SMBus Host address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SMBus Host address */
```

```
i2c_smbus_host_addr_enable(I2C0);
```

i2c_smbus_host_addr_disable

The description of i2c_smbus_host_addr_disable is shown as below:

Table 3-927. Function i2c_smbus_host_addr_disable

Function name	i2c_smbus_host_addr_disable
Function prototype	void i2c_smbus_host_addr_disable(uint32_t i2c_periph);
Function descriptions	disable SMBus Host address

Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SMBus Host address */
```

```
i2c_smbus_host_addr_disable(I2C0);
```

i2c_extented_clock_timeout_enable

The description of i2c_extented_clock_timeout_enable is shown as below:

Table 3-928. Function i2c_extented_clock_timeout_enable

Function name	i2c_extented_clock_timeout_enable
Function prototype	void i2c_extented_clock_timeout_enable(uint32_t i2c_periph);
Function descriptions	enable extended clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable extended clock timeout detection */
```

```
i2c_extented_clock_timeout_enable(I2C0);
```

i2c_extented_clock_timeout_disable

The description of i2c_extented_clock_timeout_disable is shown as below:

Table 3-929. Function i2c_extented_clock_timeout_disable

Function name	i2c_extented_clock_timeout_disable
Function prototype	void i2c_extented_clock_timeout_disable(uint32_t i2c_periph);

Function descriptions	disable extended clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable extended clock timeout detection */
i2c_extented_clock_timeout_disable(I2C0);
```

i2c_clock_timeout_enable

The description of i2c_clock_timeout_enable is shown as below:

Table 3-930. Function i2c_clock_timeout_enable

Function name	i2c_clock_timeout_enable
Function prototype	void i2c_clock_timeout_enable(uint32_t i2c_periph);
Function descriptions	enable clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable clock timeout detection */
i2c_clock_timeout_enable(I2C0);
```

i2c_clock_timeout_disable

The description of i2c_clock_timeout_disable is shown as below:

Table 3-931. Function i2c_clock_timeout_disable

Function name	i2c_clock_timeout_disable
----------------------	---------------------------

Function prototype	void i2c_clock_timeout_disable(uint32_t i2c_periph);
Function descriptions	disable clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable clock timeout detection */
```

```
i2c_clock_timeout_disable(I2C0);
```

i2c_bus_timeout_b_config

The description of i2c_bus_timeout_b_config is shown as below:

Table 3-932. Function i2c_bus_timeout_b_config

Function name	i2c_bus_timeout_b_config
Function prototype	void i2c_bus_timeout_b_config(uint32_t i2c_periph, uint32_t timeout);
Function descriptions	configure bus timeout B
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
timeout	0-0xfff, bus timeout B
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure bus timeout B */
```

```
i2c_bus_timeout_b_config(I2C0, 0xff);
```

i2c_bus_timeout_a_config

The description of i2c_bus_timeout_a_config is shown as below:

Table 3-933. Function `i2c_bus_timeout_a_config`

Function name	<code>i2c_bus_timeout_a_config</code>
Function prototype	<code>void i2c_bus_timeout_a_config(uint32_t i2c_periph, uint32_t timeout);</code>
Function descriptions	configure bus timeout A
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(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 3-934. Function `i2c_idle_clock_timeout_config`

Function name	<code>i2c_idle_clock_timeout_config</code>
Function prototype	<code>void i2c_idle_clock_timeout_config(uint32_t i2c_periph, uint32_t timeout);</code>
Function descriptions	configure idle clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
timeout	bus timeout A
<i>BUSTOA_DETECT_SCL_LOW</i>	BUSTOA is used to detect SCL low timeout
<i>BUSTOA_DETECT_IDLE</i>	BUSTOA is used to detect both SCL and SDA high timeout when the bus is idle
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure idle clock timeout detection */

i2c_idle_clock_timeout_config(I2C0, BUSTOA_DETECT_SCL_LOW);
```

i2c_flag_get

The description of i2c_flag_get is shown as below:

Table 3-935. Function i2c_flag_get

Function name	i2c_flag_get
Function prototype	FlagStatus i2c_flag_get(uint32_t i2c_periph, uint32_t flag);
Function descriptions	get I2C flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
flag	I2C flags
<i>I2C_FLAG_TBE</i>	I2C_TDATA is empty during transmitting
<i>I2C_FLAG_TI</i>	transmit interrupt
<i>I2C_FLAG_RBNE</i>	I2C_RDATA is not empty during receiving
<i>I2C_FLAG_ADDSEND</i>	address received matches in slave mode
<i>I2C_FLAG_NACK</i>	not acknowledge flag
<i>I2C_FLAG_STPDET</i>	STOP condition detected in slave mode
<i>I2C_FLAG_TC</i>	transfer complete in master mode
<i>I2C_FLAG_TCR</i>	transfer complete reload
<i>I2C_FLAG_BERR</i>	bus error
<i>I2C_FLAG_LOSTARB</i>	arbitration Lost
<i>I2C_FLAG_OUERR</i>	overflow/underrun error in slave mode
<i>I2C_FLAG_PECERR</i>	PEC error
<i>I2C_FLAG_TIMEOUT</i>	timeout flag
<i>I2C_FLAG_SMBALT</i>	SMBus Alert
<i>I2C_FLAG_I2CBSY</i>	busy flag
<i>I2C_FLAG_TR</i>	whether the I2C is a transmitter or a receiver in slave mode
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get I2C flag status */
```



```
FlagStatus flag_state = RESET;
```

```
flag_state = i2c_flag_get(I2C0, I2C_FLAG_TBE);
```

i2c_flag_clear

The description of i2c_flag_clear is shown as below:

Table 3-936. Function i2c_flag_clear

Function name	i2c_flag_clear
Function prototype	void i2c_flag_clear(uint32_t i2c_periph, uint32_t flag);
Function descriptions	clear I2C flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
flag	I2C flags
<i>I2C_FLAG_ADDSEND</i>	address received matches in slave mode
<i>I2C_FLAG_NACK</i>	not acknowledge flag
<i>I2C_FLAG_STPDET</i>	STOP condition detected in slave mode
<i>I2C_FLAG_BERR</i>	bus error
<i>I2C_FLAG_LOSTARB</i>	arbitration Lost
<i>I2C_FLAG_OUERR</i>	overrun/underrun error in slave mode
<i>I2C_FLAG_PECERR</i>	PEC error
<i>I2C_FLAG_TIMEOUT</i>	timeout flag
<i>I2C_FLAG_SMBALT</i>	SMBus Alert
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear a bus error flag*/
```

```
i2c_flag_clear(I2C0, I2C_FLAG_BERR);
```

i2c_interrupt_enable

The description of i2c_interrupt_enable is shown as below:

Table 3-937. Function i2c_interrupt_enable

Function name	i2c_interrupt_enable
Function prototype	void i2c_interrupt_enable(uint32_t i2c_periph, uint32_t interrupt);

Function descriptions	enable I2C interrupt
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
interrupt	I2C interrupts
<i>I2C_INT_ERR</i>	error interrupt
<i>I2C_INT_TC</i>	transfer complete interrupt
<i>I2C_INT_STPDET</i>	stop detection interrupt
<i>I2C_INT_NACK</i>	not acknowledge received interrupt
<i>I2C_INT_ADDM</i>	address match interrupt
<i>I2C_INT_RBNE</i>	receive interrupt
<i>I2C_INT_TI</i>	transmit interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C0 transmit interrupt */
```

```
i2c_interrupt_enable(I2C0, I2C_INT_TI);
```

i2c_interrupt_disable

The description of i2c_interrupt_disable is shown as below:

Table 3-938. Function i2c_interrupt_disable

Function name	i2c_interrupt_disable
Function prototype	void i2c_interrupt_disable(uint32_t i2c_periph, uint32_t interrupt);
Function descriptions	disable I2C interrupt
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
interrupt	I2C interrupts
<i>I2C_INT_ERR</i>	error interrupt
<i>I2C_INT_TC</i>	transfer complete interrupt
<i>I2C_INT_STPDET</i>	stop detection interrupt
<i>I2C_INT_NACK</i>	not acknowledge received interrupt

<i>I2C_INT_ADDM</i>	address match interrupt
<i>I2C_INT_RBNE</i>	receive interrupt
<i>I2C_INT_TI</i>	transmit interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C0 transmit interrupt */
```

```
i2c_interrupt_disable(I2C0, I2C_INT_TI);
```

i2c_interrupt_flag_get

The description of i2c_interrupt_flag_get is shown as below:

Table 3-939. Function i2c_interrupt_flag_get

Function name	i2c_interrupt_flag_get
Function prototype	FlagStatus i2c_interrupt_flag_get(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag);
Function descriptions	get I2C interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
int_flag	I2C interrupt flags, refer to Table 3-877. Enum i2c_interrupt_flag_enum.
<i>I2C_INT_FLAG_TI</i>	transmit interrupt flag
<i>I2C_INT_FLAG_RBNE</i>	I2C_RDATA is not empty during receiving interrupt flag
<i>I2C_INT_FLAG_ADDS END</i>	address received matches in slave mode interrupt flag
<i>I2C_INT_FLAG_NACK</i>	not acknowledge interrupt flag
<i>I2C_INT_FLAG_STPD ET</i>	stop condition detected in slave mode interrupt flag
<i>I2C_INT_FLAG_TC</i>	transfer complete in master mode interrupt flag
<i>I2C_INT_FLAG_TCR</i>	transfer complete reload interrupt flag
<i>I2C_INT_FLAG_BERR</i>	bus error interrupt flag
<i>I2C_INT_FLAG_LOSTA RB</i>	arbitration lost interrupt flag
<i>I2C_INT_FLAG_OUER R</i>	overflow/underrun error in slave mode interrupt flag
<i>I2C_INT_FLAG_PEC</i>	PEC error interrupt flag

<i>RR</i>	
<i>I2C_INT_FLAG_TIMEO</i> <i>UT</i>	timeout interrupt flag
<i>I2C_INT_FLAG_SMBA</i> <i>LT</i>	SMBus Alert interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get I2C interrupt flag status */
```

```
FlagStatus flag_state = RESET;
```

```
flag_state = i2c_interrupt_flag_get(I2C0, I2C_INT_FLAG_TI);
```

i2c_interrupt_flag_clear

The description of i2c_interrupt_flag_clear is shown as below:

Table 3-940. Function i2c_interrupt_flag_clear

Function name	i2c_interrupt_flag_clear
Function prototype	void i2c_interrupt_flag_clear(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag);
Function descriptions	clear I2C interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
int_flag	I2C interrupt flags, refer to Table 3-877. Enum i2c_interrupt_flag_enum.
<i>I2C_INT_FLAG_ADDS</i> <i>END</i>	address received matches in slave mode interrupt flag
<i>I2C_INT_FLAG_NACK</i>	not acknowledge interrupt flag
<i>I2C_INT_FLAG_STPD</i> <i>ET</i>	stop condition detected in slave mode interrupt flag
<i>I2C_INT_FLAG_BERR</i>	bus error interrupt flag
<i>I2C_INT_FLAG_LOSTA</i> <i>RB</i>	arbitration lost interrupt flag
<i>I2C_INT_FLAG_OUER</i> <i>R</i>	overflow/underrun error in slave mode interrupt flag
<i>I2C_INT_FLAG_PECE</i> <i>RR</i>	PEC error interrupt flag

<i>I2C_INT_FLAG_TIMEO</i> <i>UT</i>	timeout interrupt flag
<i>I2C_INT_FLAG_SMBA</i> <i>LT</i>	SMBus Alert interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear a bus error flag */
```

```
i2c_interrupt_flag_clear(I2C0, I2C_INT_FLAG_BERR);
```

3.26. IPA

The IPA provides a configurable and flexible image format conversion from one or two source image to the destination image. The IPA registers are listed in chapter [3.26.1](#), the IPA firmware functions are introduced in chapter [3.26.2](#).

3.26.1. Descriptions of Peripheral registers

Table 3-941. IPA Registers

Registers	Descriptions
IPA_CTL	IPA control register
IPA_INTF	IPA interrupt flag register
IPA_INTC	IPA interrupt flag clear register
IPA_FMADDR	IPA foreground memory base address register
IPA_FLOFF	IPA foreground line offset register
IPA_BMADDR	IPA background memory base address register
IPA_BLOFF	IPA background line offset register
IPA_FPCTL	IPA foreground pixel control register
IPA_FPV	IPA foreground pixel value register
IPA_BPCTL	IPA background pixel control register
IPA_BPV	IPA background pixel value register
IPA_FLMADDR	IPA foreground LUT memory base address register
IPA_BLMADDR	IPA background LUT memory base address register
IPA_DPCTL	IPA destination pixel control register
IPA_DPV	IPA destination pixel value register
IPA_DMADDR	IPA destination memory base address register
IPA_DLOFF	IPA destination line offset register
IPA_IMS	IPA image size register

Registers	Descriptions
IPA_LM	IPA line mark register
IPA_ITCTL	IPA inter-timer control register
IPA_BSCTL	IPA bilinear scaling control register
IPA_DIMS	IPA scaling image size register
IPA_EF_UV_MADDR	IPA foreground even frame/UV memory base address register
IPA_CSCC_CFG0	IPA color space conversion coefficient 0
IPA_CSCC_CFG1	IPA color space conversion coefficient 1
IPA_CSCC_CFG2	IPA color space conversion coefficient 2

3.26.2. Descriptions of Peripheral functions

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

Table 3-942. IPA firmware function

Function name	Function description
ipa_deinit	deinitialize IPA
ipa_transfer_enable	enable IPA transfer
ipa_transfer_hangup_enable	enable IPA transfer hang up
ipa_transfer_hangup_disable	disable IPA transfer hang up
ipa_transfer_stop_enable	enable IPA transfer stop
ipa_transfer_stop_disable	disable IPA transfer stop
ipa_foreground_lut_loading_enable	enable IPA foreground LUT loading
ipa_background_lut_loading_enable	enable IPA background LUT loading
ipa_pixel_format_convert_mode_set	set pixel format convert mode, the function is invalid when the IPA transfer is enabled
ipa_foreground_interlace_mode_enable	enable foreground interlace mode
ipa_foreground_interlace_mode_disable	disable foreground interlace mode
ipa_foreground_struct_para_init	initialize the structure of IPA foreground parameter struct with the default values, it is suggested that call this function after an ipa_foreground_parameter_struct structure is defined
ipa_foreground_init	initialize foreground parameters
ipa_background_struct_para_init	initialize the structure of IPA background parameter struct with the default values, it is suggested that call this function after an ipa_background_parameter_struct structure is defined
ipa_background_init	initialize background parameters
ipa_destination_struct_para_init	initialize the structure of IPA destination parameter struct with the default values, it is suggested that call this function after an ipa_destination_parameter_struct structure is defined

Function name	Function description
ipa_destination_init	initialize destination parameters
ipa_foreground_lut_init	initialize IPA foreground LUT parameters
ipa_background_lut_init	initialize IPA background LUT parameters
ipa_line_mark_config	configure IPA line mark
ipa_inter_timer_config	IPA inter-timer enable or disable
ipa_interval_clock_num_config	configure the number of clock cycles interval
ipa_color_conversion_struct_para_init	initialize the structure of IPA color conversion parameter struct with the YUV or YCbCr conversion parameter, it is suggested that call this function after an ipa_conversion_parameter_struct structure is defined
ipa_color_conversion_config	configure the color space conversion parameter
ipa_foreground_scaling_config	configure IPA foreground scaling, including horizontal/vertical pre-decimation factors and X/Y scaling factors
ipa_destination_scaling_config	configure IPA destination scaling, including width/height of image to be processed
ipa_flag_get	get IPA flag status in IPA_INTF register
ipa_flag_clear	clear IPA flag in IPA_INTF register
ipa_interrupt_enable	enable IPA interrupt
ipa_interrupt_disable	disable IPA interrupt
ipa_interrupt_flag_get	get IPA interrupt flag
ipa_interrupt_flag_clear	clear IPA interrupt flag

Structure ipa_foreground_parameter_struct

Table3-943. Structure ipa_foreground_parameter_struct

Member name	Function description
foreground_memaddr	foreground memory base address
foreground_lineoff	foreground line offset
foreground_prealpha	foreground pre-defined alpha value
foreground_alpha_algorithm	foreground alpha value calculation algorithm
foreground_pf	foreground pixel format
foreground_prered	foreground pre-defined red value
foreground_pregreen	foreground pre-defined green value
foreground_preblue	foreground pre-defined blue value
foreground_interlace_mode	foreground input interlace mode enable
foreground_efuv_memaddr	foreground even frame / UV memory base address

Structure ipa_background_parameter_struct

Table 3-944. Structure ipa_background_parameter_struct

Member name	Function description
background_memaddr	background memory base address

Member name	Function description
background_lineoff	background line offset
background_prealpha	background pre-defined alpha value
background_alpha_algorithm	background alpha value calculation algorithm
background_pf	background pixel format
background_prered	background pre-defined red value
background_pregreen	background pre-defined green value
background_preblue	background pre-defined blue value

Structure ipa_destination_parameter_struct

Table 3-945. Structure ipa_destination_parameter_struct

Member name	Function description
destination_memaddr	destination memory base address
destination_lineoff	destination line offset
destination_prealpha	destination pre-defined alpha value
destination_pf	destination pixel format
destination_prered	destination pre-defined red value
destination_pregreen	destination pre-defined green value
destination_preblue	destination pre-defined blue value
image_width	width of the image to be processed
image_height	height of the image to be processed
image_rotate	angle of image rotation
image_hor_decimation	image horizontal pre-decimation in width
image_ver_decimation	image vertical pre-decimation in height
image_bilinear_xscale	bilinear scaling x factor
image_bilinear_yscale	bilinear scaling y factor
image_scaling_width	width of the image after scaling
image_scaling_height	height of the image after scaling

Structure ipa_conversion_parameter_struct

Table 3-946. Structure ipa_conversion_parameter_struct

Member name	Function description
color_space	color space convert mode
y_offset	offset implicit in the Y data
uv_offset	offset implicit in the UV data
coef_c0	Y multiplier coefficient
coef_c1	V/Cr red multiplier coefficient
coef_c2	V/Cr green multiplier coefficient
coef_c3	U/Cb green multiplier coefficient
coef_c4	U/Cb blue multiplier coefficient

Enum ipa_dpf_enum

Table 3-947. Enum ipa_dpf_enum

enum name	Function description
IPA_DPF_ARGB8888	destination pixel format ARGB8888
IPA_DPF_RGB888	destination pixel format RGB888
IPA_DPF_RGB565	destination pixel format RGB565
IPA_DPF_ARGB1555	destination pixel format ARGB1555
IPA_DPF_ARGB4444	destination pixel format ARGB4444

Enum ipa_colorspace_enum

Table 3-948. Enum ipa_colorspace_enum

enum name	Function description
IPA_COLORSPACE_Y UV	IPA color conversion using YUV parameter
IPA_COLORSPACE_Y CBCR	IPA color conversion using YCbCr parameter

ipa_deinit

The description of ipa_deinit is shown as below:

Table 3-949. Function ipa_deinit

Function name	ipa_deinit
Function prototype	void ipa_deinit(void);
Function descriptions	deinitialize IPA registers
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize IPA */
ipa_deinit();
```

ipa_transfer_enable

The description of ipa_transfer_enable is shown as below:

Table 3-950. Function ipa_transfer_enable

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

Example:

```
/* enable IPA transfer */
ipa_transfer_enable();
```

ipa_transfer_hangup_enable

The description of ipa_transfer_hangup_enable is shown as below:

Table 3-951. Function ipa_transfer_hangup_enable

Function name	ipa_transfer_hangup_enable
Function prototype	void ipa_transfer_hangup_enable(void);
Function descriptions	enable IPA transfer hang up
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable IPA transfer hang up */
ipa_transfer_hangup_enable();
```

ipa_transfer_hangup_disable

The description of ipa_transfer_hangup_disable is shown as below:

Table 3-952. Function ipa_transfer_hangup_disable

Function name	ipa_transfer_hangup_disable
Function prototype	void ipa_transfer_hangup_disable(void);
Function descriptions	disable IPA transfer hang up
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable IPA transfer hang up */
ipa_transfer_hangup_disable();
```

ipa_transfer_stop_enable

The description of ipa_transfer_stop_enable is shown as below:

Table 3-953. Function ipa_transfer_stop_enable

Function name	ipa_transfer_stop_enable
Function prototype	void ipa_transfer_stop_enable(void);
Function descriptions	enable IPA transfer stop
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable IPA transfer stop */
ipa_transfer_stop_enable();
```

ipa_transfer_stop_disable

The description of ipa_transfer_stop_disable is shown as below:

Table 3-954. Function ipa_transfer_stop_disable

Function name	ipa_transfer_stop_disable
Function prototype	void ipa_transfer_stop_disable(void);
Function descriptions	disable IPA transfer stop
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable IPA transfer stop */
ipa_transfer_stop_disable();
```

ipa_foreground_lut_loading_enable

The description of ipa_foreground_lut_loading_enable is shown as below:

Table 3-955. Function ipa_foreground_lut_loading_enable

Function name	ipa_foreground_lut_loading_enable
Function prototype	void ipa_foreground_lut_loading_enable(void);
Function descriptions	enable IPA foreground LUT loading
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable IPA foreground LUT loading */
ipa_foreground_lut_loading_enable();
```

ipa_background_lut_loading_enable

The description of ipa_background_lut_loading_enable is shown as below:

Table 3-956. Function `ipa_background_lut_loading_enable`

Function name	<code>ipa_background_lut_loading_enable</code>
Function prototype	<code>void ipa_background_lut_loading_enable(void);</code>
Function descriptions	enable IPA background LUT loading
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable IPA background LUT loading */
ipa_background_lut_loading_enable();
```

`ipa_pixel_format_convert_mode_set`

The description of `ipa_pixel_format_convert_mode_set` is shown as below:

Table 3-957. Function `ipa_pixel_format_convert_mode_set`

Function name	<code>ipa_pixel_format_convert_mode_set</code>
Function prototype	<code>void ipa_pixel_format_convert_mode_set(uint32_t pfcmm);</code>
Function descriptions	set pixel format convert mode, the function is invalid when the IPA transfer is enabled
Precondition	-
The called functions	-
Input parameter{in}	
pfcmm	pixel format convert mode
<code>IPA_FGTODE</code>	foreground memory to destination memory without pixel format convert
<code>IPA_FGTODE_PF_CONVERT</code>	foreground memory to destination memory with pixel format convert
<code>IPA_FGBGTODE</code>	foreground and background memory to destination memory
<code>IPA_FILL_UP_DE</code>	fill up destination memory with specific color
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure foreground memory to destination memory with pixel format convert */
ipa_pixel_format_convert_mode_set(IPA_FGTODE_PF_CONVERT);
```

ipa_foreground_interlace_mode_enable

The description of ipa_foreground_interlace_mode_enable is shown as below:

Table 3-958. Function ipa_foreground_interlace_mode_enable

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

Example:

```
/* enable foreground interlace mode */
ipa_foreground_interlace_mode_enable();
```

ipa_foreground_interlace_mode_disable

The description of ipa_foreground_interlace_mode_disable is shown as below:

Table 3-959. Function ipa_foreground_interlace_mode_disable

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

Example:

```
/* disable foreground interlace mode */
ipa_foreground_interlace_mode_disable();
```

ipa_foreground_struct_para_init

The description of ipa_foreground_struct_para_init is shown as below:

Table 3-960. Function ipa_foreground_struct_para_init

Function name	ipa_foreground_struct_para_init
Function prototype	void ipa_foreground_struct_para_init(ipa_foreground_parameter_struct* foreground_struct);
Function descriptions	initialize the structure of IPA foreground parameter struct with the default values, it is suggested that call this function after an ipa_foreground_parameter_struct structure is defined
Precondition	-
The called functions	-
Input parameter{in}	
*foreground_struct	a pointer to ipa_foreground_parameter_struct, refer to Structure ipa_foreground_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
ipa_foreground_parameter_struct fg_struct;
```

```
/* initialize the structure of IPA foreground parameter struct with the default values */
```

```
ipa_foreground_struct_para_init(&fg_struct);
```

ipa_foreground_init

The description of ipa_foreground_init is shown as below:

Table 3-961. Function ipa_foreground_init

Function name	ipa_foreground_init
Function prototype	void ipa_foreground_init(ipa_foreground_parameter_struct* foreground_struct);
Function descriptions	initialize foreground parameters
Precondition	-
The called functions	-
Input parameter{in}	
*foreground_struct	a pointer to ipa_foreground_parameter_struct, refer to Structure ipa_foreground_parameter_struct
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
ipa_foreground_parameter_struct ipa_fg_init_struct;

ipa_foreground_struct_para_init(&ipa_fg_init_struct);

/* configure IPA foreground */

ipa_fg_init_struct.foreground_memaddr = (uint32_t)&gBuffer;

ipa_fg_init_struct.foreground_pf = FOREGROUND_PPF_RGB565;

ipa_fg_init_struct.foreground_alpha_algorithm = IPA_FG_ALPHA_MODE_1;

ipa_fg_init_struct.foreground_prealpha = 0x75;

ipa_fg_init_struct.foreground_lineoff = 0x00;

ipa_fg_init_struct.foreground_preblue = 0x00;

ipa_fg_init_struct.foreground_pregreen = 0x00;

ipa_fg_init_struct.foreground_prered = 0x00;

ipa_fg_init_struct.foreground_efuv_memaddr = 0x00;

ipa_fg_init_struct.foreground_interlace_mode = DISABLE;

/* foreground initialization */

ipa_foreground_init(&ipa_fg_init_struct);
```

ipa_background_struct_para_init

The description of ipa_background_struct_para_init is shown as below:

Table 3-962. Function ipa_background_struct_para_init

Function name	ipa_background_struct_para_init
Function prototype	void ipa_background_struct_para_init(ipa_background_parameter_struct* background_struct);
Function descriptions	initialize the structure of IPA background parameter struct with the default values , it is suggested that call this function after an ipa_background_parameter_struct structure is defined
Precondition	-
The called functions	-
Input parameter{in}	
*background_struct	a pointer to ipa_background_parameter_struct, refer to Structure ipa_background_parameter_struct
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
ipa_background_parameter_struct bg_struct;
```

```
/* initialize the structure of IPA background parameter struct with the default values */
```

```
ipa_background_struct_para_init(&bg_struct);
```

ipa_background_init

The description of ipa_background_init is shown as below:

Table 3-963. Function ipa_background_init

Function name	ipa_background_init
Function prototype	void ipa_background_init(ipa_background_parameter_struct* background_struct);
Function descriptions	initialize background parameters
Precondition	-
The called functions	-
Input parameter{in}	
*background_struct	a pointer to ipa_background_parameter_struct, refer to Structure ipa_background_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
ipa_background_parameter_struct ipa_bg_init_struct;
```

```
ipa_background_struct_para_init(&ipa_bg_init_struct);
```

```
/* configure IPA background */
```

```
ipa_bg_init_struct.background_memaddr = (uint32_t)&gBuffer;
```

```
ipa_bg_init_struct.background_pf = BACKGROUND_PPF_RGB565;
```

```
ipa_bg_init_struct.background_alpha_algorithm = IPA_BG_ALPHA_MODE_0;
```

```
ipa_bg_init_struct.background_prealpha = 255;
```

```
ipa_bg_init_struct.background_lineoff = 0x00;
```

```
ipa_bg_init_struct.background_preblue = 0x00;
```

```
ipa_bg_init_struct.background_pregreen = 0x00;
```

```
ipa_bg_init_struct.background_prered = 0x00;
```

```
/* background initialization */
```

```
ipa_background_init(&ipa_bg_init_struct);
```

ipa_destination_struct_para_init

The description of ipa_destination_struct_para_init is shown as below:

Table 3-964. Function ipa_destination_struct_para_init

Function name	ipa_destination_struct_para_init
Function prototype	void ipa_destination_struct_para_init(ipa_destination_parameter_struct* destination_struct);
Function descriptions	initialize the structure of IPA destination parameter struct with the default values, it is suggested that call this function after an ipa_destination_parameter_struct structure is defined
Precondition	-
The called functions	-
Input parameter{in}	
*destination_struct	a pointer to ipa_destination_parameter_struct, refer to Structure ipa_destination_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
ipa_destination_parameter_struct destination_struct;
```

```
/* initialize the structure of IPA destination parameter struct with the default values */
```

```
ipa_destination_struct_para_init(&destination_struct);
```

ipa_destination_init

The description of ipa_destination_init is shown as below:

Table 3-965. Function ipa_destination_init

Function name	ipa_destination_init
Function prototype	void ipa_destination_init(ipa_destination_parameter_struct* destination_struct);
Function descriptions	initialize destination parameters
Precondition	-
The called functions	-
Input parameter{in}	

*destination_struct	a pointer to ipa_destination_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

ipa_destination_parameter_struct ipa_destination_init_struct;

ipa_destination_struct_para_init(&ipa_destination_init_struct);

/* configure destination pixel format */

ipa_destination_init_struct.destination_pf = IPA_DPF_RGB565;

/* configure destination memory base address */

ipa_destination_init_struct.destination_memaddr = (uint32_t)&gBuffer;

/* configure destination pre-defined alpha value RGB */

ipa_destination_init_struct.destination_pregreen = 0;

ipa_destination_init_struct.destination_preblue = 0;

ipa_destination_init_struct.destination_prered = 0;

ipa_destination_init_struct.destination_prealpha = 0;

/* configure destination line offset */

ipa_destination_init_struct.destination_lineoff = 0;

/* configure height of the image to be processed */

ipa_destination_init_struct.image_height = 160;

/* configure width of the image to be processed */

ipa_destination_init_struct.image_width = 229;

ipa_destination_init_struct.image_rotate = DESTINATION_ROTATE_0;

ipa_destination_init_struct.image_hor_decimation =
DESTINATION_HORDECIMATE_DISABLE;

ipa_destination_init_struct.image_ver_decimation =
DESTINATION_VERDECIMATE_DISABLE;

ipa_destination_init_struct.image_scaling_height = 160;

ipa_destination_init_struct.image_scaling_width = 229;

ipa_destination_init_struct.image_bilinear_xscale = 0x1000;

```

```
ipa_destination_init_struct.image_bilinear_yscale = 0x1000;
```

```
/* ipa destination initialization */
```

```
ipa_destination_init(&ipa_destination_init_struct);
```

ipa_foreground_lut_init

The description of ipa_foreground_lut_init is shown as below:

Table 3-966. Function ipa_foreground_lut_init

Function name	ipa_foreground_lut_init
Function prototype	void ipa_foreground_lut_init(uint8_t fg_lut_num, uint8_t fg_lut_pf, uint32_t fg_lut_addr);
Function descriptions	initialize IPA foreground LUT parameters
Precondition	-
The called functions	-
Input parameter{in}	
fg_lut_num	foreground LUT number of pixel
Input parameter{in}	
fg_lut_pf	foreground LUT pixel format
IPA_LUT_PF_ARGB8888	LUT pixel format is ARGB8888
IPA_LUT_PF_RGB888	LUT pixel format is RGB888
Input parameter{in}	
fg_lut_addr	foreground LUT memory base address. The address must be aligned to 8-bit, 16-bit or 32-bit corresponding with the foreground LUT pixel format.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the LUT foreground LUT parameters */
```

```
ipa_foreground_lut_init(1, IPA_LUT_PF_ARGB8888, 0x20002000);
```

ipa_background_lut_init

The description of ipa_background_lut_init is shown as below:

Table 3-967. Function ipa_background_lut_init

Function name	ipa_background_lut_init
Function prototype	void ipa_background_lut_init(uint8_t bg_lut_num, uint8_t bg_lut_pf, uint32_t bg_lut_addr);
Function descriptions	initialize IPA background LUT parameters

Precondition	-
The called functions	-
Input parameter{in}	
bg_lut_num	background LUT number of pixel
Input parameter{in}	
bg_lut_pf	background LUT pixel format
<i>IPA_LUT_PF_ARGB888</i> 88	LUT pixel format is ARGB8888
<i>IPA_LUT_PF_RGB888</i>	LUT pixel format is RGB888
Input parameter{in}	
bg_lut_addr	background LUT memory base address. The address must be aligned to 8-bit, 16-bit or 32-bit corresponding with the foreground LUT pixel format.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the LUT background LUT parameters */
ipa_background_lut_init(2, IPA_LUT_PF_RGB888, 0x20001000);
```

ipa_line_mark_config

The description of ipa_line_mark_config is shown as below:

Table 3-968. Function ipa_line_mark_config

Function name	ipa_line_mark_config
Function prototype	void ipa_line_mark_config(uint16_t line_num);
Function descriptions	configure IPA line mark
Precondition	-
The called functions	-
Input parameter{in}	
line_num	line number
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure line mark */
ipa_line_mark_config(10);
```

ipa_inter_timer_config

The description of ipa_inter_timer_config is shown as below:

Table 3-969. Function ipa_inter_timer_config

Function name	ipa_inter_timer_config
Function prototype	void ipa_inter_timer_config(uint8_t timer_cfg);
Function descriptions	inter-timer enable or disable
Precondition	-
The called functions	-
Input parameter{in}	
timer_cfg	inter-timer configuration
<i>IPA_INTER_TIMER_ENABLE</i>	enable inter-timer
<i>IPA_INTER_TIMER_DISABLE</i>	disable inter-timer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable inter-timer */
```

```
ipa_inter_timer_config(IPA_INTER_TIMER_ENABLE);
```

ipa_interval_clock_num_config

The description of ipa_interval_clock_num_config is shown as below:

Table 3-970. Function ipa_interval_clock_num_config

Function name	ipa_interval_clock_num_config
Function prototype	void ipa_interval_clock_num_config(uint8_t clk_num);
Function descriptions	configure the number of clock cycles interval
Precondition	-
The called functions	-
Input parameter{in}	
clk_num	the number of clock cycles
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the number of clock cycles interval and it has no meaning if ITEN is '0' */
```

```
ipa_interval_clock_num_config(1);
```

ipa_color_conversion_struct_para_init

The description of ipa_color_conversion_struct_para_init is shown as below:

Table 3-971. Function ipa_color_conversion_struct_para_init

Function name	ipa_color_conversion_struct_para_init
Function prototype	void ipa_color_conversion_struct_para_init(ipa_conversion_parameter_struct* conversion_struct, ipa_colorspace_enum colorspace);
Function descriptions	initialize the structure of IPA foreground parameter struct with the default values, it is suggested that call this function after an ipa_foreground_parameter_struct structure is defined
Precondition	-
The called functions	-
Input parameter{in}	
colorspace	the color space, refer to Enum ipa_colorspace_enum
IPA_COLORSPACE_Y UV	using default YUV parameter to initialization struct
IPA_COLORSPACE_Y CBCR	using default YCbCr parameter to initialization struct
Output parameter{out}	
conversion_struct	the data needed to config color conversion parameters, refer to Structure ipa_conversion_parameter_struct
Return value	
-	-

Example:

```
/* initialize the structure of IPA ipa_conversion_parameter_struct with the default values */
```

```
ipa_conversion_parameter_struct conversion_struct;
```

```
ipa_color_conversion_struct_para_init(&conversion_struct, IPA_COLORSPACE_YUV);
```

ipa_color_conversion_config

The description of ipa_color_conversion_config is shown as below:

Table 3-972. Function ipa_color_conversion_config

Function name	ipa_color_conversion_config
Function prototype	void ipa_color_conversion_config(ipa_conversion_parameter_struct* conversion_struct);
Function descriptions	configure the color space conversion parameter
Precondition	-
The called functions	-

Input parameter{in}	
conversion_struct	the data needed to config color conversion parameters, refer to Structure ipa_conversion_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure the color space conversion parameters */

ipa_conversion_parameter_struct conversion_struct;

conversion_struct.color_space = IPA_COLORSPACE_YUV;

conversion_struct.y_offset = 0x0;

conversion_struct.uv_offset = 0x0;

conversion_struct.coef_c0 = 0x100;

conversion_struct.coef_c1 = 0x123;

conversion_struct.coef_c2 = 0x76B;

conversion_struct.coef_c3 = 0x79C;

conversion_struct.coef_c4 = 0x208;

ipa_color_conversion_config(&conversion_struct);

```

ipa_foreground_scaling_config

The description of ipa_foreground_scaling_config is shown as below:

Table 3-973. Function ipa_foreground_scaling_config

Function name	ipa_foreground_scaling_config
Function prototype	void ipa_foreground_scaling_config(uint32_t horizontal_decimation, uint32_t vertical_decimation, uint32_t image_scaling_width, uint32_t image_scaling_height);
Function descriptions	configure IPA foreground scaling, including horizontal/vertical pre-decimation factors and X/Y scaling factors
Precondition	-
The called functions	-
Input parameter{in}	
horizontal_decimation	horizontal scaling value
<i>DESTINATION_HORD ECIMATE_DISABLE</i>	disable horizontal decimate
<i>DESTINATION_HORD</i>	horizontal decimated by 2

<i>ECIMATE_2</i>	
<i>DESTINATION_HORD</i> <i>ECIMATE_4</i>	horizontal decimated by 4
<i>DESTINATION_HORD</i> <i>ECIMATE_8</i>	horizontal decimated by 8
Input parameter{in}	
vertical_decimation	vertical scaling value
<i>DESTINATION_VERD</i> <i>ECIMATE_DISABLE</i>	disable vertical decimate
<i>DESTINATION_VERD</i> <i>ECIMATE_2</i>	vertical decimated by 2
<i>DESTINATION_VERD</i> <i>ECIMATE_4</i>	vertical decimated by 4
<i>DESTINATION_VERD</i> <i>ECIMATE_8</i>	vertical decimated by 8
Input parameter{in}	
image_scaling_width	image scaling factor of width
Input parameter{in}	
image_scaling_height	image scaling factor of height
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure IPA foreground scaling */
```

```
ipa_foreground_scaling_config(DESTINATION_HORDECIMATE_2,  
DESTINATION_VERDECIMATE_2, 320, 240);
```

ipa_destination_scaling_config

The description of ipa_destination_scaling_config is shown as below:

Table 3-974. Function ipa_destination_scaling_config

Function name	ipa_destination_scaling_config
Function prototype	void ipa_destination_scaling_config(uint32_t dest_scaling_width, uint32_t dest_scaling_height);
Function descriptions	configure IPA destination scaling, including width/height of image to be processed
Precondition	-
The called functions	-
Input parameter{in}	
dest_scaling_width	width of destination image after scaling

Input parameter{in}	
dest_scaling_height	height of destination image after scaling
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure IPA destination scaling */
ipa_destination_scaling_config(320, 240);
```

ipa_flag_get

The description of ipa_flag_get is shown as below:

Table 3-975. Function ipa_flag_get

Function name	ipa_flag_get
Function prototype	FlagStatus ipa_flag_get(uint32_t flag);
Function descriptions	get IPA flag status in IPA_INTF register
Precondition	-
The called functions	-
Input parameter{in}	
flag	IPA flags
<i>IPA_FLAG_TAE</i>	transfer access error interrupt flag
<i>IPA_FLAG_FTF</i>	full transfer finish interrupt flag
<i>IPA_FLAG_TLM</i>	transfer line mark interrupt flag
<i>IPA_FLAG_LAC</i>	LUT access conflict interrupt flag
<i>IPA_FLAG_LLF</i>	LUT loading finish interrupt flag
<i>IPA_FLAG_WCF</i>	wrong configuration interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* wait for full transfer finish flag set */
while(RESET == ipa_flag_get(IPA_FLAG_FTF));
```

ipa_flag_clear

The description of ipa_flag_clear is shown as below:

Table 3-976. Function ipa_flag_clear

Function name	ipa_flag_clear
Function prototype	void ipa_flag_clear(uint32_t flag);
Function descriptions	clear IPA flag in IPA_INTF register
Precondition	-
The called functions	-
Input parameter{in}	
flag	IPA flags
<i>IPA_FLAG_TAE</i>	transfer access error interrupt flag
<i>IPA_FLAG_FTF</i>	full transfer finish interrupt flag
<i>IPA_FLAG_TLM</i>	transfer line mark interrupt flag
<i>IPA_FLAG_LAC</i>	LUT access conflict interrupt flag
<i>IPA_FLAG_LLF</i>	LUT loading finish interrupt flag
<i>IPA_FLAG_WCF</i>	wrong configuration interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait for full transfer finish flag set */
while(RESET ==ipa_flag_get(IPA_FLAG_FTF));
/* clear full transfer finish flag */
ipa_flag_clear(IPA_FLAG_FTF);
```

ipa_interrupt_enable

The description of ipa_interrupt_enable is shown as below:

Table 3-977. Function ipa_interrupt_enable

Function name	ipa_interrupt_enable
Function prototype	void ipa_interrupt_enable(uint32_t int_flag);
Function descriptions	enable IPA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	IPA interrupt flags
<i>IPA_INT_TAE</i>	transfer access error interrupt
<i>IPA_INT_FTF</i>	full transfer finish interrupt
<i>IPA_INT_TLM</i>	transfer line mark interrupt
<i>IPA_INT_LAC</i>	LUT access conflict interrupt
<i>IPA_INT_LLF</i>	LUT loading finish interrupt

<i>IPA_INT_WCF</i>	wrong configuration interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable full transfer finish interrupt */
ipa_interrupt_enable(IPA_INT_FTF);
```

ipa_interrupt_disable

The description of ipa_interrupt_disable is shown as below:

Table 3-978. Function ipa_interrupt_disable

Function name	ipa_interrupt_disable
Function prototype	void ipa_interrupt_disable(uint32_t int_flag);
Function descriptions	disable IPA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	IPA interrupt flags
<i>IPA_INT_TAE</i>	transfer access error interrupt
<i>IPA_INT_FTF</i>	full transfer finish interrupt
<i>IPA_INT_TLM</i>	transfer line mark interrupt
<i>IPA_INT_LAC</i>	LUT access conflict interrupt
<i>IPA_INT_LLF</i>	LUT loading finish interrupt
<i>IPA_INT_WCF</i>	wrong configuration interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable full transfer finish interrupt */
ipa_interrupt_disable(IPA_INT_FTF);
```

ipa_interrupt_flag_get

The description of ipa_interrupt_flag_get is shown as below:

Table 3-979. Function ipa_interrupt_flag_get

Function name	ipa_interrupt_flag_get
----------------------	------------------------

Function prototype	FlagStatus ipa_interrupt_flag_get(uint32_t int_flag);
Function descriptions	get IPA interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	IPA interrupt flag flags
<i>IPA_INT_FLAG_TAE</i>	transfer access error interrupt flag
<i>IPA_INT_FLAG_FTF</i>	full transfer finish interrupt flag
<i>IPA_INT_FLAG_TLM</i>	transfer line mark interrupt flag
<i>IPA_INT_FLAG_LAC</i>	LUT access conflict interrupt flag
<i>IPA_INT_FLAG_LLF</i>	LUT loading finish interrupt flag
<i>IPA_INT_FLAG_WCF</i>	wrong configuration interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check whether full transfer finish interrupt flag is SET */
while(RESET ==ipa_interrupt_flag_get(IPA_INT_FLAG_FTF));
```

ipa_interrupt_flag_clear

The description of ipa_interrupt_flag_clear is shown as below:

Table 3-980. Function ipa_interrupt_flag_clear

Function name	ipa_interrupt_flag_clear
Function prototype	void ipa_interrupt_flag_clear(uint32_t int_flag);
Function descriptions	clear IPA interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	IPA interrupt flag flags
<i>IPA_INT_FLAG_TAE</i>	transfer access error interrupt flag
<i>IPA_INT_FLAG_FTF</i>	full transfer finish interrupt flag
<i>IPA_INT_FLAG_TLM</i>	transfer line mark interrupt flag
<i>IPA_INT_FLAG_LAC</i>	LUT access conflict interrupt flag
<i>IPA_INT_FLAG_LLF</i>	LUT loading finish interrupt flag
<i>IPA_INT_FLAG_WCF</i>	wrong configuration interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
if(ipa_interrupt_flag_get(IPA_INT_FLAG_FTF) != RESET){
    /* clear full transfer finish interrupt flag */
    ipa_interrupt_flag_clear(IPA_INT_FLAG_FTF);
}
```

3.27. 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 [3.27.1](#), the LPDTS firmware functions are introduced in chapter [3.27.2](#).

3.27.1. Descriptions of Peripheral registers

Table 3-981. 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

3.27.2. Descriptions of Peripheral functions

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

Table 3-982. 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
lpdts_soft_trigger_disable	disable LPDTS software trigger

Function name	Function description
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

Table3-983. 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_11, SPT_CLOCK_12, SPT_CLOCK_13, SPT_CLOCK_14, SPT_CLOCK_15)

lpdts_deinit

The description of lpdts_deinit is shown as below:

Table3-984. 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:

Table3-985. Function lpdts_struct_para_init

Function name	lpdts_struct_para_init
Function prototype	void lpdts_struct_para_init(lpdts_parameter_struct* init_struct);
Function descriptions	initialize the parameters of LPDTS struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	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 Table3-983. Structure lpdts_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-986. Function lpdts_init

Function name	lpdts_init
Function prototype	void lpdts_init(lpdts_parameter_struct* init_struct);
Function descriptions	initialize LPDTS peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	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 Table3-983. Structure lpdts_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-987. Function lpdts_enable

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

Example:

```
/* enable LPDTS temperature sensor */

lpdts_enable();
```

lpdts_disable

The description of lpdts_disable is shown as below:

Table3-988. Function lpdts_disable

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

Example:

```
/* disable LPDTS temperature sensor */

lpdts_disable();
```

lpdts_soft_trigger_enable

The description of lpdts_soft_trigger_enable is shown as below:

Table3-989. Function lpdts_soft_trigger_enable

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

Example:

```
/* software trigger start */

lpdts_soft_trigger_enable();
```

lpdts_soft_trigger_disable

The description of lpdts_soft_trigger_disable is shown as below:

Table3-990. Function lpdts_soft_trigger_disable

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

Example:

```
/* disable LPDTS software trigger */
```

```
lpdts_soft_trigger_disable();
```

lpdts_high_threshold_set

The description of lpdts_high_threshold_set is shown as below:

Table3-991. Function lpdts_high_threshold_set

Function name	lpdts_high_threshold_set
Function prototype	void lpdts_high_threshold_set(uint16_t value);
Function descriptions	configure LPDTS interrupt high threshold
Precondition	-
The called functions	-
Input parameter{in}	
value	interrupt high threshold
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set threshold value */
```

```
lpdts_high_threshold_set(0U);
```

lpdts_low_threshold_set

The description of lpdts_low_threshold_set is shown as below:

Table3-992. Function lpdts_low_threshold_set

Function name	lpdts_low_threshold_set
Function prototype	void lpdts_low_threshold_set(uint16_t value);
Function descriptions	configure LPDTS interrupt low threshold
Precondition	-
The called functions	-
Input parameter{in}	
value	interrupt low threshold
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-993. Function lpdts_ref_clock_source_config

Function name	lpdts_ref_clock_source_config
Function prototype	void lpdts_ref_clock_source_config(uint32_t source);
Function descriptions	configure LPDTS clock source
Precondition	-
The called functions	-
Input parameter{in}	
source	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:

Table3-994. Function lpdts_temperature_get

Function name	lpdts_temperature_get
Function prototype	int32_t lpdts_temperature_get(void);
Function descriptions	get LPDTS collection temperature value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
int32_t	The resulting temperature value

Example:

```
int32_t temperature;
```

```

/* wait till TSRF flag is set */

while(RESET == lpdts_flag_get(LPPTS_FLAG_TSRF)){

}

temperature = lpdts_temperature_get();

```

lpdts_flag_get

The description of lpdts_flag_get is shown as below:

Table3-995. Function lpdts_flag_get

Function name	lpdts_flag_get
Function prototype	FlagStatus lpdts_flag_get(uint32_t flag);
Function descriptions	get LPPTS status flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	interrupt flag
<i>LPPTS_FLAG_TSR</i>	temperature sensor ready flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```

/* wait till TSRF flag is set */

while(RESET == lpdts_flag_get(LPPTS_FLAG_TSR)) {

}

```

lpdts_interrupt_enable

The description of lpdts_interrupt_enable is shown as below:

Table3-996. Function lpdts_interrupt_enable

Function name	lpdts_interrupt_enable
Function prototype	void lpdts_interrupt_enable(uint32_t interrupt);
Function descriptions	enable LPPTS interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	interrupt type
<i>LPPTS_INT_EM</i>	end of measurement interrupt (synchronized on PCLK)
<i>LPPTS_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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable end of measurement interrupt */
```

```
lpdts_interrupt_enable(LPDTS_INT_EM);
```

lpdts_interrupt_disable

The description of lpdts_interrupt_disable is shown as below:

Table3-997. Function lpdts_interrupt_disable

Function name	lpdts_interrupt_disable
Function prototype	void lpdts_interrupt_disable(uint32_t interrupt);
Function descriptions	disable LPDTS interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable end of measurement interrupt */
```

```
lpdts_interrupt_disable(LPDTS_INT_EM);
```

lpdts_interrupt_flag_get

The description of lpdts_interrupt_flag_get is shown as below:

Table3-998. Function `lpdts_interrupt_flag_get`

Function name	<code>lpdts_interrupt_flag_get</code>
Function prototype	<code>FlagStatus lpdts_interrupt_flag_get(uint32_t flag);</code>
Function descriptions	get LPDTS interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	interrupt flag
<code>LPDTS_INT_FLAG_EM</code>	end of measurement interrupt flag
<code>LPDTS_INT_FLAG_LT</code>	low threshold interrupt flag
<code>LPDTS_INT_FLAG_HT</code>	high threshold interrupt flag
<code>LPDTS_INT_FLAG_EMA</code>	end of measurement asynchronous interrupt flag
<code>LPDTS_INT_FLAG_LTA</code>	low threshold asynchronous interrupt flag
<code>LPDTS_INT_FLAG-HTA</code>	high threshold asynchronous interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* wait till HTA flag is set */
while(RESET == lpdts_interrupt_flag_get(LPDTES_INT_FLAG-HTA)) {
}
```

`lpdts_interrupt_flag_clear`

The description of `lpdts_interrupt_flag_clear` is shown as below:

Table3-999. Function `lpdts_interrupt_flag_clear`

Function name	<code>lpdts_interrupt_flag_clear</code>
Function prototype	<code>void lpdts_interrupt_flag_clear(uint32_t flag);</code>
Function descriptions	clear LPDTS interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	interrupt flag
<code>LPDTS_INT_FLAG_EM</code>	end of measurement interrupt flag
<code>LPDTS_INT_FLAG_LT</code>	low threshold interrupt flag
<code>LPDTS_INT_FLAG-HT</code>	high threshold interrupt flag
<code>LPDTS_INT_FLAG_EMA</code>	end of measurement asynchronous interrupt flag
<code>LPDTS_INT_FLAG_LTA</code>	low threshold asynchronous interrupt flag
<code>LPDTS_INT_FLAG-HTA</code>	high threshold asynchronous interrupt flag

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear HTA flag */
```

```
lpdts_interrupt_flag_clear(LPDTTS_INT_FLAG_HTA);
```

3.28. MDIO

The MDIO interface can receive complete MDIO frames. The MDIO registers are listed in chapter [3.28.1](#), the MDIO firmware functions are introduced in chapter [3.28.2](#).

3.28.1. Descriptions of Peripheral registers

MDIO registers are listed in the table shown as below:

Table 3-1000. MDIO Registers

Registers	Descriptions
MDIO_CTL	MDIO control register
MDIO_RFRM	MDIO received frame information register
MDIO_RDATA	MDIO received data register
MDIO_RADDR	MDIO received address register
MDIO_TDATA	MDIO transfer data register
MDIO_CFG	MDIO configuration register
MDIO_STAT	MDIO status register
MDIO_INTEN	MDIO interrupt enable register
MDIO_PIN	MDIO pin value register
MDIO_TO	MDIO timeout register

3.28.2. Descriptions of Peripheral functions

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

Table 3-1001. MDIO firmware function

Function name	Function description
mdio_deinit	reset MDIO
mdio_software_reset	reset MDIO block
mdio_init	initialize MDIO for communication
mdio_phy_length_config	configure MDIO phy bit length
mdio_soft_phyadr_set	set the software PHYADR value
mdio_framefield_phyadr_config	select the expected frame field PHYADR

Function name	Function description
mdio_framefield_devadd_config	configure the expected frame field DEVADD
mdio_phy_pin_read	read the hardware PRTADR[4:0] value
mdio_timeout_config	configure the expected frame bit timeout
mdio_timeout_enable	enable MDIO frame bit timeout
mdio_timeout_disable	disable MDIO frame bit timeout
mdio_op_receive	read the received frame field OP
mdio_phyadr_receive	read the received frame field PHYADR
mdio_devadd_receive	read the received frame field DEVADD
mdio_ta_receive	read the received frame field TA
mdio_data_receive	read the received frame field DATA
mdio_address_receive	read the received frame field ADDRESS
mdio_data_transmit	transmit the frame field DATA
mdio_flag_get	get the flag status of the frame
mdio_flag_clear	clear MDIO flag status
mdio_interrupt_enable	enable MDIO interrupt
mdio_interrupt_disable	disable MDIO interrupt

mdio_deinit

The description of mdio_deinit is shown as below:

Table 3-1002. Function mdio_deinit

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

Example:

```
/* reset MDIO */
mdio_deinit();
```

mdio_software_reset

The description of mdio_software_reset is shown as below:

Table 3-1003. Function `mdio_software_reset`

Function name	<code>mdio_software_reset</code>
Function prototype	<code>void mdio_software_reset(void);</code>
Function descriptions	reset MDIO block
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset MDIO block */
mdio_software_reset();
```

mdio_init

The description of `mdio_init` is shown as below:

Table 3-1004. Function `mdio_init`

Function name	<code>mdio_init</code>
Function prototype	<code>uint32_t mdio_init(uint32_t phy_size, uint32_t phy_softaddr, uint32_t phy_sel, uint16_t devaddr);</code>
Function descriptions	initialize MDIO for communication
Precondition	-
The called functions	-
Input parameter{in}	
phy_size	PHY bit length
<code>MDIO_PHY_BITS_3</code>	PHY use 3 bits
<code>MDIO_PHY_BITS_5</code>	PHY use 5 bits
Input parameter{in}	
phy_softaddr	software provided PHYADR
<code>0-31</code>	software PHYADR value
Input parameter{in}	
phy_sel	PHYADR select
<code>MDIO_PHYADR_HARWARE</code>	sets expected PHYADR = PHYPIN[4:0]
<code>MDIO_PHYADR_HW_SW_MIX</code>	sets Software address valid bits
<code>MDIO_PHYADR_SOFTWARE</code>	sets expected PHYADR = PHYSW[4:0]

1-30	other user defined value
Input parameter{in}	
devadd	device type
<i>DEVADD_PMA_PMD</i>	device type PMA/PMD
<i>DEVADD_WIS</i>	device type WIS
<i>DEVADD_PCS</i>	device type PCS
<i>DEVADD_PHY_XS</i>	device type PHY XS
<i>DEVADD_DTE_XS</i>	device type DTE XS
Output parameter{out}	
-	-
Return value	
uint32_t	the PHYADR that the device will respond to 0 - 31

Example:

```
/* PHY use 5 bits, select software provided address '0x0' as PHYADR, DEVADD is 0x1 */
mdio_init(MDIO_PHY_BITS_5, 0x0, MDIO_PHYADR_SOFTWARE, DEVADD_PMA_PMD);
```

mdio_phy_length_config

The description of mdio_phy_length_config is shown as below:

Table 3-1005. Function mdio_phy_length_config

Function name	mdio_phy_length_config
Function prototype	void mdio_phy_length_config(uint32_t phy_bit);
Function descriptions	configure MDIO phy bit length
Precondition	-
The called functions	-
Input parameter{in}	
phy_bit	PHY bit length
<i>MDIO_PHY_BITS_3</i>	PHY use 3 bits
<i>MDIO_PHY_BITS_5</i>	PHY use 5 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure MDIO phy bit length */
mdio_phy_length_config(MDIO_PHY_BITS_3);
```

mdio_soft_phyadr_set

The description of mdio_soft_phyadr_set is shown as below:

Table 3-1006. Function `mdio_soft_phyadr_set`

Function name	<code>mdio_soft_phyadr_set</code>
Function prototype	<code>void mdio_soft_phyadr_set(uint32_t phy_soft);</code>
Function descriptions	set the software PHYADR value
Precondition	-
The called functions	-
Input parameter{in}	
phy_soft	software provided PHYADR
0-31	software PHYADR value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the software PHYADR value */
```

```
mdio_soft_phyadr_set (1);
```

`mdio_framefield_phyadr_config`

The description of `mdio_framefield_phyadr_config` is shown as below:

Table 3-1007. Function `mdio_framefield_phyadr_config`

Function name	<code>mdio_framefield_phyadr_config</code>
Function prototype	<code>void mdio_framefield_phyadr_config(uint32_t phy_sel);</code>
Function descriptions	select the expected frame field PHYADR
Precondition	-
The called functions	-
Input parameter{in}	
phy_sel	PHYADR select
<i>MDIO_PHYADR_HARDWARE</i>	sets expected PHYADR = PHYPIN[4:0]
<i>MDIO_PHYADR_SOFTWARE</i>	sets expected PHYADR = PHYSW[4:0]
1-30	expected PHYADR selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select the expected PHYADR */
```

```
mdio_framefield_phyadr_config(MDIO_PHYADR_HARDWARE);
```

mdio_framefield_devadd_config

The description of mdio_framefield_devadd_config is shown as below:

Table 3-1008. Function mdio_framefield_devadd_config

Function name	mdio_framefield_devadd_config
Function prototype	void mdio_framefield_devadd_config(uint16_t type);
Function descriptions	configure the expected frame field DEVADD
Precondition	-
The called functions	-
Input parameter{in}	
type	device type
DEVADD_PMA_PMD	device type PMA/PMD
DEVADD_WIS	device type WIS
DEVADD_PCS	device type PCS
DEVADD_PHY_XS	device type PHY XS
DEVADD_DTE_XS	device type DTE XS
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DEVADD */
```

```
mdio_framefield_devadd_config(DEVADD_PMA_PMD);
```

mdio_phy_pin_read

The description of mdio_phy_pin_read is shown as below:

Table 3-1009. Function mdio_phy_pin_read

Function name	mdio_phy_pin_read
Function prototype	uint32_t mdio_phy_pin_read(void);
Function descriptions	read the hardware PRTADR[4:0] value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	PRTADR[4:0] pin value 0 – 0x1F

Example:

```
uint32_t hard_phy = 0;

/* read the hardware PRTADR[4:0] value */

hard_phy = mdio_phy_pin_read();
```

mdio_timeout_config

The description of mdio_timeout_config is shown as below:

Table 3-1010. Function mdio_timeout_config

Function name	mdio_timeout_config
Function prototype	void mdio_timeout_config(uint16_t timeout);
Function descriptions	configure the expected frame bit timeout
Precondition	-
The called functions	-
Input parameter{in}	
timeout	timeout counter among frame bits
0 - 0xFFFF	timeout counter value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the frame bit timeout */

mdio_timeout_config(0xFFFF);
```

mdio_timeout_enable

The description of mdio_timeout_enable is shown as below:

Table 3-1011. Function mdio_timeout_enable

Function name	mdio_timeout_enable
Function prototype	void mdio_timeout_enable(void);
Function descriptions	enable MDIO frame bit timeout
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable MDIO frame bit timeout */
```

```
mdio_timeout_enable();
```

mdio_timeout_disable

The description of mdio_timeout_disable is shown as below:

Table 3-1012. Function mdio_timeout_disable

Function name	mdio_timeout_disable
Function prototype	void mdio_timeout_disable(void);
Function descriptions	disable MDIO frame bit timeout
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable MDIO frame bit timeout */
```

```
mdio_timeout_disable();
```

mdio_op_receive

The description of mdio_op_receive is shown as below:

Table 3-1013. Function mdio_op_receive

Function name	mdio_op_receive
Function prototype	uint16_t mdio_op_receive(void);
Function descriptions	read the received frame field OP
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	OP of received 0x0-0x11

Example:

```
uint16_t rx_op = 0;
```

```
/* read the received frame field OP */
```

```
rx_op = mdio_op_receive();
```

mdio_phyadr_receive

The description of mdio_phyadr_receive is shown as below:

Table 3-1014. Function mdio_phyadr_receive

Function name	mdio_phyadr_receive
Function prototype	uint16_t mdio_phyadr_receive(void);
Function descriptions	read the received frame field PHYADR
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	PHYADR of received 0 – 0x1F

Example:

```
uint16_t rx_phyadr = 0;
```

```
/* read the received frame field PHYADR */
```

```
rx_phyadr = mdio_phyadr_receive();
```

mdio_devadd_receive

The description of mdio_devadd_receive is shown as below:

Table 3-1015. Function mdio_devadd_receive

Function name	mdio_devadd_receive
Function prototype	uint16_t mdio_devadd_receive(void);
Function descriptions	read the received frame field DEVADD
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	DEVADD of received 0 – 0x1F

Example:


```
/* read the received frame field DEVADD */
```

```
uint16_t rx_devadd = 0;
rx_devadd = mdio_devadd_receive();
```

mdio_ta_receive

The description of mdio_ta_receive is shown as below:

Table 3-1016. Function mdio_ta_receive

Function name	mdio_ta_receive
Function prototype	uint16_t mdio_ta_receive(void);
Function descriptions	read the received frame field TA
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	TA of received 0 – 3

Example:

```
/* read the received frame field TA */
```

```
uint16_t rx_ta = 0;
rx_ta = mdio_ta_receive();
```

mdio_data_receive

The description of mdio_data_receive is shown as below:

Table 3-1017. Function mdio_data_receive

Function name	mdio_data_receive
Function prototype	uint16_t mdio_data_receive(void);
Function descriptions	read the received frame field DATA
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	DATA of received 0 – 0xFFFF

Example:

```
/* read the received frame field DATA */
```

```
uint16_t rx_data = 0;
rx_data = mdio_data_receive();
```

mdio_address_receive

The description of mdio_address_receive is shown as below:

Table 3-1018. Function mdio_address_receive

Function name	mdio_address_receive
Function prototype	uint16_t mdio_address_receive(void);
Function descriptions	read the received frame field ADDRESS
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	ADDRESS of received 0 – 0xFFFF

Example:

```
/* read the received frame field ADDRESS */
```

```
uint16_t rx_address = 0;
rx_address = mdio_address_receive();
```

mdio_data_transmit

The description of mdio_data_transmit is shown as below:

Table 3-1019. Function mdio_data_transmit

Function name	mdio_data_transmit
Function prototype	void mdio_data_transmit(uint16_t data);
Function descriptions	transmit the frame field DATA
Precondition	-
The called functions	-
Input parameter{in}	
data	data to put in a read or post read increment address frame for transmission
0 - 0xFFFF	data to transmit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* transmit data 1 */
mdio_data_transmit(0x1);
```

mdio_flag_get

The description of mdio_flag_get is shown as below:

Table 3-1020. Function mdio_flag_get

Function name	mdio_flag_get
Function prototype	FlagStatus mdio_flag_get(uint32_t flag);
Function descriptions	get the flag status of the frame
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag status to get
MDIO_FLAG_WFRM	a write data frame flag status
MDIO_FLAG_ADDRFRM M	an address frame flag status
MDIO_FLAG_RDINCF RM	a post read increment address frame flag status
MDIO_FLAG_RDFRM	a read data frame flag status
MDIO_FLAG_DEVM	a DEVADD match frame flag status
MDIO_FLAG_DEVM	a DEVADD nonmatch frame flag status
MDIO_FLAG_PHYM	a PHYADR match frame flag status
MDIO_FLAG_PHYNM	a PHYADR nonmatch frame flag status
MDIO_FLAG_TANM	a TA nonmatch frame flag status
MDIO_FLAG_TIMEOUT T	timeout flag
MDIO_FLAG_TX_UNDE ERRUN	transmit underrun flag
MDIO_FLAG_RX_OVE RRUN	receive overrun flag
MDIO_FLAG_RBNE	read data buffer not empty flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the address frame state */

FlagStatus flag;
flag = mdio_flag_get(MDIO_FLAG_ADDRFRM);
```

mdio_flag_clear

The description of mdio_flag_clear is shown as below:

Table 3-1021. Function mdio_flag_clear

Function name	mdio_flag_clear
Function prototype	void mdio_flag_clear(uint32_t flag);
Function descriptions	clear the flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	MDIO flag
MDIO_FLAG_WRFRM	a write data frame flag status
MDIO_FLAG_ADDRFRM	an address frame flag status
MDIO_FLAG_RDINCFRM	a post read increment address frame flag status
MDIO_FLAG_RDFRM	a read data frame flag status
MDIO_FLAG_DEVM	a DEVADD match frame flag status
MDIO_FLAG_DEVNM	a DEVADD nonmatch frame flag status
MDIO_FLAG_PHYM	a PHYADR match frame flag status
MDIO_FLAG_PHYNM	a PHYADR nonmatch frame flag status
MDIO_FLAG_TANM	a TA nonmatch frame flag status
MDIO_FLAG_TIMEOUT	timeout flag
MDIO_FLAG_TX_UNDEERRUN	transmit underrun flag
MDIO_FLAG_RX_OVERRUN	receive overrun flag
MDIO_FLAG_RBNE	read data buffer not empty flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear read data buffer not empty flag */
mdio_flag_clear(MDIO_FLAG_RBNE);
```

mdio_interrupt_enable

The description of mdio_interrupt_enable is shown as below:

Table 3-1022. Function `mdio_interrupt_enable`

Function name	<code>mdio_interrupt_enable</code>
Function prototype	<code>void mdio_interrupt_enable(uint32_t interrupt);</code>
Function descriptions	enable MDIO interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	MDIO interrupt
<code>MDIO_INT_WRFRM</code>	a write data frame interrupt
<code>MDIO_INT_ADDRFRM</code>	an address frame interrupt
<code>MDIO_INT_RDINCFRM</code>	a post read increment address frame interrupt
<code>MDIO_INT_RDFRM</code>	a read data frame interrupt
<code>MDIO_INT_DEVM</code>	a DEVADD match frame interrupt
<code>MDIO_INT_DEVNM</code>	a DEVADD nonmatch frame interrupt
<code>MDIO_INT_PHYM</code>	a PHYADR match frame interrupt
<code>MDIO_INT_PHYNM</code>	a PHYADR nonmatch frame interrupt
<code>MDIO_INT_TANM</code>	a TA nonmatch frame flag interrupt
<code>MDIO_INT_TIMEOUT</code>	a timeout interrupt
<code>MDIO_INT_TX_UNDE RRUN</code>	a transmit underrun interrupt
<code>MDIO_INT_RX_OVER RUN</code>	a receive overrun interrupt
<code>MDIO_INT_RBNE</code>	a read data buffer not empty interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable address frame interrupt */
mdio_interrupt_enable (MDIO_INT_ADDRFRM);
```

`mdio_interrupt_disable`

The description of `mdio_interrupt_disable` is shown as below:

Table 3-1023. Function `mdio_interrupt_disable`

Function name	<code>mdio_interrupt_disable</code>
Function prototype	<code>void mdio_interrupt_disable(uint32_t interrupt);</code>
Function descriptions	disable MDIO interrupt
Precondition	-
The called functions	-
Input parameter{in}	

interrupt	MDIO interrupt
<i>MDIO_INT_WRFRM</i>	a write data frame interrupt
<i>MDIO_INT_ADDRFRM</i>	an address frame interrupt
<i>MDIO_INT_RDINCFRM</i>	a post read increment address frame interrupt
<i>MDIO_INT_RDFRM</i>	a read data frame interrupt
<i>MDIO_INT_DEVM</i>	a DEVADD match frame interrupt
<i>MDIO_INT_DEVNM</i>	a DEVADD nonmatch frame interrupt
<i>MDIO_INT_PHYM</i>	a PHYADR match frame interrupt
<i>MDIO_INT_PHYNM</i>	a PHYADR nonmatch frame interrupt
<i>MDIO_INT_TANM</i>	a TA nonmatch frame flag interrupt
<i>MDIO_INT_TIMEOUT</i>	a timeout interrupt
<i>MDIO_INT_TX_UNDE RRUN</i>	a transmit underrun interrupt
<i>MDIO_INT_RX_OVER RUN</i>	a receive overrun interrupt
<i>MDIO_INT_RBNE</i>	a read data buffer not empty interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable address frame interrupt */
mdio_interrupt_disable (MDIO_INT_ADDRFRM);
```

3.29. 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 [3.29.1](#), the MDMA firmware functions are introduced in chapter [3.29.2](#).

3.29.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

Table 3-1024. MDMA Registers

Registers	Descriptions
MDMA_GINTF	global interrupt flag register
MDMA_CHxSTAT0	channel x status register 0
MDMA_CHxSTATC	channel x status clear register
MDMA_CHxSTAT1	channel x status register 1
MDMA_CHxCTL0	channel x control register 0

Registers	Descriptions
MDMA_CHxCFG	channel x configure register
MDMA_CHxBTCFG	channel x block transfer configure register
MDMA_CHxSADDR	channel x source address register
MDMA_CHxDADDR	channel x destination address register
MDMA_CHxMBAD DRU	channel x multi-block address update register
MDMA_CHxLADDR	channel x link address register
MDMA_CHxCTL1	channel x control register 1
MDMA_CHxMADDR	channel x mask address register
MDMA_CHxMDATA	channel x mask data register

3.29.2. Descriptions of Peripheral functions

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

Table 3-1025. 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

Function name	Function description
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
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 3-1026. 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
destination_addr	specifies the destination address
tbytes_num_in_block	specifies the transfer bytes number in a buffer or block transfer

Member name	Function description
source_bus	specifies the source bus
destination_bus	specifies the destination bus
bufferable_write_mode	specifies the bufferable write mode

Structure mdma_multi_block_parameter_struct

Table 3-1027. 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 3-1028. 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 3-1029. Enum mdma_add_update_dir_enum

Member name	Function description
UPDATE_DIR_INCREMENT	MDMA address update increase
UPDATE_DIR_DECREMENT	MDMA address update decrease

Enum mdma_channel_enum

Table 3-1030. 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 3-1031. Function mdma_deinit

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

Example:

```
/* deinitialize MDMA */
mdma_deinit();
```

mdma_channel_deinit

The description of mdma_channel_deinit is shown as below:

Table 3-1032. Function mdma_channel_deinit

Function name	mdma_channel_deinit
Function prototype	void mdma_channel_deinit(mdma_channel_enum channelx);
Function descriptions	deinitialize MDMA registers of a channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Output parameter{out}	
-	-
Return value	
-	-

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 3-1033. Function mdma_para_struct_init

Function name	mdma_para_struct_init
Function prototype	void mdma_para_struct_init(mdma_parameter_struct *init_struct);
Function descriptions	initialize the MDMA parameters struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
init_struct	the initialization data needed to initialize MDMA channel, refer to Table 3-1026. Structure mdma_parameter_struct .
Return value	
-	-

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 3-1034. Function mdma_multi_block_para_struct_init

Function name	mdma_multi_block_para_struct_init
Function prototype	void mdma_multi_block_para_struct_init (mdma_multi_block_parameter_struct *block_init_struct);
Function descriptions	initialize the MDMA multi block transfer mode parameters struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
block_init_struct	the initialization data needed to initialize MDMA channel, refer to Table 3-1027. Structure mdma_multi_block_parameter_struct .
Return value	
-	-

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 3-1035. Function mdma_link_node_para_struct_init

Function name	mdma_link_node_para_struct_init
Function prototype	void mdma_link_node_para_struct_init(mdma_link_node_parameter_struct *node);
Function descriptions	initialize the MDMA link node configuration struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
node	initialize the MDMA link node configuration struct with the default values,

	refer to Table 3-1028. Structure mdma_link_node_parameter_struct.
Return value	
-	-

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 3-1036. Function mdma_init

Function name	mdma_init
Function prototype	void mdma_init(mdma_channel_enum channelx, mdma_parameter_struct *init_struct);
Function descriptions	initialize MDMA channel with MDMA parameter structure
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum.
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to Table 3-1026. Structure mdma_parameter_struct.
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.destination_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.destination_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 3-1037. Function mdma_buffer_block_mode_config

Function name	mdma_buffer_block_mode_config
Function prototype	void mdma_buffer_block_mode_config(mdma_channel_enum channelx, uint32_t saddr, uint32_t daddr, uint32_t tbnun);
Function descriptions	configure MDMA buffer/block transfer mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
saddr	source address, 0x00000000-0xFFFFFFFF
Input parameter{in}	
daddr	destination address, 0x00000000-0xFFFFFFFF
Input parameter{in}	
tbnun	number of bytes to transfer, 0x00000000-0x00010000

Output parameter{out}	
-	-
Return value	
-	-

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 3-1038. Function mdma_multi_block_mode_config

Function name	mdma_multi_block_mode_config
Function prototype	void mdma_multi_block_mode_config(mdma_channel_enum channelx, uint32_t tbnun, mdma_multi_block_parameter_struct *block_init_struct);
Function descriptions	configure MDMA multi block transfer mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
tbnun	number of bytes to transfer in block, 0x00000000-0x00000FFF
Input parameter{in}	
block_init_struct	Structure for MDMA multi block mode, the structure members can refer to Table 3-1027. Structure mdma_multi_block_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

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 3-1039. Function mdma_node_create

Function name	mdma_node_create
Function prototype	void mdma_node_create(mdma_link_node_parameter_struct *node, mdma_multi_block_parameter_struct *block_init_struct, mdma_parameter_struct *init_struct);
Function descriptions	MDMA link list node create
Precondition	-
The called functions	-
Input parameter{in}	
block_init_struct	Structure for MDMA multi block mode, the structure members can refer to Table 3-1027. Structure mdma_multi_block_parameter_struct.
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to Table 3-1026. Structure mdma_parameter_struct.
Output parameter{out}	
node	Structure for link node, the structure members can refer to Table 3-1028. Structure mdma_link_node_parameter_struct.
Return value	
-	-

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 3-1040. 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 Table 3-1028. Structure mdma_link_node_parameter_struct.
Input parameter{in}	

new_node	Structure for new node, the structure members can refer to Table 3-1028. Structure mdma link node parameter struct.
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 3-1041. 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}	
pre_node	Structure for previous node, the structure members can refer to Table 3-1028. Structure mdma link node parameter struct.
Input parameter{in}	
new_node	Structure for new node, the structure members can refer to Table 3-1028. Structure mdma link node parameter struct.
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-1042. Function mdma_destination_address_config

Function name	mdma_destination_address_config
Function prototype	void mdma_destination_address_config(mdma_channel_enum channelx, uint32_t address);
Function descriptions	configure MDMA destination base address
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
address	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 3-1043. Function mdma_source_address_config

Function name	mdma_source_address_config
Function prototype	void mdma_source_address_config(mdma_channel_enum channelx, uint32_t address);
Function descriptions	configure MDMA source base address
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
address	Source base address, 0x00000000-0xFFFFFFFF
Output parameter{out}	

-	-
Return value	
-	-

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 3-1044. Function mdma_destination_bus_config

Function name	mdma_destination_bus_config
Function prototype	void mdma_destination_bus_config(mdma_channel_enum channelx, uint32_t bus);
Function descriptions	configure MDMA destination bus
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
bus	destination bus
MDMA_DESTINATION_AXI	destination bus is the system bus or AXI bus
MDMA_DESTINATION_AHB_TCM	destination bus is AHB bus or TCM
Output parameter{out}	
-	-
Return value	
-	-

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 3-1045. Function mdma_source_bus_config

Function name	mdma_source_bus_config
----------------------	------------------------

Function prototype	void mdma_source_bus_config(mdma_channel_enum channelx, uint32_t bus);
Function descriptions	configure MDMA source bus
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum.
Input parameter{in}	
bus	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1046. Function mdma_priority_config

Function name	mdma_priority_config
Function prototype	void mdma_priority_config(mdma_channel_enum channelx, uint32_t priority);
Function descriptions	configure priority level of MDMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum.
Input parameter{in}	
priority	priority Level of this channel
<i>MDMA_PRIORITY_LO</i> <i>W</i>	low priority
<i>MDMA_PRIORITY_ME</i>	medium priority

<i>DIUM</i>	
<i>MDMA_PRIORITY_HIGH</i>	high priority
<i>MDMA_PRIORITY_ULTRA_HIGH</i>	ultra high priority
Output parameter{out}	
-	-
Return value	
-	-

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 3-1047. Function mdma_endianness_config

Function name	mdma_endianness_config
Function prototype	void mdma_endianness_config(mdma_channel_enum channelx, uint32_t endianness);
Function descriptions	configure endianness of MDMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
endianness	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
Output parameter{out}	
-	-
Return value	

-	-
---	---

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 3-1048. Function mdma_alignment_config

Function name	mdma_alignment_config
Function prototype	void mdma_endianness_config(mdma_channel_enum channelx, uint32_t endianness);
Function descriptions	configure data alignment of MDMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
alignment	MDMA data alignment
MDMA_DATAALIGN_PACKEN	pack/unpack the source data to match the destination data size
MDMA_DATAALIGN_RIGHT	right aligned, padded with 0s (default)
MDMA_DATAALIGN_RIGHT_SIGNED	right aligned with sign extended, note: this mode is allowed only if the source data size is smaller than destination data size
MDMA_DATAALIGN_LEFT	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1049. Function mdma_source_burst_beats_config

Function name	mdma_source_burst_beats_config
Function prototype	void mdma_source_burst_beats_config(mdma_channel_enum channelx, uint32_t sbeat);
Function descriptions	configure transfer burst beats of source
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
sbeat	source transfer burst beats
MDMA_SOURCE_BURST_SINGLE	single burst
MDMA_SOURCE_BURST_2BEATS	2-beat incrementing burst
MDMA_SOURCE_BURST_4BEATS	4-beat incrementing burst
MDMA_SOURCE_BURST_8BEATS	8-beat incrementing burst
MDMA_SOURCE_BURST_16BEATS	16-beat incrementing burst
MDMA_SOURCE_BURST_32BEATS	32-beat incrementing burst
MDMA_SOURCE_BURST_64BEATS	64-beat incrementing burst
MDMA_SOURCE_BURST_128BEATS	128-beat incrementing burst
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 3-1050. Function mdma_destination_burst_beats_config

Function name	mdma_destination_burst_beats_config
Function prototype	void mdma_destination_burst_beats_config(mdma_channel_enum channelx, uint32_t dbeat);
Function descriptions	configure transfer burst beats of destination
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
dbeat	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
Output parameter{out}	
-	-
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 3-1051. Function mdma_source_width_config

Function name	mdma_source_width_config
Function prototype	void mdma_source_width_config(mdma_channel_enum channelx, uint32_t swidth);
Function descriptions	configure data size of source
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
swidth	source data size
MDMA_SOURCE_DATASIZE_8BIT	data size of source is 8-bit
MDMA_SOURCE_DATASIZE_16BIT	data size of source is 16-bit
MDMA_SOURCE_DATASIZE_32BIT	data size of source is 32-bit
MDMA_SOURCE_DATASIZE_64BIT	data size of source is 64-bit
Output parameter{out}	
-	-
Return value	
-	-

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 3-1052. Function mdma_destination_width_config

Function name	mdma_destination_width_config
Function prototype	void mdma_destination_width_config(mdma_channel_enum channelx, uint32_t dwidth);
Function descriptions	configure data size of destination
Precondition	-

The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
dwidth	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1053. Function mdma_source_increment_config

Function name	mdma_source_increment_config
Function prototype	void mdma_source_increment_config(mdma_channel_enum channelx, uint32_t sinc);
Function descriptions	configure source adress increment mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
sinc	source adress increment mode
MDMA_SOURCE_INCREMENT_DISABLE	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 3-1054. Function mdma_destination_increment_config

Function name	mdma_destination_increment_config
Function prototype	void mdma_destination_increment_config(mdma_channel_enum channelx, uint32_t dinc);
Function descriptions	configure destination adress increment mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
dinc	destination adress increment mode
<i>MDMA_DESTINATION</i>	no increment

<code>_INCREASE_DISABLE</code>	
<code>MDMA_DESTINATION_INCREMENT_8BIT</code>	destination address is increased by 8-bit
<code>MDMA_DESTINATION_INCREMENT_16BIT</code>	destination address is increased by 16-bit
<code>MDMA_DESTINATION_INCREMENT_32BIT</code>	destination address is increased by 32-bit
<code>MDMA_DESTINATION_INCREMENT_64BIT</code>	destination address is increased by 64-bit
<code>MDMA_DESTINATION_DECREMENT_8BIT</code>	destination address is decreased by 8-bit
<code>MDMA_DESTINATION_DECREMENT_16BIT</code>	destination address is decreased by 16-bit
<code>MDMA_DESTINATION_DECREMENT_32BIT</code>	destination address is decreased by 32-bit
<code>MDMA_DESTINATION_DECREMENT_64BIT</code>	destination address is decreased by 64-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure source address increment mode */
```

```
mdma_destination_increment_config (MDMA_CH0, MDMA_DESTINATION_INCREMENT_16BIT);
```

mdma_channel_bufferable_write_enable

The description of `mdma_channel_bufferable_write_enable` is shown as below:

Table 3-1055. Function `mdma_channel_bufferable_write_enable`

Function name	<code>mdma_channel_bufferable_write_enable</code>
Function prototype	<code>void mdma_channel_bufferable_write_enable(mdma_channel_enum channelx);</code>
Function descriptions	enable MDMA channel bufferable write mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<code>MDMA_CHx(x=0..15)</code>	MDMA channel selection, refer to Table 3-1030. Enum <code>mdma_channel_enum</code> .
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 3-1056. 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 Table 3-1030. Enum mdma_channel_enum .
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 3-1057. Function mdma_channel_software_request_enable

Function name	mdma_channel_software_request_enable
Function prototype	void mdma_channel_software_request_enable(mdma_channel_enum channelx);
Function descriptions	enable MDMA channel software request
Precondition	-
The called functions	-
Input parameter{in}	

channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum <i>mdma_channel_enum</i> .
Output parameter{out}	
-	-
Return value	
-	-

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 3-1058. Function mdma_channel_enable

Function name	mdma_channel_enable
Function prototype	void mdma_channel_enable(mdma_channel_enum channelx);
Function descriptions	enable MDMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum <i>mdma_channel_enum</i> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable MDMA channel */
mdma_channel_enable(MDMA_CH0);
```

mdma_channel_disable

The description of mdma_channel_disable is shown as below:

Table 3-1059. Function mdma_channel_disable

Function name	mdma_channel_disable
Function prototype	void mdma_channel_disable(mdma_channel_enum channelx);
Function descriptions	disable MDMA channel

Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum.
Output parameter{out}	
-	-
Return value	
-	-

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 3-1060. Function mdma_transfer_error_direction_get

Function name	mdma_transfer_error_direction_get
Function prototype	uint32_t mdma_transfer_error_direction_get(mdma_channel_enum channelx);
Function descriptions	get MDMA transfer error direction
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum.
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1061. Function mdma_transfer_error_address_get

Function name	mdma_transfer_error_address_get
Function prototype	uint32_t mdma_transfer_error_address_get(mdma_channel_enum channelx);
Function descriptions	get MDMA transfer error address
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1062. Function mdma_flag_get

Function name	mdma_flag_get
Function prototype	FlagStatus mdma_flag_get(mdma_channel_enum channelx, uint32_t flag);
Function descriptions	get MDMA flag
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
flag	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
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-1063. Function mdma_flag_clear

Function name	mdma_flag_clear
Function prototype	void mdma_flag_clear(mdma_channel_enum channelx, uint32_t flag);
Function descriptions	clear MDMA flag
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
flag	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>	link data transfer error flag in the last transfer of the channel

<i>R</i>	
<i>MDMA_FLAG_MDER</i>	mask data error flag
<i>R</i>	
<i>MDMA_FLAG_ASERR</i>	address and size error flag
<i>MDMA_FLAG_BZERR</i>	block size error flag
Output parameter{out}	
-	-
Return value	
-	-

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 3-1064. Function mdma_interrupt_enable

Function name	mdma_interrupt_enable
Function prototype	void mdma_interrupt_enable(mdma_channel_enum channelx, uint32_t interrupt);
Function descriptions	enable MDMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Output parameter{out}	
interrupt	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
Return value	
-	-

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 3-1065. Function mdma_interrupt_disable

Function name	mdma_interrupt_disable
Function prototype	void mdma_interrupt_disable(mdma_channel_enum channelx, uint32_t interrupt);
Function descriptions	disable MDMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to Table 3-1030. Enum mdma_channel_enum .
Input parameter{in}	
interrupt	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1066. Function mdma_interrupt_flag_get

Function name	mdma_interrupt_flag_get
Function prototype	FlagStatus mdma_interrupt_flag_get(mdma_channel_enum channelx, uint32_t int_flag);
Function descriptions	get MDMA interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel

<i>MDMA_CHx</i> (<i>x</i> =0..15)	MDMA channel selection, refer to Table 3-1030. Enum <i>mdma_channel_enum</i> .
Input parameter{in}	
int_flag	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
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-1067. Function `mdma_interrupt_flag_clear`

Function name	<code>mdma_interrupt_flag_clear</code>
Function prototype	<code>void mdma_interrupt_flag_clear(mdma_channel_enum channelx, uint32_t int_flag);</code>
Function descriptions	clear MDMA interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
<i>MDMA_CHx</i> (<i>x</i> =0..15)	MDMA channel selection, refer to Table 3-1030. Enum <i>mdma_channel_enum</i> .
Input parameter{in}	
int_flag	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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear MDMA interrupt flag */
```

```
mdma_interrupt_flag_clear(MDMA_CH0, MDMA_INT_FLAG_TCF);
```

3.30. 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 [3.30.1](#), the OSPI firmware functions are introduced in chapter [3.30.2](#).

3.30.1. Descriptions of Peripheral registers

OSPI registers are listed in the table shown as below:

Table 3-1068. 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

Registers	Descriptions
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

3.30.2. Descriptions of Peripheral functions

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

Table 3-1069. 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

Function name	Function description
ospi_data_receive	OSPI receive data
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

Table3-1070. 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`

Table3-1071. 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,

Member name	Function description
	OSPI_ALTERNATE_BYTES_2_LINES, OSPI_ALTERNATE_BYTES_4_LINES, OSPI_ALTERNATE_BYTES_8_LINES)
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)
nbddata	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

Table3-1072. 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 3-1073. 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:

Table3-1074. Function ospi_deinit

Function name	ospi_deinit
Function prototype	void ospi_deinit(uint32_t ospi_periph);
Function descriptions	reset the OSPI peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the OSPI peripheral */
ospi_deinit();
```

ospi_struct_init

The description of ospi_struct_init is shown as below:

Table3-1075. Function ospi_struct_init

Function name	ospi_struct_init
Function prototype	void ospi_struct_init(ospi_parameter_struct *ospi_struct);
Function descriptions	initialize the parameters of OSPI struct with default values
Precondition	-
The called functions	-
Input parameter{in}	
ospi_struct	OSPI parameter struct, the structure members can refer to members of the structure Table3-1070. Structure ospi_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1076. Function ospi_init

Function name	ospi_init
Function prototype	void ospi_init(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct);
Function descriptions	initialize OSPI parameter
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
ospi_struct	OSPI parameter struct, the structure members can refer to members of the structure Table3-1070. Structure ospi_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

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;

ospi_init(OSPI0, &ospi_struct);

```

ospi_enable

The description of ospi_enable is shown as below:

Table3-1077. Function ospi_enable

Function name	ospi_enable
Function prototype	void ospi_enable(uint32_t ospi_periph);

Function descriptions	enable OSPI
Precondition	-
The called functions	-
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:

Table3-1078. 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:

Table3-1079. 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}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
dtysel	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
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1080. Function ospi_device_memory_size_config

Function name	ospi_device_memory_size_config
Function prototype	void ospi_device_memory_size_config(uint32_t ospi_periph, uint32_t mesz);
Function descriptions	configure device memory size
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
mesz	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
Output parameter{out}	
-	-

Return value	
-	-

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:

Table3-1081. Function ospi_functional_mode_config

Function name	ospi_functional_mode_config
Function prototype	void ospi_functional_mode_config(uint32_t ospi_periph, uint32_t fmod);
Function descriptions	select functional mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
fmod	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
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1082. Function ospi_status_polling_config

Function name	ospi_status_polling_config
---------------	----------------------------

Function prototype	void ospi_status_polling_config(uint32_t ospi_periph, uint32_t stop, uint32_t mode);
Function descriptions	configure status polling mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
stop	OSPI automatic stop
OSPI_AUTOMATIC_STOP_MATCH	status polling mode stop in match
Input parameter{in}	
mode	OSPI match mode
OSPI_MATCH_MODE_AND	status polling match mode and
OSPI_MATCH_MODE_OR	status polling match mode or
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure status polling mode */
```

```
ospi_status_polling_config(OSPI0, OSPI_AUTOMATIC_STOP_ABORT,
ospi_status_polling_config(OSPI0, OSPI_MATCH_MODE_OR);
```

ospi_status_mask_config

The description of ospi_status_mask_config is shown as below:

Table3-1083. Function ospi_status_mask_config

Function name	ospi_status_mask_config
Function prototype	void ospi_status_mask_config(uint32_t ospi_periph, uint32_t mask);
Function descriptions	configure status mask
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
mask	status mask (0-0xFFFFFFFF)
Output parameter{out}	
-	-

Return value	
-	-

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:

Table3-1084. Function ospi_status_match_config

Function name	ospi_status_match_config
Function prototype	void ospi_status_match_config(uint32_t ospi_periph, uint32_t match);
Function descriptions	configure status match
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
match	status match (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1085. Function ospi_interval_cycle_config

Function name	ospi_interval_cycle_config
Function prototype	void ospi_interval_cycle_config(uint32_t ospi_periph, uint16_t interval);
Function descriptions	configure interval cycle
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	

interval	interval cycle (0-0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1086. Function ospi_fifo_level_config

Function name	ospi_fifo_level_config
Function prototype	void ospi_fifo_level_config(uint32_t ospi_periph, uint32_t ftl)
Function descriptions	configure OSPI fifo threshold level
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
ftl	FIFO threshold level
OSPI_FIFO_THRESHOLD_x	threshold level set to x (x = 1, 2, ..., 31, 32)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1087. Function ospi_chip_select_high_cycle_config

Function name	ospi_chip_select_high_cycle_config
Function prototype	void ospi_chip_select_high_cycle_config(uint32_t ospi_periph, uint32_t cshc);

Function descriptions	configure chip select high cycle
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
cs hc	OSPI chip select high cycle
OSPI_CS_HIGH_TIME _x_CYCLE	chip select high cycle set to x (x = 1, 2, ..., 63, 64)
Output parameter{out}	
-	-
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:

Table3-1088. Function ospi_prescaler_config

Function name	ospi_prescaler_config
Function prototype	void ospi_prescaler_config(uint32_t ospi_periph, uint32_t psc);
Function descriptions	configure OSPI prescaler
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
psc	the scaler factor for generating SCK based on the kernel clock (0-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure OSPI prescaler */
ospi_prescaler_config(OSPI0, 0x08);
```

ospi_dummy_cycles_config

The description of ospi_dummy_cycles_config is shown as below:

Table3-1089. Function ospi_dummy_cycles_config

Function name	ospi_dummy_cycles_config
Function prototype	void ospi_dummy_cycles_config(uint32_t ospi_periph, uint32_t dumyc);
Function descriptions	configure dummy cycles number
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
dumyc	number of dummy cycles
<i>OSPI_DUMYC_CYCLE</i> <i>S_x</i>	dummy cycles set to x (x = 0, 1, 2, ..., 30, 31)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1090. Function ospi_delay_hold_cycle_config

Function name	ospi_delay_hold_cycle_config
Function prototype	void ospi_delay_hold_cycle_config(uint32_t ospi_periph, uint32_t dehqc);
Function descriptions	delay hold 1/4 cycle
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
dehqc	OSPI delay hold quarter cycle
<i>OSPI_DELAY_HOLD_</i> <i>NONE</i>	OSPI no delay hold cycle
<i>OSPI_DELAY_HOLD_</i> <i>QUARTER_CYCLE</i>	OSPI delay hold 1/4 cycle
Output parameter{out}	

-	-
Return value	
-	-

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:

Table3-1091. Function ospi_sample_shift_config

Function name	ospi_sample_shift_config
Function prototype	void ospi_sample_shift_config(uint32_t ospi_periph, uint32_t ssample);
Function descriptions	configure sample shift
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
ssample	OSPI sample shift
OSPI_SAMPLE_SHIFTING_NONE	OSPI no sample shift
OSPI_SAMPLE_SHIFTING_HALF_CYCLE	OSPI have 1/2 cycle sample shift
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1092. Function ospi_data_length_config

Function name	ospi_data_length_config
Function prototype	void ospi_data_length_config(uint32_t ospi_periph, uint32_t dtlen);
Function descriptions	configure data length

Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx (x=0,1)
Input parameter{in}	
dtlen	data length (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure data length */
```

```
ospi_data_length_config(OSPI0, 0x00005555);
```

ospi_instruction_config

The description of ospi_instruction_config is shown as below:

Table3-1093. Function ospi_instruction_config

Function name	ospi_instruction_config
Function prototype	void ospi_instruction_config(uint32_t ospi_periph, uint32_t imod, uint32_t inssz, uint32_t instruction);
Function descriptions	configure instruction mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx (x=0,1)
Input parameter{in}	
imod	OSPI instruction mode
<i>OSPI_INSTRUCTION_NONE</i>	no instruction mode
<i>OSPI_INSTRUCTION_1_LINE</i>	instruction mode on a single line
<i>OSPI_INSTRUCTION_2_LINES</i>	instruction mode on two lines
<i>OSPI_INSTRUCTION_4_LINES</i>	instruction mode on four lines
<i>OSPI_INSTRUCTION_8_LINES</i>	instruction mode on eight lines
Input parameter{in}	
inssz	OSPI instruction size
<i>OSPI_INSTRUCTION_8_BIT</i>	8-bit instruction

8_BITS	
OSPI_INSTRUCTION_16_BITS	16-bit instruction
OSPI_INSTRUCTION_24_BITS	24-bit instruction
OSPI_INSTRUCTION_32_BITS	32-bit instruction
Input parameter{in}	
instruction	the instruction to be sent (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1094. Function ospi_address_config

Function name	ospi_address_config
Function prototype	void ospi_address_config(uint32_t ospi_periph, uint32_t addrmod, uint32_t addrdrtr, uint32_t addrsh, uint32_t addr);
Function descriptions	configure 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

NES	
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
OSPI_ADDRESS_24_BITS	address size on 24-bit address
OSPI_ADDRESS_32_BITS	address size on 32-bit address
Input parameter{in}	
addr	the address to be sent (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address mode */
```

```
ospi_address_config(OSPI0, OSPI_ADDRESS_8_LINES,
OSPI_ADDRDTR_MODE_DISABLE, OSPI_ADDRESS_32_BITS, 0xFFFFFFFF);
```

ospi_alternate_bytes_config

The description of ospi_alternate_bytes_config is shown as below:

Table3-1095. Function ospi_alternate_bytes_config

Function name	ospi_alternate_bytes_config
Function prototype	void ospi_alternate_bytes_config(uint32_t ospi_periph, uint32_t atlemod, uint32_t abdtr, uint32_t altesz, uint32_t alte)
Function descriptions	configure alternate bytes mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx (x=0,1)
Input parameter{in}	

atlemod	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
Input parameter{in}	
abdtr	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
Input parameter{in}	
altesz	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}	
alte	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:

Table3-1096. Function `ospi_data_config`

Function name	<code>ospi_data_config</code>
Function prototype	<code>void ospi_data_config(uint32_t ospi_periph, uint32_t datamod, uint32_t dadtr);</code>
Function descriptions	configure data mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
datamod	OSPI data mode
<code>OSPI_DATA_NONE</code>	no data mode
<code>OSPI_DATA_1_LINE</code>	data mode on a single line
<code>OSPI_DATA_2_LINES</code>	data mode on two lines
<code>OSPI_DATA_4_LINES</code>	data mode on four lines
<code>OSPI_DATA_8_LINES</code>	data mode on eight lines
Input parameter{in}	
dadtr	OSPI data double transfer rate
<code>OSPI_DADTR_MODE_DISABLE</code>	data double transfer rate mode disable
<code>OSPI_DADTR_MODE_ENABLE</code>	data double transfer rate mode enable
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1097. Function `ospi_data_transmit`

Function name	<code>ospi_data_transmit</code>
Function prototype	<code>void ospi_data_transmit(uint32_t ospi_periph, uint32_t data);</code>
Function descriptions	OSPI transmit data
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx (x=0,1)

Input parameter{in}	
data	transmit data (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* OSPI transmit data */
```

```
ospi_data_transmit(OSPI0, 0x00FF0000);
```

ospi_data_receive

The description of ospi_data_receive is shown as below:

Table3-1098. Function ospi_data_receive

Function name	ospi_data_receive
Function prototype	uint32_t ospi_data_receive(uint32_t ospi_periph);
Function descriptions	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:

Table3-1099. 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:

Table3-1100. 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}	
ospi_periph	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:

Table3-1101. Function ospi_wrap_size_config

Function name	ospi_wrap_size_config
Function prototype	void ospi_wrap_size_config(uint32_t ospi_periph, uint32_t wpsz);
Function descriptions	configure wrap size
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)

Input parameter{in}	
wpsz	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:

Table3-1102. Function `ospi_wrap_instruction_config`

Function name	<code>ospi_wrap_instruction_config</code>
Function prototype	<code>void ospi_wrap_instruction_config(uint32_t ospi_periph, uint32_t imod, uint32_t inssz, uint32_t instruction);</code>
Function descriptions	configure wrap instruction mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
imod	OSPI instruction mode
<i>OSPI_INSTRUCTION_NONE</i>	no instruction mode
<i>OSPI_INSTRUCTION_1_LINE</i>	instruction mode on a single line
<i>OSPI_INSTRUCTION_2_LINES</i>	instruction mode on two lines
<i>OSPI_INSTRUCTION_4_LINES</i>	instruction mode on four lines

<code>OSPI_INSTRUCTION_8_LINES</code>	instruction mode on eight lines
Input parameter{in}	
inssz	OSPI instruction size
<code>OSPI_INSTRUCTION_8_BITS</code>	8-bit instruction
<code>OSPI_INSTRUCTION_16_BITS</code>	16-bit instruction
<code>OSPI_INSTRUCTION_24_BITS</code>	24-bit instruction
<code>OSPI_INSTRUCTION_32_BITS</code>	32-bit instruction
Input parameter{in}	
instruction	the instruction to be sent (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1103. Function `ospi_wrap_address_config`

Function name	<code>ospi_wrap_address_config</code>
Function prototype	<code>void ospi_wrap_address_config (uint32_t ospi_periph, uint32_t addrmod, uint32_t addrdr, uint32_t addrsz, uint32_t addr);</code>
Function descriptions	Configure wrap address mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx (x=0,1)
Input parameter{in}	
addrmod	OSPI address mode
<code>OSPI_ADDRESS_NONE</code>	no address mode
<code>OSPI_ADDRESS_1_LINE</code>	address mode on a single line

<i>OSPI_ADDRESS_2_LINES</i>	address mode on two lines
<i>OSPI_ADDRESS_4_LINES</i>	address mode on four lines
<i>OSPI_ADDRESS_8_LINES</i>	address mode on eight lines
Input parameter{in}	
addrdtr	OSPI address double transfer rate
<i>OSPI_ADDRDTR_MODE_DISABLE</i>	address double transfer rate mode disable
<i>OSPI_ADDRDTR_MODE_ENABLE</i>	address double transfer rate mode enable
Input parameter{in}	
addrsz	OSPI address size
<i>OSPI_ADDRESS_8_BITS</i>	8-bit address
<i>OSPI_ADDRESS_16_BITS</i>	16-bit address
<i>OSPI_ADDRESS_24_BITS</i>	24-bit address
<i>OSPI_ADDRESS_32_BITS</i>	32-bit address
Input parameter{in}	
addr	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:

Table3-1104. Function `ospi_wrap_alternate_bytes_config`

Function name	<code>ospi_wrap_alternate_bytes_config</code>
Function prototype	<code>void ospi_wrap_alternate_bytes_config(uint32_t ospi_periph, uint32_t atlemod, uint32_t abdtr, uint32_t altesz, uint32_t alte)</code>
Function descriptions	configure wrap alternate bytes mode

Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx (x=0,1)
Input parameter{in}	
atlemod	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
Input parameter{in}	
abdtr	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
Input parameter{in}	
altesz	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}	
alte	The alternate bytes to be sent (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1105. Function ospi_wrap_data_config

Function name	ospi_wrap_data_config
Function prototype	void ospi_wrap_data_config(uint32_t ospi_periph, uint32_t datamod, uint32_t dadtr);
Function descriptions	configure wrap data mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
datamod	OSPI data mode
OSPI_DATA_NONE	no data mode
OSPI_DATA_1_LINE	data mode on a single line
OSPI_DATA_2_LINES	data mode on two lines
OSPI_DATA_4_LINES	data mode on four lines
OSPI_DATA_8_LINES	data mode on eight lines
Input parameter{in}	
dadtr	OSPI data double transfer rate
OSPI_DADTR_MODE_DISABLE	data double transfer rate mode disable
OSPI_DADTR_MODE_ENABLE	data double transfer rate mode enable
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1106. Function ospi_wrap_dummy_cycles_config

Function name	ospi_wrap_dummy_cycles_config
Function prototype	void ospi_wrap_dummy_cycles_config(uint32_t ospi_periph, uint32_t

	dumyc);
Function descriptions	configure wrap dummy cycles number
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
dumyc	dummy cycles number (0-0x1F)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1107. Function ospi_wrap_delay_hold_cycle_config

Function name	ospi_wrap_delay_hold_cycle_config
Function prototype	void ospi_wrap_delay_hold_cycle_config(uint32_t ospi_periph, uint32_t dehqc);
Function descriptions	delay hold 1/4 cycle in wrap
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
dehqc	OSPI delay hold quarter cycle
OSPI_DELAY_HOLD_NONE	OSPI no delay hold cycle
OSPI_DELAY_HOLD_QUARTER_CYCLE	OSPI delay hold 1/4 cycle
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1108. Function `ospi_wrap_sample_shift_config`

Function name	<code>ospi_wrap_sample_shift_config</code>
Function prototype	<code>void ospi_wrap_sample_shift_config(uint32_t ospi_periph, uint32_t ssample);</code>
Function descriptions	configure sample shift in wrap
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
ssample	OSPI sample shift
<code>OSPI_SAMPLE_SHIFTING_NONE</code>	OSPI no sample shift
<code>OSPI_SAMPLE_SHIFTING_HALF_CYCLE</code>	OSPI have 1/2 cycle sample shift
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1109. Function `ospi_write_instruction_config`

Function name	<code>ospi_write_instruction_config</code>
Function prototype	<code>void ospi_write_instruction_config(uint32_t ospi_periph, uint32_t imod, uint32_t inssz, uint32_t instruction);</code>
Function descriptions	configure write instruction mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	

imod	OSPI instruction mode
<i>OSPI_INSTRUCTION_NONE</i>	no instruction mode
<i>OSPI_INSTRUCTION_1_LINE</i>	instruction mode on a single line
<i>OSPI_INSTRUCTION_2_LINES</i>	instruction mode on two lines
<i>OSPI_INSTRUCTION_4_LINES</i>	instruction mode on four lines
<i>OSPI_INSTRUCTION_8_LINES</i>	instruction mode on eight lines
Input parameter{in}	
inssz	OSPI instruction size
<i>OSPI_INSTRUCTION_8_BITS</i>	instruction size on 8-bit address
<i>OSPI_INSTRUCTION_16_BITS</i>	instruction size on 16-bit address
<i>OSPI_INSTRUCTION_24_BITS</i>	instruction size on 24-bit address
<i>OSPI_INSTRUCTION_32_BITS</i>	instruction size on 32-bit address
Input parameter{in}	
instruction	the instruction to be sent (0-0xFFFFFFFF)
Output parameter{out}	
-	-
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:

Table3-1110. Function `ospi_write_address_config`

Function name	<code>ospi_write_address_config</code>
Function prototype	<code>void ospi_write_address_config(uint32_t ospi_periph, uint32_t addrmod, uint32_t addrdrtr, uint32_t addrsz, uint32_t addr);</code>
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
OSPI_ADDRESS_24_BITS	address size on 24-bit address
OSPI_ADDRESS_32_BITS	address size on 32-bit address
Input parameter{in}	
addr	the address to be sent (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1111. Function ospi_write_alternate_bytes_config

Function name	ospi_write_alternate_bytes_config
Function prototype	void ospi_write_alternate_bytes_config(uint32_t ospi_periph, uint32_t atlemod, uint32_t abdtr, uint32_t altesz, uint32_t alte);
Function descriptions	configure write alternate bytes mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx (x=0,1)
Input parameter{in}	
atlemod	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}	
abdtr	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}	
altesz	OSPI alternate bytes size
OSPI_ALTERNATE_BYTES_8_BITS	alternate bytes size on 8-bit address
OSPI_ALTERNATE_BYTES_16_BITS	alternate bytes size on 16-bit address
OSPI_ALTERNATE_BYTES_24_BITS	alternate bytes size on 24-bit address
OSPI_ALTERNATE_BYTES_32_BITS	alternate bytes size on 32-bit address
Input parameter{in}	
alte	The alternate bytes to be sent (0-0xFFFFFFFF)

Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1112. Function ospi_write_data_config

Function name	ospi_write_data_config
Function prototype	void ospi_write_data_config(uint32_t ospi_periph, uint32_t datamod, uint32_t dadtr);
Function descriptions	configure write data mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
datamod	OSPI data mode
OSPI_DATA_NONE	no data mode
OSPI_DATA_1_LINE	data mode on a single line
OSPI_DATA_2_LINES	data mode on two lines
OSPI_DATA_4_LINES	data mode on four lines
OSPI_DATA_8_LINES	data mode on eight lines
Input parameter{in}	
dadtr	OSPI data double transfer rate
OSPI_DADTR_MODE_DISABLE	data double transfer rate mode disable
OSPI_DADTR_MODE_ENABLE	data double transfer rate mode enable
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1113. Function `ospi_write_dummy_cycles_config`

Function name	<code>ospi_write_dummy_cycles_config</code>
Function prototype	<code>void ospi_write_dummy_cycles_config(uint32_t ospi_periph, uint32_t dumyc);</code>
Function descriptions	configure write dummy cycles number
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
dumyc	number of dummy cycles
OSPI_DUMYC_CYCLE S_x	dummy cycles set to x (x=0,1,2,...,30,31)
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1114. Function `ospi_write_dummy_cycles_config`

Function name	<code>ospi_write_dummy_cycles_config</code>
Function prototype	<code>void ospi_command_config(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, ospi_regular_cmd_struct *cmd_struct);</code>
Function descriptions	configure OSPI regular command parameter
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
ospi_struct	OSPI parameter struct, the structure members can refer to members of the strcture Table3-1070. Structure <code>ospi_parameter_struct</code>

Input parameter{in}	
cmd_struct	OSPI command struct, the structure members can refer to members of the stricture Table3-1071. Structure ospi_regular_cmd_struct
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1115. Function ospi_transmit

Function name	ospi_transmit
Function prototype	void ospi_transmit(uint32_t ospi_periph, uint8_t *pdata);
Function descriptions	transmit data
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
pdata	pointer to data buffer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* transmit data */
ospi_transmit(OSPI0, pdata);
```

ospi_receive

The description of ospi_receive is shown as below:

Table3-1116. Function ospi_receive

Function name	ospi_receive
Function prototype	void ospi_receive(uint32_t ospi_periph, uint8_t *pdata);
Function descriptions	receive data

Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
pdata	pointer to data buffer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* receive data */
ospi_receive(OSPI0, pdata);
```

ospi_autopolling_mode

The description of ospi_autopolling_mode is shown as below:

Table3-1117. Function ospi_autopolling_mode

Function name	ospi_autopolling_mode
Function prototype	void ospi_autopolling_mode(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, ospi_autopolling_struct *autopl_cfg_struct);
Function descriptions	configure the OSPI automatic polling mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
ospi_struct	OSPI parameter struct, the structure members can refer to members of the strcture Table3-1070. Structure ospi_parameter_struct
Input parameter{in}	
ospi_autopolling_struct	OSPI autopolling stuct, the structure members can refer to members of the strcture Table3-1072. Structure ospi_autopolling_struct
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1118. Function ospi_interrupt_enable

Function name	ospi_interrupt_enable
Function prototype	void ospi_interrupt_enable(uint32_t ospi_periph, uint8_t interrupt);
Function descriptions	enable OSPI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
interrupt	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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable OSPI interrupt */
ospi_interrupt_enable(OSPI0, OSPI_INT_TERR);
```

ospi_interrupt_disable

The description of ospi_interrupt_disable is shown as below:

Table3-1119. Function ospi_interrupt_disable

Function name	ospi_interrupt_disable
Function prototype	void ospi_interrupt_disable(uint32_t ospi_periph, uint8_t interrupt);
Function descriptions	disable OSPI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
interrupt	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
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:

Table3-1120. 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:

Table3-1121. Function `ospi_flag_get`

Function name	<code>ospi_flag_get</code>
Function prototype	<code>FlagStatus ospi_flag_get(uint32_t ospi_periph, uint32_t flag);</code>
Function descriptions	get OSPI flag status
Precondition	-
The called functions	-
Input parameter{in}	
<code>ospi_periph</code>	OSPIx(x=0,1)

Input parameter{in}	
flag	OSPI flag status
<i>OSPI_FLAG_TERR</i>	transfer error flag
<i>OSPI_FLAG_TC</i>	transfer complete flag
<i>OSPI_FLAG_FT</i>	fifo threshold flag
<i>OSPI_FLAG_SM</i>	status match flag
<i>OSPI_FLAG_BUSY</i>	busy flag
Output parameter{out}	
FlagStatus	SET or RESET
Return value	
-	-

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:

Table3-1122. Function `ospi_flag_clear`

Function name	<code>ospi_flag_clear</code>
Function prototype	<code>void ospi_flag_clear(uint32_t ospi_periph, uint32_t flag);</code>
Function descriptions	clear OSPI flag status
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
flag	OSPI clear flag status
<i>OSPI_FLAG_TERR</i>	transfer error flag
<i>OSPI_FLAG_TC</i>	transfer complete flag
<i>OSPI_FLAG_SM</i>	status match flag
Output parameter{out}	
-	-
Return value	
-	-

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:

Table3-1123. Function ospi_interrupt_flag_get

Function name	ospi_interrupt_flag_get
Function prototype	FlagStatus ospi_interrupt_flag_get(uint32_t ospi_periph, uint8_t int_flag);
Function descriptions	get OSPI interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
int_flag	OSPI interrupt flag status
<i>OSPI_INT_FLAG_TER</i> <i>R</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
Output parameter{out}	
FlagStatus	SET or RESET
Return value	
-	-

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:

Table3-1124. Function ospi_interrupt_flag_clear

Function name	ospi_interrupt_flag_clear
Function prototype	void ospi_interrupt_flag_clear(uint32_t ospi_periph, uint32_t int_flag);
Function descriptions	clear OSPI interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Input parameter{in}	
int_flag	clear OSPI interrupt flag status
<i>OSPI_INT_FLAG_TER</i> <i>R</i>	transfer error interrupt flag

OSPI_INT_FLAG_TC	transfer complete interrupt flag
OSPI_INT_FLAG_SM	status match interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear OSPI interrupt flag status */
```

```
ospi_interrupt_flag_clear(OSPI0, OSPI_STATC_TERRC);
```

3.31. OSPIM

OSPI supports OSPI pin assignment with full matrix before alternate function map. The OSPIM registers are listed in chapter [3.31.1](#), the OSPIM firmware functions are introduced in chapter [3.31.2](#).

3.31.1. Descriptions of Peripheral registers

OSPI registers are listed in the table shown as below:

Table 3-1125. OSPIM Registers

Registers	Descriptions
OSPIM_PCFG0	OSPIM I/O manager port configuration register 0
OSPIM_PCFG1	OSPIM I/O manager port configuration register 1

3.31.2. Descriptions of Peripheral functions

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

Table 3-1126. OSPIM firmware function

Function name	Function description
ospim_deinit	reset the OSPIM peripheral
ospim_port_sck_config	configure SCK for port
ospim_port_sck_source_select	select source of SCK for port
ospim_port_csn_config	configure CSN for port
ospim_port_csn_source_select	select source of CSN for port
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:

Table3-1127. 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:

Table3-1128. Function ospim_port_sck_config

Function name	ospim_port_sck_config
Function prototype	void ospim_port_sck_config(uint8_t port, uint32_t sckconfig);
Function descriptions	configure SCK for port
Precondition	-
The called functions	-
Input parameter{in}	
port	number of port
OSPIM_PORT0	port 0
OSPIM_PORT1	port 1
Input parameter{in}	
sckconfig	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_PORT0, OSPIM_PORT_SCK_ENABLE);
```

ospim_port_sck_source_select

The description of ospim_port_sck_source_select is shown as below:

Table3-1129. Function ospim_port_sck_source_select

Function name	ospim_port_sck_source_select
Function prototype	void ospim_port_sck_source_select(uint8_t port, uint32_t sck_source);
Function descriptions	select source of SCK for port
Precondition	-
The called functions	-
Input parameter{in}	
port	number of port
OSPIM_PORT0	port 0
OSPIM_PORT1	port 1
Input parameter{in}	
sck_source	source of SCK
OSPIM_SCK_SOURCE_OSPI0_SCK	the source of SCK is OSPI0_SCK
OSPIM_SCK_SOURCE_OSPI1_SCK	the source of SCK is OSPI1_SCK
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select source of SCK for port */
```

```
ospim_port_sck_source_select(OSPIM_PORT0, OSPIM_SCK_SOURCE_OSPI0_SCK);
```

ospim_port_csn_config

The description of ospim_port_csn_config is shown as below:

Table3-1130. Function ospim_port_csn_config

Function name	ospim_port_csn_config
Function prototype	void ospim_port_csn_config(uint8_t ospim_port, uint32_t csconfig);
Function descriptions	configurate CSN for port

Precondition	-
The called functions	-
Input parameter{in}	
port	number of port
<i>OSPIM_PORT0</i>	port 0
<i>OSPIM_PORT1</i>	port 1
Input parameter{in}	
csnconfig	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_PORT0, OSPIM_PORT_CSN_ENABLE);
```

ospim_port_csn_source_select

The description of ospim_port_csn_source_select is shown as below:

Table3-1131. Function ospim_port_csn_source_select

Function name	ospim_port_csn_source_select
Function prototype	void ospim_port_csn_source_select(uint8_t port, uint32_t csn_source);
Function descriptions	select source of CSN for port
Precondition	-
The called functions	-
Input parameter{in}	
port	number of port
<i>OSPIM_PORT0</i>	port 0
<i>OSPIM_PORT1</i>	port 1
Input parameter{in}	
csn_source	source of CSN
<i>OSPIM_CSN_SOURCE_OSPI0_CSN</i>	the source of CSN is OSPI0_CSN
<i>OSPIM_CSN_SOURCE_OSPI1_CSN</i>	the source of CSN is OSPI1_CSN
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* select source of CSN for port */
```

```
ospim_port_csn_source_select(OSPIM_PORT0, OSPIM_CSN_SOURCE_OSPI0_CSN);
```

ospim_port_io3_0_config

The description of ospim_port_io3_0_config is shown as below:

Table3-1132. Function ospim_port_io3_0_config

Function name	ospim_port_io3_0_config
Function prototype	void ospim_port_io3_0_config(uint8_t port, uint32_t ioconfig);
Function descriptions	configure IO[3:0] for port
Precondition	-
The called functions	-
Input parameter{in}	
port	number of port
OSPIM_PORT0	port 0
OSPIM_PORT1	port 1
Input parameter{in}	
ioconfig	enable or disable IO[3:0]
OSPIM_IO_LOW_DISABLE	disable IO[3:0]
OSPIM_IO_LOW_ENABLE	enable IO[3:0]
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure IO[3:0] for port */
```

```
ospim_port_io3_0_config(OSPIM_PORT0, OSPIM_IO_LOW_ENABLE);
```

ospim_port_io3_0_source_select

The description of ospim_port_io3_0_source_select is shown as below:

Table3-1133. Function ospim_port_io3_0_source_select

Function name	ospim_port_io3_0_source_select
Function prototype	void ospim_port_io3_0_source_select(uint8_t port, uint32_t io_source);

Function descriptions	select source of IO[3:0] for port
Precondition	-
The called functions	-
Input parameter{in}	
port	number of port
<i>OSPIM_PORT0</i>	port 0
<i>OSPIM_PORT1</i>	port 1
Input parameter{in}	
csn_source	source of IO[3:0]
<i>OSPIM_SRCPLIO_OS</i> <i>PI0_IO_LOW</i>	select OSPI0_IO[3:0]
<i>OSPIM_SRCPLIO_OS</i> <i>PI0_IO_HIGH</i>	select OSPI0_IO[7:4]
<i>OSPIM_SRCPLIO_OS</i> <i>PI1_IO_LOW</i>	select OSPI1_IO[3:0]
<i>OSPIM_SRCPLIO_OS</i> <i>PI1_IO_HIGH</i>	select OSPI1_IO[7:4]
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select source of IO[3:0] for port */
```

```
ospim_port_io3_0_source_select(OSPIM_PORT0, OSPIM_SRCPLIO_OSPI0_IO_LOW);
```

ospim_port_io7_4_config

The description of ospim_port_io7_4_config is shown as below:

Table3-1134. Function ospim_port_io7_4_config

Function name	ospim_port_io7_4_config
Function prototype	void ospim_port_io7_4_config(uint8_t port, uint32_t ioconfig);
Function descriptions	configure IO[7:4] for port
Precondition	-
The called functions	-
Input parameter{in}	
port	number of port
<i>OSPIM_PORT0</i>	port 0
<i>OSPIM_PORT1</i>	port 1
Input parameter{in}	
ioconfig	enable or disable IO[7:4]
<i>OSPIM_IO_HIGH_DIS</i>	disable IO[7:4]

<i>ABLE</i>	
<i>OSPIM_IO_HIGH_ENABLE</i>	enable IO[7:4]
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configurate IO[7:4] for port */
```

```
ospim_port_io7_4_config(OSPIM_PORT0, OSPIM_IO_HIGH_ENABLE);
```

ospim_port_io7_4_source_select

The description of ospim_port_io7_4_source_select is shown as below:

Table3-1135. Function ospim_port_io7_4_source_select

Function name	ospim_port_io7_4_source_select
Function prototype	void ospim_port_io7_4_source_select(uint8_t port, uint32_t io_source);
Function descriptions	select source of IO[7:4] for port
Precondition	-
The called functions	-
Input parameter{in}	
port	number of port
<i>OSPIM_PORT0</i>	port 0
<i>OSPIM_PORT1</i>	port 1
Input parameter{in}	
csn_source	source of IO[3:0]
<i>OSPIM_SRCPHIO_OSPI0_IO_LOW</i>	select OSPI0_IO[3:0]
<i>OSPIM_SRCPHIO_OSPI0_IO_HIGH</i>	select OSPI0_IO[7:4]
<i>OSPIM_SRCPHIO_OSPI1_IO_LOW</i>	select OSPI1_IO[3:0]
<i>OSPIM_SRCPHIO_OSPI1_IO_HIGH</i>	select OSPI1_IO[7:4]
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select source of IO[7:4] for port */
```

```
ospim_port_io7_4_source_select(OSPIM_PORT0, OSPIM_SRCPIO_OSPI0_IO_LOW);
```

3.32. MISC

MISC is a software package that provide the interfaces for NVIC and SysTick. The NVIC and SysTick registers are listed in chapter [3.32.1](#), the MISC firmware functions are introduced in chapter [3.32.2](#).

3.32.1. Descriptions of Peripheral registers

Table 3-1136. NVIC Registers

Registers	Descriptions
ISER ⁽¹⁾	Interrupt Set Enable Register
ICER ⁽¹⁾	Interrupt Clear Enable Register
ISPR ⁽¹⁾	Interrupt Set Pending Register
ICPR ⁽¹⁾	Interrupt Clear Pending Register
IABR ⁽¹⁾	Interrupt Active bit Register
IP ⁽¹⁾	Interrupt Priority Register
STIR ⁽¹⁾	Software Trigger Interrupt Register
CPUID ⁽²⁾	CPUID Base Register
ICSR ⁽²⁾	Interrupt Control and State Register
VTOR ⁽²⁾	Vector Table Offset Register
AIRCR ⁽²⁾	Application Interrupt and Reset Control Register
SCR ⁽²⁾	System Control Register
CCR ⁽²⁾	Configuration Control Register
SHPR ⁽²⁾	System Handlers Priority Registers
SHCSR ⁽²⁾	System Handler Control and State Register
CFSR ⁽²⁾	Configurable Fault Status Register
HFSR ⁽²⁾	HardFault Status Register
DFSR ⁽²⁾	Debug Fault Status Register
MMFAR ⁽²⁾	MemManage Fault Address Register
BFAR ⁽²⁾	BusFault Address Register
AFSR ⁽²⁾	Auxiliary Fault Status Register
ID_PFR ⁽²⁾	Processor Feature Register
ID_DFR ⁽²⁾	Debug Feature Register
ID_AFR ⁽²⁾	Auxiliary Feature Register
ID_MFR ⁽²⁾	Memory Model Feature Register
ID_ISAR ⁽²⁾	Instruction Set Attributes Register
CLIDR ⁽²⁾	Cache Level ID register
CTR ⁽²⁾	Cache Type register
CCSIDR ⁽²⁾	Cache Size ID Register
CSSELR ⁽²⁾	Cache Size Selection Register

Registers	Descriptions
CPACR ⁽²⁾	Coprocessor Access Control Register
STIR ⁽²⁾	Software Triggered Interrupt Register
MVFR0 ⁽²⁾	Media and VFP Feature Register 0
MVFR1 ⁽²⁾	Media and VFP Feature Register 1
MVFR2 ⁽²⁾	Media and VFP Feature Register 2
ICIALLU ⁽²⁾	I-Cache Invalidate All to PoU
ICIMVAU ⁽²⁾	I-Cache Invalidate by MVA to PoU
DCIMVAC ⁽²⁾	D-Cache Invalidate by MVA to PoC
DCISW ⁽²⁾	D-Cache Invalidate by Set-way
DCCMVAU ⁽²⁾	D-Cache Clean by MVA to PoU
DCCMVAC ⁽²⁾	D-Cache Clean by MVA to PoC
DCCSW ⁽²⁾	D-Cache Clean by Set-way
DCCIMVAC ⁽²⁾	D-Cache Clean and Invalidate by MVA to PoC
DCCISW ⁽²⁾	D-Cache Clean and Invalidate by Set-way
ITCMCR ⁽²⁾	Instruction Tightly-Coupled Memory Control Register
DTCMCR ⁽²⁾	Data Tightly-Coupled Memory Control Registers
AHBPCR ⁽²⁾	AHB Control Register
CACR ⁽²⁾	L1 Cache Control Register
AHBSCR ⁽²⁾	AHB Slave Control Register
ABFSR ⁽²⁾	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 3-1137. SysTick Registers

Registers	Descriptions
CTRL ⁽¹⁾	SysTick Control and Status Register
LOAD ⁽¹⁾	SysTick Reload Value Register
VAL ⁽¹⁾	SysTick Current Value Register
CALIB ⁽¹⁾	SysTick Calibration Register

1. refer to the structure SysTick_Type, is defined in the core_cm7.h file

3.32.2. Descriptions of Peripheral functions

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

Table 3-1138. 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

Function name	Function description
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 3-1139. 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 3-1140. Enum IRQn_Type

Member name	Function description
WWDGT_IRQn	WWDGT interrupt
AVD_PVD_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
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

Member name	Function description
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
SDIO0_IRQn	SDIO0 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
DMA1_Channel2_IRQn	DMA1 Channel 2 global interrupt
DMA1_Channel3_IRQn	DMA1 Channel 3 global interrupt
DMA1_Channel4_IRQn	DMA1 Channel 4 global interrupt

Member name	Function description
ENET0_IRQn	Ethernet0 global interrupt
ENET0_WKUP_IRQn	Ethernet0 wakeup from EXTI 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
DCI_IRQn	DCI global interrupt
CAU_IRQn	CAU global interrupt
HAU_TRNG_IRQn	HAU and 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
SAI0_IRQn	SAI0 global interrupt
TLI_IRQn	TLI global interrupt
TLI_ER_IRQn	TLI error interrupt
IPA_IRQn	IPA global interrupt
SAI1_IRQn	SAI1 global interrupt
QSPI0_IRQn	QSPI0 global interrupt
I2C3_EV_IRQn	I2C3 event interrupt
I2C3_ER_IRQn	I2C3 error interrupt
RSPDIF_IRQn	RSPDIF global 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
SAI2_IRQn	SAI2 global interrupt
TIMER14_IRQn	TIMER14 global interrupt
TIMER15_IRQn	TIMER15 global interrupt
TIMER16_IRQn	TIMER16 global interrupt
MDIO_IRQn	MDIO global interrupt

Member name	Function description
MDMA_IRQn	MDMA global interrupt
SDIO1_IRQn	SDIO1 global interrupt
HWSEM_IRQn	HWSEM 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
RTDEC0_IRQn	RTDEC0 global interrupt
RTDEC1_IRQn	RTDEC1 global interrupt
FAC_IRQn	FAC global interrupt
TMU_IRQn	TMU global interrupt
TIMER22_IRQn	TIMER22 global interrupt
TIMER23_IRQn	TIMER23 global interrupt
TIMER30_IRQn	TIMER30 global interrupt
TIMER31_IRQn	TIMER31 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
ENET1_IRQn	Ethernet1 global interrupt
ENET1_WKUP_IRQn	Ethernet1 wakeup from EXTI 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

Member name	Function description
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
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
TIMER30_DEC_IRQn	TIMER30 DEC interrupt
TIMER31_DEC_IRQn	TIMER31 DEC interrupt

nvic_priority_group_set

The description of nvic_priority_group_set is shown as below:

Table 3-1141. Function nvic_priority_group_set

Function name	nvic_priority_group_set
Function prototype	void nvic_priority_group_set(uint32_t nvic_prigroup);
Function descriptions	set the priority group
Precondition	-
The called functions	-
Input parameter{in}	

nvic_prigroup	priority group
<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 3-1142. Function nvic_irq_enable

Function name	nvic_irq_enable
Function prototype	void nvic_irq_enable(uint8_t nvic_irq, uint8_t nvic_irq_pre_priority, uint8_t nvic_irq_sub_priority);
Function descriptions	enable NVIC interrupt request
Precondition	-
The called functions	nvic_priority_group_set
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to Table 3-1140. Enum IRQn_Type
Input parameter{in}	
nvic_irq_pre_priority	the pre-emption priority needed to set
Input parameter{in}	
nvic_irq_sub_priority	the subpriority needed to set
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable window watchDog timer interrupt, pre-priority is 1, sub-priority is 1 */
```

```
nvic_irq_enable(WWDGT_IRQn, 1, 1);
```

nvic_irq_disable

The description of nvic_irq_disable is shown as below:

Table 3-1143. Function nvic_irq_disable

Function name	nvic_irq_disable
Function prototype	void nvic_irq_disable(uint8_t nvic_irq);
Function descriptions	disable NVIC interrupt request
Precondition	-
The called functions	-
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to enum Table 3-1140. Enum IRQn_Type
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 3-1144. Function nvic_vector_table_set

Function name	nvic_vector_table_set
Function prototype	void nvic_vector_table_set(uint32_t nvic_vect_tab, uint32_t offset);
Function descriptions	set the NVIC vector table base address
Precondition	-
The called functions	-
Input parameter{in}	
nvic_vect_tab	the RAM or FLASH base address
<i>NVIC_VECTTAB_RAM</i>	RAM base address
<i>NVIC_VECTTAB_FLASH</i>	Flash base address
Input parameter{in}	
offset	vector table offset
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set vector table address = NVIC_VECTTAB_FLASH + 0x200 */
nvic_vector_table_set(NVIC_VECTTAB_FLASH, 0x200);
```

system_lowpower_set

The description of system_lowpower_set is shown as below:

Table 3-1145. Function system_lowpower_set

Function name	system_lowpower_set
Function prototype	void system_lowpower_set(uint8_t lowpower_mode);
Function descriptions	set the state of the low power mode
Precondition	-
The called functions	-
Input parameter{in}	
lowpower_mode	the low power mode state
SCB_LPM_SLEEP_EXIT_ISR	if chose this para, the system always enter low power mode by exiting from ISR
SCB_LPM_DEEPSLEEP	if chose this para, the system will enter the DEEPSLEEP mode
SCB_LPM_WAKE_BY_ALL_INT	if chose this para, the lowpower mode can be woke up by all the enable and disable interrupts
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the system always enter low power mode by exiting from ISR */
system_lowpower_set(SCB_LPM_SLEEP_EXIT_ISR);
```

system_lowpower_reset

The description of system_lowpower_reset is shown as below:

Table 3-1146. Function system_lowpower_reset

Function name	system_lowpower_reset
Function prototype	void system_lowpower_reset(uint8_t lowpower_mode);
Function descriptions	reset the state of the low power mode
Precondition	-
The called functions	-
Input parameter{in}	
lowpower_mode	the low power mode state
SCB_LPM_SLEEP_EXIT_ISR	if chose this para, the system will exit low power mode by exiting from ISR

<i>T_ISR</i>	
<i>SCB_LPM_DEEPSLEEP</i>	if chose this para, the system will enter the SLEEP mode
<i>P</i>	
<i>SCB_LPM_WAKE_BY_</i>	if chose this para, the lowpower mode only can be woke up by the enable
<i>ALL_INT</i>	interrupts
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the system will exit low power mode by exiting from ISR */
system_lowpower_reset(SCB_LPM_SLEEP_EXIT_ISR);
```

systick_clksource_set

The description of systick_clksource_set is shown as below:

Table 3-1147. Function systick_clksource_set

Function name	systick_clksource_set
Function prototype	void systick_clksource_set(uint32_t systick_clksource);
Function descriptions	set the systick clock source
Precondition	-
The called functions	-
Input parameter{in}	
systick_clksource	the systick clock source needed to choose
<i>SYSTICK_CLKSOURC</i>	systick clock source is from CK_SYS
<i>E_CKSYS</i>	
<i>SYSTICK_CLKSOURC</i>	systick clock source is from CK_SYS / 2
<i>E_CKSYS_DIV2</i>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* systick clock source is CK_SYS / 2 */
systick_clksource_set(SYSTICK_CLKSOURCE_CKSYS_DIV2);
```

mpu_region_struct_para_init

The description of mpu_region_struct_para_init is shown as below:

Table 3-1148. Function mpu_region_struct_para_init

Function name	mpu_region_struct_para_init
----------------------	-----------------------------

Function prototype	void mpu_region_struct_para_init(mpu_region_init_struct *mpu_init_struct);
Function descriptions	initialize mpu_region_init_struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
mpu_init_struct	MPU initialization structure, refer to Table 3-1139. Structure mpu_region_init_struct.
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 3-1149. Function mpu_region_config

Function name	mpu_region_config
Function prototype	void mpu_region_config(mpu_region_init_struct *mpu_init_struct);
Function descriptions	configure the MPU region
Precondition	-
The called functions	-
Input parameter{in}	
mpu_init_struct	MPU initialization structure, refer to Table 3-1139. Structure mpu_region_init_struct.
Output parameter{out}	
-	-
Return value	
-	-

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 3-1150. Function mpu_region_enable

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

Example:

```
/* enable MPU region */
```

```
mpu_region_enable();
```

3.33. 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 [3.33.1](#), the PMU firmware functions are introduced in chapter [3.33.2](#).

3.33.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

Table 3-1151. 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

Registers	Descriptions
PMU_PAR	PMU parameter register

3.33.2. Descriptions of Peripheral functions

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

Table 3-1152. PMU firmware function

Function name	Function description
pmu_deinit	reset PMU registers
pmu_lvd_select	select low voltage detector threshold
pmu_lvd_enable	enable PMU lvd
pmu_lvd_disable	disable PMU lvd
pmu_avd_select	select analog voltage detector threshold
pmu_avd_enable	enable PMU analog voltage detector
pmu_avd_disable	disable PMU analog voltage detector
pmu_cvd_enable	enable PMU V _{0.9V} core voltage detector
pmu_cvd_disable	disable PMU V _{0.9V} core voltage detector
pmu_ldo_output_select	control the V _{0.9V} core voltage level
pmu_sldo_output_select	Deep-sleep mode V _{0.9V} core voltage select
pmu_vbat_charging_select	PMU V _{BAT} battery charging resistor selection
pmu_vbat_charging_enable	enable V _{BAT} battery charging
pmu_vbat_charging_disable	disable V _{BAT} battery charging
pmu_vbat_temp_monitor_enable	enable V _{BAT} and temperature monitoring
pmu_vbat_temp_monitor_disable	disable V _{BAT} and temperature monitoring
pmu_usb_regulator_enable	enable USB regulator
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 3-1153. Function pmu_deinit

Function name	pmu_deinit
Function prototype	void pmu_deinit(void);
Function descriptions	reset PMU registers
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset PMU register */
```

```
pmu_deinit();
```

pmu_lvd_select

The description of pmu_lvd_select is shown as below:

Table 3-1154. Function pmu_lvd_select

Function name	pmu_lvd_select
Function prototype	void pmu_lvd_select(uint32_t lvd_t_n);
Function descriptions	select low voltage detector threshold
Precondition	-
The called functions	-
Input parameter{in}	
lvd_t_n	voltage threshold value
<i>PMU_LVDT_0</i>	voltage threshold is 2.1V
<i>PMU_LVDT_1</i>	voltage threshold is 2.3V
<i>PMU_LVDT_2</i>	voltage threshold is 2.4V
<i>PMU_LVDT_3</i>	voltage threshold is 2.6V
<i>PMU_LVDT_4</i>	voltage threshold is 2.7V
<i>PMU_LVDT_5</i>	voltage threshold is 2.9V
<i>PMU_LVDT_6</i>	voltage threshold is 3.0V
<i>PMU_LVDT_7</i>	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 3-1155. 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 3-1156. Function pmu_lvd_disable

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

Example:

```
/* disable PMU lvd */

pmu_lvd_disable();
```

pmu_avd_select

The description of pmu_avd_select is shown as below:

Table 3-1157. Function pmu_avd_select

Function name	pmu_avd_select
Function prototype	void pmu_avd_select(uint32_t avdt_n);
Function descriptions	select analog voltage detector threshold
Precondition	-
The called functions	-
Input parameter{in}	
avdt_n	select analog voltage detector threshold
PMU_VAVDVC_0	voltage threshold of analog voltage detector is 1.7V
PMU_VAVDVC_1	voltage threshold of analog voltage detector is 2.1V
PMU_VAVDVC_2	voltage threshold of analog voltage detector is 2.5V
PMU_VAVDVC_3	voltage threshold of analog voltage detector is 2.8V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select analog voltage detector threshold 2.8V */

pmu_avd_select(PMU_VAVDVC_3);
```

pmu_avd_enable

The description of pmu_avd_enable is shown as below:

Table 3-1158. Function pmu_avd_enable

Function name	pmu_avd_enable
Function prototype	void pmu_avd_enable(void);
Function descriptions	enable PMU analog voltage detector
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* enable PMU analog voltage detector */
```

```
pmu_avd_enable();
```

pmu_avd_disable

The description of pmu_avd_disable is shown as below:

Table 3-1159. Function pmu_avd_disable

Function name	pmu_avd_disable
Function prototype	void pmu_avd_disable(void);
Function descriptions	disable PMU analog voltage detector
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable PMU analog voltage detector */
```

```
pmu_avd_disable();
```

pmu_cvd_enable

The description of pmu_cvd_enable is shown as below:

Table 3-1160. Function pmu_cvd_enable

Function name	pmu_cvd_enable
Function prototype	void pmu_cvd_enable(void);
Function descriptions	enable PMU V _{0.9V} voltage detector
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable PMU V0.9V voltage detector */
```

```
pmu_cvd_enable();
```

pmu_cvd_disable

The description of pmu_cvd_disable is shown as below:

Table 3-1161. Function pmu_cvd_disable

Function name	pmu_cvd_disable
Function prototype	void pmu_cvd_disable(void);
Function descriptions	disable PMU V _{0.9V} core voltage detector
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1162. 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 V _{0.9V} core voltage level
Precondition	-
The called functions	-
Input parameter{in}	
ldo_n	Select LDO output voltage
<i>PMU_LDOVS_0</i>	LDO output voltage 0.8V mode
<i>PMU_LDOVS_1</i>	LDO output voltage 0.85V mode
<i>PMU_LDOVS_2</i>	LDO output voltage 0.9V mode
<i>PMU_LDOVS_3</i>	LDO output voltage 0.95V mode

<i>PMU_LDOVS_4</i>	LDO output voltage 0.975V mode
<i>PMU_LDOVS_5</i>	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 3-1163. 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 $V_{0.9V}$ core voltage select
Precondition	-
The called functions	-
Input parameter{in}	
slido_n	select Stop mode voltage
<i>PMU_SLDOVS_0</i>	SLDOVS scale 0.6V
<i>PMU_SLDOVS_1</i>	SLDOVS scale 0.7V
<i>PMU_SLDOVS_2</i>	SLDOVS scale 0.8V
<i>PMU_SLDOVS_3</i>	SLDOVS scale 0.9V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select Deep-sleep mode voltage 0.9V */
```

```
pmu_slido_output_select(PMU_SLDOVS_3);
```

pmu_vbat_charging_select

The description of pmu_vbat_charging_select is shown as below:

Table 3-1164. Function pmu_vbat_charging_select

Function name	pmu_vbat_charging_select
Function prototype	void pmu_vbat_charging_select(uint32_t resistor);

Function descriptions	PMU V _{BAT} battery charging resistor selection
Precondition	-
The called functions	-
Input parameter{in}	
resistor	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1165. Function pmu_vbat_charging_enable

Function name	pmu_vbat_charging_enable
Function prototype	void pmu_vbat_charging_enable(void);
Function descriptions	enable V _{BAT} battery charging
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1166. Function pmu_vbat_charging_disable

Function name	pmu_vbat_charging_disable
----------------------	---------------------------

Function prototype	void pmu_vbat_charging_disable(void);
Function descriptions	disable V _{BAT} battery charging
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1167. Function pmu_vbat_temp_monitor_enable

Function name	pmu_vbat_temp_monitor_enable
Function prototype	void pmu_vbat_temp_monitor_enable(void);
Function descriptions	enable V _{BAT} and temperature monitoring
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1168. Function pmu_vbat_temp_monitor_disable

Function name	pmu_vbat_temp_monitor_disable
Function prototype	void pmu_vbat_temp_monitor_disable(void);

Function descriptions	disable V _{BAT} and temperature monitoring
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1169. Function pmu_usb_regulator_enable

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

Example:

```
/* enable USB regulator */
pmu_usb_regulator_enable();
```

pmu_usb_regulator_disable

The description of pmu_usb_regulator_disable is shown as below:

Table 3-1170. Function pmu_usb_regulator_disable

Function name	pmu_usb_regulator_disable
Function prototype	void pmu_usb_regulator_disable(void);
Function descriptions	disable USB regulator

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

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 3-1171. Function pmu_usb_voltage_detector_enable

Function name	pmu_usb_voltage_detector_enable
Function prototype	void pmu_usb_voltage_detector_enable(void);
Function descriptions	enable V _{DD33USB} voltage level detector
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1172. Function pmu_usb_voltage_detector_disable

Function name	pmu_usb_voltage_detector_disable
Function prototype	void pmu_usb_voltage_detector_disable(void);
Function descriptions	disable V _{DD33USB} voltage level detector
Precondition	-

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

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 3-1173. Function pmu_smpps_ldo_supply_config

Function name	pmu_smpps_ldo_supply_config
Function prototype	void pmu_smpps_ldo_supply_config(uint32_t smppsmode);
Function descriptions	power supply configurations
Precondition	-
The called functions	-
Input parameter{in}	
smppsmode	power supply mode
PMU_LDO_SUPPLY	V _{0.9V} domains are supplied from the LDO
PMU_DIRECT_SMPS_SUPPLY	V _{0.9V} domains are supplied from the SMPS only
PMU_BYPASS	The SMPS disabled and the LDO Bypass. The V _{0.9V} domains are supplied from an external source
Output parameter{out}	
-	-
Return value	
-	-

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 3-1174. Function pmu_to_sleepmode

Function name	pmu_to_sleepmode
Function prototype	void pmu_to_sleepmode(uint8_t sleepmodecmd);
Function descriptions	enter sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
sleepmodecmd	command to enter sleep mode
<i>WFI_CMD</i>	use WFI command
<i>WFE_CMD</i>	use WFE command
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PMU work at sleep mode */
pmu_to_sleepmode(WFI_CMD);
```

pmu_to_deepsleepmode

The description of pmu_to_deepsleepmode is shown as below:

Table 3-1175. Function pmu_to_deepsleepmode

Function name	pmu_to_deepsleepmode
Function prototype	void pmu_to_deepsleepmode(uint8_t deepsleepmodecmd);
Function descriptions	enter deepsleep mode
Precondition	-
The called functions	-
Input parameter{in}	
deepsleepmodecmd	command to enter deepsleep mode
<i>WFI_CMD</i>	use WFI command
<i>WFE_CMD</i>	use WFE command
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PMU work in deepsleep mode */
pmu_to_deepsleepmode(WFI_CMD);
```

pmu_to_standbymode

The description of pmu_to_standbymode is shown as below:

Table 3-1176. Function pmu_to_standbymode

Function name	pmu_to_standbymode
Function prototype	void pmu_to_standbymode(void);
Function descriptions	enter standby mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PMU work at standby mode */
pmu_to_standby();
```

pmu_wakeup_pin_enable

The description of pmu_wakeup_pin_enable is shown as below:

Table 3-1177. Function pmu_wakeup_pin_enable

Function name	pmu_wakeup_pin_enable
Function prototype	void pmu_wakeup_pin_enable(uint32_t wakeup_pin);
Function descriptions	enable PMU wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_pin	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)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable wakeup pin0 */
```

```
pmu_wakeup_pin_enable(PMU_WAKEUP_PIN0);
```

pmu_wakeup_pin_disable

The description of pmu_wakeup_pin_disable is shown as below:

Table 3-1178. Function pmu_wakeup_pin_disable

Function name	pmu_wakeup_pin_disable
Function prototype	void pmu_wakeup_pin_disable(uint32_t wakeup_pin);
Function descriptions	disable PMU wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_pin	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)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable wakeup pin0 */
pmu_wakeup_pin_disable(PMU_WAKEUP_PIN0);
```

pmu_backup_write_enable

The description of pmu_backup_write_enable is shown as below:

Table 3-1179. Function pmu_backup_write_enable

Function name	pmu_backup_write_enable
Function prototype	void pmu_backup_write_enable(void);
Function descriptions	enable backup domain write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:


```
/* enable backup domain write */
```

```
pmu_backup_write_enable();
```

pmu_backup_write_disable

The description of pmu_backup_write_disable is shown as below:

Table 3-1180. Function pmu_backup_write_disable

Function name	pmu_backup_write_disable
Function prototype	void pmu_backup_write_disable(void);
Function descriptions	disable backup domain write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable backup domain write */
```

```
pmu_backup_write_disable();
```

pmu_backup_voltage_stabilizer_enable

The description of pmu_backup_voltage_stabilizer_enable is shown as below:

Table 3-1181. Function pmu_backup_voltage_stabilizer_enable

Function name	pmu_backup_voltage_stabilizer_enable
Function prototype	void pmu_backup_voltage_stabilizer_enable(void);
Function descriptions	enable backup voltage stabilizer
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1182. Function pmu_backup_voltage_stabilizer_disable

Function name	pmu_backup_voltage_stabilizer_disable
Function prototype	void pmu_backup_voltage_stabilizer_disable(void);
Function descriptions	disable backup voltage stabilizer
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1183. Function pmu_enter_deepsleep_wait_time_config

Function name	pmu_enter_deepsleep_wait_time_config
Function prototype	void pmu_enter_deepsleep_wait_time_config (uint32_t wait_time);
Function descriptions	configure IRC counter before enter Deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
wait_time	IRC counter before enter Deep-sleep mode(0x00~0x1F)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1184. Function pmu_exit_deepsleep_wait_time_config

Function name	pmu_exit_deepsleep_wait_time_config
Function prototype	void pmu_exit_deepsleep_wait_time_config (uint32_t wait_time);
Function descriptions	IRC counter before exit Deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
wait_time	IRC counter before exit Deep-sleep mode(0x00~0xFF)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1185. Function pmu_flag_get

Function name	pmu_flag_get
Function prototype	FlagStatus pmu_flag_get(uint32_t flag);
Function descriptions	get flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag
<i>PMU_FLAG_WAKEUP</i>	wakeup flag
<i>PMU_FLAG_STANDBY</i>	standby flag
<i>PMU_FLAG_LVDF</i>	low voltage detector status flag
<i>PMU_FLAG_VAVDF</i>	V _{DDA} analog voltage detector voltage output on V _{DDA} flag
<i>PMU_FLAG_VOVD</i>	peripheral voltage on V _{DDA} detector flag
<i>PMU_FLAG_VBATLF</i>	V _{BAT} level monitoring versus low threshold
<i>PMU_FLAG_VBATHF</i>	V _{BAT} 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.
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

```
FlagStatus status;
```

```
status = pmu_flag_get(PMU_FLAG_WAKEUP);
```

pmu_flag_clear

The description of pmu_flag_clear is shown as below:

Table 3-1186. Function pmu_flag_clear

Function name	pmu_flag_clear
Function prototype	void pmu_flag_clear(uint32_t flag);
Function descriptions	clear flag bit
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag
<i>PMU_FLAG_WAKEUP</i>	wakeup flag
<i>PMU_FLAG_STANDBY</i>	standby flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear flag bit */
```

```
pmu_flag_clear(PMU_FLAG_STANDBY);
```

3.34. 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 [3.34.1](#), the RAMECCMU firmware functions are introduced in chapter [3.34.2](#).

3.34.1. Descriptions of Peripheral registers

Table 3-1187. RAMECCMU Registers

Registers	Descriptions
RAMECCMU_INT	RAMECCMU global interrupt register
RAMECCMU_MxCTL	RAMECCMU monitor x control register
RAMECCMU_MxSTATUS	RAMECCMU monitor x status register
RAMECCMU_MxFADDR	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_MxFECCCODE	RAMECCMU monitor x failing ECC error code register

3.34.2. Descriptions of Peripheral functions

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

Table 3-1188. 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 3-1189. 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 3-1190. Function rameccmu_deinit

Function name	rameccmu_deinit
Function prototype	void rameccmu_deinit(uint32_t rameccmu_periph);
Function descriptions	deinit RAMECCMU unit
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_periph	rameccmu
<i>RAMECCMU0</i>	RAMECCMU for Region 0
<i>RAMECCMU1</i>	RAMECCMU for Region 1
Output parameter{out}	
-	-
Return value	
-	-

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 3-1191. Function rameccmu_monitor_failing_address_get

Function name	rameccmu_monitor_failing_address_get
Function prototype	uint32_t rameccmu_monitor_failing_address_get(rameccmu_monitor_enum rameccmu_monitor);
Function descriptions	get RAMECCMU monitor ECC failing address
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_monitor	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
Output parameter{out}	
-	-
Return value	
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 3-1192. Function `rameccmu_monitor_failing_data_low_bits_get`

Function name	<code>rameccmu_monitor_failing_data_low_bits_get</code>
Function prototype	<code>uint32_t rameccmu_monitor_failing_data_low_bits_get(rameccmu_monitor_enum rameccmu_monitor);</code>
Function descriptions	get RAMECCMU monitor ECC failing data low 32 bits
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_monitor	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
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1193. Function `rameccmu_monitor_failing_data_high_bits_get`

Function name	<code>rameccmu_monitor_failing_data_high_bits_get</code>
Function prototype	<code>uint32_t</code>

	rameccmu_monitor_failing_data_high_bits_get(rameccmu_monitor_enum rameccmu_monitor);
Function descriptions	get RAMECCMU monitor ECC failing data high 32 bits
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_monitor	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
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1194. Function rameccmu_monitor_failing_ecc_error_code_get

Function name	rameccmu_monitor_failing_ecc_error_code_get
Function prototype	uint32_t rameccmu_monitor_failing_ecc_error_code_get(rameccmu_monitor_enum rameccmu_monitor);
Function descriptions	

Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_monitor	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
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1195. Function rameccmu_global_interrupt_enable

Function name	rameccmu_global_interrupt_enable
Function prototype	void rameccmu_global_interrupt_enable(uint32_t rameccmu_periph, uint32_t interrupt);
Function descriptions	enable RAMECCMU global ECC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_periph	rameccmu

<i>RAMECCMU0</i>	RAMECCMU for Region 0
<i>RAMECCMU1</i>	RAMECCMU for Region 1
Input parameter{in}	
interrupt	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1196. Function `rameccmu_global_interrupt_disable`

Function name	<code>rameccmu_global_interrupt_disable</code>
Function prototype	<code>void rameccmu_global_interrupt_disable(uint32_t rameccmu_periph, uint32_t interrupt);</code>
Function descriptions	disable RAMECCMU global ECC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_periph	rameccmu
<i>RAMECCMU0</i>	RAMECCMU for Region 0
<i>RAMECCMU1</i>	RAMECCMU for Region 1
Input parameter{in}	
interrupt	global ECC interrupt
<i>RAMECCMU_INT_EC C_GLOBAL_ERROR</i>	ECC global error interrupt
<i>RAMECCMU_INT_EC</i>	ECC single error interrupt

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

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 3-1197. Function `rameccmu_monitor_interrupt_enable`

Function name	<code>rameccmu_monitor_interrupt_enable</code>
Function prototype	<code>void rameccmu_monitor_interrupt_enable(rameccmu_monitor_enum rameccmu_monitor, uint32_t monitor_interrupt);</code>
Function descriptions	enable RAMECCMU monitor ECC error interrupt
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_monitor	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}	
monitor_interrupt	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
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 3-1198. 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}	
rameccmu_monitor	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>	RAMECCMU0 monitor 3

OR3	
RAMECCMU0_MONIT OR4	RAMECCMU0 monitor 4
RAMECCMU1_MONIT OR0	RAMECCMU1 monitor 0
RAMECCMU1_MONIT OR1	RAMECCMU1 monitor 1
RAMECCMU1_MONIT OR2	RAMECCMU1 monitor 2
Input parameter{in}	
monitor_interrupt	monitor interrupt
RAMECCMU_INT_EC C_SINGLE_ERROR	ECC single error interrupt
RAMECCMU_INT_EC C_DOUBLE_ERROR	ECC double error interrupt
RAMECCMU_INT_EC C_DOUBLE_ERROR_ BYTE_WRITE	ECC double error on byte write interrupt
RAMECCMU_INT_EC C_ERROR_LATCHING	ECC error latching
Output parameter{out}	
-	-
Return value	
-	-

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 3-1199. Function rameccmu_monitor_flag_get

Function name	rameccmu_monitor_flag_get
Function prototype	FlagStatus rameccmu_monitor_flag_get(rameccmu_monitor_enum rameccmu_monitor, uint32_t flag);
Function descriptions	get RAMECCMU monitor ECC error flag
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_monitor	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
Input parameter{in}	
flag	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
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-1200. Function rameccmu_monitor_flag_clear

Function name	rameccmu_monitor_flag_clear
Function prototype	void rameccmu_monitor_flag_clear(rameccmu_monitor_enum

	rameccmu_monitor, uint32_t flag);
Function descriptions	clear RAMECCMU monitor ECC error flag
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_monitor	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
Input parameter{in}	
flag	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
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 3-1201. Function rameccmu_monitor_interrupt_flag_get

Function name	rameccmu_monitor_interrupt_flag_get
Function prototype	FlagStatus rameccmu_monitor_interrupt_flag_get(rameccmu_monitor_enum rameccmu_monitor, uint32_t flag);
Function descriptions	get RAMECCMU monitor ECC interrupt error flag
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_monitor	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
Input parameter{in}	
flag	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
Output parameter{out}	
-	-

Return value	
FlagStatus	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 3-1202. Function rameccmu_monitor_interrupt_flag_clear

Function name	rameccmu_monitor_interrupt_flag_clear
Function prototype	void rameccmu_monitor_interrupt_flag_clear(rameccmu_monitor_enum rameccmu_monitor, uint32_t flag);
Function descriptions	clear RAMECCMU monitor interrupt ECC error flag
Precondition	-
The called functions	-
Input parameter{in}	
rameccmu_monitor	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
Input parameter{in}	
flag	RAMECCMU monitor flag
RAMECCMU_INT_FL AG_ECC_SINGLE_ERR OR	ECC single error detected and corrected flag

RAMECCMU_INT_FLAG_ECC_DOUBLE_ERROR	ECC double error detected flag
RAMECCMU_INT_FLAG_ECC_DOUBLE_ERROR_BYTE_WRITE	ECC double error on byte write detected flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear RAMECCMU monitor ECC error flag */
rameccmu_monitor_interrupt_flag_clear(RAMECCMU0_MONITOR0,
RAMECCMU_INT_FLAG_ECC_SINGLE_ERROR);
```

3.35. 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 [3.35.1](#), the RCU firmware functions are introduced in chapter [3.35.2](#).

3.35.1. Descriptions of Peripheral registers

RCU registers are listed in the table shown as below:

Table 3-1203. 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

Registers	Descriptions
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
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

3.35.2. Descriptions of Peripheral functions

Table 3-1204. RCU firmware function

Function name	Function description
rcu_deinit	deinitialize the RCU
rcu_periph_clock_enable	enable the peripherals clock
rcu_periph_clock_disable	disable the peripherals clock
rcu_periph_clock_sleep_enable	enable the peripherals clock when sleep mode
rcu_periph_clock_sleep_disable	disable the peripherals clock when sleep mode
rcu_periph_reset_enable	reset the peripherals
rcu_periph_reset_disable	disable reset the peripheral
rcu_bkp_reset_enable	enable the BKP domain reset
rcu_bkp_reset_disable	disable the BKP domain reset
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 pll p pllq pll r divider output
rcu_pll_clock_output_disable	disable the pll p pllq pll r 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

Function name	Function description
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_sdio_clock_config	configure the SDIO clock source selection
rcu_deepsleep_wakeup_sys_clock_config	configure the Deep-sleep wakeup system clock source selection
rcu_tli_clock_div_config	configure the TLI clock division 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_sai_clock_config	configure the SAIx(x=0,1) clock source selection
rcu_sai2_block_clock_config	configure the SAI2Bx(x=0,1) clock source selection
rcu_rspdif_clock_config	configure the RSPDIF clock source selection
rcu_exmc_clock_config	configure the EXMC clock source selection
rcu_hpdf_clock_config	configure the HPDF clock source selection
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

Function name	Function description
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 3-1205. Enum rcu_periph_enum

enum name	Function description
RCU_ENET1	ENET1 clock
RCU_ENET1TX	ENET1 TX clock
RCU_ENET1RX	ENET1 RX clock
RCU_ENET1PTP	ENET1 PTP clock
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_ENET0TX	ENET0 TX clock
RCU_ENET0RX	ENET0 RX clock
RCU_ENET0PTP	ENET0 PTP clock
RCU_USBHS1	USBHS1 clock
RCU_USBHS1ULPI	USBHS1HSULPI clock
RCU_DCI	DCI clock
RCU_FAC	FAC clock
RCU_SDIO1	SDIO1 clock
RCU_CAU	CAU clock
RCU_HAU	HAU clock
RCU_TRNG	TRNG clock
RCU_TMU	TMU clock
RCU_RAMECCMU1	RAMECCMU1 clock
RCU_EXMC	EXMC clock
RCU_IPA	IPA 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

enum name	Function description
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_HWSEM	HWSEM 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_TIMER30	TIMER30 clock
RCU_TIMER31	TIMER31 clock
RCU_TIMER50	TIMER50 clock
RCU_TIMER51	TIMER51 clock
RCU_RSPDIF	RSPDIF clock
RCU_SPI1	SPI1 clock
RCU_SPI2	SPI2 clock
RCU_MDIO	MDIO clock
RCU_USART1	USART1 clock
RCU_USART2	USART2 clock
RCU_UART3	UART3 clock
RCU_UART4	UART4 clock
RCU_I2C0	I2C0 clock
RCU_I2C1	I2C1 clock
RCU_I2C2	I2C2 clock
RCU_I2C3	I2C3 clock
RCU_CTC	CTC clock
RCU_DACHOLD	DACHOLD clock
RCU_DAC	DAC clock
RCU_UART6	UART6 clock

enum name	Function description
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_SAI0	SAI0 clock
RCU_SAI1	SAI1 clock
RCU_SAI2	SAI2 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_TLI	TLI clock
RCU_WWDGT	WWDGT clock
RCU_SYSCFG	SYSCFG clock
RCU_CMP	CMP clock
RCU_VREF	VREF clock
RCU_LPPTS	LPPTS 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 3-1206. Enum rcu_periph_sleep_enum

enum name	Function description
RCU_ENET1_SLP	ENET1 clock
RCU_ENET1TX_SLP	ENET1 TX clock
RCU_ENET1RX_SLP	ENET1 RX clock
RCU_ENET1PTP_SLP	ENET1 PTP clock
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_ENET0_SLP	ENET0 clock
RCU_ENET0TX_SLP	ENET0 TX clock
RCU_ENET0RX_SLP	ENET0 RX clock
RCU_ENET0PTP_SLP	ENET0 PTP clock
RCU_USBHS1_SLP	USBHS1 clock
RCU_USBHS1ULPI_SLP	USBHS1ULPI clock
RCU_DCI_SLP	DCI clock
RCU_FAC_SLP	FAC clock
RCU_SDIO1_SLP	SDIO1 clock
RCU_CAU_SLP	CAU clock
RCU_HAU_SLP	HAU clock
RCU_TRNG_SLP	TRNG clock
RCU_TMU_SLP	TMU clock
RCU_RAMECCMU1_SLP	RAMECCMU1 clock
RCU_EXMC_SLP	EXMC clock
RCU_IPA_SLP	IPA clock
RCU_SDIO0_SLP	SDIO0 clock
RCU_MDMA_SLP	MDMMA 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

enum name	Function description
LP	
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
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_TIMER30_SLP	TIMER30 clock
RCU_TIMER31_SLP	TIMER31 clock
RCU_TIMER50_SLP	TIMER50 clock
RCU_TIMER51_SLP	TIMER51 clock
RCU_RSPDIF_SLP	RSPDIF clock
RCU_SPI1_SLP	SPI1 clock
RCU_SPI2_SLP	SPI2 clock
RCU_MDIO_SLP	MDIO 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

enum name	Function description
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_SAI0_SLP	SAI0 clock
RCU_SAI1_SLP	SAI1 clock
RCU_SAI2_SLP	SAI2 clock
RCU_TIMER40_SLP	TIMER40 clock
RCU_TIMER41_SLP	TIMER41 clock
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_TLI_SLP	TLI 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 3-1207. Enum rcu_periph_reset_enum

enum name	Function description
RCU_ENET1RST	ENET1 clock reset
RCU_USBHS0RST	USBHS0 clock reset
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_SDIO1RST	SDIO1 clock reset
RCU_CAURST	CAU clock reset
RCU_HAURST	HAU clock reset
RCU_TRNGRST	TRNG clock reset
RCU_TMURST	TMU clock reset
RCU_EXMCRST	EXMC clock reset
RCU_IPARST	IPA clock reset
RCU_SDIO0RST	SDIO0 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
RCU_GPIOHRST	GPIOH clock reset
RCU_GPIOJRST	GPIOJ clock reset
RCU_GPIOKRST	GPIOK clock reset
RCU_CRCRST	CRC clock reset
RCU_HWSEMRST	HWSEM clock reset
RCU_TIMER1RST	TIMER1 clock reset
RCU_TIMER2RST	TIMER2 clock reset
RCU_TIMER3RST	TIMER3 clock reset

enum name	Function description
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_TIMER30RST	TIMER30 clock reset
RCU_TIMER31RST	TIMER31 clock reset
RCU_TIMER50RST	TIMER50 clock reset
RCU_TIMER51RST	TIMER51 clock reset
RCU_RSPDIFRST	RSPDIF clock reset
RCU_SPI1RST	SPI1 clock reset
RCU_SPI2RST	SPI2 clock reset
RCU_MDIORST	MDIO 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
RCU_SPI4RST	SPI4 clock reset
RCU_SPI5RST	SPI5 clock reset

enum name	Function description
RCU_SAI0RST	SAI0 clock reset
RCU_SAI1RST	SAI1 clock reset
RCU_SAI2RST	SAI2 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_TLIRST	TLI 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 3-1208. 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	PLLUSBH0 stabilization flags

enum name	Function description
RCU_FLAG_PLLUSBH S1STB	PLLUSBHS1 stabilization flags
RCU_FLAG_LCKMD	LXTAL clock failure detection flags
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 3-1209. 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 3-1210. 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 3-1211. 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

enum name	Function description
RCU_INT_PLLUSBHS0 STB	PLLUSBHS0 stabilization interrupt
RCU_INT_PLLUSBHS1 STB	PLLUSBHS1 stabilization interrupt

Enum rcu_osci_type_enum

Table 3-1212. 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 3-1213. 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

enum name	Function description
CK_USART1	USART1 clock
CK_USART2	USART2 clock
CK_USART5	USART5 clock
CK_IRC64MDIV	IRC64MDIV clock
CK_HXTAL	HXTAL clock
CK_LPIRC4M	LPIRC4M clock

Enum usart_idx_enum

Table 3-1214. 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 3-1215. 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 3-1216. Enum can_idx_enum

enum name	Function description
IDX_CAN0	idnex of CAN0
IDX_CAN1	idnex of CAN1
IDX_CAN2	idnex of CAN2

Enum sai_idx_enum

Table 3-1217. Enum sai_idx_enum

enum name	Function description
IDX_SAI0	idnex of SAI0
IDX_SAI1	idnex of SAI1

Enum sai2b_idx_enum

Table 3-1218. Enum sai2b_idx_enum

enum name	Function description
IDX_SAI2B0	idnex of SAI2B0
IDX_SAI2B1	idnex of SAI2B1

Enum adc_idx_enum

Table 3-1219. 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 3-1220. Enum usbhs_idx_enum

enum name	Function description
IDX_USBHS0	idnex of USBHS0
IDX_USBHS1	idnex of USBHS1

Enum pll_idx_enum

Table 3-1221. Enum pll_idx_enum

enum name	Function description
IDX_PLL0	idnex of PLL0
IDX_PLL1	idnex of PLL1
IDX_PLL2	idnex of PLL2

Enum sdio_idx_enum

Table 3-1222. Enum sdio_idx_enum

enum name	Function description
IDX_SDIO0	idnex of SDIO0
IDX_SDIO1	idnex of SDIO1

Enum spi_idx_enum

Table 3-1223. 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

enum name	Function description
IDX_SPI4	idnex of SPI4
IDX_SPI5	idnex of SPI5

rcu_deinit

The description of rcu_deinit is shown as below:

Table 3-1224. Function rcu_deinit

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

Example:

```
/* reset RCU */
rcu_deinit();
```

rcu_periph_clock_enable

The description of rcu_periph_clock_enable is shown as below:

Table 3-1225. Function rcu_periph_clock_enable

Function name	rcu_periph_clock_enable
Function prototype	void rcu_periph_clock_enable(rcu_periph_enum periph);
Function descriptions	enable the peripherals clock
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to Table 3-1205. Enum rcu_periph_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the USART0 clock */
```

```
rcu_periph_clock_enable(RCU_USART0);
```

rcu_periph_clock_disable

The description of rcu_periph_clock_disable is shown as below:

Table 3-1226. Function rcu_periph_clock_disable

Function name	rcu_periph_clock_disable
Function prototype	void rcu_periph_clock_disable(rcu_periph_enum periph);
Function descriptions	disable the peripherals clock
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to Table 3-1205. Enum rcu_periph_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the USART0 clock */
rcu_periph_clock_disable(RCU_USART0);
```

rcu_periph_clock_sleep_enable

The description of rcu_periph_clock_sleep_enable is shown as below:

Table 3-1227. Function rcu_periph_clock_sleep_enable

Function name	rcu_periph_clock_sleep_enable
Function prototype	void rcu_periph_clock_sleep_enable(rcu_periph_sleep_enum periph);
Function descriptions	enable the peripherals clock when in sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to Table 3-1206. Enum rcu_periph_sleep_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the FMC clock when in sleep mode */
rcu_periph_clock_sleep_enable(RCU_FMC_SLP);
```

rcu_periph_clock_sleep_disable

The description of rcu_periph_clock_sleep_disable is shown as below:

Table 3-1228. Function rcu_periph_clock_sleep_disable

Function name	rcu_periph_clock_sleep_disable
Function prototype	void rcu_periph_clock_sleep_disable(rcu_periph_sleep_enum periph);
Function descriptions	disable the peripherals clock when in sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to Table 3-1206. Enum rcu_periph_sleep_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the FMC clock when in sleep mode */
```

```
rcu_periph_clock_sleep_disable(RCU_FMC_SLP);
```

rcu_periph_reset_enable

The description of rcu_periph_reset_enable is shown as below:

Table 3-1229. Function rcu_periph_reset_enable

Function name	rcu_periph_reset_enable
Function prototype	void rcu_periph_reset_enable(rcu_periph_reset_enum periph_reset);
Function descriptions	enable the peripherals reset
Precondition	-
The called functions	-
Input parameter{in}	
periph_reset	RCU peripherals reset, refer to Table 3-1207. Enum rcu_periph_reset_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 reset */
```

```
rcu_periph_reset_enable(RCU_SPI0RST);
```

rcu_periph_reset_disable

The description of rcu_periph_reset_disable is shown as below:

Table 3-1230. Function rcu_periph_reset_disable

Function name	rcu_periph_reset_disable
Function prototype	void rcu_periph_reset_disable(rcu_periph_reset_enum periph_reset);
Function descriptions	disable the peripheral reset
Precondition	-
The called functions	-
Input parameter{in}	
periph_reset	RCU peripherals reset, refer to Table 3-1207. Enum rcu_periph_reset_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 reset */
rcu_periph_reset_disable(RCU_SPI0RST);
```

rcu_bkp_reset_enable

The description of rcu_bkp_reset_enable is shown as below:

Table 3-1231. Function rcu_bkp_reset_enable

Function name	rcu_bkp_reset_enable
Function prototype	void rcu_bkp_reset_enable(void);
Function descriptions	enable the BKP domain reset
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the BKP domain */
rcu_bkp_reset_enable();
```


rcu_bkp_reset_disable

The description of rcu_bkp_reset_disable is shown as below:

Table 3-1232. Function rcu_bkp_reset_disable

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

Example:

```
/* disable the BKP domain reset */
rcu_bkp_reset_disable();
```

rcu_system_clock_source_config

The description of rcu_system_clock_source_config is shown as below:

Table 3-1233. Function rcu_system_clock_source_config

Function name	rcu_system_clock_source_config
Function prototype	void rcu_system_clock_source_config(uint32_t ck_sys);
Function descriptions	configure the system clock source
Precondition	-
The called functions	-
Input parameter{in}	
ck_sys	system clock source select
<i>RCU_CKSYSSRC_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
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* configure the CK_HXTAL as the CK_SYS source */
rcu_system_clock_source_config(RCU_CKSYSSRC_HXTAL);
```

rcu_system_clock_source_get

The description of rcu_system_clock_source_get is shown as below:

Table 3-1234. Function rcu_system_clock_source_get

Function name	rcu_system_clock_source_get
Function prototype	uint32_t rcu_system_clock_source_get(void);
Function descriptions	get the system clock source
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	RCU_SCSS_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 3-1235. Function rcu_ahb_clock_config

Function name	rcu_ahb_clock_config
Function prototype	void rcu_ahb_clock_config(uint32_t ck_ahb);
Function descriptions	configure the AHB clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_ahb	AHB clock prescaler selection
RCU_AHB_CKSYS_DIVx	select CK_SYS / x, (x = 1, 2, 4, 8, 16, 64, 128, 256, 512)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_SYS/128 */
rcu_ahb_clock_config(RCU_AHB_CKSYS_DIV128);
```

rcu_apb1_clock_config

The description of rcu_apb1_clock_config is shown as below:

Table 3-1236. Function rcu_apb1_clock_config

Function name	rcu_apb1_clock_config
Function prototype	void rcu_apb1_clock_config(uint32_t ck_apb1);
Function descriptions	configure the APB1 clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_apb1	APB1 clock prescaler selection
<i>RCU_APB1_CK_AHB_D</i> <i>IVx</i>	select (CK_AHB / x) as CK_APB1 (x = 1, 2, 4, 8, 16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/16 as CK_APB1 */
rcu_apb1_clock_config(RCU_APB1_CK_AHB_DIV16);
```

rcu_apb2_clock_config

The description of rcu_apb2_clock_config is shown as below:

Table 3-1237. Function rcu_apb2_clock_config

Function name	rcu_apb2_clock_config
Function prototype	void rcu_apb2_clock_config(uint32_t ck_apb2);
Function descriptions	configure the APB2 clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	

ck_apb2	APB2 clock prescaler selection
<i>RCU_APB2_CK_AHB_D</i> <i>IVx</i>	select (CK_AHB / x) as CK_APB2 clock (x = 1, 2, 4, 8, 16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB2 */
rcu_apb2_clock_config(RCU_APB2_CK_AHB_DIV8);
```

rcu_apb3_clock_config

The description of rcu_apb3_clock_config is shown as below:

Table 3-1238. Function rcu_apb3_clock_config

Function name	rcu_apb3_clock_config
Function prototype	void rcu_apb3_clock_config(uint32_t ck_apb3);
Function descriptions	configure the APB3 clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_apb3	APB3 clock prescaler selection
<i>RCU_APB3_CK_AHB_D</i> <i>IVx</i>	select (CK_AHB / x) as CK_APB3 clock (x = 1, 2, 4, 8, 16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB3 */
rcu_apb3_clock_config(RCU_APB3_CK_AHB_DIV8);
```

rcu_apb4_clock_config

The description of rcu_apb4_clock_config is shown as below:

Table 3-1239. Function rcu_apb4_clock_config

Function name	rcu_apb4_clock_config
Function prototype	void rcu_apb4_clock_config(uint32_t ck_apb4);
Function descriptions	configure the APB4 clock prescaler selection

Precondition	-
The called functions	-
Input parameter{in}	
ck_apb4	APB4 clock prescaler selection
<i>RCU_APB4_CK_AHB_D</i> <i>IVx</i>	select (CK_AHB / x) as CK_APB4 clock (x = 1, 2, 4, 8, 16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB4 */
```

```
rcu_apb4_clock_config(RCU_APB4_CK_AHB_DIV8);
```

rcu_ckout0_config

The description of rcu_ckout0_config is shown as below:

Table 3-1240. Function rcu_ckout0_config

Function name	rcu_ckout0_config
Function prototype	void rcu_ckout0_config(uint32_t ckout0_src, uint32_t ckout0_div);
Function descriptions	configure the CK_OUT0 clock source and divider
Precondition	-
The called functions	-
Input parameter{in}	
ckout0_src	CK_OUT0 clock source selection
<i>RCU_CKOUT0SRC_IR</i> <i>C64MDIV</i>	IRC64M selected
<i>RCU_CKOUT0SRC_LX</i> <i>TAL</i>	LXTAL selected
<i>RCU_CKOUT0SRC_H</i> <i>XTAL</i>	HXTAL selected
<i>RCU_CKOUT0SRC_PL</i> <i>L0P</i>	PLL0P selected
<i>RCU_CKOUT0SRC_IR</i> <i>C48M</i>	IRC48M selected
<i>RCU_CKOUT0SRC_P</i> <i>ER</i>	PER selected
<i>RCU_CKOUT0SRC_U</i> <i>SBHS060M</i>	USBHS0 60M selected
<i>RCU_CKOUT0SRC_U</i> <i>SBHS160M</i>	USBHS1 60M selected

ckout0_div	CK_OUT0 divider
<i>RCU_CKOUT0_DIVx</i>	CK_OUT is divided by x(x = 1, 2, 3...15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the HXTAL as CK_OUT0 clock source */
```

```
rcu_ckout0_config(RCU_CKOUT0SRC_HXTAL, RCU_CKOUT0_DIV1);
```

rcu_ckout1_config

The description of rcu_ckout1_config is shown as below:

Table 3-1241. Function rcu_ckout1_config

Function name	rcu_ckout1_config
Function prototype	void rcu_ckout1_config(uint32_t ckout1_src, uint32_t ckout1_div);
Function descriptions	configure the CK_OUT1 clock source and divider
Precondition	-
The called functions	-
Input parameter{in}	
ckout1_src	CK_OUT1 clock source selection
<i>RCU_CKOUT1SRC_SYSTEMCLOCK</i>	system clock selected
<i>RCU_CKOUT1SRC_PLL1R</i>	PLL1R selected
<i>RCU_CKOUT1SRC_HXTAL</i>	HXTAL selected
<i>RCU_CKOUT1SRC_PLL0P</i>	PLL0P selected
<i>RCU_CKOUT1SRC_LP1RC4M</i>	LPIRC4M selected
<i>RCU_CKOUT1SRC_IRC32K</i>	IRC32K selected
<i>RCU_CKOUT1SRC_PLL2R</i>	PLL2R selected
Input parameter{in}	
ckout1_div	CK_OUT1 divider
<i>RCU_CKOUT1_DIVx</i>	CK_OUT1 is divided by x(x = 1, 2, 3...15)
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* configure the HXTAL as CK_OUT1 clock source */
```

```
rcu_ckout1_config(RCU_CKOUT1SRC_HXTAL, RCU_CKOUT1_DIV1);
```

rcu_pll_input_output_clock_range_config

The description of rcu_pll_input_output_clock_range_config is shown as below:

Table 3-1242. Function rcu_pll_input_output_clock_range_config

Function name	rcu_pll_input_output_clock_range_config
Function prototype	void rcu_pll_input_output_clock_range_config(pll_idx_enum pll_idx, uint32_t ck_input, uint32_t ck_output);
Function descriptions	configure the pll input and output clock range
Precondition	-
The called functions	-
Input parameter{in}	
pll_idx	pll index, refer to Table 3-1221. Enum pll_idx_enum
Input parameter{in}	
ck_input	input clock range
RCU_PLLRNG_1M_2M	input clock frequency: 1-2MHz
RCU_PLLRNG_2M_4M	input clock frequency: 2-4MHz
RCU_PLLRNG_4M_8M	input clock frequency: 4-8MHz
RCU_PLLRNG_8M_16M	input clock frequency: 8-16MHz
Input parameter{in}	
RCU_PLLVCO_192M_836M	select wide VCO, range: 192-836MHz
RCU_PLLVCO_150M_420M	select narrow VCO, range: 150-420MHz
Output parameter{out}	
-	-
Return value	
-	-

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 3-1243. Function rcu_pll_fractional_config

Function name	rcu_pll_fractional_config
Function prototype	void rcu_pll_fractional_config(pll_idx_enum pll_idx ,uint32_t pll_fracn);
Function descriptions	configure fractional part of the multiplication factor for PLL VCO
Precondition	-
The called functions	-
Input parameter{in}	
pll_idx	pll index, refer to Table 3-1221. Enum pll_idx_enum
Input parameter{in}	
pll_fracn	fractional part of the multiplication factor
uint32_t	0x00-0x00001fff
Output parameter{out}	
-	-
Return value	
-	-

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 3-1244. Function rcu_pll_fractional_latch_enable

Function name	rcu_pll_fractional_latch_enable
Function prototype	void rcu_pll_fractional_latch_enable(pll_idx_enum pll_idx);
Function descriptions	PLL fractional latch enable
Precondition	-
The called functions	-
Input parameter{in}	
pll_idx	pll index, refer to Table 3-1221. Enum pll_idx_enum
Output parameter{out}	
-	-
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 3-1245. 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 Table 3-1221. Enum pll_idx_enum
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 3-1246. 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
<i>RCU_PLLSRC_IRC64MDIV</i>	select IRC64MDIV as PLL source clock
<i>RCU_PLLSRC_LPIRC4M</i>	select LPIRC4M as PLL source clock
<i>RCU_PLLSRC_HXTAL</i>	select HXTAL as PLL source clock
Output parameter{out}	
-	-
Return value	
-	-

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 3-1247. Function rcu_pll0_config

Function name	rcu_pll0_config
Function prototype	ErrStatus rcu_pll0_config(uint32_t pll0_psc, uint32_t pll0_n, uint32_t pll0_p, uint32_t pll0_q, uint32_t pll0_r);
Function descriptions	configure the main PLL0 clock
Precondition	-
The called functions	-
Input parameter{in}	
pll0_psc	the PLL0 VCO source clock prescaler
<i>uint32_t</i>	1-63
Input parameter{in}	
pll0_n	the PLL0 VCO clock multi factor
<i>uint32_t</i>	9-512
Input parameter{in}	
pll0_p	the PLL0P output frequency division factor from PLL0 VCO clock
<i>uint32_t</i>	1-128
Input parameter{in}	
pll0_q	the PLL0Q output frequency division factor from PLL0 VCO clock
<i>uint32_t</i>	1-128
Input parameter{in}	
pll0_r	the PLL0R output frequency division factor from PLL0 VCO clock
<i>uint32_t</i>	1-128
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-1248. Function rcu_pll1_config

Function name	rcu_pll1_config
Function prototype	ErrStatus rcu_pll1_config(uint32_t pll1_psc, uint32_t pll1_n, uint32_t pll1_p, uint32_t pll1_q, uint32_t pll1_r);
Function descriptions	configure the PLL1 clock
Precondition	-
The called functions	-
Input parameter{in}	
pll1_psc	the PLL1 VCO source clock prescaler
uint32_t	1-63
Input parameter{in}	
pll1_n	the PLL1 VCO clock multi factor
uint32_t	9-512
Input parameter{in}	
PLL1_p	the PLL1 P output frequency division factor from PLL1 VCO clock
uint32_t	1-128
Input parameter{in}	
PLL1_q	the PLL1 Q output frequency division factor from PLL1 VCO clock
uint32_t	1-128
Input parameter{in}	
PLL1_r	the PLL1 R output frequency division factor from PLL1 VCO clock
uint32_t	1-128
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-1249. Function rcu_pll2_config

Function name	rcu_pll2_config
Function prototype	ErrStatus rcu_pll2_config(uint32_t pll2_psc, uint32_t pll2_n, uint32_t pll2_p, uint32_t pll2_q, uint32_t pll2_r);
Function descriptions	configure the PLL2 clock
Precondition	-
The called	-

functions	
Input parameter{in}	
pll2_psc	the PLL2 VCO source clock prescaler
<i>uint32_t</i>	1-63
Input parameter{in}	
pll2_n	the PLL2 VCO clock multi factor
<i>uint32_t</i>	9-512
Input parameter{in}	
pll2_p	the PLL2 P output frequency division factor from PLL2 VCO clock
<i>uint32_t</i>	1-128
Input parameter{in}	
pll2_q	the PLL2 Q output frequency division factor from PLL2 VCO clock
<i>uint32_t</i>	1-128
Input parameter{in}	
pll2_r	the PLL2 R output frequency division factor from PLL2 VCO clock
<i>uint32_t</i>	1-128
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-1250. Function rcu_pll_clock_output_enable

Function name	rcu_pll_clock_output_enable
Function prototype	void rcu_pll_clock_output_enable(uint32_t pllxy);
Function descriptions	enable the pll p pll q pll r divider output
Precondition	-
The called functions	-
Input parameter{in}	
pllxy	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1251. Function rcu_pll_clock_output_disable

Function name	rcu_pll_clock_output_disable
Function prototype	void rcu_pll_clock_output_disable(uint32_t pllxy);
Function descriptions	disable the pll p pllq pllr divider output
Precondition	-
The called functions	-
Input parameter{in}	
pllxy	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1252. Function rcu_pllusb0_config

Function name	rcu_pllusb0_config
Function prototype	void rcu_pllusb0_config(uint32_t pllusb_presel, uint32_t pllusb_predv, uint32_t pllusb_mf, uint32_t usbhsv);
Function descriptions	configure the PLLUSBH0 clock
Precondition	-
The called functions	-
Input parameter{in}	
pllusb_presel	PLLUSBH0PRE clock selection
<i>RCU_PLLUSBHSPRE_HXTAL</i>	PLLUSBHSPRE select HXTAL as clock
<i>RCU_PLLUSBHSPRE_IRC48M</i>	PLLUSBHSPRE select IRC48M as clock
Input parameter{in}	
pllusb_predv	the divider factor from PLLUSBH0 clock
<i>RCU_PLLUSBHSPRE_DIVx (x= 1...15)</i>	select RCU_PLLUSBHSPRE_DIVx as PLLUSBH0 clock divider factor
Input parameter{in}	
pllusb_mf	PLLUSBH0 clock multiplication factor
<i>RCU_PLLUSBHS_MULx (x = 16,17...127)</i>	select RCU_PLLUSBHS_MULx as PLLUSBH0 clock multiplication factor
Input parameter{in}	
usbhsv	the divider factor from USBH0DV clock
<i>RCU_USBHS_DIVx(x = 2,4...16)</i>	select RCU_USBHS_DIVx as USBHSDV clock divider factor
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR-

Example:

```
/* configure the PLLUSBH0 */
```

```
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 3-1253. Function rcu_pllusb1_config

Function name	rcu_pllusb1_config
Function prototype	void rcu_pllusb1_config(uint32_t pllusb_presel, uint32_t pllusb_predv, uint32_t pllusb_mf, uint32_t usbhsv);
Function descriptions	configure the PLLUSBHS1 clock
Precondition	-
The called functions	-
Input parameter{in}	
pllusb_presel	PLLUSBHS1PRE clock selection
RCU_PLLUSBHSPRE_HXTAL	PLLUSBHSPRE select HXTAL as clock
RCU_PLLUSBHSPRE_IRC48M	PLLUSBHSPRE select IRC48M as clock
Input parameter{in}	
pllusb_predv	the divider factor from PLLUSBHS1 clock
RCU_PLLUSBHSPRE_DIVx (x = 1...15)	select RCU_PLLUSBHSPRE_DIVx as PLLUSBHS clock divider factor
Input parameter{in}	
pllusb_mf	PLLUSBHS1 clock multiplication factor
RCU_PLLUSBHS_MULx (x = 16,17...127)	select RCU_PLLUSBHS_MULx as PLLUSBHS clock multiplication factor
Input parameter{in}	
usbhsv	the divider factor from USBHS1DV clock
RCU_USBHS_DIVx(x = 2,4...16)	select RCU_USBHS_DIVx as USBHSDV clock divider factor
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-1254. Function rcu_rtc_clock_config

Function name	rcu_rtc_clock_config
Function prototype	void rcu_rtc_clock_config(uint32_t rtc_clock_source);
Function descriptions	configure the RTC clock source selection

Precondition	-
The called functions	-
Input parameter{in}	
rtc_clock_source	RTC clock source selection
<i>RCU_RTCSRC_NONE</i>	no clock selected
<i>RCU_RTCSRC_LXTAL</i>	CK_LXTAL selected as RTC source clock
<i>RCU_RTCSRC_IRC32K</i>	CK_IRC32K selected as RTC source clock
<i>RCU_RTCSRC_HXTAL_DIV_RTCDIV</i>	CK_HXTAL/RTCDIV selected as RTC source clock
Output parameter{out}	
-	-
Return value	
-	-

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 3-1255. Function rcu_rtc_div_config

Function name	rcu_rtc_div_config
Function prototype	void rcu_rtc_div_config(uint32_t rtc_div);
Function descriptions	configure the frequency division of RTC clock when HXTAL was selected as its clock source
Precondition	-
The called functions	-
Input parameter{in}	
rtc_div	RTC clock frequency division
<i>RCU_RTC_HXTAL_NONE</i>	no clock for RTC
<i>RCU_RTC_HXTAL_DIV</i> <i>x(x = 2,3...63)</i>	RTCDIV clock select CK_HXTAL/x
Output parameter{out}	
-	-
Return value	
-	-

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 3-1256. Function rcu_ck48m_clock_config

Function name	rcu_ck48m_clock_config
Function prototype	void rcu_ck48m_clock_config(uint32_t ck48m_clock_source);
Function descriptions	configure the CK48M clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
ck48m_clock_source	CK48M clock source selection
RCU_CK48MSRC_PLL48M	CK_PLL48M selected as CK48M source clock
RCU_CK48MSRC_IRC48M	CK_IRC48M selected as CK48M source clock
Output parameter{out}	
-	-
Return value	
-	-

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 3-1257. Function rcu_pll48m_clock_config

Function name	rcu_pll48m_clock_config
Function prototype	void rcu_pll48m_clock_config(uint32_t pll48m_clock_source);
Function descriptions	configure the PLL48M clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
pll48m_clock_source	PLL48M clock source selection
RCU_PLL48MSRC_PL_L0Q	CK_PLL0Q selected as PLL48M source clock
RCU_PLL48MSRC_PL_L2P	CK_PLL2P selected as PLL48M source clock
Output parameter{out}	

-	-
Return value	
-	-

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 3-1258. Function rcu_irc64mdiv_clock_config

Function name	rcu_irc64mdiv_clock_config
Function prototype	void rcu_irc64mdiv_clock_config(uint32_t ck_irc64mdiv);
Function descriptions	configure the IRC64M clock divider selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_irc64mdiv	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1259. Function rcu_irc64mdiv_freq_get

Function name	rcu_irc64mdiv_freq_get
Function prototype	uint32_t rcu_irc64mdiv_freq_get(void);
Function descriptions	get the irc64mdiv clock
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
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 3-1260. Function rcu_timer_clock_prescaler_config

Function name	rcu_timer_clock_prescaler_config
Function prototype	void rcu_timer_clock_prescaler_config(uint32_t timer_clock_prescaler);
Function descriptions	configure the TIMER clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_clock_prescaler	TIMER clock selection
RCU_TIMER_PSC_MUL2	If CK_APBx = CK_AHB or CK_APBx = CK_AHB/2, CK_TIMERx = CK_AHB, else CK_TIMERx = 2 x CK_APBx
RCU_TIMER_PSC_MUL4	If CK_APBx = CK_AHB or CK_APBx = CK_AHB/2 or CK_APBx = CK_AHB/4, CK_TIMERx = CK_AHB, else CK_TIMERx = 4 x CK_APBx
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER clock source */
```

```
rcu_timer_clock_prescaler_config(RCU_TIMER_PSC_MUL4);
```

rcu_spi_clock_config

The description of rcu_spi_clock_config is shown as below:

Table 3-1261. Function rcu_spi_clock_config

Function name	rcu_spi_clock_config
Function prototype	void rcu_spi_clock_config(spi_idx_enum spi_idx, uint32_t ck_spi);

Function descriptions	configure the SPI / I2S clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
spi_idx	SPI index, refer to Table 3-1223. Enum spi_idx_enum
Input parameter{in}	
ck_spi	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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PLL0Q as SPI0 clock */
rcu_spi_clock_config(IDX_SPI0, RCU_SPISRC_PLL0Q);
```

rcu_sdio_clock_config

The description of rcu_sdio_clock_config is shown as below:

Table 3-1262. Function rcu_sdio_clock_config

Function name	rcu_sdio_clock_config
Function prototype	void rcu_sdio_clock_config(sdio_idx_enum sdio_idx, uint32_t ck_sdio);
Function descriptions	configure the SPI / I2S clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
sdio_idx	SDIO index, refer to Table 3-1222. Enum sdio_idx_enum

Input parameter{in}	
ck_sdio	SDIO clock source selection
<i>RCU_SDIO0SRC_PLL0Q</i>	select CK_PLL0Q as SDIO0 clock
<i>RCU_SDIO0SRC_PLL1R</i>	select CK_PLL1R as SDIO0 clock
<i>RCU_SDIO1SRC_PLL0Q</i>	select CK_PLL0Q as SDIO1 clock
<i>RCU_SDIO1SRC_PLL1R</i>	select CK_PLL1R as SDIO1 clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PLL0Q as SDIO0 clock */
rcu_sdio_clock_config(IDX_SDIO0, RCU_SDIO0SRC_PLL0Q);
```

rcu_deepsleep_wakeup_sys_clock_config

The description of rcu_deepsleep_wakeup_sys_clock_config is shown as below:

Table 3-1263. Function rcu_deepsleep_wakeup_sys_clock_config

Function name	rcu_deepsleep_wakeup_sys_clock_config
Function prototype	void rcu_deepsleep_wakeup_sys_clock_config(uint32_t ck_dspwussel);
Function descriptions	configure the Deep-sleep wakeup system clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_dspwussel	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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_LPIRC4M as Deep-sleep wakeup system clock source */
rcu_deepsleep_wakeup_sys_clock_config(RCU_DSPWUSSEL_LPIRC4M);
```

rrcu_tli_clock_div_config

The description of rrcu_tli_clock_div_config is shown as below:

Table 3-1264. Function rrcu_tli_clock_div_config

Function name	rrcu_tli_clock_div_config
Function prototype	void rrcu_tli_clock_div_config(uint32_t pll2_r_div);
Function descriptions	configure the PLL2R divider used as input of TLI
Precondition	-
The called functions	-
Input parameter{in}	
pll2_r_div	PLL2R divider used as input of TLI
RCU_PLL2R_DIV2	PLL2R / 2
RCU_PLL2R_DIV4	PLL2R / 4
RCU_PLL2R_DIV8	PLL2R / 8
RCU_PLL2R_DIV16	PLL2R / 16
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the TLI prescaler factor from PLL2R clock */
rrcu_tli_clock_div_config(RCU_PLL2R_DIV4);
```

rrcu_usart_clock_config

The description of rrcu_usart_clock_config is shown as below:

Table 3-1265. Function rrcu_usart_clock_config

Function name	rrcu_usart_clock_config
Function prototype	void rrcu_usart_clock_config(usart_idx_enum usart_idx, uint32_t ck_usart);
Function descriptions	configure the USARTx(x=0,1,2,5) clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
usart_idx	USART index, refer to Table 3-1214. Enum usart_idx_enum
Input parameter{in}	
ck_usart	USART clock source selection
RCU_USARTSRC_APB	select CK_APB as USART clock
RCU_USARTSRC_AHB	select CK_AHB as USART clock
RCU_USARTSRC_LXTAL	select CK_LXTAL as USART clock

<i>RCU_USARTSRC_IRC64MDIV</i>	select CK_IRC64MDIV as USART clock
Output parameter{out}	
-	-
Return value	
-	-

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 3-1266. Function rcu_i2c_clock_config

Function name	rcu_i2c_clock_config
Function prototype	void rcu_i2c_clock_config(i2c_idx_enum i2c_idx, uint32_t ck_i2c);
Function descriptions	configure the I2Cx(x=0,1,2,3) clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_idx	I2C index, refer to Table 3-1215. Enum i2c_idx_enum
Input parameter{in}	
ck_i2c	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_IRC64MDIV</i>	select CK_IRC64MDIV as I2C clock
<i>RCU_I2CSRC_LPIRC4M</i>	select CK_LPIRC4M as I2C clock
Output parameter{out}	
-	-
Return value	
-	-

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 3-1267. Function rcu_can_clock_config

Function name	rcu_can_clock_config
Function prototype	void rcu_can_clock_config(can_idx_enum can_idx, uint32_t ck_can);
Function descriptions	configure the CANx(x=0,1,2) clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
can_idx	CAN index, refer to Table 3-1216. Enum can_idx_enum
Input parameter{in}	
ck_can	CAN clock source selection
<i>RCU_CANSRC_HXTAL</i> <i>L</i>	select CK_HXTAL as CAN clock
<i>RCU_CANSRC_APB2</i>	select CK_APB2 as CAN clock
<i>RCU_CANSRC_APB2_</i> <i>DIV2</i>	select CK_APB2 / 2 as CAN clock
<i>RCU_CANSRC_IRC64</i> <i>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 3-1268. 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 Table 3-1219. Enum adc_idx_enum
Input parameter{in}	
ck_adc	ADC clock source selection
<i>RCU_ADCSRC_PLL1P</i>	select CK_PLL1P as ADC clock
<i>RCU_ADCSRC_PLL2R</i>	select CK_PLL2R as ADC clock
<i>RCU_ADCSRC_PER</i>	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_sai_clock_config

The description of rcu_sai_clock_config is shown as below:

Table 3-1269. Function rcu_sai_clock_config

Function name	rcu_sai_clock_config
Function prototype	void rcu_sai_clock_config(sai_idx_enum sai_idx, uint32_t ck_sai);
Function descriptions	configure the SAIx(x=0,1) clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
sai_idx	SAI index, refer to Table 3-1217. Enum sai_idx_enum
Input parameter{in}	
ck_sai	SAI clock source selection
RCU_SAISRC_PLL0Q	select CK_PLL0Q as SAI clock
RCU_SAISRC_PLL1P	select CK_PLL1P as SAI clock
RCU_SAISRC_PLL2P	select CK_PLL2P as SAI clock
RCU_SAISRC_CKIN	select CK_I2S_CKIN as SAI clock
RCU_SAISRC_PER	select CK_I2S_PER as SAI clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_PLLQ as the SAI0 clock source */
rcu_sai_clock_config(IDX_SAI0, RCU_SAISRC_PLLQ);
```

rcu_sai2_block_clock_config

The description of rcu_sai2_block_clock_config is shown as below:

Table 3-1270. Function rcu_sai2_block_clock_config

Function name	rcu_sai2_block_clock_config
---------------	-----------------------------

Function prototype	void rcu_sai2_block_clock_config(sai2b_idx_enum sai2b_idx, uint32_t ck_sai2b);
Function descriptions	configure the SAI2Bx(x=0,1) clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
sai2b_idx	SAI2 index, refer to Table 3-1218. Enum sai2b_idx_enum
Input parameter{in}	
ck_sai2b	SAI2 clock source selection
<i>RCU_SAI2BSRC_PLL0Q</i>	select CK_PLL0Q as SAI2 clock
<i>RCU_SAI2BSRC_PLL1P</i>	select CK_PLL1P as SAI2 clock
<i>RCU_SAI2BSRC_PLL2P</i>	select CK_PLL2P as SAI2 clock
<i>RCU_SAI2BSRC_CKIN</i>	select I2S_CKIN as SAI2 clock
<i>RCU_SAI2BSRC_PER</i>	select CK_PER as SAI2 clock
<i>RCU_SAI2BSRC_RSPDIF_SYMB</i>	select CK_RSPDIF_SYMB as SAI2 clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_PLLQ as the SAI2B0 clock source */
```

```
rcu_sai2_block_clock_config(IDX_SAI2B0, RCU_SAI2SRC_PLLQ);
```

rcu_rspdif_clock_config

The description of rcu_rspdif_clock_config is shown as below:

Table 3-1271. Function rcu_rspdif_clock_config

Function name	rcu_rspdif_clock_config
Function prototype	void rcu_rspdif_clock_config(uint32_t ck_rspdif);
Function descriptions	configure the RSPDIF clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_rspdif	RSPDIF clock source selection
<i>RCU_RSPDIFSRC_PLLOQ</i>	select CK_PLL0Q as RSPDIF clock
<i>RCU_RSPDIFSRC_PL</i>	select CK_PLL1R as RSPDIF clock

<i>L1R</i>	
<i>RCU_RSPDIFSRC_PL</i> <i>L2R</i>	select CK_PLL2R as RSPDIF clock
<i>RCU_RSPDIFSRC_IR</i> <i>C64MDIV</i>	select CK_IRC64MDIV as RSPDIF clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_PLLQ as the RSPDIF clock source */
```

```
rcu_rspdif_clock_config(RCU_RSPDIFSRC_PLLQ);
```

rcu_exmc_clock_config

The description of rcu_exmc_clock_config is shown as below:

Table 3-1272. 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	-
The called functions	-
Input parameter{in}	
ck_exmc	EXMC clock source selection
<i>RCU_EXMCSRC_AHB</i>	select CK_AHB as EXMC clock
<i>RCU_EXMCSRC_PLL0</i> <i>Q</i>	select CK_PLL0Q as EXMC clock
<i>RCU_EXMCSRC_PLL1</i> <i>R</i>	select CK_PLL1R as EXMC clock
<i>RCU_EXMCSRC_PER</i>	select CK_PER as EXMC clock
Output parameter{out}	
-	-
Return value	
-	-

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 3-1273. Function rcu_hpdf_clock_config

Function name	rcu_hpdf_clock_config
Function prototype	void rcu_hpdf_clock_config(uint32_t ck_hpdf);
Function descriptions	configure the HPDF clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_hpdf	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1274. 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
<i>RCU_PERSRC_IRC64MDIV</i>	select CK_IRC64MDIV as PER clock
<i>RCU_PERSRC_LPIRC4M</i>	select CK_LPIRC4M as PER clock
<i>RCU_PERSRC_HXTAL</i>	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 3-1275. 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 Table 3-1220. Enum usbhs_idx_enum
Input parameter{in}	
ck_usbhspsc	USBHS clock prescaler from CK_PLL1Q
<i>RCU_USBHSPSC_DIV</i> <i>x(x=1,2...8)</i>	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 3-1276. 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 Table 3-1220. Enum usbhs_idx_enum

Input parameter{in}	
ck_usb48m	USBHS48M clock source selection
<i>RCU_USB48MSRC_PL L0R</i>	select CK_PLL0R as USBHS48M clock
<i>RCU_USB48MSRC_PL LUSBHS</i>	select CK_PLLUSBHS as USBHS48M clock
<i>RCU_USB48MSRC_PL L1Q</i>	select CK_PLL1Q as USBHS48M clock
<i>RCU_USB48MSRC_IR C48M</i>	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 3-1277. 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);
Function descriptions	configure the USBHSSEL clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
usbhs_idx	USBHS index, refer to Table 3-1220. Enum usbhs_idx_enum
Input parameter{in}	
ck_usbhs	USBHS clock source selection
<i>RCU_USBHSSEL_48M</i>	select 48M as USBHS clock
<i>RCU_USBHSSEL_60M</i>	select 60M as USBHS clock
Output parameter{out}	
-	-
Return value	
-	-

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 3-1278. Function rcu_usbhs_clock_selection_enable

Function name	rcu_usbhs_clock_selection_enable
Function prototype	void rcu_usbhs_clock_selection_enable(usbhs_idx_enum usbhs_idx);
Function descriptions	enable the USBHS clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
usbhs_idx	USBHS index, refer to Table 3-1220. Enum usbhs_idx_enum
Output parameter{out}	
-	-
Return value	
-	-

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 3-1279. Function rcu_usbhs_clock_selection_disable

Function name	rcu_usbhs_clock_selection_disable
Function prototype	void rcu_usbhs_clock_selection_disable(usbhs_idx_enum usbhs_idx);
Function descriptions	disable the USBHS clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
usbhs_idx	USBHS index, refer to Table 3-1220. Enum usbhs_idx_enum
Output parameter{out}	
-	-
Return value	
-	-

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 3-1280. Function rcu_lxtal_drive_capability_config

Function name	rcu_lxtal_drive_capability_config
Function prototype	void rcu_lxtal_drive_capability_config(uint32_t lxtal_dricap);
Function descriptions	configure the LXTAL drive capability
Precondition	-
The called functions	-
Input parameter{in}	
lxtal_dricap	drive capability of LXTAL
<i>RCU_LXTAL_LOWDRI</i>	lower driving capability
<i>RCU_LXTAL_MED_LOWDRI</i>	medium low driving capability
<i>RCU_LXTAL_MED_HIGHDRI</i>	medium high driving capability
<i>RCU_LXTAL_HIGHDRI</i>	higher driving capability
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the LXTAL lower driving capability */
```

```
rcu_lxtal_drive_capability_config(RCU_LXTAL_LOWDRI);
```

rcu_oscstap_wait

The description of rcu_oscstap_wait is shown as below:

Table 3-1281. Function rcu_oscstap_wait

Function name	rcu_oscstap_wait
Function prototype	ErrStatus rcu_oscstap_wait(rcu_oscstap_type_enum osci);
Function descriptions	wait for oscillator stabilization flags is SET or oscillator startup is timeout
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to Table 3-1212. Enum rcu_oscstap_type_enum
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* wait for oscillator stabilization flag */

if(SUCCESS == rcu_osc_stab_wait(RCU_HXTAL)){

}
```

rcu_osc_on

The description of rcu_osc_on is shown as below:

Table 3-1282. Function rcu_osc_on

Function name	rcu_osc_on
Function prototype	void rcu_osc_on(rcu_osc_type_enum osci);
Function descriptions	turn on the oscillator
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to Table 3-1212. Enum rcu_osc_type_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn on the high speed crystal oscillator */

rcu_osc_on(RCU_HXTAL);
```

rcu_osc_off

The description of rcu_osc_off is shown as below:

Table 3-1283. Function rcu_osc_off

Function name	rcu_osc_off
Function prototype	void rcu_osc_off(rcu_osc_type_enum osci);
Function descriptions	turn off the oscillator
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to Table 3-1212. Enum rcu_osc_type_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn off the high speed crystal oscillator */
```

```
rcu_osci_off(RCU_HXTAL);
```

rcu_osci_bypass_mode_enable

The description of rcu_osci_bypass_mode_enable is shown as below:

Table 3-1284. Function rcu_osci_bypass_mode_enable

Function name	rcu_osci_bypass_mode_enable
Function prototype	void rcu_osci_bypass_mode_enable(rcu_osci_type_enum osci);
Function descriptions	enable the oscillator bypass mode
Precondition	HXTALEN or LXTALEN must be reset before it
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to Table 3-1212. Enum rcu_osci_type_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the high speed crystal oscillator bypass mode */
```

```
rcu_osci_bypass_mode_enable(RCU_HXTAL);
```

rcu_osci_bypass_mode_disable

The description of rcu_osci_bypass_mode_disable is shown as below:

Table 3-1285. Function rcu_osci_bypass_mode_disable

Function name	rcu_osci_bypass_mode_disable
Function prototype	void rcu_osci_bypass_mode_disable(rcu_osci_type_enum osci);
Function descriptions	disable the oscillator bypass mode
Precondition	HXTALEN or LXTALEN must be reset before it
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to Table 3-1212. Enum rcu_osci_type_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the high speed crystal oscillator bypass mode */
```

```
rcu_osci_bypass_mode_disable(RCU_HXTAL);
```

rcu_irc64m_adjust_value_set

The description of rcu_irc64m_adjust_value_set is shown as below:

Table 3-1286. Function rcu_irc64m_adjust_value_set

Function name	rcu_irc64m_adjust_value_set
Function prototype	void rcu_irc64m_adjust_value_set(uint32_t irc64m_adjval);
Function descriptions	set the IRC64M adjust value
Precondition	-
The called functions	-
Input parameter{in}	
irc64m_adjval	IRC64M adjust value, must be between 0 and 0x7F
Output parameter{out}	
-	-
Return value	
-	-

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 3-1287. Function rcu_lpirc4m_adjust_value_set

Function name	rcu_lpirc4m_adjust_value_set
Function prototype	void rcu_lpirc4m_adjust_value_set(uint32_t lpirc4m_adjval);
Function descriptions	set the LPIRC4M adjust value
Precondition	-
The called functions	-
Input parameter{in}	
lpirc4m_adjval	LPIRC4M adjust value, must be between 0 and 0x3F
Output parameter{out}	
-	-
Return value	
-	-

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 3-1288. Function rcu_hxtal_clock_monitor_enable

Function name	rcu_hxtal_clock_monitor_enable
Function prototype	void rcu_hxtal_clock_monitor_enable(void);
Function descriptions	enable the HXTAL clock monitor
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the HXTAL clock monitor */
```

```
rcu_hxtal_clock_monitor_enable();
```

rcu_hxtal_clock_monitor_disable

The description of rcu_hxtal_clock_monitor_disable is shown as below:

Table 3-1289. Function rcu_hxtal_clock_monitor_disable

Function name	rcu_hxtal_clock_monitor_disable
Function prototype	void rcu_hxtal_clock_monitor_disable(void);
Function descriptions	disable the HXTAL clock monitor
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the HXTAL clock monitor */
```

```
rcu_hxtal_clock_monitor_disable();
```

rcu_lxtal_clock_monitor_enable

The description of rcu_lxtal_clock_monitor_enable is shown as below:

Table 3-1290. Function rcu_lxtal_clock_monitor_enable

Function name	rcu_lxtal_clock_monitor_enable
Function prototype	void rcu_lxtal_clock_monitor_enable(void);
Function descriptions	enable the LXTAL clock monitor
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the LXTAL clock monitor */
rcu_lxtal_clock_monitor_enable();
```

rcu_lxtal_clock_monitor_disable

The description of rcu_lxtal_clock_monitor_disable is shown as below:

Table 3-1291. Function rcu_lxtal_clock_monitor_disable

Function name	rcu_lxtal_clock_monitor_disable
Function prototype	void rcu_lxtal_clock_monitor_disable(void);
Function descriptions	disable the LXTAL clock monitor
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the LXTAL clock monitor */
rcu_lxtal_clock_monitor_disable();
```

rcu_clock_freq_get

The description of rcu_clock_freq_get is shown as below:

Table 3-1292. Function rcu_clock_freq_get

Function name	rcu_clock_freq_get
Function prototype	uint32_t rcu_clock_freq_get(rcu_clock_freq_enum clock);
Function descriptions	get the system clock, bus clock and peripheral clock frequency
Precondition	-
The called functions	-
Input parameter{in}	
clock	the clock frequency which to get, refer to Table 3-1213. Enum rcu_clock_freq_enum
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 3-1293. Function rcu_flag_get

Function name	rcu_flag_get
Function prototype	FlagStatus rcu_flag_get(rcu_flag_enum flag);
Function descriptions	get the clock stabilization and peripheral reset flags
Precondition	-
The called functions	-
Input parameter{in}	
flag	the clock stabilization and peripheral reset flags, refer to Table 3-1208. Enum rcu_flag_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the clock stabilization flag */
```

```
if(RESET != rcu_flag_get(RCU_FLAG_LXTALSTB)){
```

```
}
```

rcu_all_reset_flag_clear

The description of rcu_all_reset_flag_clear is shown as below:

Table 3-1294. 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 3-1295. Function rcu_interrupt_enable

Function name	rcu_interrupt_enable
Function prototype	void rcu_interrupt_enable(rcu_int_enum interrupt);
Function descriptions	enable the stabilization interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	clock stabilization interrupt, refer to Table 3-1211. Enum rcu_int_enum
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 3-1296. Function rcu_interrupt_disable

Function name	rcu_interrupt_disable
Function prototype	void rcu_interrupt_disable(rcu_int_enum interrupt);
Function descriptions	disable the stabilization interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	clock stabilization interrupt, refer to Table 3-1211. Enum rcu_int_enum
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 3-1297. Function rcu_interrupt_flag_get

Function name	rcu_interrupt_flag_get
Function prototype	FlagStatus rcu_interrupt_flag_get(rcu_int_flag_enum int_flag);
Function descriptions	get the clock stabilization interrupt and ckm flags
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt and ckm flags, refer to Table 3-1209. Enum rcu_int_flag_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the clock stabilization interrupt flag */
```



```
if(SET == rcu_interrupt_flag_get(RCU_INT_FLAG_HXTALSTB)){
}
```

rcu_interrupt_flag_clear

The description of rcu_interrupt_flag_clear is shown as below:

Table 3-1298. Function rcu_interrupt_flag_clear

Function name	rcu_interrupt_flag_clear
Function prototype	void rcu_interrupt_flag_clear(rcu_int_flag_clear_enum int_flag)
Function descriptions	clear the interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	clock stabilization and stuck interrupt flags clear, refer to Table 3-1210. Enum rcu_int_flag_clear_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the interrupt HXTAL stabilization interrupt flag */
rcu_interrupt_flag_clear(RCU_INT_FLAG_HXTALSTB_CLR);
```

3.36. RSPDIF

The receiver of Sony/Philips Digital Interface (RSPDIF) module provides the function of receiving and decoding RSPDIF audio data streams. The RSPDIF registers are listed in chapter [3.36.1](#), the RSPDIF firmware functions are introduced in chapter [3.36.2](#).

3.36.1. Descriptions of Peripheral registers

RSPDIF registers are listed in the table shown as below:

Table 3-1299. RSPDIF Registers

Registers	Descriptions
RSPDIF_CTL	RSPDIF control register
RSPDIF_INTEN	RSPDIF interrupt enable register
RSPDIF_STAT	RSPDIF status register
RSPDIF_STATC	RSPDIF status flag clear register
RSPDIF_DATA	RSPDIF RX data register
RSPDIF_CHSTAT	RSPDIF RX channel status register

Registers	Descriptions
RSPDIF_DTH	RSPDIF RX data threshold register

3.36.2. Descriptions of Peripheral functions

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

Table 3-1300. RSPDIF firmware function

Function name	Function description
rspdif_deinit	reset the RSPDIF
rspdif_struct_para_init	initialize the parameters of RSPDIF structure with the default values
rspdif_init	initialize the RSPDIF parameters
rspdif_enable	specifies the RSPDIF peripheral state
rspdif_disable	disable RSPDIF
rspdif_symbol_clock_enable	enable RSPDIF symbol clock
rspdif_symbol_clock_disable	disable RSPDIF symbol clock
rspdif_backup_symbol_clock_enable	enable RSPDIF backup symbol clock
rspdif_backup_symbol_clock_disable	disable RSPDIF backup symbol clock
rspdif_dma_enable	enable the RSPDIF receiver DMA
rspdif_dma_disable	disable the RSPDIF receiver DMA
rspdif_control_buffer_dma_enable	enable the RSPDIF control buffer DMA
rspdif_control_buffer_dma_disable	disable the RSPDIF control buffer DMA
rspdif_data_read	RSPDIF read data
rspdif_duration_of_symbols_get	get duration of 5 symbols counted using rspdif_ck
rspdif_user_data_get	get user data information
rspdif_channel_status_get	get channel status information
rspdif_start_block_status_get	get start of block
rspdif_low_threshold_get	get threshold low estimation
rspdif_high_threshold_get	get threshold high estimation
rspdif_flag_get	get RSPDIF flag status
rspdif_flag_clear	clear RSPDIF flag
rspdif_interrupt_enable	enable RSPDIF interrupt
rspdif_interrupt_disable	disable RSPDIF interrupt
rspdif_interrupt_flag_get	get RSPDIF interrupt flag status
rspdif_interrupt_flag_clear	clear RSPDIF interrupt flag status

Structure `rspdif_parameter_struct`

Table 3-1301. Structure `rspdif_parameter_struct`

Member name	Function description
input_sel	the RSPDIF input selection (RSPDIF_INPUT_INx, x = 0,1,2,3)

Member name	Function description
max_retrie	the RSPDIF maximum allowed re-tries during synchronization phase (RSPDIF_MAXRETRIES_NONE, RSPDIF_MAXRETRIES_3, RSPDIF_MAXRETRIES_15, RSPDIF_MAXRETRIES_63)
wait_activity	the RSPDIF wait for activity on the selected input (RSPDIF_WAIT_FOR_ACTIVITY_OFF, RSPDIF_WAIT_FOR_ACTIVITY_ON)
channel_sel	whether swapping the channel status from channel A or B (RSPDIF_CHANNEL_A, RSPDIF_CHANNEL_B)
sample_format	the RSPDIF data samples format (RSPDIF_DATAFORMAT_LSB, RSPDIF_DATAFORMAT_MSB, RSPDIF_DATAFORMAT_32BITS)
sound_mode	the RSPDIF is in stereo or mono mode (RSPDIF_STEREOMODE_DISABLE, RSPDIF_STEREOMODE_ENABLE)
pre_type	whether copy the preamble type value into the RSPDIF_DATA (RSPDIF_PREAMBLE_TYPE_MASK_OFF, RSPDIF_PREAMBLE_TYPE_MASK_ON)
channel_status_bit	whether the channel status and user bits are copied or not into the received frame (RSPDIF_CHANNEL_STATUS_MASK_OFF, RSPDIF_CHANNEL_STATUS_MASK_ON)
validity_bit	whether the validity bit is copied or not into the received frame (RSPDIF_VALIDITY_MASK_OFF, RSPDIF_VALIDITY_MASK_ON)
parity_error_bit	whether the parity error bit is copied or not into the received frame (RSPDIF_PERROR_MASK_OFF, RSPDIF_PERROR_MASK_ON)
symbol_clk	the RSPDIF symbol clock generation (RSPDIF_SYMBOL_CLK_OFF, RSPDIF_SYMBOL_CLK_ON)
bak_symbol_clk	the RSPDIF backup symbol clock generation (RSPDIF_BACKUP_SYMBOL_CLK_OFF, RSPDIF_BACKUP_SYMBOL_CLK_ON)

Structure rspdif_data_struct

Table 3-1302. Structure rspdif_data_struct

Member name	Function description
format	the data format
preamble	the preamble type
channel_status	channel status bit
user_bit	user bit
validity	validity bit
parity_err	parity error bit
data0	data value 0
data1	data value 1

rspdif_deinit

The description of rspdif_deinit is shown as below:

Table 3-1303. Function rspdif_deinit

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

Example:

```
/* reset RSPDIF */
rspdif_deinit ();
```

rspdif_struct_para_init

The description of rspdif_struct_para_init is shown as below:

Table 3-1304. Function rspdif_struct_para_init

Function name	rspdif_struct_para_init
Function prototype	void rspdif_struct_para_init(rspdif_parameter_struct *rspdif_struct);
Function descriptions	initialize the parameters of RSPDIF structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
rspdif_parameter_struct	the initialized structure rspdif_parameter_struct pointer, the structure members can refer to members of the structure Table 3-1301. Structure <u>rspdif_parameter</u> struct.
Return value	
-	-

Example:

```
/* initialize the parameters of RSPDIF structure with the default values */
rspdif_parameter_struct rspdif_structure;
```

```
rspdif_struct_para_init(&rspdif_structure);
```

rspdif_init

The description of `rspdif_init` is shown as below:

Table 3-1305. Function `rspdif_init`

Function name	<code>rspdif_init</code>
Function prototype	<code>void rspdif_init(rspdif_parameter_struct *rspdif_struct);</code>
Function descriptions	initialize the RSPDIF parameters
Precondition	-
The called functions	-
Input parameter{in}	
rspdif_parameter_struct	RSPDIF parameter initialization structure, the structure members can refer to members of the structure Table 3-1301. Structure <code>rspdif_parameter_struct</code> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the RSPDIF */

rspdif_parameter_struct rspdif_structure;

rspdif_struct_para_init(&rspdif_structure);

rspdif_structure.input_sel      = RSPDIF_INPUT_IN0;
rspdif_structure.max_retrie    = RSPDIF_MAXRETRIES_15;
rspdif_structure.wait_activity = RSPDIF_WAIT_FOR_ACTIVITY_ON;
rspdif_structure.channel_sel    = RSPDIF_CHANNEL_A;
rspdif_structure.sample_format  = RSPDIF_DATAFORMAT_MSB;
rspdif_structure.sound_mode     = RSPDIF_STEREOMODE_ENABLE;
rspdif_structure.pre_type       = RSPDIF_PREAMBLE_TYPE_MASK_OFF;
rspdif_structure.channel_status_bit = RSPDIF_CHANNEL_STATUS_MASK_OFF;
rspdif_structure.validity_bit   = RSPDIF_VALIDITY_MASK_OFF;
rspdif_structure.parity_error_bit = RSPDIF_PERROR_MASK_OFF;
rspdif_structure.symbol_clk     = RSPDIF_SYMBOL_CLK_OFF;
rspdif_structure.bak_symbol_clk = RSPDIF_BACKUP_SYMBOL_CLK_OFF;
```

```
rspdif_init(&rspdif_structure);
```

rspdif_enable

The description of rspdif_enable is shown as below:

Table 3-1306. Function rspdif_enable

Function name	rspdif_enable
Function prototype	void rspdif_enable(uint32_t mode);
Function descriptions	specifies the RSPDIF peripheral state
Precondition	-
The called functions	-
Input parameter{in}	
mode	RSPDIF state
RSPDIF_STATE_SYN C	enable RSPDIF synchronization only
RSPDIF_STATE_RCV	enable RSPDIF receiver
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RSPDIF */
```

```
rspdif_enable(RSPDIF_STATE_RCV);
```

rspdif_disable

The description of rspdif_disable is shown as below:

Table 3-1307. Function rspdif_disable

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

Example:

```
/* disable RSPDIF */
```

```
rspdif_disable();
```

rspdif_symbol_clock_enable

The description of rspdif_symbol_clock_enable is shown as below:

Table 3-1308. Function rspdif_symbol_clock_enable

Function name	rspdif_symbol_clock_enable
Function prototype	void rspdif_symbol_clock_enable(void);
Function descriptions	enable RSPDIF symbol clock
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RSPDIF symbol clock */
```

```
rspdif_symbol_clock_enable();
```

rspdif_symbol_clock_disable

The description of rspdif_symbol_clock_disable is shown as below:

Table 3-1309. Function rspdif_symbol_clock_disable

Function name	rspdif_symbol_clock_disable
Function prototype	void rspdif_symbol_clock_disable(void);
Function descriptions	disable RSPDIF symbol clock
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RSPDIF symbol clock */
```

```
rspdif_symbol_clock_disable();
```

rspdif_backup_symbol_clock_enable

The description of `rspdif_backup_symbol_clock_enable` is shown as below:

Table 3-1310. Function `rspdif_backup_symbol_clock_enable`

Function name	<code>rspdif_backup_symbol_clock_enable</code>
Function prototype	<code>void rspdif_backup_symbol_clock_enable(void);</code>
Function descriptions	enable RSPDIF backup symbol clock
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RSPDIF backup symbol clock */
```

```
rspdif_backup_symbol_clock_enable();
```

rspdif_backup_symbol_clock_disable

The description of `rspdif_backup_symbol_clock_disable` is shown as below:

Table 3-1311. Function `rspdif_backup_symbol_clock_disable`

Function name	<code>rspdif_backup_symbol_clock_disable</code>
Function prototype	<code>void rspdif_backup_symbol_clock_disable(void);</code>
Function descriptions	disable RSPDIF backup symbol clock
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RSPDIF backup symbol clock */
```

```
rspdif_backup_symbol_clock_disable();
```


rspdif_dma_enable

The description of rspdif_dma_enable is shown as below:

Table 3-1312. Function rspdif_dma_enable

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

Example:

```
/* enable the RSPDIF receiver DMA */
rspdif_dma_enable();
```

rspdif_dma_disable

The description of rspdif_dma_disable is shown as below:

Table 3-1313. Function rspdif_dma_disable

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

Example:

```
/* disable the RSPDIF receiver DMA */
rspdif_dma_disable();
```

rspdif_control_buffer_dma_enable

The description of rspdif_control_buffer_dma_enable is shown as below:

Table 3-1314. Function rspdif_control_buffer_dma_enable

Function name	rspdif_control_buffer_dma_enable
Function prototype	void rspdif_control_buffer_dma_enable(void);
Function descriptions	enable the RSPDIF control buffer DMA
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the RSPDIF control buffer DMA */
```

```
rspdif_control_buffer_dma_enable();
```

rspdif_control_buffer_dma_disable

The description of rspdif_control_buffer_dma_disable is shown as below:

Table 3-1315. Function rspdif_control_buffer_dma_disable

Function name	rspdif_control_buffer_dma_disable
Function prototype	void rspdif_control_buffer_dma_disable(void);
Function descriptions	disable the RSPDIF control buffer DMA
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the RSPDIF control buffer DMA */
```

```
rspdif_control_buffer_dma_disable();
```

rspdif_data_read

The description of rspdif_data_read is shown as below:

Table 3-1316. Function rspdif_data_read

Function name	rspdif_data_read
Function prototype	void rspdif_data_read(rspdif_data_struct *data_struct);
Function descriptions	RSPDIF read data
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
data_struct	RSPDIF data structure and the structure members can refer to members of the structure Table 3-1302. Structure rspdif_data_struct
Return value	
-	-

Example:

```
/* RSPDIF read data */
rspdif_data_struct data_structure;
rspdif_data_read(&data_structure);
```

rspdif_duration_of_symbols_get

The description of rspdif_duration_of_symbols_get is shown as below:

Table 3-1317. Function rspdif_duration_of_symbols_get

Function name	rspdif_duration_of_symbols_get
Function prototype	uint32_t rspdif_duration_of_symbols_get(void);
Function descriptions	get duration of 5 symbols counted using rspdif_ck
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	0x0-0x7FFF

Example:

```
/* get duration of 5 symbols counted using rspdif_ck */
uint32_t cnt;
```

```
cnt = rspdif_duration_of_symbols_get();
```

rspdif_user_data_get

The description of `rspdif_user_data_get` is shown as below:

Table 3-1318. Function `rspdif_user_data_get`

Function name	<code>rspdif_user_data_get</code>
Function prototype	<code>uint32_t rspdif_user_data_get(void);</code>
Function descriptions	get user data information
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<code>uint32_t</code>	0x0-0xFFFF

Example:

```
/* get user data information */

uint32_t user_data;

user_data = rspdif_user_data_get();
```

rspdif_channel_status_get

The description of `rspdif_channel_status_get` is shown as below:

Table 3-1319. Function `rspdif_channel_status_get`

Function name	<code>rspdif_channel_status_get</code>
Function prototype	<code>uint32_t rspdif_channel_status_get(void);</code>
Function descriptions	get channel status information
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<code>uint32_t</code>	0x0-0xFF

Example:

```
/* get channel status information */
```

```
uint32_t ch_status;
```

```
ch_status = rspdif_channel_status_get();
```

rspdif_start_block_status_get

The description of `rspdif_start_block_status_get` is shown as below:

Table 3-1320. Function `rspdif_start_block_status_get`

Function name	<code>rspdif_start_block_status_get</code>
Function prototype	<code>FlagStatus rspdif_start_block_status_get(void);</code>
Function descriptions	get start of block
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get start of block */
```

```
FlagStatus flag;
```

```
flag = rspdif_start_block_status_get();
```

rspdif_low_threshold_get

The description of `rspdif_low_threshold_get` is shown as below:

Table 3-1321. Function `rspdif_low_threshold_get`

Function name	<code>rspdif_low_threshold_get</code>
Function prototype	<code>uint32_t rspdif_low_threshold_get(void);</code>
Function descriptions	get threshold low estimation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	0x0~0x1FF

Example:

```
/* get threshold low estimation */
```

```
uint32_t low_th;
```

```
low_th = rspdif_low_threshold_get();
```

rspdif_high_threshold_get

The description of `rspdif_high_threshold_get` is shown as below:

Table 3-1322. Function `rspdif_high_threshold_get`

Function name	<code>rspdif_high_threshold_get</code>
Function prototype	<code>uint32_t rspdif_high_threshold_get(void);</code>
Function descriptions	get threshold high estimation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	0x0~0xFF

Example:

```
/* get threshold high estimation */
```

```
uint32_t high_th;
```

```
high_th = rspdif_high_threshold_get();
```

rspdif_flag_get

The description of `rspdif_flag_get` is shown as below:

Table 3-1323. Function `rspdif_flag_get`

Function name	<code>rspdif_flag_get</code>
Function prototype	<code>FlagStatus rspdif_flag_get(uint16_t flag);</code>
Function descriptions	get RSPDIF flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	RSPDIF flag status
<code>RSPDIF_FLAG_RBNE</code>	RSPDIF RX buffer is not empty
<code>RSPDIF_FLAG_CBNE</code>	RSPDIF RX control buffer is not empty
<code>RSPDIF_FLAG_PERR</code>	RSPDIF parity error
<code>RSPDIF_FLAG_RXORER</code>	RSPDIF RX overrun error

RSPDIF_FLAG_SYND B	RSPDIF synchronization block detected
RSPDIF_FLAG_SYND O	RSPDIF synchronization done
RSPDIF_FLAG_FRER R	RSPDIF frame error
RSPDIF_FLAG_SYNE RR	RSPDIF synchronization error
RSPDIF_FLAG_TMOU TERR	RSPDIF time out error
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get RSPDIF flag status */
```

```
FlagStatus flag;
```

```
flag = rspdif_flag_get(RSPDIF_FLAG_TMOUERR);
```

rspdif_flag_clear

The description of `rspdif_flag_clear` is shown as below:

Table 3-1324. Function `rspdif_flag_clear`

Function name	<code>rspdif_flag_clear</code>
Function prototype	<code>void rspdif_flag_clear(uint16_t flag);</code>
Function descriptions	clear RSPDIF flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	RSPDIF flag status
RSPDIF_FLAG_PERR	RSPDIF parity error
RSPDIF_FLAG_RXOR ER	RSPDIF RX overrun error
RSPDIF_FLAG_SYND B	RSPDIF synchronization block detected
RSPDIF_FLAG_SYND O	RSPDIF synchronization done
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear RSPDIF flag */
rspdif_flag_clear(RSPDIF_FLAG_PERR);
```

rspdif_interrupt_enable

The description of rspdif_interrupt_enable is shown as below:

Table 3-1325. Function rspdif_interrupt_enable

Function name	rspdif_interrupt_enable
Function prototype	void rspdif_interrupt_enable(uint8_t interrupt);
Function descriptions	enable RSPDIF interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	RSPDIF interrupt
RSPDIF_INT_RBNE	RSPDIF RX buffer is not empty interrupt
RSPDIF_INT_CBNE	RSPDIF RX control buffer is not empty interrupt
RSPDIF_INT_PERR	RSPDIF parity error interrupt
RSPDIF_INT_RXORER R	RSPDIF RX overrun error interrupt
RSPDIF_INT_SYNDB	RSPDIF synchronization block detected interrupt
RSPDIF_INT_SYNDO	RSPDIF synchronization done interrupt
RSPDIF_INT_RXDCER R	RSPDIF data decoding error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RSPDIF interrupt */
rspdif_interrupt_enable(RSPDIF_INT_RBNE);
```

rspdif_interrupt_disable

The description of rspdif_interrupt_disable is shown as below:

Table 3-1326. Function rspdif_interrupt_disable

Function name	rspdif_interrupt_disable
Function prototype	void rspdif_interrupt_disable(uint8_t interrupt);
Function descriptions	disable RSPDIF interrupt
Precondition	-

The called functions	-
Input parameter{in}	
interrupt	RSPDIF interrupt
RSPDIF_INT_RBNE	RSPDIF RX buffer is not empty interrupt
RSPDIF_INT_CBNE	RSPDIF RX control buffer is not empty interrupt
RSPDIF_INT_PERR	RSPDIF parity error interrupt
RSPDIF_INT_RXORER R	RSPDIF RX overrun error interrupt
RSPDIF_INT_SYNDB	RSPDIF synchronization block detected interrupt
RSPDIF_INT_SYNDO	RSPDIF synchronization done interrupt
RSPDIF_INT_RXDCER R	RSPDIF data decoding error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RSPDIF interrupt */
rspdif_interrupt_disable(RSPDIF_INT_RBNE);
```

rspdif_interrupt_flag_get

The description of rspdif_interrupt_flag_get is shown as below:

Table 3-1327. Function rspdif_interrupt_flag_get

Function name	rspdif_interrupt_flag_get
Function prototype	FlagStatus rspdif_interrupt_flag_get(uint16_t int_flag);
Function descriptions	get RSPDIF interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	RSPDIF interrupt flag status
RSPDIF_INT_FLAG_RBNE	RSPDIF RX buffer is not empty interrupt flag
RSPDIF_INT_FLAG_CBNE	RSPDIF RX control buffer is not empty interrupt flag
RSPDIF_INT_FLAG_PERR	RSPDIF parity error interrupt flag
RSPDIF_INT_FLAG_RXORER	RSPDIF RX overrun error interrupt flag
RSPDIF_INT_FLAG_SYNDB	RSPDIF synchronization block detected interrupt flag

RSPDIF_INT_FLAG_S YNDO	RSPDIF synchronization done interrupt flag
RSPDIF_INT_FLAG_F RERR	RSPDIF frame error interrupt flag
RSPDIF_INT_FLAG_S YNERR	RSPDIF synchronization error interrupt flag
RSPDIF_INT_FLAG_T MOUTERR	RSPDIF time out error interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get RSPDIF interrupt flag status */
```

```
FlagStatus flag;
```

```
flag = rspdif_interrupt_flag_get(RSPDIF_INT_FLAG_TMOUTERR);
```

rspdif_interrupt_flag_clear

The description of `rspdif_interrupt_flag_clear` is shown as below:

Table 3-1328. Function `rspdif_interrupt_flag_clear`

Function name	<code>rspdif_interrupt_flag_clear</code>
Function prototype	<code>void rspdif_interrupt_flag_clear(uint16_t int_flag);</code>
Function descriptions	clear RSPDIF interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	SPDIF interrupt flag status
RSPDIF_INT_FLAG_P ERR	RSPDIF parity error interrupt flag
RSPDIF_INT_FLAG_R XORER	RSPDIF RX overrun error interrupt flag
RSPDIF_INT_FLAG_S YNDB	RSPDIF synchronization block detected interrupt flag
RSPDIF_INT_FLAG_S YNDO	RSPDIF synchronization done interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear RSPDIF interrupt flag status */

rspdif_interrupt_flag_clear(RSPDIF_INT_FLAG_PERR);
```

3.37. 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 [3.37.1](#), the FWDGT firmware functions are introduced in chapter [3.37.2](#).

3.37.1. Descriptions of Peripheral registers

RTC registers are listed in the table shown as below:

Table 3-1329. 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

Registers	Descriptions
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
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

3.37.2. Descriptions of Peripheral functions

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

Table 3-1330. 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

Function name	Function description
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 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 3-1331. 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 3-1332. 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 3-1333. 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 3-1334. 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_	RTC tamper precharge feature during a voltage level detection

enable	
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 3-1335. Function rtc_deinit

Function name	rtc_deinit
Function prototype	ErrStatus rtc_deinit(void);
Function descriptions	reset most of the RTC registers
Precondition	-
The called functions	rcu_periph_reset_enable/ rcu_periph_reset_disable -
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* reset most of the RTC registers*/
```

```
ErrStatus error_status = rtc_deinit();
```

rtc_init

The description of rtc_init is shown as below:

Table 3-1336. Function rtc_init

Function name	rtc_init
Function prototype	ErrStatus rtc_init(rtc_parameter_struct* rtc_initpara_struct);
Function descriptions	initialize RTC registers
Precondition	-
The called functions	-
Input parameter{in}	
rtc_initpara_struct	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure Table 3-1331. Structure rtc_parameter_struct
Output parameter{out}	
-	-

Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* initialize RTC registers */

rtc_parameter_struct rtc_initpara;

rtc_interrupt_disable(RTC_INT_SECOND);

rtc_initpara.factor_asyn = prescaler_a;

rtc_initpara.factor_syn = prescaler_s;

rtc_initpara.year = 0x16;

rtc_initpara.day_of_week = RTC_SATURDAY;

rtc_initpara.month = RTC_APR;

rtc_initpara.date = 0x30;

rtc_initpara.display_format = RTC_24HOUR;

rtc_initpara.am_pm = RTC_AM;

rtc_init(&rtc_initpara);
```

rtc_init_mode_enter

The description of rtc_init_mode_enter is shown as below:

Table 3-1337. Function rtc_init_mode_enter

Function name	rtc_init_mode_enter
Function prototype	ErrStatus rtc_init_mode_enter(void);
Function descriptions	enter RTC init mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/*enter RTC init mode*/

ErrStatus error_status = rtc_init_mode_enter();
```


rtc_init_mode_exit

The description of rtc_init_mode_exit is shown as below:

Table 3-1338. Function rtc_init_mode_exit

Function name	rtc_init_mode_exit
Function prototype	void rtc_init_mode_exit(void);
Function descriptions	exit RTC init mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*exit RTC init mode*/
rtc_init_mode_exit();
```

rtc_register_sync_wait

The description of rtc_register_sync_wait is shown as below:

Table 3-1339. Function rtc_register_sync_wait

Function name	rtc_register_sync_wait
Function prototype	ErrStatus rtc_register_sync_wait(void);
Function descriptions	wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/*wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the
shadow registers are updated*/

ErrStatus error_status = rtc_register_sync_wait();
```

rtc_current_time_get

The description of rtc_current_time_get is shown as below:

Table 3-1340. Function rtc_current_time_get

Function name	rtc_current_time_get
Function prototype	void rtc_current_time_get(rtc_parameter_struct* rtc_initpara_struct);
Function descriptions	get current time and date
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
rtc_initpara_struct	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure Table 3-1331. Structure rtc_parameter_struct
Return value	
-	-

Example:

```
/*get current time and date*/
rtc_parameter_struct rtc_initpara_struct;
rtc_current_time_get(&rtc_initpara_struct);
```

rtc_subsecond_get

The description of rtc_subsecond_get is shown as below:

Table 3-1341. Function rtc_subsecond_get

Function name	rtc_subsecond_get
Function prototype	uint32_t rtc_subsecond_get(void);
Function descriptions	get current subsecond value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	current subsecond value(0x00-0xFFFF)

Example:

```
/*get current subsecond value*/
```

```
uint32_t sub_second = rtc_subsecond_get();
```

rtc_alarm_config

The description of rtc_alarm_config is shown as below:

Table 3-1342. Function rtc_alarm_config

Function name	rtc_alarm_config
Function prototype	void rtc_alarm_config(uint8_t rtc_alarm, rtc_alarm_struct *rtc_alarm_time);
Function descriptions	configure RTC alarm
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	RTC_ALARM0 or RTC_ALARM1
Input parameter{in}	
rtc_alarm_time	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to members of the structure Table 3-1332. Structure rtc_alarm_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*rtc_alarm_config*/
```

```
rtc_alarm_struct rtc_alarm_time;
```

```
rtc_alarm_config(RTC_ALARM0,&rtc_alarm_time);
```

rtc_alarm_subsecond_config

The description of rtc_alarm_subsecond_config is shown as below:

Table 3-1343. 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
<i>RTC_MASKSSC_0_14</i>	mask alarm subsecond configuration
<i>RTC_MASKSSC_1_14</i>	mask RTC_ALARM0SS_SSC[14:1], and RTC_ALARM0SS_SSC[0] is to be compared
<i>RTC_MASKSSC_2_14</i>	mask RTC_ALARM0SS_SSC[14:2], and RTC_ALARM0SS_SSC[1:0] is to be compared
<i>RTC_MASKSSC_3_14</i>	mask RTC_ALARM0SS_SSC[14:3], and RTC_ALARM0SS_SSC[2:0] is to be compared
<i>RTC_MASKSSC_4_14</i>	mask RTC_ALARM0SS_SSC[14:4], and RTC_ALARM0SS_SSC[3:0] is to be compared
<i>RTC_MASKSSC_5_14</i>	mask RTC_ALARM0SS_SSC[14:5], and RTC_ALARM0SS_SSC[4:0] is to be compared
<i>RTC_MASKSSC_6_14</i>	mask RTC_ALARM0SS_SSC[14:6], and RTC_ALARM0SS_SSC[5:0] is to be compared
<i>RTC_MASKSSC_7_14</i>	mask RTC_ALARM0SS_SSC[14:7], and RTC_ALARM0SS_SSC[6:0] is to be compared
<i>RTC_MASKSSC_8_14</i>	mask RTC_ALARM0SS_SSC[14:8], and RTC_ALARM0SS_SSC[7:0] is to be compared
<i>RTC_MASKSSC_9_14</i>	mask RTC_ALARM0SS_SSC[14:9], and RTC_ALARM0SS_SSC[8:0] is to be compared
<i>RTC_MASKSSC_10_14</i> 4	mask RTC_ALARM0SS_SSC[14:10], and RTC_ALARM0SS_SSC[9:0] is to be compared
<i>RTC_MASKSSC_11_14</i> 4	mask RTC_ALARM0SS_SSC[14:11], and RTC_ALARM0SS_SSC[10:0] is to be compared
<i>RTC_MASKSSC_12_14</i> 4	mask RTC_ALARM0SS_SSC[14:12], and RTC_ALARM0SS_SSC[11:0] is to be compared
<i>RTC_MASKSSC_13_14</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
Input parameter{in}	
subsecond	alarm subsecond value(0x000 - 0x7FFF)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1344. Function rtc_alarm_enable

Function name	rtc_alarm_enable
Function prototype	void rtc_alarm_enable(uint8_t rtc_alarm);
Function descriptions	enable RTC alarm
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	RTC_ALARM0 or RTC_ALARM1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*enable RTC alarm*/
rtc_alarm_enable(RTC_ALARM0);
```

rtc_alarm_disable

The description of rtc_alarm_disable is shown as below:

Table 3-1345. Function rtc_alarm_disable

Function name	rtc_alarm_disable
Function prototype	ErrStatus rtc_alarm_disable(uint8_t rtc_alarm);
Function descriptions	disable RTC alarm
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	RTC_ALARM0 or RTC_ALARM1
Output parameter{out}	
-	-
Return value	
ErrStatus	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 3-1346. Function rtc_alarm_get

Function name	rtc_alarm_get
Function prototype	void rtc_alarm_get(uint8_t rtc_alarm, rtc_alarm_struct *rtc_alarm_time);
Function descriptions	get RTC alarm
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	RTC_ALARM0 or RTC_ALARM1
Output parameter{out}	
rtc_alarm_time	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to members of the structure Table 3-1332. Structure rtc_alarm_struct
Return value	
-	-

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 3-1347. Function rtc_alarm_subsecond_get

Function name	rtc_alarm_subsecond_get
Function prototype	uint32_t rtc_alarm_subsecond_get(uint8_t rtc_alarm);
Function descriptions	get RTC alarm subsecond
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	RTC_ALARM0 or RTC_ALARM1
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1348. Function rtc_timestamp_enable

Function name	rtc_timestamp_enable
Function prototype	void rtc_timestamp_enable(uint32_t edge);
Function descriptions	enable RTC time-stamp
Precondition	-
The called functions	-
Input parameter{in}	
edge	specify which edge to detect of time-stamp
<i>RTC_TIMESTAMP_RISING_EDGE</i>	rising edge is valid event edge for timestamp event
<i>RTC_TIMESTAMP_FALLING_EDGE</i>	falling edge is valid event edge for timestamp event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*enable RTC time-stamp*/
```

```
rtc_timestamp_enable(RTC_TIMESTAMP_RISING_EDGE);
```

rtc_timestamp_disable

The description of rtc_timestamp_disable is shown as below:

Table 3-1349. Function rtc_timestamp_disable

Function name	rtc_timestamp_disable
Function prototype	void rtc_timestamp_disable(void);
Function descriptions	disable RTC time-stamp
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*disable RTC time-stamp*/
```

```
rtc_timestamp_disable();
```

rtc_timestamp_internalevent_config

The description of rtc_timestamp_internalevent_config is shown as below:

Table 3-1350. Function rtc_timestamp_internalevent_config

Function name	rtc_timestamp_internalevent_config
Function prototype	void rtc_timestamp_internalevent_config(uint32_t mode)
Function descriptions	configure RTC time-stamp internal event
Precondition	-
The called functions	-
Input parameter{in}	
mode	specify which internal or external event to be detected
<i>RTC_ITSEN_DISABLE</i>	disable RTC time-stamp internal event
<i>RTC_ITSEN_ENABLE</i>	enable RTC time-stamp internal event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC time-stamp internal event */
```

```
rtc_timestamp_internalevent_config(RTC_ITSEN_DISABLE);
```

rtc_timestamp_get

The description of rtc_timestamp_get is shown as below:

Table 3-1351. Function rtc_timestamp_get

Function name	rtc_timestamp_get
Function prototype	void rtc_timestamp_get(rtc_timestamp_struct* rtc_timestamp);
Function descriptions	get RTC timestamp time and date
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
rtc_timestamp	Pointer to a rtc_timestamp_struct structure which contains parameters for RTC time-stamp configuration, the structure members can refer to members of the structure Table 3-1334. Structure rtc_tamper_struct

Return value	
-	-

Example:

```
/* get RTC timestamp time and date */

rtc_timestamp_struct rtc_timestamp;

rtc_timestamp_get(& rtc_timestamp);
```

rtc_timestamp_subsecond_get

The description of rtc_timestamp_subsecond_get is shown as below:

Table 3-1352. Function rtc_timestamp_subsecond_get

Function name	rtc_timestamp_subsecond_get
Function prototype	uint32_t rtc_timestamp_subsecond_get(void);
Function descriptions	get RTC time-stamp subsecond
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	RTC time-stamp subsecond value

Example:

```
/* get RTC time-stamp subsecond */

uint32_t subsecond = rtc_timestamp_subsecond_get();
```

rtc_tamper_enable

The description of rtc_tamper_enable is shown as below:

Table 3-1353. Function rtc_tamper_enable

Function name	rtc_tamper_enable
Function prototype	void rtc_tamper_enable(rtc_tamper_struct* rtc_tamper);
Function descriptions	enable RTC tamper
Precondition	-
The called functions	-
Input parameter{in}	
rtc_tamper	pointer to a rtc_tamper_struct structure which contains parameters for RTC tamper configuration, the structure members can refer to members of the structure Table 3-1334. Structure rtc_tamper_struct

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RTC tamper */
rtc_tamper_struct rtc_tamper
rtc_tamper_enable(& rtc_tamper);
```

rtc_tamper_disable

The description of rtc_tamper_disable is shown as below:

Table 3-1354. Function rtc_tamper_disable

Function name	rtc_tamper_disable
Function prototype	void rtc_tamper_disable(uint32_t source);
Function descriptions	disable RTC tamper
Precondition	-
The called functions	-
Input parameter{in}	
source	specify which tamper source to be disabled
RTC_TAMPER0	RTC tamper0
RTC_TAMPER1	RTC tamper1
RTC_TAMPER2	RTC tamper2
Output parameter{out}	
-	-
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 3-1355. 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 3-1356. 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
Input parameter{in}	
mode	specify the output pin mode when output alarm signal or auto wakeup signal
<i>RTC_ALARM_OUTPUT_OD</i>	open drain mode
<i>RTC_ALARM_OUTPUT_PP</i>	push pull mode
Output parameter{out}	
-	-
Return value	
-	-

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 3-1357. Function rtc_calibration_output_config

Function name	rtc_calibration_output_config
Function prototype	void rtc_calibration_output_config(uint32_t source);
Function descriptions	configure rtc calibration output source
Precondition	-
The called functions	-
Input parameter{in}	
source	specify signal to output
<i>RTC_CALIBRATION_5</i> <i>12HZ</i>	when the LSE frequency is 32768Hz and the RTC_PSC is the default value, output 512Hz signal
<i>RTC_CALIBRATION_1</i> <i>HZ</i>	when the LSE frequency is 32768Hz and the RTC_PSC is the default value, output 1Hz signal
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* 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 3-1358. Function rtc_hour_adjust

Function name	rtc_hour_adjust
Function prototype	void rtc_hour_adjust(uint32_t operation);
Function descriptions	adjust the daylight saving time by adding or subtracting one hour from the current time
Precondition	-
The called functions	-
Input parameter{in}	
operation	hour adjustment operation
<i>RTC_CTL_A1H</i>	add one hour

<i>RTC_CTL_S1H</i>	subtract one hour
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* adjust the daylight saving time by adding one hour from the current time */
```

```
rtc_hour_adjust(RTC_CTL_A1H);
```

rtc_second_adjust

The description of rtc_second_adjust is shown as below:

Table 3-1359. Function rtc_second_adjust

Function name	rtc_second_adjust
Function prototype	ErrStatus rtc_second_adjust(uint32_t add, uint32_t minus);
Function descriptions	adjust RTC second or subsecond value of current time
Precondition	-
The called functions	-
Input parameter{in}	
add	add 1s to current time or not
<i>RTC_SHIFT_ADD1S_R ESET</i>	no effect
<i>RTC_SHIFT_ADD1S_S ET</i>	add 1s to current time
Input parameter{in}	
minus	number of subsecond to minus from current time(0x0 - 0x7FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* adjust RTC second or subsecond value of current time */
```

```
ErrStatus error_status = rtc_second_adjust(RTC_SHIFT_ADD1S_SET, 0);
```

rtc_bypass_shadow_enable

The description of rtc_bypass_shadow_enable is shown as below:

Table 3-1360. Function rtc_bypass_shadow_enable

Function name	rtc_bypass_shadow_enable
----------------------	--------------------------

Function prototype	void rtc_bypass_shadow_enable(void);
Function descriptions	enable RTC bypass shadow registers function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RTC bypass shadow registers function*/
```

```
rtc_bypass_shadow_enable();
```

rtc_bypass_shadow_disable

The description of rtc_bypass_shadow_disable shown as below:

Table 3-1361. Function rtc_bypass_shadow_disable

Function name	rtc_bypass_shadow_disable
Function prototype	void rtc_bypass_shadow_disable(void);
Function descriptions	disable RTC bypass shadow registers function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC bypass shadow registers function */
```

```
rtc_bypass_shadow_disable();
```

rtc_refclock_detection_enable

The description of rtc_refclock_detection_enable shown as below:

Table 3-1362. Function rtc_refclock_detection_enable

Function name	rtc_refclock_detection_enable
Function prototype	ErrStatus rtc_refclock_detection_enable(void);

Function descriptions	enable RTC reference clock detection function
Precondition	-
The called functions	rtc_init_mode_enter/rtc_init_mode_exit
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* enable RTC reference clock detection function*/
```

```
ErrStatus error_status = rtc_refclock_detection_enable();
```

rtc_refclock_detection_disable

The description of rtc_refclock_detection_disable is shown as below:

Table 3-1363. Function rtc_refclock_detection_disable

Function name	rtc_refclock_detection_disable
Function prototype	ErrStatus rtc_refclock_detection_disable(void);
Function descriptions	disable RTC reference clock detection function
Precondition	-
The called functions	rtc_init_mode_enter/rtc_init_mode_exit
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* disable RTC reference clock detection function*/
```

```
ErrStatus error_status = rtc_refclock_detection_disable();
```

rtc_wakeup_enable

The description of rtc_refclock_detection_disable is shown as below:

Table 3-1364. Function rtc_wakeup_enable

Function name	rtc_wakeup_enable
Function prototype	void rtc_wakeup_enable(void);
Function descriptions	enable RTC auto wakeup function

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

Example:

```
/* enable RTC auto wakeup function */
```

```
rtc_wakeup_enable();
```

rtc_wakeup_disable

The description of rtc_wakeup_disable is shown as below:

Table 3-1365. Function rtc_wakeup_disable

Function name	rtc_wakeup_disable
Function prototype	ErrStatus rtc_wakeup_disable(void);
Function descriptions	disable RTC auto wakeup function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* disable RTC auto wakeup function */
```

```
ErrStatus error_status = rtc_wakeup_disable();
```

rtc_wakeup_clock_set

The description of rtc_wakeup_clock_set is shown as below:

Table 3-1366. Function rtc_wakeup_clock_set

Function name	rtc_wakeup_clock_set
Function prototype	ErrStatus rtc_wakeup_clock_set(uint8_t wakeup_clock);
Function descriptions	set RTC auto wakeup timer clock
Precondition	-

The called functions	-
Input parameter{in}	
wakeup_clock	wakeup timer clock is RTC clock divided factor
<i>WAKEUP_RTCK_DIV</i> 16	RTC auto wakeup timer clock is RTC clock divided by 16
<i>WAKEUP_RTCK_DIV</i> 8	RTC auto wakeup timer clock is RTC clock divided by 8
<i>WAKEUP_RTCK_DIV</i> 4	RTC auto wakeup timer clock is RTC clock divided by 4
<i>WAKEUP_RTCK_DIV</i> 2	RTC auto wakeup timer clock is RTC clock divided by 2
<i>WAKEUP_CKSPRE</i>	RTC auto wakeup timer clock is ckspre
<i>WAKEUP_CKSPRE_2</i> <i>EXP16</i>	RTC auto wakeup timer clock is ckspre and wakeup timer add 2exp16
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* RTC auto wakeup timer clock is ckspre */
```

```
ErrStatus error_status = rtc_wakeup_clock_set(WAKEUP_RTCK_DIV8);
```

rtc_wakeup_timer_set

The description of rtc_wakeup_timer_set is shown as below:

Table 3-1367. Function rtc_wakeup_timer_set

Function name	rtc_wakeup_timer_set
Function prototype	ErrStatus rtc_wakeup_timer_set(uint16_t wakeup_timer);
Function descriptions	set wakeup timer value
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_timer	wakeup timer value
<i>uint16_t</i>	0x0000-0xffff
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* set wakeup timer value */
```

```
ErrStatus error_status = rtc_wakeup_timer_set(0XFFEE);
```

rtc_wakeup_timer_get

The description of rtc_wakeup_timer_set is shown as below:

Table 3-1368. Function rtc_wakeup_timer_get

Function name	rtc_wakeup_timer_get
Function prototype	uint16_t rtc_wakeup_timer_get(void);
Function descriptions	set wakeup timer value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	0-0xFFFF

Example:

```
/* get wakeup timer value */
```

```
uint32_t wakeup_time = rtc_wakeup_timer_get();
```

rtc_smooth_calibration_config

The description of rtc_smooth_calibration_config is shown as below:

Table 3-1369. Function rtc_smooth_calibration_config

Function name	rtc_smooth_calibration_config
Function prototype	ErrStatus rtc_smooth_calibration_config(uint32_t window, uint32_t plus, uint32_t minus);
Function descriptions	configure RTC smooth calibration
Precondition	-
The called functions	-
Input parameter{in}	
window	select calibration window
RTC_CALIBRATION_WINDOW_32S	2exp20 RTCCLK cycles, 32s if RTCCLK = 32768 Hz
RTC_CALIBRATION_WINDOW_16S	2exp19 RTCCLK cycles, 16s if RTCCLK = 32768 Hz
RTC_CALIBRATION_WINDOW_8S	2exp18 RTCCLK cycles, 8s if RTCCLK = 32768 Hz
Input parameter{in}	
plus	add RTC clock or not

<i>RTC_CALIBRATION_PLUS_SET</i>	add one RTC clock every 2048 rtc clock
<i>RTC_CALIBRATION_PLUS_RESET</i>	no effect
Input parameter{in}	
minus	the RTC clock to minus during the calibration window(0x0 - 0xFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* configure RTC smooth calibration */
```

```
ErrStatus error_status;
```

```
error_status = rtc_smooth_calibration_config(RTC_CALIBRATION_WINDOW_32S,
RTC_CALIBRATION_PLUS_SET, 0x10);
```

rtc_interrupt_enable

The description of rtc_interrupt_enable is shown as below:

Table 3-1370. Function rtc_interrupt_enable

Function name	rtc_interrupt_enable
Function prototype	void rtc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable specified RTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify which interrupt source to be enabled
<i>RTC_INT_TIMESTAMP</i>	timestamp interrupt
<i>RTC_INT_ALARM0</i>	Alarm0 interrupt
<i>RTC_INT_ALARM1</i>	Alarm1 interrupt
<i>RTC_INT_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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1371. Function rtc_interrupt_disable

Function name	rtc_interrupt_disable
Function prototype	void rtc_interrupt_disable(uint32_t interrupt);
Function descriptions	disble specified RTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify which RTC interrupt to disable
<i>RTC_INT_TIMESTAMP</i>	timestamp interrupt
<i>RTC_INT_ALARM0</i>	Alarm0 interrupt
<i>RTC_INT_ALARM1</i>	Alarm1 interrupt
<i>RTC_INT_TAMP_ALL</i>	All tamp interrupt
<i>RTC_INT_TAMP0</i>	Tamper0 detection interrupt
<i>RTC_INT_TAMP1</i>	Tamper1 detection interrupt
<i>RTC_INT_TAMP2</i>	Tamper2 detection interrupt
<i>RTC_INT_WAKEUP</i>	wakeup timer interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disble specified RTC interrupt */
```

```
rtc_interrupt_disable(RTC_INT_TAMP0);
```

rtc_flag_get

The description of rtc_flag_get is shown as below:

Table 3-1372. 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
<i>RTC_FLAG_SCP</i>	smooth calibration pending flag
<i>RTC_FLAG_TP2</i>	RTC tamper 2 detected flag
<i>RTC_FLAG_TP1</i>	RTC tamper 1 detected flag
<i>RTC_FLAG_TP0</i>	RTC tamper 0 detected flag
<i>RTC_FLAG_TSOVR</i>	time-stamp overflow event flag
<i>RTC_FLAG_TS</i>	time-stamp event flag
<i>RTC_FLAG_ALARM0</i>	Alarm0 event flag
<i>RTC_FLAG_ALARM1</i>	Alarm1 event flag
<i>RTC_FLAG_WT</i>	wakeup timer occurs flag
<i>RTC_FLAG_INIT</i>	init mode event flag
<i>RTC_FLAG_RSYN</i>	time and date registers synchronized event flag
<i>RTC_FLAG_YCM</i>	year parameter configured event flag
<i>RTC_FLAG_SOP</i>	shift function operation pending flag
<i>RTC_FLAG_ALARM0</i> <i>W</i>	Alarm0 written available flag
<i>RTC_FLAG_ALARM1</i> <i>W</i>	Alarm1 written available flag
<i>RTC_FLAG_WTW</i>	wakeup timer can be written flag
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 3-1373. Function `rtc_flag_clear`

Function name	<code>rtc_flag_clear</code>
Function prototype	<code>void rtc_flag_clear(uint32_t flag);</code>
Function descriptions	clear specified flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag to clear
<i>RTC_FLAG_TP1</i>	RTC tamper 1 detected flag
<i>RTC_FLAG_TP0</i>	RTC tamper 0 detected flag
<i>RTC_FLAG_TSOVR</i>	time-stamp overflow event flag

<i>RTC_FLAG_TS</i>	time-stamp event flag
<i>RTC_FLAG_WT</i>	wakeup timer occurs flag
<i>RTC_FLAG_ALARM0</i>	Alarm0 event flag
<i>RTC_FLAG_ALARM1</i>	Alarm1 event flag
<i>RTC_FLAG_RSYN</i>	time and date registers synchronized event flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear time-stamp event flag */
rtc_flag_clear(RTC_FLAG_TS);
```

3.38. RTDEC

The real-time decryption (RTDEC) allows to decrypt in real-time according to information of the read request address. RTDEC can configure four independent and different encrypted areas. And each area has the option of execute-only or execute-never enforcement to choose. RTDEC is usually used with an external flash. The RTDEC registers are listed in chapter [3.38.1](#), the RTDEC firmware functions are introduced in chapter [3.38.2](#).

3.38.1. Descriptions of Peripheral registers

RTDEC registers are listed in the table shown as below:

Table 3-1374. RTDEC Registers

Registers	Descriptions
RTDEC_ARE_CFG	area configuration register
RTDEC_ARE_SADDR	area start address register
RTDEC_ARE_EADDR	area end address register
RTDEC_ARE_NONCE0	area random number register 0
RTDEC_ARE_NONCE1	area random number register 1
RTDEC_ARE_KEY0	area key register 0
RTDEC_ARE_KEY1	area key register 1
RTDEC_ARE_KEY2	area key register 2
RTDEC_ARE_KEY3	area key register 3
RTDEC_INTF	interrupt flag register
RTDEC_INTC	interrupt flag clear register
RTDEC_INTEN	interrupt enable register

3.38.2. Descriptions of Peripheral functions

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

Table 3-1375. RTDEC firmware function

Function name	Function description
rtdec_deinit	reset RTDEC
rtdec_struct_para_init	initialize the parameters of RTDEC struct with the default values
rtdec_init	initialize RTDEC
rtdec_config	configure RTDEC area data attribute
rtdec_lock	configure RTDEC key or register lock
rtdec_addr_init	initialize RTDEC area address
rtdec_nonce_init	initialize RTDEC nonce, nonce follows little endian format
rtdec_key_init	initialize RTDEC key, key follows little endian format
rtdec_key_crc_get	get crc value of RTDEC key data
rtdec_enable	enable RTDEC area
rtdec_disable	disable RTDEC area
rtdec_flag_get	get RTDEC error flag
rtdec_flag_clear	clear RTDEC error flag
rtdec_interrupt_enable	enable RTDEC interrupt
rtdec_interrupt_disable	disable RTDEC interrupt
rtdec_interrupt_flag_get	get RTDEC interrupt flag
rtdec_interrupt_flag_clear	clear RTDEC interrupt flag

Structure rtdec_parameter_struct

Table 3-1376. Structure rtdec_parameter_struct

Member name	Function description
access_mode	area access mode
key_crc	key crc value
fw_version	area firmware version
key	area key bits
nonce	area nonce bits
start_addr	area start address
end_addr	area end address

rtdec_deinit

The description of rtdec_deinit is shown as below:

Table 3-1377. Function rtdec_deinit

Function name	rtdec_deinit
Function prototype	void rtdec_deinit(uint32_t rtdec_periph);

Function descriptions	reset RTDEC
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset RTDEC */
rtdec_deinit(RTDEC0);
```

rtdec_struct_para_init

The description of rtdec_init is shown as below:

Table 3-1378. Function rtdec_struct_para_init

Function name	rtdec_struct_para_init
Function prototype	void rtdec_struct_para_init(rtdec_parameter_struct *rtdec_struct);
Function descriptions	initialize the parameters of RTDEC struct with default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
rtdec_struct	the initialized struct rtdec_parameter_struct pointer, refer to Table 3-1376. Structure rtdec_parameter_struct
Return value	
-	-

Example:

```
rtdec_parameter_struct rtdec_struct;

/* initialize rtdec_struct parameter with default values */
rtdec_struct_para_init(&rtdec_struct);
```

rtdec_init

The description of rtdec_init is shown as below:

Table 3-1379. Function rtdec_init

Function name	rtdec_init
Function prototype	ErrStatus rtdec_init(uint32_t rtdec_periph, uint32_t rtdec_area, rtdec_parameter_struct *rtdec_struct);
Function descriptions	initialize RTDEC
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
RTDECx	x = 0, 1
rtdec_area	RTDEC area
RTDEC_AREAx	x = 0, 1, 2, 3
rtdec_struct	RTDEC parameter initialization struct members of the structure, refer to Table 3-1376. Structure rtdec_parameter_struct
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
rtdec_parameter_struct rtdec_struct;
```

```
/* configure RTDEC area0 */
```

```
rtdec_init(RTDEC0, RTDEC_AREA0, &rtdec_struct);
```

rtdec_config

The description of rtdec_config is shown as below:

Table 3-1380. Function rtdec_config

Function name	rtdec_config
Function prototype	void rtdec_config(uint32_t rtdec_periph, uint32_t rtdec_area, uint8_t access_mode, uint16_t firmware_version);
Function descriptions	configure RTDEC area data attribute
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
RTDECx	x = 0, 1
rtdec_area	RTDEC area
RTDEC_AREAx	x = 0, 1, 2, 3
access_mode	allowed access mode of data
RTDEC_MODE_CODE	code / instruction access only

<code>_ACCESS</code>	
<code>RTDEC_MODE_DATA_ACCESS</code>	data access only
<code>RTDEC_MODE_BOTH_ACCESS</code>	code and data access
<code>firmware_version</code>	16-bit number, version of data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure RTDEC area0 */
```

```
rtdec_config(RTDEC0, RTDEC_AREA0, RTDEC_MODE_CODE_ACCESS, 0x1234);
```

rtdec_lock

The description of `rtdec_lock` is shown as below:

Table 3-1381. Function `rtdec_lock`

Function name	<code>rtdec_lock</code>
Function prototype	<code>void rtdec_lock(uint32_t rtdec_periph, uint32_t rtdec_area, uint32_t lock_type);</code>
Function descriptions	configure RTDEC key or register lock
Precondition	-
The called functions	-
Input parameter{in}	
<code>rtdec_periph</code>	RTDEC peripheral
<code>RTDECx</code>	x = 0, 1
<code>rtdec_area</code>	RTDEC area
<code>RTDEC_AREAx</code>	x = 0, 1, 2, 3
<code>lock_type</code>	key lock or register lock
<code>RTDEC_ARE_CFG_LK</code>	register lock
<code>RTDEC_ARE_K_LK</code>	key lock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock RTDEC area0 key */
```

```
rtdec_lock(RTDEC0, RTDEC_AREA0, RTDEC_ARE_K_LK);
```

rtdec_addr_init

The description of rtdec_addr_init is shown as below:

Table 3-1382. Function rtdec_addr_init

Function name	rtdec_addr_init
Function prototype	void rtdec_addr_init(uint32_t rtdec_periph, uint32_t rtdec_area, uint32_t saddr, uint32_t eaddr);
Function descriptions	initialize RTDEC area address
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
rtdec_area	RTDEC area
<i>RTDEC_AREAx</i>	x = 0, 1, 2, 3
saddr	area start address, the 4 MSB bits and the 12 LSB bits are ignored
eaddr	area end address, the 4 MSB bits and the 12 LSB bits are ignored
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
uint32_t start_addr = 0x90000000;
uint32_t end_addr = 0x90001000;
/* initialize RTDEC area0 address */
rtdec_addr_init(RTDEC0, RTDEC_AREA0, start_addr, end_addr);
```

rtdec_nonce_init

The description of rtdec_nonce_init is shown as below:

Table 3-1383. Function rtdec_nonce_init

Function name	rtdec_nonce_init
Function prototype	void rtdec_nonce_init(uint32_t rtdec_periph, uint32_t rtdec_area, uint32_t* nonce);
Function descriptions	initialize RTDEC nonce, nonce follows little endian format
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1

rtdec_area	RTDEC area
<i>RTDEC_AREAx</i>	x = 0, 1, 2, 3
nonce	an array containing 64-bit nonce data, little endian format
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
uint32_t nonce[2] = {0x89ABCDEF, 0x01234567};
```

```
/* initialize RTDEC area0 nonce */
```

```
rtdec_addr_init(RTDEC0, RTDEC_AREA0, nonce);
```

rtdec_key_init

The description of rtdec_key_init is shown as below:

Table 3-1384. Function rtdec_key_init

Function name	rtdec_key_init
Function prototype	void rtdec_key_init(uint32_t rtdec_periph, uint32_t rtdec_area, uint32_t* key);
Function descriptions	initialize RTDEC key, key follows little endian format
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
rtdec_area	RTDEC area
<i>RTDEC_AREAx</i>	x = 0, 1, 2, 3
key	an array containing 128-bit key data, little endian format
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
uint32_t rtdec_key[4] = {0x01234567, 0x89ABCDEF, 0x01234567, 0x89ABCDEF};
```

```
/* initialize RTDEC area0 key */
```

```
rtdec_key_init(RTDEC0, RTDEC_AREA0, rtdec_key);
```

rtdec_key_crc_get

The description of rtdec_key_crc_get is shown as below:

Table 3-1385. Function rtdec_key_crc_get

Function name	rtdec_key_crc_get
Function prototype	uint8_t rtdec_key_crc_get(uint32_t rtdec_periph, uint32_t rtdec_area);
Function descriptions	get crc value of RTDEC key data
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
rtdec_area	RTDEC area
<i>RTDEC_AREAx</i>	x = 0, 1, 2, 3
Output parameter{out}	
-	-
Return value	
uint8_t	CRC value

Example:

```
uint8_t key_crc;
```

```
/* get RTDEC area0 key crc */
```

```
key_crc = rtdec_key_crc_get(RTDEC0, RTDEC_AREA0);
```

rtdec_enable

The description of rtdec_enable is shown as below:

Table 3-1386. Function rtdec_enable

Function name	rtdec_enable
Function prototype	void rtdec_enable(uint32_t rtdec_periph, uint32_t rtdec_area);
Function descriptions	enable RTDEC area
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
rtdec_area	RTDEC area
<i>RTDEC_AREAx</i>	x = 0, 1, 2, 3
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable RTDEC area0 */
```

```
rtdec_enable(RTDEC0, RTDEC_AREA0);
```

rtdec_disable

The description of rtdec_disable is shown as below:

Table 3-1387. Function rtdec_disable

Function name	rtdec_disable
Function prototype	void rtdec_disable(uint32_t rtdec_periph, uint32_t rtdec_area);
Function descriptions	disable RTDEC area
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
rtdec_area	RTDEC area
<i>RTDEC_AREAx</i>	x = 0, 1, 2, 3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTDEC area0 */
```

```
rtdec_disable(RTDEC0, RTDEC_AREA0);
```

rtdec_flag_get

The description of rtdec_flag_get is shown as below:

Table 3-1388. Function rtdec_flag_get

Function name	rtdec_flag_get
Function prototype	FlagStatus rtdec_flag_get(uint32_t rtdec_periph, uint32_t flag);
Function descriptions	get RTDEC error flag
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1

flag	error flag
<i>RTDEC_FLAG_SEC_ERROR</i>	security error flag
<i>RTDEC_FLAG_MODE_ERROR</i>	access mode error flag
<i>RTDEC_FLAG_KEY_ERROR</i>	key error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

FlagStatus status;

/* get RTDEC error flag */

status = rtdec_flag_get(RTDEC0, RTDEC_FLAG_SEC_ERROR);

rtdec_flag_clear

The description of rtdec_flag_clear is shown as below:

Table 3-1389. Function rtdec_flag_clear

Function name	rtdec_flag_clear
Function prototype	void rtdec_flag_clear(uint32_t rtdec_periph, uint32_t flag);
Function descriptions	clear RTDEC error flag
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
flag	error flag
<i>RTDEC_FLAG_SEC_ERROR</i>	security error flag
<i>RTDEC_FLAG_MODE_ERROR</i>	access mode error flag
<i>RTDEC_FLAG_KEY_ERROR</i>	key error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear RTDEC error flag */
```

```
rtdec_flag_clear(RTDEC0, RTDEC_FLAG_SEC_ERROR);
```

rtdec_interrupt_enable

The description of rtdec_interrupt_enable is shown as below:

Table 3-1390. Function rtdec_interrupt_enable

Function name	rtdec_interrupt_enable
Function prototype	void rtdec_interrupt_enable(uint32_t rtdec_periph, uint32_t interrupt);
Function descriptions	enable RTDEC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
interrupt	interrupt type
<i>RTDEC_INT_SEC</i>	security error interrupt
<i>RTDEC_INT_MODE</i>	access mode error interrupt
<i>RTDEC_INT_KEY</i>	key error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable security error interrupt */
```

```
rtdec_interrupt_enable(RTDEC0, RTDEC_INT_SEC);
```

rtdec_interrupt_disable

The description of rtdec_interrupt_disable is shown as below:

Table 3-1391. Function rtdec_interrupt_disable

Function name	rtdec_interrupt_disable
Function prototype	void rtdec_interrupt_disable(uint32_t rtdec_periph, uint32_t interrupt);
Function descriptions	disable RTDEC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
interrupt	interrupt type

<i>RTDEC_INT_SEC</i>	security error interrupt
<i>RTDEC_INT_MODE</i>	access mode error interrupt
<i>RTDEC_INT_KEY</i>	key error interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable security error interrupt */
```

```
rtdec_interrupt_disable(RTDEC0, RTDEC_INT_SEC);
```

rtdec_interrupt_flag_get

The description of `rtdec_interrupt_flag_get` is shown as below:

Table 3-1392. Function `rtdec_interrupt_flag_get`

Function name	<code>rtdec_interrupt_flag_get</code>
Function prototype	<code>FlagStatus rtdec_interrupt_flag_get(uint32_t rtdec_periph, uint32_t int_flag);</code>
Function descriptions	get RTDEC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
int_flag	interrupt flag
<i>RTDEC_INT_FLAG_SEC_ERROR</i>	security error interrupt flag
<i>RTDEC_INT_FLAG_MODE_ERROR</i>	access mode error interrupt flag
<i>RTDEC_INT_FLAG_KEY_ERROR</i>	key error interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus status;
```

```
/* get RTDEC interrupt flag */
```

```
status = rtdec_interrupt_flag_get(RTDEC0, RTDEC_INT_FLAG_SEC_ERROR);
```

rtdec_interrupt_flag_clear

The description of rtdec_interrupt_flag_clear is shown as below:

Table 3-1393. Function rtdec_interrupt_flag_clear

Function name	rtdec_interrupt_flag_clear
Function prototype	void rtdec_interrupt_flag_clear(uint32_t rtdec_periph, uint32_t int_flag);
Function descriptions	clear RTDEC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
rtdec_periph	RTDEC peripheral
<i>RTDECx</i>	x = 0, 1
flag	interrupt flag
<i>RTDEC_INT_FLAG_SEC_ERR</i>	security error interrupt flag
<i>RTDEC_INT_FLAG_MODE_ERR</i>	access mode error interrupt flag
<i>RTDEC_INT_FLAG_KEY_ERR</i>	key error interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear RTDEC interrupt flag */
```

```
rtdec_interrupt_flag_clear(RTDEC0, RTDEC_INT_FLAG_SEC_ERROR);
```

3.39. SAI

The serial audio interface(SAI) is designed to target a wide range of commonly used audio protocols, both in mono and stereo modes, such as I2S, PCM/DSP, AC'97, LSB or MSB justified and TDM. The SAI registers are listed in chapter [3.39.1](#), the SAI firmware functions are introduced in chapter [3.39.2](#).

3.39.1. Descriptions of Peripheral registers

SAI registers are listed in the table shown as below:

Table 3-1394. SAI Registers

Registers	Descriptions
SAI_SYNCFG	SAI synchronize configuration register

Registers	Descriptions
SAI_B0CFG0	SAI block 0 configuration register0
SAI_B0CFG1	SAI block 0 configuration register1
SAI_B0FCFG	SAI block 0 frame configuration register
SAI_B0SCFG	SAI block 0 slot configuration register
SAI_B0INTEN	SAI block 0 interrupt enable register
SAI_B0STAT	SAI block 0 status register
SAI_B0INTC	SAI block 0 interrupt flag clear register
SAI_B0DATA	SAI block 0 data register
SAI_B1CFG0	SAI block 1 configuration register0
SAI_B1CFG1	SAI block 1 configuration register1
SAI_B1FCFG	SAI block 1 frame configuration register
SAI_B1SCFG	SAI block 1 slot configuration register
SAI_B1INTEN	SAI block 1 interrupt enable register
SAI_B1STAT	SAI block 1 status register
SAI_B1INTC	SAI block 1 interrupt flag clear register
SAI_B1DATA	SAI block 1 data register
SAI_PDMCTL	SAI PDM control register
SAI_PDMCFG	SAI PDM configuration register

3.39.2. Descriptions of Peripheral functions

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

Table 3-1395. SAI firmware function

Function name	Function description
sai_deinit	reset SAI
sai_struct_para_init	initialize SAI parameter struct with the default values
sai_frame_struct_para_init	initialize SAI frame parameter struct with the default values
sai_slot_struct_para_init	initialize SAI slot parameter struct with the default values
sai_init	initialize SAI
sai_frame_init	initialize SAI frame
sai_slot_init	initialize SAI slot
sai_enable	sai enable
sai_disable	sai disable
sai_sdoutput_config	SAI serial data near inactive slot output management
sai_monomode_config	configure SAI mono mode
sai_companing_config	configure SAI companing mode
sai_mute_enable	enable SAI mute detected or mute send
sai_mute_disable	disable SAI mute detected or mute send
sai_mute_value_config	configure SAI mute value
sai_mute_count_config	configure SAI mute frame count
sai_data_transmit	SAI transmit data

Function name	Function description
sai_data_receive	SAI receive data
sai_fifo_status_get	get SAI fifo status
sai_fifo_flush	SAI fifo flush
sai_dma_enable	enable SAI dma
sai_dma_disable	disable SAI dma
sai_sync_input_config	synchronization input select
sai_sync_output_config	synchronization output select
sai_pdm_enable	enable SAI pdm mode
sai_pdm_disable	disable SAI pdm mode
sai_pdm_microphone_number_config	configure SAI pdm mode microphone number
sai_pdm_delay_config	configure SAI pdm mode microphone delay
sai_pdm_clk0_enable	enable SAI pdm mode clock line 0
sai_pdm_clk0_disable	disable SAI pdm mode clock line 0
sai_pdm_clk1_enable	enable SAI pdm mode clock line 1
sai_pdm_clk1_disable	disable SAI pdm mode clock line 1
sai_interrupt_enable	enable the SAI interrupt
sai_interrupt_disable	disable the SAI interrupt
sai_interrupt_flag_get	get SAI interrupt flag status
sai_interrupt_flag_clear	clear SAI interrupt flag status
sai_flag_get	get SAI flag status
sai_flag_clear	clear SAI flag status

Structure sai_parameter_struct

Table 3-1396. Structure sai_parameter_struct

Member name	Function description
operating_mode	operating mode
protocol	protocol selection
data_width	data width
shift_dir	shift direction
sample_edge	sampling clock edge
sync_mode	synchronization mode
output_drive	output Drive
clk_div_bypass	clock divider logic bypass
mclk_div	master clock divider ratio
mclk_oversampling	the master clock oversampling rate
mclk_enable	the master clock enable
fifo_threshold	FIFO threshold

Structure sai_frame_parameter_struct

Table 3-1397. Structure sai_frame_parameter_struct

Member name	Function description
frame_width	frame width
frame_sync_width	frame synchronization active width
frame_sync_function	frame synchronization function
frame_sync_polarity	frame synchronization active polarity
frame_sync_offset	frame synchronization offset

Structure sai_slot_parameter_struct

Table 3-1398. Structure sai_slot_parameter_struct

Member name	Function description
slot_number	slot number
slot_width	slot width
data_offset	data offset
slot_active	slot activation vector

Enum sai_fifo_state_enum

Table 3-1399. Enum sai_fifo_state_enum

Member name	Function description
FIFO_EMPTY	empty
FIFO_EMPTY_TO_1_4_FULL	empty < fifo_level <= 1/4_full
FIFO_1_4_FULL_TO_1_2_FULL	1/4_full < fifo_level <= 1/2_full
FIFO_1_2_FULL_TO_3_4_FULL	1/2_full < fifo_level <= 3/4_full
FIFO_3_4_FULL_TO_FULL	3/4_full < fifo_level < full
FIFO_FULL	full

sai_deinit

The description of sai_deinit is shown as below:

Table 3-1400. Function sai_deinit

Function name	sai_deinit
Function prototype	void sai_deinit(uint32_t sai_periph);
Function descriptions	reset SAI
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the SAI0 */
```

```
sai_deinit(SAI0);
```

sai_struct_para_init

The description of sai_struct_para_init is shown as below:

Table 3-1401. Function sai_struct_para_init

Function name	sai_struct_para_init
Function prototype	void sai_struct_para_init(sai_parameter_struct* initpara);
Function descriptions	initialize SAI parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
initpara	the initialization data needed to initialize SAI, refer to Table 3-1396. Structure sai_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of SAI structure */
```

```
sai_struct_para_init sai_init_struct;
```

```
sai_struct_para_init (&sai_init_struct);
```

sai_frame_struct_para_init

The description of sai_frame_struct_para_init is shown as below:

Table 3-1402. Function sai_frame_struct_para_init

Function name	sai_frame_struct_para_init
Function prototype	void sai_frame_struct_para_init(sai_frame_parameter_struct* initpara);
Function descriptions	initialize the parameter of SAI frame structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	

initpara	the initialization data needed to initialize SAI frame, refer to Table 3-1397. Structure sai_frame_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of SAI frame structure */
sai_frame_struct_para_init sai_frame_init_struct;
sai_frame_struct_para_init (&sai_frame_init_struct);
```

sai_slot_struct_para_init

The description of sai_slot_struct_para_init is shown as below:

Table 3-1403. Function sai_slot_struct_para_init

Function name	sai_slot_struct_para_init
Function prototype	void sai_slot_struct_para_init(sai_slot_parameter_struct* initpara);
Function descriptions	initialize the parameter of SAI slot structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
initpara	the initialization data needed to initialize SAI slot, refer to Table 3-1398. Structure sai_slot_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of SAI slot structure */
sai_slot_struct_para_init sai_slot_init_struct;
sai_slot_struct_para_init (&sai_slot_init_struct);
```

sai_init

The description of sai_init is shown as below:

Table 3-1404. Function sai_init

Function name	sai_init
Function prototype	void sai_init(uint32_t sai_periph, uint32_t block, sai_parameter_struct* initpara);

Function descriptions	initialize SAI
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
initpara	the initialization data needed to initialize SAI, refer to Table 3-1396. Structure sai_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize SAI frame */
sai_parameter_struct sai_init_structure;
sai_init_structure.operating_mode = SAI_MASTER_TRANSMITTER;
sai_init_structure.protocol = SAI_PROTOCOL_POLYMORPHIC;
sai_init_structure.data_width = SAI_DATAWIDTH_16BIT;
sai_init_structure.shift_dir = SAI_SHIFT_MSB;
sai_init_structure.sample_edge = SAI_SAMPEDGE_FALLING;
sai_init_structure.sync_mode = SAI_SYNCMODE_ASYNC;
sai_init_structure.output_drive = SAI_OUTPUT_WITH_SAIEN;
sai_init_structure.clk_div_bypass = SAI_CLKDIV_BYPASS_OFF;
sai_init_structure.mclk_div = SAI_MCLKDIV_2;
sai_init_structure.mclk_oversampling = SAI_MASTERCLK_OVERSAMP_256;
sai_init_structure.mclk_enable = SAI_MASTERCLK_DISABLE;
sai_init_structure.fifo_threshold = SAI_FIFOTH_QUARTER;
sai_init(SAI0, SAI_BLOCK0, sai_init_structure);

```

sai_frame_init

The description of sai_frame_init is shown as below:

Table 3-1405. Function sai_frame_init

Function name	sai_frame_init
Function prototype	void sai_frame_init(uint32_t sai_periph, uint32_t block, sai_frame_parameter_struct* initpara);
Function descriptions	initialize SAI frame
Precondition	-

The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
initpara	the initialization data needed to initialize SAI frame, refer to Table 3-1397 . Structure sai_frame_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize SAI frame */
sai_frame_parameter_struct sai_frame_init_structure;
sai_frame_init_structure.frame_width = 128;
sai_frame_init_structure.frame_sync_width = 1;
sai_frame_init_structure.frame_sync_function = SAI_FSFUNC_START;
sai_frame_init_structure.frame_sync_polarity = SAI_FSPOLR_LOW;
sai_frame_init_structure.frame_sync_offset = SAI_FSOST_BEGINNING;
sai_frame_init(SAI0, SAI_BLOCK0, sai_frame_init_structure);

```

sai_slot_init

The description of sai_slot_init is shown as below:

Table 3-1406. Function sai_slot_init

Function name	sai_slot_init
Function prototype	void sai_slot_init(uint32_t sai_periph, uint32_t block, sai_slot_parameter_struct* initpara);
Function descriptions	initialize SAI slot
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
initpara	the initialization data needed to initialize SAI slot, refer to Table 3-1398 .

	<u>Structure sai_slot_parameter_struct</u>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize SAI0 slot */
sai_slot_parameter_struct sai_slot_init_structure;
sai_slot_init_structure.slot_number = 8;
sai_slot_init_structure.slot_width = SAI_SLOTWIDTH_16BIT;
sai_slot_init_structure.data_offset = 0;
sai_slot_init_structure.slot_active = SAI_SLOT_ACTIVE_ALL;
void sai_slot_init(SAI0, SAI_BLOCK0, sai_slot_init_structure);
```

sai_enable

The description of sai_enable is shown as below:

Table 3-1407. Function sai_enable

Function name	sai_enable
Function prototype	void sai_enable(uint32_t sai_periph, uint32_t block);
Function descriptions	SAI enable
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SAI0 */
sai_enable(SAI0, SAI_BLOCK0);
```

sai_disable

The description of sai_disable is shown as below:

Table 3-1408. Function sai_disable

Function name	sai_disable
Function prototype	void sai_disable(uint32_t sai_periph, uint32_t block);
Function descriptions	SAI disable
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SAI0 */
```

```
sai_disable (SAI0, SAI_BLOCK0);
```

sai_sdoutput_config

The description of sai_sdoutput_config is shown as below:

Table 3-1409. Function sai_sdoutput_config

Function name	sai_disable
Function prototype	void sai_sdoutput_config(uint32_t sai_periph, uint32_t block, uint32_t sdout);
Function descriptions	SAI serial data near inactive slot output management
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
sdout	serial data output management mode selection
SAI_SDLIN_DRIVE	SD line output is driven entirely during the audio frame
SAI_SDLIN_RELEASE	SD line output is released near inactive slots

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SAI0 serial data output management mode selection */
sai_sdoutput_config (SAI0, SAI_BLOCK0, SAI_SDLINE_DRIVE);
```

sai_monomode_config

The description of sai_monomode_config is shown as below:

Table 3-1410. Function sai_monomode_config

Function name	sai_monomode_config
Function prototype	void sai_monomode_config(uint32_t sai_periph, uint32_t block, uint32_t mono);
Function descriptions	configure SAI mono mode
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
mono	stereo and mono mode selection
SAI_STEREO_MODE	stereo mode
SAI_MONO_MODE	mono mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SAI0 mono mode */
sai_monomode_config(SAI0, SAI_BLOCK0, SAI_STEREO_MODE);
```

sai_companding_config

The description of sai_companding_config is shown as below:

Table 3-1411. Function sai_companding_config

Function name	sai_companding_config
Function prototype	void sai_companding_config(uint32_t sai_periph, uint32_t block, uint32_t compander, uint32_t complement);
Function descriptions	configure SAI companding mode
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
compander	compander mode
SAI_COMPANDER_OFF	no compansion applies
SAI_COMPANDER_ULAW	u-law algorithm
SAI_COMPANDER_ALAW	A-law algorithm
Input parameter{in}	
complement	complement mode.
SAI_COMPLEMENT_1S	data represented in 1's complement form
AI_COMPLEMENT_2S	data represented in 2's complement form
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SAI0 companding mode */
```

```
sai_companding_config(SAI0, SAI0_BLOCK0, SAI_MONO_MODE, SAI_COMPLEMENT_1S);
```

sai_mute_enable

The description of sai_mute_enable is shown as below:

Table 3-1412. Function sai_mute_enable

Function name	sai_mute_enable
Function prototype	void sai_mute_enable(uint32_t sai_periph, uint32_t block);

Function descriptions	SAI mute detected enable or mute send enable
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SAI0 mute */
```

```
sai_mute_enable(SAI0, SAI_BLOCK0);
```

sai_mute_disable

The description of sai_mute_disable is shown as below:

Table 3-1413. Function sai_mute_disable

Function name	sai_mute_disable
Function prototype	void sai_mute_disable(uint32_t sai_periph, uint32_t block);
Function descriptions	SAI mute detected disable or mute send disable
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SAI0 mute */
```

```
sai_mute_disable(SAI0, SAI_BLOCK0);
```

sai_mute_value_config

The description of sai_mute_value_config is shown as below:

Table 3-1414. Function sai_mute_value_config

Function name	sai_mute_value_config
Function prototype	void sai_mute_value_config(uint32_t sai_periph, uint32_t block, uint32_t value);
Function descriptions	configure SAI mute value
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
value	mute value
SAI_MUTESSENT_0	0 is sent via the Serial Data (SD) line when mute is on
SAI_MUTESSENT_LASTFRAME	If SLOTNB is less or equals to two, last frame is sent via the Serial Data (SD) line
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SAI0 mute value is 0*/
```

```
sai_mute_value_config(SAI0, SAI_BLOCK0, SAI_MUTESSENT_0);
```

sai_mute_count_config

The description of sai_mute_count_config is shown as below:

Table 3-1415. Function sai_mute_count_config

Function name	sai_mute_count_config
Function prototype	void sai_mute_count_config(uint32_t sai_periph, uint32_t block, uint32_t count);
Function descriptions	configure SAI mute frame count
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral

SAI/x	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
count	0~63, mute frame count
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SAI0 mute frame count 16 */
sai_mute_count_config (SAI0, SAI_BLOCK0, 16);
```

sai_data_transmit

The description of sai_data_transmit is shown as below:

Table 3-1416. Function sai_data_transmit

Function name	sai_data_transmit
Function prototype	void sai_data_transmit(uint32_t sai_periph, uint32_t block, uint32_t data);
Function descriptions	SAI transmit data
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
data	32-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SAI0 transmit data */
sai_data_transmit (SAI0, SAI_BLOCK0, sai0_send_array[send_n]);
```


sai_data_receive

The description of sai_data_receive is shown as below:

Table 3-1417. Function sai_data_receive

Function name	sai_data_receive
Function prototype	uint32_t sai_data_receive(uint32_t sai_periph, uint32_t block);
Function descriptions	SAI receive data
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Output parameter{out}	
-	-
Return value	
uint32_t	0x00000000-0xFFFFFFFF

Example:

```
/* SAI0 receive data */
uint32_t sai_receiver;

sai_receiver = sai_data_receive (SAI0);
```

sai_fifo_status_get

The description of sai_fifo_status_get is shown as below:

Table 3-1418. Function sai_fifo_status_get

Function name	sai_fifo_status_get
Function prototype	sai_fifo_state_enum sai_fifo_status_get(uint32_t sai_periph, uint32_t block);
Function descriptions	get SAI fifo status
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Output parameter{out}	

-	-
Return value	
sai_fifo_state_enum	fifo status, refer to Table 3-1399. Enum sai_fifo_state_enum

Example:

```
/* get SAI0 fifo status */
```

```
sai_fifo_state_enum fifo_state;
```

```
fifo_state =sai_fifo_status_get(SAI0, SAI_BLOCK0);
```

sai_fifo_flush

The description of sai_fifo_flush is shown as below:

Table 3-1419. Function sai_fifo_flush

Function name	sai_fifo_flush
Function prototype	void sai_fifo_flush(uint32_t sai_periph, uint32_t block);
Function descriptions	SAI fifo flush
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SAI0 fifo flush */
```

```
sai_fifo_flush(SAI0);
```

sai_dma_enable

The description of sai_dma_enable is shown as below:

Table 3-1420. Function sai_dma_enable

Function name	sai_dma_enable
Function prototype	void sai_dma_enable(uint32_t sai_periph, uint32_t block);
Function descriptions	enable SAI dma
Precondition	-

The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SAI dma */
```

```
sai_dma_enable (SAI0, SAI_BLOCK0);
```

sai_dma_disable

The description of sai_dma_disable is shown as below:

Table 3-1421. Function sai_dma_disable

Function name	sai_dma_disable
Function prototype	void sai_dma_disable(uint32_t sai_periph, uint32_t block);
Function descriptions	disable SAI dma
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SAI dma */
```

```
sai_dma_disable (SAI0, SAI_BLOCK0);
```

sai_sync_input_config

The description of sai_sync_input_config is shown as below:

Table 3-1422. Function sai_sync_input_config

Function name	sai_sync_input_config
Function prototype	void sai_sync_input_config(uint32_t sai_periph, uint32_t input);
Function descriptions	SAI synchronization input select
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Input parameter{in}	
input	specify which external SAI to be select for synchronization
SAI_SYNCINPUT_SAI0	SAI1 or SAI2 selects the synchronization coming from SAI0
SAI_SYNCINPUT_SAI1	SAI0 or SAI2 selects the synchronization coming from SAI1
SAI_SYNCINPUT_SAI2	SAI0 or SAI1 selects the synchronization coming from SAI2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SAI1 selects the synchronization coming from SAI0 */
```

```
sai_sync_input_config (SAI1, SAI_SYNCINPUT_SAI0);
```

sai_sync_output_config

The description of sai_sync_output_config is shown as below:

Table 3-1423. Function sai_sync_output_config

Function name	sai_sync_output_config
Function prototype	void sai_sync_output_config(uint32_t sai_periph, uint32_t output);
Function descriptions	SAI synchronization output select
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Input parameter{in}	
output	specify which block to be used for further synchronization for others SAI
SAI_SYNCOUTPUT_OFF	no synchronization output signals

<i>SAI_SYNCOUTPUT_BLOCK0</i>	block 0 used for further synchronization for others SAI
<i>SAI_SYNCOUTPUT_BLOCK1</i>	block 1 used for further synchronization for others SAI
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SAI 0 block 0 used for further synchronization for others SAI */
```

```
sai_sync_output_config (SAI0, SAI_SYNCOUTPUT_BLOCK0);
```

sai_pdm_enable

The description of sai_pdm_enable is shown as below:

Table 3-1424. Function sai_pdm_enable

Function name	sai_pdm_enable
Function prototype	void sai_pdm_enable(uint32_t sai_periph);
Function descriptions	enable SAI pdm mode
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SAI0 pdm mode */
```

```
sai_pdm_enable(SAI0);
```

sai_pdm_disable

The description of sai_pdm_disable is shown as below:

Table 3-1425. Function sai_pdm_disable

Function name	sai_pdm_disable
Function prototype	void sai_pdm_disable(uint32_t sai_periph);
Function descriptions	disable SAI pdm mode

Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SAI0 pdm mode */
sai_pdm_disable(SAI0);
```

sai_pdm_microphone_number_config

The description of sai_pdm_microphone_num_config is shown as below:

Table 3-1426. Function sai_pdm_microphone_num_config

Function name	sai_pdm_microphone_num_config
Function prototype	void sai_pdm_microphone_num_config(uint32_t sai_periph, uint32_t microphonenum);
Function descriptions	configure SAI pdm mode microphone number
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Input parameter{in}	
microphonenum	2, 4, 6 or 8, select microphones number
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SAI0 pdm mode microphone number 2 */
sai_pdm_microphone_num_config(SAI0, 2);
```

sai_pdm_delay_config

The description of sai_pdm_delay_config is shown as below:

Table 3-1427. Function sai_pdm_delay_config

Function name	sai_pdm_delay_config
Function prototype	void sai_pdm_delay_config(uint32_t sai_periph, uint32_t microphone, uint32_t delay);
Function descriptions	configure SAI pdm mode microphone delay
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Input parameter{in}	
microphone	
SAI_PDM_MICROPHO NE0_L	microphone 0 channel left
SAI_PDM_MICROPHO NE0_R	microphone 0 channel right
SAI_PDM_MICROPHO NE1_L	microphone 1 channel left
SAI_PDM_MICROPHO NE1_R	microphone 1 channel right
SAI_PDM_MICROPHO NE2_L	microphone 2 channel left
SAI_PDM_MICROPHO NE2_R	microphone 2 channel right
SAI_PDM_MICROPHO NE3_L	microphone 3 channel left
SAI_PDM_MICROPHO NE3_R	microphone 3 channel right
Input parameter{in}	
delay	0~7, the microphone data flow delay period
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SAI pdm mode microphone 0 left delay 1 */
sai_pdm_delay_config(SAI0, SAI_PDM_MICROPHONE0_L, 1);
```

sai_pdm_clk0_enable

The description of sai_pdm_clk0_enable is shown as below:

Table 3-1428. Function sai_pdm_clk0_enable

Function name	sai_pdm_clk0_enable
Function prototype	void sai_pdm_clk0_enable(uint32_t sai_periph);
Function descriptions	enable SAI pdm mode clock line 0
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SAI0 pdm mode clock line 0 */
sai_pdm_clk0_enable(SAI0);
```

sai_pdm_clk0_disable

The description of sai_pdm_clk0_disable is shown as below:

Table 3-1429. Function sai_pdm_clk0_disable

Function name	sai_pdm_clk0_disable
Function prototype	void sai_pdm_clk0_disable(uint32_t sai_periph);
Function descriptions	disable SAI pdm mode clock line 0
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SAI0 pdm mode clock line 0 */
sai_pdm_clk0_disable(SAI0);
```

sai_pdm_clk1_enable

The description of sai_pdm_clk1_enable is shown as below:

Table 3-1430. Function sai_pdm_clk1_enable

Function name	sai_pdm_clk1_enable
Function prototype	void sai_pdm_clk1_enable(uint32_t sai_periph);
Function descriptions	enable SAI pdm mode clock line 1
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SAI0 pdm mode clock line 1 */
sai_pdm_clk1_enable(SAI0);
```

sai_pdm_clk1_disable

The description of sai_pdm_clk1_disable is shown as below:

Table 3-1431. Function sai_pdm_clk1_disable

Function name	sai_pdm_clk1_disable
Function prototype	void sai_pdm_clk1_disable(uint32_t sai_periph);
Function descriptions	disable SAI pdm mode clock line 1
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SAI0 pdm mode clock line 1 */
sai_pdm_clk1_disable(SAI0);
```

sai_interrupt_enable

The description of sai_interrupt_enable is shown as below:

Table 3-1432. Function sai_interrupt_enable

Function name	sai_interrupt_enable
Function prototype	void sai_interrupt_enable(uint32_t sai_periph, uint32_t block, uint32_t interrupt);
Function descriptions	enable the SAI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAI/x	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
interrupt	specify which interrupt to enable
SAI_INT_OUERR	FIFO overrun or underrun interrupt enable
SAI_INT_MTDET	mute detection interrupt enable
SAI_INT_ERRCK	error clock interrupt enable
SAI_INT_FFREQ	FIFO request interrupt enable
SAI_INT_ACNRDY	audio codec not ready interrupt enable
SAI_INT_FSADET	frame synchronization advanced detection interrupt enable
SAI_INT_FSPDET	frame synchronization postpone detection interrupt enable
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* enable SAI0 block0 mute detection interrupt */
```

```
sai_interrupt_enable(SAI0, SAI_BLOCK0, SAI_INT_MTDET);
```

sai_interrupt_disable

The description of sai_interrupt_disable is shown as below:

Table 3-1433. Function sai_interrupt_disable

Function name	sai_interrupt_disable
Function prototype	void sai_interrupt_disable(uint32_t sai_periph, uint32_t block, uint32_t interrupt);
Function descriptions	disable the SAI interrupt
Precondition	-
The called functions	-
Input parameter{in}	

sai_periph	SAI peripheral
<i>SAIx</i>	(x=0,1,2)
Input parameter{in}	
block	SAI block
<i>SAI_BLOCKx</i>	(x=0,1)
Input parameter{in}	
interrupt	specify which interrupt to enable
<i>SAI_INT_OUERR</i>	FIFO overrun or underrun interrupt enable
<i>SAI_INT_MTDET</i>	mute detection interrupt enable
<i>SAI_INT_ERRCK</i>	error clock interrupt enable
<i>SAI_INT_FFREQ</i>	FIFO request interrupt enable
<i>SAI_INT_ACNRDY</i>	audio codec not ready interrupt enable
<i>SAI_INT_FSADET</i>	frame synchronization advanced detection interrupt enable
<i>SAI_INT_FSPDET</i>	frame synchronization postpone detection interrupt enable
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* disable SAI0 block0 mute detection interrupt */
```

```
sai_interrupt_disable(SAI0, SAI_BLOCK0, SAI_INT_MTDET);
```

sai_interrupt_flag_get

The description of sai_interrupt_flag_get is shown as below:

Table 3-1434. Function sai_interrupt_flag_get

Function name	sai_interrupt_flag_get
Function prototype	FlagStatus sai_interrupt_flag_get(uint32_t sai_periph, uint32_t block, uint32_t interrupt);
Function descriptions	get the SAI interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
<i>SAIx</i>	(x=0,1,2)
Input parameter{in}	
block	SAI block
<i>SAI_BLOCKx</i>	(x=0,1)
Input parameter{in}	
interrupt	specify which interrupt flag to get
<i>SAI_FLAG_OUERR</i>	FIFO overrun or underrun interrupt flag

<i>SAI_FLAG_MTDET</i>	mute detection interrupt flag
<i>SAI_FLAG_ERRCK</i>	error clock interrupt enable
<i>SAI_FLAG_FFREQ</i>	FIFO request interrupt flag
<i>SAI_FLAG_ACNRDY</i>	audio codec not ready interrupt flag
<i>SAI_FLAG_FSADET</i>	frame synchronization advanced detection interrupt flag
<i>SAI_FLAG_FSPDET</i>	frame synchronization postpone detection interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SAI0 block0 intrerrupt flag status*/
```

```
FlagStatus flag_state = RESET;
```

```
flag_state = sai_interrupt_flag_get(SAI0, SAI_BLOCK0, SAI_FLAG_MTDET);
```

sai_interrupt_flag_clear

The description of sai_interrupt_flag_clear is shown as below:

Table 3-1435. Function sai_interrupt_flag_clear

Function name	sai_interrupt_flag_clear
Function prototype	void sai_interrupt_flag_clear(uint32_t sai_periph, uint32_t block, uint32_t interrupt);
Function descriptions	clear the SAI interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
<i>SAIx</i>	(x=0,1,2)
Input parameter{in}	
block	SAI block
<i>SAI_BLOCKx</i>	(x=0,1)
Input parameter{in}	
interrupt	specify which interrupt flag to clear
<i>SAI_FLAG_OUERR</i>	FIFO overrun or underrun interrupt flag
<i>SAI_FLAG_MTDET</i>	mute detection interrupt flag
<i>SAI_FLAG_ERRCK</i>	error clock interrupt enable
<i>SAI_FLAG_FFREQ</i>	FIFO request interrupt flag
<i>SAI_FLAG_ACNRDY</i>	audio codec not ready interrupt flag
<i>SAI_FLAG_FSADET</i>	frame synchronization advanced detection interrupt flag
<i>SAI_FLAG_FSPDET</i>	frame synchronization postpone detection interrupt flag

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear SAI0 block 0 error clock interrupt flag */
```

```
sai_flag_clear (SAI0, SAI_BLOCK0, SAI_FLAG_ERRCK);
```

sai_flag_get

The description of sai_flag_get is shown as below:

Table 3-1436. Function sai_flag_get

Function name	sai_interrupt_flag_get
Function prototype	FlagStatus sai_flag_get(uint32_t sai_periph, uint32_t block, uint32_t flag);
Function descriptions	get the SAI flag
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
interrupt	specify which flag to get
SAI_FLAG_OUERR	FIFO overrun or underrun flag
SAI_FLAG_MTDET	mute detection flag
SAI_FLAG_ERRCK	error clock enable
SAI_FLAG_FFREQ	FIFO request flag
SAI_FLAG_ACNRDY	audio codec not ready flag
SAI_FLAG_FSADET	frame synchronization advanced detection flag
SAI_FLAG_FSPDET	frame synchronization postpone detection flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SAI0 block0 flag status*/
```

```
FlagStatus flag_state = RESET;
```

```
flag_state = sai_flag_get(SAI0, SAI_BLOCK0, SAI_FLAG_MTDET);
```

sai_flag_clear

The description of sai_flag_clear is shown as below:

Table 3-1437. Function sai_flag_clear

Function name	sai_flag_clear
Function prototype	void sai_flag_clear(uint32_t sai_periph, uint32_t block, uint32_t flag);
Function descriptions	clear the SAI flag
Precondition	-
The called functions	-
Input parameter{in}	
sai_periph	SAI peripheral
SAIx	(x=0,1,2)
Input parameter{in}	
block	SAI block
SAI_BLOCKx	(x=0,1)
Input parameter{in}	
interrupt	specify which flag to clear
SAI_FLAG_OUERR	FIFO overrun or underrun flag
SAI_FLAG_MTDET	mute detection flag
SAI_FLAG_ERRCK	error clock flag
SAI_FLAG_ACNRDY	audio codec not ready flag
SAI_FLAG_FSADET	frame synchronization advanced detection flag
SAI_FLAG_FSPDET	frame synchronization postpone detection flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear SAI0 block 0 error clock flag */
```

```
sai_flag_clear (SAI0, SAI_BLOCK0, SAI_FLAG_ERRCK);
```

3.40. SDIO

The secure digital input/output interface (SDIO) defines the SD/SD I/O / e•MMC card host interface, which provides command/data transfer between the AHB system bus and SD memory cards, SD I/O cards and embedded Multimedia Card (e•MMC). The SDIO registers are listed in chapter [3.40.1](#), the SDIO firmware functions are introduced in chapter [3.40.2](#).

3.40.1. Descriptions of Peripheral registers

SDIO registers are listed in the table shown as below:

Table 3-1438. SDIO Registers

Registers	Descriptions
SDIO_PWRCTL	Power control register
SDIO_CLKCTL	Clock control register
SDIO_CMDAGMT	Command argument register
SDIO_CMDCTL	Command control register
SDIO_RSPCMDIDX	Command index response register
SDIO_RESPx x=0..3	Response register
SDIO_DATATO	Data timeout register
SDIO_DATALEN	Data length register
SDIO_DATACTL	Data control register
SDIO_DATACNT	Data counter register
SDIO_STAT	Status register
SDIO_INTC	Interrupt clear register
SDIO_INTEN	Interrupt enable register
SDIO_ACKTO	ACK timeout register
SDIO_FIFO	FIFO data register
SDIO_IDMACTL	IDMA control register
SDIO_IDMASIZE	IDMA buffer size register
SDIO_IDMAADDR0	IDMA buffer 0 base address register
SDIO_IDMAADDR1	IDMA buffer 1 base address register

3.40.2. Descriptions of Peripheral functions

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

Table 3-1439. SDIO firmware function

Function name	Function description
sdio_deinit	deinitialize the SDIO
sdio_clock_config	configure the SDIO clock
sdio_clock_receive_set	set receive clock
sdio_hardware_clock_enable	enable hardware clock control
sdio_hardware_clock_disable	disable hardware clock control
sdio_bus_mode_set	set different SDIO card bus mode
sdio_bus_speed_set	set SDIO bus speed
sdio_data_rate_set	set SDIO data rate
sdio_direction_polarity_set	set direction polarity of data and command
sdio_power_state_set	set the SDIO power state

Function name	Function description
sdio_power_state_get	get the SDIO power state
sdio_command_response_config	configure the command and response
sdio_wait_type_set	set the command state machine wait type
sdio_trans_start_enable	enable the CSM transfer command mode(treats the command as a data transfer command)
sdio_trans_start_disable	disable the CSM transfer command mode(treats the command as a data transfer command)
sdio_trans_stop_enable	enable the CSM stop command mode(treats the command as a data stop command)
sdio_trans_stop_disable	disable the CSM stop command mode(treats the command as a data stop command)
sdio_csm_enable	enable the CSM(command state machine)
sdio_csm_disable	disable the CSM(command state machine)
sdio_command_index_get	get the last response command index
sdio_response_get	get the response for the last received command
sdio_hold_enable	enable the DSM status hold
sdio_hold_disable	disable the DSM status hold
sdio_suspend_enable	enable the SDIO command suspend mode (the CSM treats the command as a suspend command or resume command)
sdio_suspend_disable	disable the SDIO command suspend mode (the CSM treats the command as a suspend command or resume command)
sdio_data_config	configure the data timeout, data length and data block size
sdio_data_transfer_config	configure the data transfer mode and direction
sdio_dsm_enable	enable the DSM(data state machine) for data transfer
sdio_dsm_disable	disable the DSM(data state machine)
sdio_data_write	write data(one word) to the transmit FIFO
sdio_data_read	read data(one word) from the receive FIFO
sdio_data_counter_get	get the number of remaining data bytes to be transferred to card
sdio_fifo_reset_enable	enable reset the FIFO
sdio_fifo_reset_disable	disable reset the FIFO
sdio_idma_set	set IDMA buffer mode and size
sdio_idma_buffer0_address_set	set IDMA buffer0 address
sdio_idma_buffer1_address_set	set IDMA buffer1 address
sdio_buffer_selection_get	get IDMA buffer selection
sdio_idma_buffer_select	select IDMA buffer
sdio_idma_enable	enable the IDMA request for SDIO
sdio_idma_disable	disable the IDMA request for SDIO
sdio_flag_get	get the flags state of SDIO
sdio_flag_clear	clear the pending flags of SDIO
sdio_interrupt_enable	enable the SDIO interrupt

Function name	Function description
sdio_interrupt_disable	disable the SDIO interrupt
sdio_interrupt_flag_get	get the interrupt flags state of SDIO
sdio_interrupt_flag_clear	clear the interrupt pending flags of SDIO
sdio_voltage_switch_enable	enable voltage switch
sdio_voltage_switch_disable	disable voltage switch
sdio_voltage_switch_sequence_enable	enable voltage switch sequence
sdio_voltage_switch_sequence_disable	disable voltage switch sequence
sdio_boot_mode_set	set boot mode
sdio_boot_ack_enable	enable DSM(data state machine) boot acknowledgment
sdio_boot_ack_disable	disable DSM(data state machine) boot acknowledgment
sdio_boot_acktimeout_set	set boot ACK timeout period
sdio_boot_enable	enable boot operation
sdio_boot_disable	disable boot operation

sdio_deinit

The description of sdio_deinit is shown as below:

Table 3-1440. Function sdio_deinit

Function name	sdio_deinit
Function prototype	void sdio_deinit(uint32_t sdio_periph);
Function descriptions	deinitialize the SDIO
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
sdio_periph	SDIO peripheral
SDIO0	use SDIO0
SDIO1	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the SDIO0 */
sdio_deinit(SDIO0);
```

sdio_clock_config

The description of sdio_clock_config is shown as below:

Table 3-1441. Function `sdio_clock_config`

Function name	<code>sdio_clock_config</code>
Function prototype	<code>void sdio_clock_config(uint32_t sdio_periph, uint32_t clock_edge, uint32_t clock_powersave, uint32_t clock_division);</code>
Function descriptions	configure the SDIO clock
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<code>SDIO0</code>	use SDIO0
<code>SDIO1</code>	use SDIO1
Input parameter{in}	
clock_edge	SDIO_CLK clock edge
<code>SDIO_SDIOLCKEDGE_RISING</code>	select the rising edge of the SDIOCLK to generate SDIO_CLK
<code>SDIO_SDIOLCKEDGE_FALLING</code>	select the falling edge of the SDIOCLK to generate SDIO_CLK
Input parameter{in}	
clock_powersave	SDIO_CLK clock dynamic switch on/off for power saving
<code>SDIO_CLOCKPWRSERVE_ENABLE</code>	SDIO_CLK closed when bus is idle
<code>SDIO_CLOCKPWRSERVE_DISABLE</code>	SDIO_CLK clock is always on
Input parameter{in}	
clock_division	clock division, 0 - 1023
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the SDIO0 clock */
```

```
sdio_clock_config(SDIO0, SDIO_SDIOLCKEDGE_RISING,
SDIO_CLOCKPWRSERVE_DISABLE, SD_CLK_DIV_TRANS_DSPEED);
```

sdio_clock_receive_set

The description of `sdio_clock_receive_set` is shown as below:

Table 3-1442. Function `sdio_clock_receive_set`

Function name	<code>sdio_clock_receive_set</code>
Function prototype	<code>void sdio_clock_receive_set(uint32_t sdio_periph, uint32_t clock_receive);</code>
Function descriptions	set receive clock

Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
clock_receive	receive clock selection
<i>SDIO_RECEIVECLOCK_INCLK</i>	select SDIO_IN_CLK clock
<i>SDIO_RECEIVECLOCK_CLKIN</i>	select SDIO_CLKIN clock
<i>SDIO_RECEIVECLOCK_FBCLK</i>	select SDIO_FB_CLK clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SDIO0 select SDIO_FB_CLK as receive clock */
```

```
sdio_clock_receive_set(SDIO0, SDIO_RECEIVECLOCK_FBCLK);
```

sdio_hardware_clock_enable

The description of sdio_hardware_clock_enable is shown as below:

Table 3-1443. Function sdio_hardware_clock_enable

Function name	sdio_hardware_clock_enable
Function prototype	void sdio_hardware_clock_enable(uint32_t sdio_periph);
Function descriptions	enable hardware clock control
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable hardware clock control */
```

```
sdio_hardware_clock_enable(SDIO0);
```

sdio_hardware_clock_disable

The description of sdio_hardware_clock_disable is shown as below:

Table 3-1444. Function sdio_hardware_clock_disable

Function name	sdio_hardware_clock_disable
Function prototype	void sdio_hardware_clock_disable(uint32_t sdio_periph);
Function descriptions	disable hardware clock control
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable hardware clock control */
```

```
sdio_hardware_clock_disable(SDIO0);
```

sdio_bus_mode_set

The description of sdio_bus_mode_set is shown as below:

Table 3-1445. Function sdio_bus_mode_set

Function name	sdio_bus_mode_set
Function prototype	void sdio_bus_mode_set(uint32_t sdio_periph, uint32_t bus_mode);
Function descriptions	set different SDIO card bus mode
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
bus_mode	SDIO card bus mode
<i>SDIO_BUSMODE_1BI</i>	1-bit SDIO card bus mode

<i>T</i>	
<i>SDIO_BUSMODE_4BI</i>	4-bit SDIO card bus mode
<i>T</i>	
<i>SDIO_BUSMODE_8BI</i>	8-bit SDIO card bus mode
<i>T</i>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO0 1-bit bus mode */
```

```
sdio_bus_mode_set(SDIO0, SDIO_BUSMODE_1BIT);
```

sdio_bus_speed_set

The description of sdio_bus_speed_set is shown as below:

Table 3-1446. Function sdio_bus_speed_set

Function name	sdio_bus_speed_set
Function prototype	void sdio_bus_speed_set(uint32_t sdio_periph, uint32_t bus_speed);
Function descriptions	set SDIO bus speed
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
bus_speed	SDIO card bus speed
<i>SDIO_BUSSPEED_LO</i> <i>W</i>	DS, HS, SDR12, SDR25 bus speed
<i>SDIO_BUSSPEED_HI</i> <i>GH</i>	SDR50, SDR104, DDR50 bus speed
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO0 bus speed mode */
```

```
sdio_bus_speed_set(SDIO0, SDIO_BUSSPEED_HIGH);
```

sdio_data_rate_set

The description of sdio_data_rate_set is shown as below:

Table 3-1447. Function sdio_data_rate_set

Function name	sdio_data_rate_set
Function prototype	void sdio_data_rate_set(uint32_t sdio_periph, uint32_t data_rate);
Function descriptions	set SDIO data rate
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
data_rate	SDIO card data rate
<i>SDIO_DATA_RATE_SDR</i>	SDR bus mode selected
<i>SDIO_DATA_RATE_DDR</i>	DDR bus mode selected
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO0 bus data rate */
```

```
sdio_data_rate_set(SDIO0, SDIO_DATA_RATE_DDR);
```

sdio_direction_polarity_set

The description of sdio_direction_polarity_set is shown as below:

Table 3-1448. Function sdio_direction_polarity_set

Function name	sdio_direction_polarity_set
Function prototype	void sdio_direction_polarity_set(uint32_t sdio_periph, uint32_t dirpl);
Function descriptions	set direction polarity of data and command
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	

dirpl	SDIO direction polarity
<i>SDIO_DIRECTION_SIGNAL_LOW</i>	direction signal is low, the voltage transceiver IOs driven as output
<i>SDIO_DIRECTION_SIGNAL_HIGH</i>	direction signal is high, the voltage transceiver IOs driven as output
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO0 bus direction polarity */
```

```
sdio_direction_polarity_set(SDIO, SDIO_DIRECTION_SIGNAL_HIGH);
```

sdio_power_state_set

The description of sdio_power_state_set is shown as below:

Table 3-1449. Function sdio_power_state_set

Function name	sdio_power_state_set
Function prototype	void sdio_power_state_set(uint32_t sdio_periph, uint32_t power_state);
Function descriptions	set the SDIO power state
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
power_state	SDIO power state
<i>SDIO_POWER_ON</i>	SDIO power on
<i>SDIO_POWER_CYCLE</i>	SDIO power cycle
<i>SDIO_POWER_OFF</i>	SDIO power off
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO0 power state */
```

```
sdio_power_state_set(SDIO0, SDIO_POWER_ON);
```

sdio_power_state_get

The description of sdio_power_state_get is shown as below:

Table 3-1450. Function sdio_power_state_get

Function name	sdio_power_state_get
Function prototype	uint32_t sdio_power_state_get(uint32_t sdio_periph);
Function descriptions	get the SDIO power state
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
uint32_t	SDIO_POWER_ON / SDIO_POWER_CYCLE / SDIO_POWER_OFF

Example:

```
/* get the SDIO0 power state */
```

```
uint32_t sdio_power_value;
```

```
sdio_power_value = sdio_power_state_get(SDIO0);
```

sdio_command_response_config

The description of sdio_command_response_config is shown as below:

Table 3-1451. Function sdio_command_response_config

Function name	sdio_command_response_config
Function prototype	void sdio_command_response_config(uint32_t sdio_periph, uint32_t cmd_index, uint32_t cmd_argument, uint32_t response_type);
Function descriptions	configure the command and response
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
cmd_index	command index, refer to the related specifications
Input parameter{in}	
cmd_argument	command argument, refer to the related specifications

Input parameter{in}	
response_type	response type
<i>SDIO_RESPONSETYPE_NO</i>	no response
<i>SDIO_RESPONSETYPE_SHORT</i>	short response
<i>SDIO_RESPONSETYPE_SHORT_NOCRC</i>	short response without CRC
<i>SDIO_RESPONSETYPE_LONG</i>	long response
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* CMD2(SD_CMD_ALL_SEND_CID) command response config*/
```

```
sdio_command_response_config(SDIO0, SD_CMD_ALL_SEND_CID, (uint32_t)0x0,
SDIO_RESPONSETYPE_LONG);
```

sdio_wait_type_set

The description of sdio_wait_type_set is shown as below:

Table 3-1452. Function sdio_wait_type_set

Function name	sdio_wait_type_set
Function prototype	void sdio_wait_type_set(uint32_t sdio_periph, uint32_t wait_type);
Function descriptions	set the command state machine wait type
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
wait_type	wait type
<i>SDIO_WAITTYPE_NO</i>	not wait interrupt
<i>SDIO_WAITTYPE_INTERRUPT</i>	wait interrupt
<i>SDIO_WAITTYPE_DATAEND</i>	wait the end of data transfer
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* set the SDIO0 the command state machine wait type */
```

```
sdio_wait_type_set(SDIO0, SDIO_WAITTYPE_NO);
```

sdio_trans_start_enable

The description of sdio_trans_start_enable is shown as below:

Table 3-1453. Function sdio_trans_start_enable

Function name	sdio_trans_start_enable
Function prototype	void sdio_trans_start_enable(uint32_t sdio_periph);
Function descriptions	enable the CSM transfer command mode(treats the command as a data transfer command)
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
SDIO0	use SDIO0
SDIO1	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CSM transfer command mode */
```

```
sdio_trans_start_enable(SDIO0);
```

sdio_trans_start_disable

The description of sdio_trans_start_disable is shown as below:

Table 3-1454. Function sdio_trans_start_disable

Function name	sdio_trans_start_disable
Function prototype	void sdio_trans_start_disable(uint32_t sdio_periph);
Function descriptions	disable the CSM transfer command mode(treats the command as a data transfer command)
Precondition	-
The called functions	-
Input parameter{in}	

sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CSM transfer command mode*/
```

```
sdio_trans_start_disable(SDIO0);
```

sdio_trans_stop_enable

The description of sdio_trans_stop_enable is shown as below:

Table 3-1455. Function sdio_trans_stop_enable

Function name	sdio_trans_stop_enable
Function prototype	void sdio_trans_stop_enable (uint32_t sdio_periph);
Function descriptions	enable the CSM stop command mode(treats the command as a data stop transfer command)
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CSM stop command mode */
```

```
sdio_trans_stop_enable(SDIO0);
```

sdio_trans_stop_disable

The description of sdio_trans_stop_disable is shown as below:

Table 3-1456. Function sdio_trans_stop_disable

Function name	sdio_trans_stop_disable
Function prototype	void sdio_trans_stop_disable(uint32_t sdio_periph);

Function descriptions	disable the CSM stop command mode(treats the command as a data stop transfer command)
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CSM stop command mode*/
```

```
sdio_trans_stop_disable(SDIO0);
```

sdio_csm_enable

The description of sdio_csm_enable is shown as below:

Table 3-1457. Function sdio_csm_enable

Function name	sdio_csm_enable
Function prototype	void sdio_csm_enable(uint32_t sdio_periph);
Function descriptions	enable the CSM(command state machine)
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the CSM(command state machine) */
```

```
sdio_csm_enable(SDIO0);
```

sdio_csm_disable

The description of sdio_csm_disable is shown as below:

Table 3-1458. Function `sdio_csm_disable`

Function name	<code>sdio_csm_disable</code>
Function prototype	<code>void sdio_csm_disable(uint32_t sdio_periph);</code>
Function descriptions	disable the CSM(command state machine)
Precondition	-
The called functions	-
Input parameter{in}	
<code>sdio_periph</code>	SDIO peripheral
<code>SDIO0</code>	use SDIO0
<code>SDIO1</code>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CSM(command state machine) */
sdio_csm_disable(SDIO0);
```

`sdio_command_index_get`

The description of `sdio_command_index_get` is shown as below:

Table 3-1459. Function `sdio_command_index_get`

Function name	<code>sdio_command_index_get</code>
Function prototype	<code>uint8_t sdio_command_index_get(uint32_t sdio_periph);</code>
Function descriptions	get the last response command index
Precondition	-
The called functions	-
Input parameter{in}	
<code>sdio_periph</code>	SDIO peripheral
<code>SDIO0</code>	use SDIO0
<code>SDIO1</code>	use SDIO1
Output parameter{out}	
-	-
Return value	
<code>uint8_t</code>	last response command index

Example:

```
/* get SDIO0 command index */
uint8_t sdio_commond_value;
sdio_commond_value = sdio_command_index_get(SDIO0);
```

sdio_response_get

The description of sdio_response_get is shown as below:

Table 3-1460. Function sdio_response_get

Function name	sdio_response_get
Function prototype	uint32_t sdio_response_get(uint32_t sdio_periph, uint32_t sdio_responsex);
Function descriptions	get the response for the last received command
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
responsex	SDIO response
<i>SDIO_RESPONSE0</i>	card response[31:0]/card response[127:96]
<i>SDIO_RESPONSE1</i>	card response[95:64]
<i>SDIO_RESPONSE2</i>	card response[63:32]
<i>SDIO_RESPONSE3</i>	card response[31:1], plus bit 0
Output parameter{out}	
-	-
Return value	
uint32_t	response for the last received command

Example:

```
/* store the CID0 numbers of SDIO0 */
uint32_t sdio_cid[0];
sdio_cid[0] = sdio_response_get(SDIO0, SDIO_RESPONSE0);
```

sdio_hold_enable

The description of sdio_hold_enable is shown as below:

Table 3-1461. Function sdio_hold_enable

Function name	sdio_hold_enable
Function prototype	void sdio_hold_enable(uint32_t sdio_periph);
Function descriptions	enable the DSM status hold
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0

<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DSM status hold */
sdio_hold_enable(SDIO0);
```

sdio_hold_disable

The description of sdio_hold_disable is shown as below:

Table 3-1462. Function sdio_hold_disable

Function name	sdio_hold_disable
Function prototype	void sdio_hold_disable(uint32_t sdio_periph);
Function descriptions	disable the DSM status hold
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*disable DSM status hold */
sdio_hold_disable(SDIO0);
```

sdio_suspend_enable

The description of sdio_suspend_enable is shown as below:

Table 3-1463. Function sdio_suspend_enable

Function name	sdio_suspend_enable
Function prototype	void sdio_suspend_enable(uint32_t sdio_periph);
Function descriptions	enable the SDIO command suspend mode (the CSM treats the command as a suspend command or resume command)
Precondition	-

The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
SDIO0	use SDIO0
SDIO1	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the SDIO0 suspend mode */
```

```
sdio_suspend_enable(SDIO0);
```

sdio_suspend_disable

The description of sdio_suspend_disable is shown as below:

Table 3-1464. Function sdio_suspend_disable

Function name	sdio_suspend_disable
Function prototype	void sdio_suspend_disable(uint32_t sdio_periph);
Function descriptions	disable the SDIO command suspend mode (the CSM treats the command as a suspend command or resume command)
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
SDIO0	use SDIO0
SDIO1	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the SDIO0 suspend mode */
```

```
sdio_suspend_disable(SDIO0);
```

sdio_data_config

The description of sdio_data_config is shown as below:

Table 3-1465. Function sdio_data_config

Function name	sdio_data_config
Function prototype	void sdio_data_config(uint32_t sdio_periph, uint32_t data_timeout, uint32_t data_length, uint32_t data_blocksize);
Function descriptions	configure the data timeout, data length and data block size
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
data_timeout	data timeout period in card bus clock periods
Input parameter{in}	
data_length	number of data bytes to be transferred
Input parameter{in}	
data_blocksize	size of data block for block transfer
<i>SDIO_DATABLOCKSIZE_1BYTE</i>	block size = 1 byte
<i>SDIO_DATABLOCKSIZE_2BYTES</i>	block size = 2 bytes
<i>SDIO_DATABLOCKSIZE_4BYTES</i>	block size = 4 bytes
<i>SDIO_DATABLOCKSIZE_8BYTES</i>	block size = 8 bytes
<i>SDIO_DATABLOCKSIZE_16BYTES</i>	block size = 16 bytes
<i>SDIO_DATABLOCKSIZE_32BYTES</i>	block size = 32 bytes
<i>SDIO_DATABLOCKSIZE_64BYTES</i>	block size = 64 bytes
<i>SDIO_DATABLOCKSIZE_128BYTES</i>	block size = 128 bytes
<i>SDIO_DATABLOCKSIZE_256BYTES</i>	block size = 256 bytes
<i>SDIO_DATABLOCKSIZE_512BYTES</i>	block size = 512 bytes
<i>SDIO_DATABLOCKSIZE_1024BYTES</i>	block size = 1024 bytes
<i>SDIO_DATABLOCKSIZE_2048BYTES</i>	block size = 2048 bytes
<i>SDIO_DATABLOCKSIZE_4096BYTES</i>	block size = 4096 bytes

<i>SDIO_DATABLOCKSIZE_8192BYTES</i>	block size = 8192 bytes
<i>SDIO_DATABLOCKSIZE_16384BYTES</i>	block size = 16384 bytes
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SDIO0 data */
```

```
sdio_data_config(SDIO0, 0, 0, SDIO_DATABLOCKSIZE_1BYTE);
```

sdio_data_transfer_config

The description of `sdio_data_transfer_config` is shown as below:

Table 3-1466. Function `sdio_data_transfer_config`

Function name	<code>sdio_data_transfer_config</code>
Function prototype	<code>void sdio_data_transfer_config(uint32_t sdio_periph, uint32_t transfer_mode, uint32_t transfer_direction);</code>
Function descriptions	configure the data transfer mode and direction
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
transfer_mode	mode of data transfer
<i>SDIO_TRANSMODE_BLOCK</i>	block count data transfer
<i>SDIO_TRANSMODE_MULTIBYTE</i>	multibyte data transfer (only SD/SD I/O)
<i>SDIO_TRANSMODE_STREAM</i>	stream transfer (only eMMC)
<i>SDIO_TRANSMODE_BLOCKCMD12</i>	block data transfer ends with CMD12
Input parameter{in}	
transfer_direction	data transfer direction, read or write
<i>SDIO_TRANSDIRECTI_ON_TOCARD</i>	write data to card
<i>SDIO_TRANSDIRECTI</i>	read data from card

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

Example:

```
/* configure SDIO0 data transmisson */

sdio_data_transfer_config(SDIO0,                                SDIO_TRANSDIRECTION_TOCARD,
SDIO_TRANSMODE_BLOCKCOUNT);
```

sdio_dsm_enable

The description of sdio_dsm_enable is shown as below:

Table 3-1467. Function sdio_dsm_enable

Function name	sdio_dsm_enable
Function prototype	void sdio_dsm_enable(uint32_t sdio_periph);
Function descriptions	enable the DSM(data state machine) for data transfer
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the DSM(data state machine) */

sdio_dsm_enable(SDIO0);
```

sdio_dsm_disable

The description of sdio_dsm_disable is shown as below:

Table 3-1468. Function sdio_dsm_disable

Function name	sdio_dsm_disable
Function prototype	void sdio_dsm_disable(uint32_t sdio_periph);
Function descriptions	disable the DSM(data state machine)
Precondition	-

The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the DSM(data state machine) */
```

```
sdio_dsm_disable(SDIO0);
```

sdio_data_write

The description of sdio_data_write is shown as below:

Table 3-1469. Function sdio_data_write

Function name	sdio_data_write
Function prototype	void sdio_data_write(uint32_t sdio_periph, uint32_t data);
Function descriptions	write data(one word) to the transmit FIFO
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
data	32-bit data write to card
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write data(one word) to the transmit FIFO */
```

```
sdio_data_write(SDIO0, 0x0000 0001);
```

sdio_data_read

The description of sdio_data_read is shown as below:

Table 3-1470. Function `sdio_data_read`

Function name	<code>sdio_data_read</code>
Function prototype	<code>uint32_t sdio_data_read(uint32_t sdio_periph);</code>
Function descriptions	read data(one word) from the receive FIFO
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
uint32_t	received data

Example:

```
/* read data(one word) from the receive FIFO */
```

```
sdio_data_read(SDIO0);
```

sdio_data_counter_get

The description of `sdio_data_counter_get` is shown as below:

Table 3-1471. Function `sdio_data_counter_get`

Function name	<code>sdio_data_counter_get</code>
Function prototype	<code>uint32_t sdio_data_counter_get(uint32_t sdio_periph);</code>
Function descriptions	get the number of remaining data bytes to be transferred to card
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
uint32_t	number of remaining data bytes to be transferred

Example:

```
/* get the number of remaining data bytes to be transferred to card */
```

```
uint32_t sdio_data_value;
```

```
sdio_data_value = sdio_data_counter_get(SDIO0);
```

sdio_fifo_reset_enable

The description of sdio_fifo_reset_enable is shown as below:

Table 3-1472. Function sdio_fifo_reset_enable

Function name	sdio_fifo_reset_enable
Function prototype	void sdio_fifo_reset_enable(uint32_t sdio_periph);
Function descriptions	enable reset the FIFO
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable reset FIFO */
sdio_fifo_reset_enable(SDIO0);
```

sdio_fifo_reset_disable

The description of sdio_fifo_reset_disable is shown as below:

Table 3-1473. Function sdio_fifo_reset_disable

Function name	sdio_fifo_reset_disable
Function prototype	void sdio_fifo_reset_disable(uint32_t sdio_periph);
Function descriptions	disable reset the FIFO
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset FIFO */
```

```
sdio_fifo_reset_disable(SDIO0);
```

sdio_idma_set

The description of sdio_idma_set is shown as below:

Table 3-1474. Function sdio_idma_set

Function name	sdio_idma_set
Function prototype	void sdio_idma_set(uint32_t sdio_periph, uint32_t buffer_mode, uint32_t buffer_size);
Function descriptions	set IDMA buffer mode and size
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
buffer_mode	set IDMA buffer mode
<i>SDIO_IDMA_SINGLE_BUFFER</i>	single buffer mode
<i>SDIO_IDMA_DOUBLE_BUFFER</i>	double buffer mode
Input parameter{in}	
buffer_size	set IDMA buffer size: 0-0xFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the SDIO0 IDMA */
```

```
sdio_idma_set(SDIO, SDIO_IDMA_SINGLE_BUFFER, (uint32_t)0x20);
```

sdio_idma_buffer0_address_set

The description of sdio_idma_buffer0_address_set is shown as below:

Table 3-1475. Function sdio_idma_buffer0_address_set

Function name	sdio_idma_buffer0_address_set
Function prototype	void sdio_idma_buffer0_address_set(uint32_t sdio_periph, uint32_t buffer_address);
Function descriptions	set IDMA buffer0 address
Precondition	-

The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
SDIO0	use SDIO0
SDIO1	use SDIO1
Input parameter{in}	
buffer_address	the address of idma buffer0, buffer_address[0:1] should be 0b'00
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set IDMA buffer0 address */
```

```
sdio_idma_buffer0_address_set(SDIO0, (uint32_t)(0x0000 0010));
```

sdio_idma_buffer1_address_set

The description of sdio_idma_buffer1_address_set is shown as below:

Table 3-1476. Function sdio_idma_buffer1_address_set

Function name	sdio_idma_buffer1_address_set
Function prototype	void sdio_idma_buffer1_address_set(uint32_t sdio_periph, uint32_t buffer_address);
Function descriptions	set IDMA buffer1 address
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
SDIO0	use SDIO0
SDIO1	use SDIO1
Input parameter{in}	
buffer_address	the address of idma buffer1, buffer_address[0:1] should be 0b'00
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set IDMA buffer1 address */
```

```
sdio_idma_buffer1_address_set(SDIO0, (uint32_t)(0x0000 0010));
```


sdio_buffer_selection_get

The description of sdio_buffer_selection_get is shown as below:

Table 3-1477. Function sdio_buffer_selection_get

Function name	sdio_buffer_selection_get
Function prototype	void sdio_buffer_selection_get(uint32_t sdio_periph);
Function descriptions	get IDMA buffer selection
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
uint32_t	SDIO_IDMA_BUFER0_SELECTION/SDIO_IDMA_BUFER1_SELECTION

Example:

```
/* get the IDMA of SDIO0 */
```

```
uint32_t sdio_buffer_selection_value;
```

```
sdio_buffer_selection_value = sdio_buffer_selection_get(SDIO0);
```

sdio_idma_buffer_select

The description of sdio_idma_buffer_select is shown as below:

Table 3-1478. Function sdio_idma_buffer_select

Function name	sdio_idma_buffer_select
Function prototype	void sdio_idma_buffer_select(uint32_t sdio_periph, uint32_t buffer_select);
Function descriptions	select IDMA buffer
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
buffer_select	the buffer to be used
<i>SDIO_IDMA_BUFFER0_SELECTION</i>	select buffer0
<i>SDIO_IDMA_BUFFER1</i>	select buffer1

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

Example:

```
/* set the IDMA of SDIO0 */
```

```
sdio_idma_buffer_select(SDIO0, SDIO_IDMA_BUFFER0_SELECTION);
```

sdio_idma_enable

The description of sdio_idma_enable is shown as below:

Table 3-1479. Function sdio_idma_enable

Function name	sdio_idma_enable
Function prototype	void sdio_idma_enable(uint32_t sdio_periph);
Function descriptions	enable IDMA
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
SDIO0	use SDIO0
SDIO1	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the IDMA of SDIO0 */
```

```
sdio_idma_enable(SDIO0);
```

sdio_idma_disable

The description of sdio_idma_disable is shown as below:

Table 3-1480. Function sdio_idma_disable

Function name	sdio_idma_disable
Function prototype	void sdio_idma_disable(uint32_t sdio_periph);
Function descriptions	disable IDMA
Precondition	-
The called functions	-

Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the IDMA of SDIO0 */
```

```
sdio_idma_disable(SDIO0);
```

sdio_flag_get

The description of sdio_flag_get is shown as below:

Table 3-1481. Function sdio_flag_get

Function name	sdio_flag_get
Function prototype	FlagStatus sdio_flag_get(uint32_t sdio_periph, uint32_t flag);
Function descriptions	get the flags state of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
flag	flags state of SDIO
<i>SDIO_FLAG_CCRCE</i> <i>R</i>	command response received (CRC check failed) flag
<i>SDIO_FLAG_DTCRCE</i> <i>RR</i>	data block sent/received (CRC check failed) flag
<i>SDIO_FLAG_CMDTMO</i> <i>UT</i>	command response timeout flag
<i>SDIO_FLAG_DTTMOU</i> <i>T</i>	data timeout flag
<i>SDIO_FLAG_TXURE</i>	transmit FIFO underrun error occurs flag
<i>SDIO_FLAG_RXORE</i>	received FIFO overrun error occurs flag
<i>SDIO_FLAG_CMDREC</i> <i>V</i>	command response received (CRC check passed) flag
<i>SDIO_FLAG_CMDSEN</i> <i>D</i>	command sent (no response required) flag

<i>SDIO_FLAG_DTEND</i>	data end (data counter, SDIO_DATACNT, is zero) flag
<i>SDIO_FLAG_DTHOLD</i>	data transfer hold
<i>SDIO_FLAG_DTBLKE</i> <i>ND</i>	data block sent/received (CRC check passed) flag
<i>SDIO_FLAG_DTABOR</i> <i>T</i>	data transfer aborted by CMD12
<i>SDIO_FLAG_CMDSTA</i>	command path active state
<i>SDIO_FLAG_DATSTA</i>	data path active state
<i>SDIO_FLAG_TFH</i>	transmit FIFO is half empty flag: at least 8 words can be written into the FIFO
<i>SDIO_FLAG_RFH</i>	receive FIFO is half full flag: at least 8 words can be read in the FIFO
<i>SDIO_FLAG_TFF</i>	transmit FIFO is full flag
<i>SDIO_FLAG_RFF</i>	receive FIFO is full flag
<i>SDIO_FLAG_TFE</i>	transmit FIFO is empty flag
<i>SDIO_FLAG_RFE</i>	receive FIFO is empty flag
<i>SDIO_FLAG_DAT0BS</i> <i>Y</i>	DAT0 line signal keep busy flag
<i>SDIO_FLAG_DAT0BS</i> <i>YEND</i>	DAT0 line signal changed from busy to ready flag
<i>SDIO_FLAG_SDIOINT</i>	SD I/O interrupt received flag
<i>SDIO_FLAG_ACKFAIL</i>	boot acknowledgment received and check fail flag
<i>SDIO_FLAG_ACKTO</i>	boot acknowledgment timeout flag
<i>SDIO_FLAG_VOLSWEN</i> <i>ND</i>	voltage switch critical timing section end flag
<i>SDIO_FLAG_CLKSTO</i> <i>P</i>	SDIO_CLK stopped in voltage switch procedure flag
<i>SDIO_FLAG_IDMAER</i> <i>R</i>	IDMA transfer error flag
<i>SDIO_FLAG_IDMAEN</i> <i>D</i>	IDMA transfer end flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the flags state of SDIO0 */
```

```
FlagStatus flag_value;
```

```
flag_value = sdio_flag_get(SDIO0, SDIO_FLAG_RFH);
```

sdio_flag_clear

The description of sdio_flag_clear is shown as below:

Table 3-1482. Function sdio_flag_clear

Function name	sdio_flag_clear
Function prototype	void sdio_flag_clear(uint32_t sdio_periph, uint32_t flag);
Function descriptions	clear the pending flags of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
flag	flags state of SDIO
<i>SDIO_FLAG_CCR CER R</i>	command response received (CRC check failed) flag
<i>SDIO_FLAG_DTCRCE RR</i>	data block sent/received (CRC check failed) flag
<i>SDIO_FLAG_CMDTMO UT</i>	command response timeout flag
<i>SDIO_FLAG_DTTMOU T</i>	data timeout flag
<i>SDIO_FLAG_TXURE</i>	transmit FIFO underrun error occurs flag
<i>SDIO_FLAG_RXORE</i>	received FIFO overrun error occurs flag
<i>SDIO_FLAG_CMDREC V</i>	command response received (CRC check passed) flag
<i>SDIO_FLAG_CMDSEN D</i>	command sent (no response required) flag
<i>SDIO_FLAG_DTEND</i>	data end (data counter, SDIO_DATACNT is zero) flag
<i>SDIO_FLAG_DTHOLD</i>	data transfer hold flag
<i>SDIO_FLAG_DTBLKE ND</i>	data block sent/received (CRC check passed) flag
<i>SDIO_FLAG_DTABOR T</i>	data transfer aborted (by CMD12) flag
<i>SDIO_FLAG_DAT0BS YEND</i>	DAT0 line signal changed from busy to ready flag
<i>SDIO_FLAG_SDIOINT</i>	SD I/O interrupt received flag
<i>SDIO_FLAG_ACKFAIL</i>	boot acknowledgment received and check fail
<i>SDIO_FLAG_ACKTO</i>	boot acknowledgment timeout flag
<i>SDIO_FLAG_VOLSWE ND</i>	voltage switch critical timing section end flag

<i>SDIO_FLAG_CLKSTO</i> <i>P</i>	SDIO_CLK stopped in voltage switch procedure flag
<i>SDIO_FLAG_IDMAER</i> <i>R</i>	IDMA transfer error occurs flag
<i>SDIO_FLAG_IDMAEN</i> <i>D</i>	IDMA transfer end flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the pending flags of SDIO0 */
```

```
sdio_flag_clear(SDIO0, SDIO_FLAG_DTCRCERR);
```

sdio_interrupt_enable

The description of sdio_interrupt_enable is shown as below:

Table 3-1483. Function sdio_interrupt_enable

Function name	sdio_interrupt_enable
Function prototype	void sdio_interrupt_enable(uint32_t sdio_periph, uint32_t int_flag);
Function descriptions	enable the SDIO interrupt
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
int_flag	interrupt flags state of SDIO
<i>SDIO_INT_CCRRCERR</i>	SDIO CCRRCERR interrupt
<i>SDIO_INT_DTCRCER</i> <i>R</i>	SDIO DTCRCERR interrupt
<i>SDIO_INT_CMDTMOU</i> <i>T</i>	SDIO CMDTMOUT interrupt
<i>SDIO_INT_DTTMOUT</i>	SDIO DTTMOUT interrupt
<i>SDIO_INT_TXURE</i>	SDIO TXURE interrupt
<i>SDIO_INT_RXORE</i>	SDIO_INT_RXORE
<i>SDIO_INT_CMDRECV</i>	SDIO CMDRECV interrupt
<i>SDIO_INT_CMDSEND</i>	SDIO CMDSEND interrupt
<i>SDIO_INT_DTEND</i>	SDIO DTEND interrupt
<i>SDIO_INT_DTHOLD</i>	SDIO DTHOLD interrupt

<i>SDIO_INT_DTBLKEND</i>	SDIO DTBLKEND interrupt
<i>SDIO_INT_DTABORT</i>	SDIO DTABORT interrupt
<i>SDIO_INT_TFH</i>	SDIO TFH interrupt
<i>SDIO_INT_RFH</i>	SDIO RFH interrupt
<i>SDIO_INT_RFF</i>	SDIO RFF interrupt
<i>SDIO_INT_TFE</i>	SDIO TFE interrupt
<i>SDIO_INT_DAT0BSYEND</i>	SDIO DAT0BSYEND interrupt
<i>SDIO_INT_SDIOINT</i>	SDIO SDIOINT interrupt
<i>SDIO_INT_ACKFAIL</i>	SDIO ACKFAIL interrupt
<i>SDIO_INT_ACKTO</i>	SDIO ACKTO interrupt
<i>SDIO_INT_VOLSWEND</i>	SDIO VOLSWEND interrupt
<i>SDIO_INT_CLKSTOP</i>	SDIO CLKSTOP interrupt
<i>SDIO_INT_IDMAERR</i>	SDIO IDMAERR interrupt
<i>SDIO_INT_IDMAEND</i>	SDIO IDMAEND interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the SDIO0 corresponding interrupts */
```

```
sdio_interrupt_enable(SDIO0, SDIO_INT_CCRCERR | SDIO_INT_DTTMOUT |  
SDIO_INT_RXORE | SDIO_INT_DTEND | SDIO_INT_STBITE);
```

sdio_interrupt_disable

The description of sdio_interrupt_disable is shown as below:

Table 3-1484. Function sdio_interrupt_disable

Function name	sdio_interrupt_disable
Function prototype	void sdio_interrupt_disable(uint32_t sdio_periph, uint32_t int_flag);
Function descriptions	disable the SDIO interrupt
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
int_flag	interrupt flags state of SDIO
<i>SDIO_INT_CCRCERR</i>	SDIO CCRCERR interrupt

<i>SDIO_INT_DTCRCERR</i>	SDIO DTCRCERR interrupt
<i>SDIO_INT_CMDTMOUT</i>	SDIO CMDTMOUT interrupt
<i>SDIO_INT_DTTMOUT</i>	SDIO DTTMOUT interrupt
<i>SDIO_INT_TXURE</i>	SDIO TXURE interrupt
<i>SDIO_INT_RXORE</i>	SDIO_INT_RXORE
<i>SDIO_INT_CMDRECV</i>	SDIO CMDRECV interrupt
<i>SDIO_INT_CMDSEND</i>	SDIO CMDSEND interrupt
<i>SDIO_INT_DTEND</i>	SDIO DTEND interrupt
<i>SDIO_INT_DTHOLD</i>	SDIO DTHOLD interrupt
<i>SDIO_INT_DTBLKEND</i>	SDIO DTBLKEND interrupt
<i>SDIO_INT_DTABORT</i>	SDIO DTABORT interrupt
<i>SDIO_INT_TFH</i>	SDIO TFH interrupt
<i>SDIO_INT_RFH</i>	SDIO RFH interrupt
<i>SDIO_INT_RFF</i>	SDIO RFF interrupt
<i>SDIO_INT_TFE</i>	SDIO TFE interrupt
<i>SDIO_INT_DAT0BSYEND</i>	SDIO DAT0BSYEND interrupt
<i>SDIO_INT_SDIOINT</i>	SDIO SDIOINT interrupt
<i>SDIO_INT_ACKFAIL</i>	SDIO ACKFAIL interrupt
<i>SDIO_INT_ACKTO</i>	SDIO ACKTO interrupt
<i>SDIO_INT_VOLSWEND</i>	SDIO VOLSWEND interrupt
<i>SDIO_INT_CLKSTOP</i>	SDIO CLKSTOP interrupt
<i>SDIO_INT_IDMAERR</i>	SDIO IDMAERR interrupt
<i>SDIO_INT_IDMAEND</i>	SDIO IDMAEND interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the SDIO0 interrupt */
```

```
sdio_interrupt_disable(SDIO0, SDIO_INT_DTCRCERR);
```

sdio_interrupt_flag_get

The description of sdio_interrupt_flag_get is shown as below:

Table 3-1485. Function sdio_interrupt_flag_get

Function name	sdio_interrupt_flag_get
Function prototype	FlagStatus sdio_interrupt_flag_get(uint32_t sdio_periph, uint32_t int_flag);

Function descriptions	get the interrupt flags state of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
int_flag	interrupt flags state of SDIO
<i>SDIO_INT_FLAG_CCR CERR</i>	SDIO CCRCERR interrupt flag
<i>SDIO_INT_FLAG_DTC RCERR</i>	SDIO DTCRCERR interrupt flag
<i>SDIO_INT_FLAG_CMD TMOUT</i>	SDIO CMDTMOUT interrupt flag
<i>SDIO_INT_FLAG_DTT MOUT</i>	SDIO DTTMOUT interrupt flag
<i>SDIO_INT_FLAG_TXU RE</i>	SDIO TXURE interrupt flag
<i>SDIO_INT_FLAG_RXO RE</i>	SDIO_INT_RXORE flag
<i>SDIO_INT_FLAG_CMD RECV</i>	SDIO CMDRECV interrupt flag
<i>SDIO_INT_FLAG_CMD SEND</i>	SDIO CMDSEND interrupt flag
<i>SDIO_INT_FLAG_DTE ND</i>	SDIO DTEND interrupt flag
<i>SDIO_INT_FLAG_DTH OLD</i>	SDIO STBITE interrupt flag
<i>SDIO_INT_FLAG_DTB LKEND</i>	SDIO DTBLKEND interrupt flag
<i>SDIO_INT_FLAG_DTA BORT</i>	SDIO DTABORT interrupt flag
<i>SDIO_INT_FLAG_TFH</i>	SDIO TFH interrupt flag
<i>SDIO_INT_FLAG_RFH</i>	SDIO RFH interrupt flag
<i>SDIO_INT_FLAG_RFF</i>	SDIO RFF interrupt flag
<i>SDIO_INT_FLAG_TFE</i>	SDIO TFE interrupt flag
<i>SDIO_INT_FLAG_DAT OBSYEND</i>	SDIO DAT0BSYEND interrupt flag
<i>SDIO_INT_FLAG_SDI OINT</i>	SDIO SDIOINT interrupt flag
<i>SDIO_INT_FLAG_ACK FAIL</i>	SDIO ACKFAIL interrupt flag

<i>SDIO_INT_FLAG_ACKTO</i>	SDIO ACKTO interrupt flag
<i>SDIO_INT_FLAG_VOLSWEND</i>	SDIO VOLSWEND interrupt flag
<i>SDIO_INT_FLAG_CLKSTOP</i>	SDIO CLKSTOP interrupt flag
<i>SDIO_INT_FLAG_IDMAEND</i>	SDIO IDMAEND interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the interrupt flags state of SDIO0 */
```

```
FlagStatus flag_value;
```

```
flag_value = sdio_interrupt_flag_get(SDIO0, SDIO_INT_FLAG_DTEND);
```

sdio_interrupt_flag_clear

The description of `sdio_interrupt_flag_clear` is shown as below:

Table 3-1486. Function `sdio_interrupt_flag_clear`

Function name	<code>sdio_interrupt_flag_clear</code>
Function prototype	<code>void sdio_interrupt_flag_clear(uint32_t sdio_periph, uint32_t int_flag);</code>
Function descriptions	clear the interrupt pending flags of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
int_flag	interrupt flags state of SDIO
<i>SDIO_INT_FLAG_CCCR</i>	command response received (CRC check failed) flag
<i>SDIO_INT_FLAG_DTCRCERR</i>	data block sent/received (CRC check failed) flag
<i>SDIO_INT_FLAG_CMDTMOUT</i>	command response timeout flag
<i>SDIO_INT_FLAG_DTTMOUT</i>	data timeout flag
<i>SDIO_INT_FLAG_TXU</i>	transmit FIFO underrun error occurs flag

<i>RE</i>	
<i>SDIO_INT_FLAG_RXO RE</i>	received FIFO overrun error occurs flag
<i>SDIO_INT_FLAG_CMD RECV</i>	command response received (CRC check passed) flag
<i>SDIO_INT_FLAG_CMD SEND</i>	command sent (no response required) flag
<i>SDIO_INT_FLAG_DTE ND</i>	data end (data counter, SDIO_DATACNT, is zero) flag
<i>SDIO_INT_FLAG_DTH OLD</i>	start bit error in the bus flag
<i>SDIO_INT_FLAG_DTB LKEND</i>	data block sent/received (CRC check passed) flag
<i>SDIO_INT_FLAG_DTA BORT</i>	SDIO DTABORT interrupt flag
<i>SDIO_INT_FLAG_DAT OBSYEND</i>	SDIO DAT0BSYEND interrupt flag
<i>SDIO_INT_FLAG_SDI OINT</i>	SD I/O interrupt received flag
<i>SDIO_INT_FLAG_ACK FAIL</i>	boot acknowledgment received and check fail flag
<i>SDIO_INT_FLAG_ACK TO</i>	boot acknowledgment timeout flag
<i>SDIO_INT_FLAG_VOL SWEND</i>	voltage switch critical timing section end flag
<i>SDIO_INT_FLAG_CLK STOP</i>	SDIO_CLK stopped in voltage switch procedure flag
<i>SDIO_INT_FLAG_IDM AERR</i>	IDMA transfer error flag
<i>SDIO_INT_FLAG_IDM AEND</i>	IDMA transfer end flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the interrupt pending flags of SDIO0 */
```

```
sdio_interrupt_flag_clear(SDIO0, SDIO_INT_FLAG_DTEND);
```

sdio_voltage_switch_enable

The description of sdio_voltage_switch_enable is shown as below:

Table 3-1487. Function `sdio_voltage_switch_enable`

Function name	<code>sdio_voltage_switch_enable</code>
Function prototype	<code>void sdio_voltage_switch_enable(uint32_t sdio_periph);</code>
Function descriptions	enable voltage switch
Precondition	-
The called functions	-
Input parameter{in}	
<code>sdio_periph</code>	SDIO peripheral
<code>SDIO0</code>	use SDIO0
<code>SDIO1</code>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SDIO0 voltage switch */
sdio_voltage_switch_enable(SDIO0);
```

`sdio_voltage_switch_disable`

The description of `sdio_voltage_switch_disable` is shown as below:

Table 3-1488. Function `sdio_voltage_switch_disable`

Function name	<code>sdio_voltage_switch_disable</code>
Function prototype	<code>void sdio_voltage_switch_disable(uint32_t sdio_periph);</code>
Function descriptions	disable voltage switch
Precondition	-
The called functions	-
Input parameter{in}	
<code>sdio_periph</code>	SDIO peripheral
<code>SDIO0</code>	use SDIO0
<code>SDIO1</code>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SDIO0 voltage switch */
sdio_voltage_switch_disable(SDIO0);
```

sdio_voltage_switch_sequence_enable

The description of sdio_voltage_switch_sequence_enable is shown as below:

Table 3-1489. Function sdio_voltage_switch_sequence_enable

Function name	sdio_voltage_switch_sequence_enable
Function prototype	void sdio_voltage_switch_sequence_enable(uint32_t sdio_periph);
Function descriptions	enable voltage switch sequence
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SDIO0 voltage switch sequence */
```

```
sdio_voltage_switch_sequence_enable(SDIO0);
```

sdio_voltage_switch_sequence_disable

The description of sdio_voltage_switch_sequence_disable is shown as below:

Table 3-1490. Function sdio_voltage_switch_sequence_disable

Function name	sdio_voltage_switch_sequence_disable
Function prototype	void sdio_voltage_switch_sequence_disable(uint32_t sdio_periph);
Function descriptions	disable voltage switch sequence
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SDIO0 voltage switch sequence */
```

```
sdio_voltage_switch_sequence_disable(SDIO0);
```

sdio_boot_mode_set

The description of sdio_boot_mode_set is shown as below:

Table 3-1491. Function sdio_boot_mode_set

Function name	sdio_boot_mode_set
Function prototype	void sdio_boot_mode_set(uint32_t sdio_periph, uint32_t boot_mode);
Function descriptions	set boot mode
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
boot_mode	mode of SDIO boot
<i>SDIO_BOOT_NORMAL</i>	normal boot mode
<i>SDIO_BOOT_ALTERNATIVE</i>	alternative boot mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO0 boot mode*/
```

```
sdio_boot_mode_set (SDIO0, SDIO_BOOT_NORMAL);
```

sdio_boot_ack_enable

The description of sdio_boot_ack_enable is shown as below:

Table 3-1492. Function sdio_boot_ack_enable

Function name	sdio_boot_ack_enable
Function prototype	void sdio_boot_ack_enable(uint32_t sdio_periph);
Function descriptions	enable DSM(data state machine) boot acknowledgment
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SDIO0 boot acknowledgment */
sdio_boot_ack_enable(SDIO0);
```

sdio_boot_ack_disable

The description of sdio_boot_ack_disable is shown as below:

Table 3-1493. Function sdio_boot_ack_disable

Function name	sdio_boot_ack_disable
Function prototype	void sdio_boot_ack_disable(uint32_t sdio_periph);
Function descriptions	disable DSM(data state machine) boot acknowledgment
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
SDIO0	use SDIO0
SDIO1	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SDIO0 boot acknowledgment */
sdio_boot_ack_disable(SDIO0);
```

sdio_boot_acktimeout_set

The description of sdio_boot_acktimeout_set is shown as below:

Table 3-1494. Function sdio_boot_acktimeout_set

Function name	sdio_boot_acktimeout_set
Function prototype	void sdio_boot_acktimeout_set(uint32_t sdio_periph, uint32_t timeout);
Function descriptions	set boot ACK timeout period
Precondition	-
The called functions	-
Input parameter{in}	

sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Input parameter{in}	
timeout	boot ACK timeout period
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO0 boot ACK timeout period */
```

```
void sdio_boot_acktimeout_set(SDIO0, uint32_t (0xFFFF));
```

sdio_boot_enable

The description of sdio_boot_enable is shown as below:

Table 3-1495. Function sdio_boot_enable

Function name	sdio_boot_enable
Function prototype	void sdio_boot_enable(uint32_t sdio_periph);
Function descriptions	enable boot operation
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SDIO0 boot operation */
```

```
sdio_boot_enable(SDIO0);
```

sdio_boot_disable

The description of sdio_boot_disable is shown as below:

Table 3-1496. Function sdio_boot_disable

Function name	sdio_boot_disable
----------------------	-------------------

Function prototype	void sdio_boot_disable(uint32_t sdio_periph);
Function descriptions	disable boot operation
Precondition	-
The called functions	-
Input parameter{in}	
sdio_periph	SDIO peripheral
<i>SDIO0</i>	use SDIO0
<i>SDIO1</i>	use SDIO1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SDIO0 boot operation */
sdio_boot_disable(SDIO0);
```

3.41. 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 [3.41.1](#), the SPI/I2S firmware functions are introduced in chapter [3.41.2](#).

3.41.1. Descriptions of Peripheral registers

SPI/I2S registers are listed in the table shown as below:

Table 3-1497. 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
SPI_RCRC	RX CRC register
SPI_URDATA	Underrun Data register

Registers	Descriptions
SPI_I2SCTL	I2S control register
SPI_QCTL	Quad_SPI mode control register
SPI_RXDLYCK	RX clock delay register

3.41.2. Descriptions of Peripheral functions

SPI/I2S firmware functions are listed in the table shown as below:

Table 3-1498. 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
spi_current_data_num_config	configure SPI current data number

Function name	Function description
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 3-1499. 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 3-1500. Function spi_i2s_deinit

Function name	spi_i2s_deinit
Function prototype	void spi_i2s_deinit(uint32_t spi_periph);
Function descriptions	reset SPI and I2S peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
spi_periph	SPI/I2S peripheral
SPIx	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset SPI0 */
spi_i2s_deinit(SPI0);
```

spi_struct_para_init

The description of spi_struct_para_init is shown as below:

Table 3-1501. Function spi_struct_para_init

Function name	spi_struct_para_init
Function prototype	void spi_struct_para_init(spi_parameter_struct* spi_struct);
Function descriptions	initialize the parameters of SPI struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
spi_struct	SPI parameter struct, the structure members can refer to members of the structure Table 3-1499. spi_parameter_struct
Return value	
-	-

Example:

```
/* initialize the parameters of SPI */
spi_parameter_struct spi_init_struct;
spi_struct_para_init(&spi_init_struct);
```

spi_init

The description of spi_init is shown as below:

Table 3-1502. Function spi_init

Function name	spi_init
Function prototype	void spi_init(uint32_t spi_periph, spi_parameter_struct* spi_struct);
Function descriptions	initialize SPI peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
spi_struct	SPI parameter initialization struct, the structure members can refer to members of the structure Table 3-1499. spi_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize SPI0 */

spi_parameter_struct spi_init_struct;

spi_init_struct.trans_mode = SPI_TRANSMODE_BDTRANSMIT;

spi_init_struct.device_mode = SPI_MASTER;

spi_init_struct.frame_size = SPI_FRAME_SIZE_8BIT;

spi_init_struct.clock_polarity_phase = SPI_CK_PL_HIGH_PH_2EDGE;

spi_init_struct.nss = SPI_NSS_SOFT;

spi_init_struct.prescale = SPI_PSC_8;

spi_init_struct.endian = SPI_ENDIAN_MSB;

spi_init(SPI0, &spi_init_struct);
```

spi_enable

The description of spi_enable is shown as below:

Table 3-1503. Function spi_enable

Function name	spi_enable
Function prototype	void spi_enable(uint32_t spi_periph);
Function descriptions	enable SPI
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 */

spi_enable(SPI0);
```

spi_disable

The description of spi_disable is shown as below:

Table 3-1504. Function spi_disable

Function name	spi_disable
Function prototype	void spi_disable(uint32_t spi_periph);
Function descriptions	disable SPIx
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 */
spi_disable(SPI0);
```

i2s_init

The description of i2s_init is shown as below:

Table 3-1505. Function i2s_init

Function name	i2s_init
Function prototype	void i2s_init(uint32_t spi_periph, uint32_t i2s_mode, uint32_t i2s_standard, uint32_t i2s_ckpl);
Function descriptions	initialize I2S peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=0,1,2,5
Input parameter{in}	
i2s_mode	I2S operation mode
I2S_MODE_SLAVETX	I2S slave transmit mode
I2S_MODE_SLAVERX	I2S slave receive mode
I2S_MODE_MASTERTX	I2S master transmit mode
I2S_MODE_MASTERRX	I2S master receive mode
Input parameter{in}	
i2s_standard	I2S standard
I2S_STD_PHILLIPS	I2S phillips standard

<i>I2S_STD_MSB</i>	I2S MSB standard
<i>I2S_STD_LSB</i>	I2S LSB standard
<i>I2S_STD_PCMSHORT</i>	I2S PCM short standard
<i>I2S_STD_PCMLONG</i>	I2S PCM long standard
Input parameter{in}	
i2s_ckpl	I2S idle state clock polarity
<i>I2S_CKPL_LOW</i>	I2S clock polarity low level
<i>I2S_CKPL_HIGH</i>	I2S clock polarity high level
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize 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 3-1506. Function i2s_psc_config

Function name	i2s_psc_config
Function prototype	void i2s_psc_config(uint32_t spi_periph, uint32_t i2s_audiosample, uint32_t i2s_frameformat, uint32_t i2s_mckout);
Function descriptions	configure I2S prescaler
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
spi_periph	I2S peripheral
<i>SPIx</i>	x=0,1,2,5
Input parameter{in}	
i2s_audiosample	I2S audio sample rate
<i>I2S_AUDIOSAMPLE_8K</i>	audio sample rate is 8KHz
<i>I2S_AUDIOSAMPLE_11K</i>	audio sample rate is 11KHz
<i>I2S_AUDIOSAMPLE_16K</i>	audio sample rate is 16KHz
<i>I2S_AUDIOSAMPLE_22K</i>	audio sample rate is 22KHz
<i>I2S_AUDIOSAMPLE_32K</i>	audio sample rate is 32KHz

<i>I2S_AUDIOSAMPLE_4</i> 4K	audio sample rate is 44KHz
<i>I2S_AUDIOSAMPLE_4</i> 8K	audio sample rate is 48KHz
<i>I2S_AUDIOSAMPLE_9</i> 6K	audio sample rate is 96KHz
<i>I2S_AUDIOSAMPLE_1</i> 92K	audio sample rate is 192KHz
Input parameter{in}	
i2s_frameformat	I2S data length and channel length
<i>I2S_FRAMEFORMAT_DT16B_CH16B</i>	I2S data length is 16 bit and channel length is 16 bit
<i>I2S_FRAMEFORMAT_DT16B_CH32B</i>	I2S data length is 16 bit and channel length is 32 bit
<i>I2S_FRAMEFORMAT_DT24B_CH32B</i>	I2S data length is 24 bit and channel length is 32 bit
<i>I2S_FRAMEFORMAT_DT32B_CH32B</i>	I2S data length is 32 bit and channel length is 32 bit
Input parameter{in}	
i2s_mckout	I2S master clock output
<i>I2S_MCKOUT_ENABL</i> E	I2S master clock output enable
<i>I2S_MCKOUT_DISABL</i> E	I2S master clock output disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure 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 3-1507. Function i2s_enable

Function name	i2s_enable
Function prototype	void i2s_enable(uint32_t spi_periph);
Function descriptions	enable I2S
Precondition	-

The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=0,1,2,5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2S0*/
i2s_enable(SPI0);
```

i2s_disable

The description of i2s_disable is shown as below:

Table 3-1508. 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 3-1509. 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
<i>SPIx</i>	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1510. Function spi_nss_idleness_delay_set

Function name	spi_nss_idleness_delay_set
Function prototype	void spi_nss_idleness_delay_set(uint32_t spi_periph, uint32_t delay_cycle);
Function descriptions	set delay between active edge of NSS and start transfer or receive data in SPI master mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
delay_cycle	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)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1511. Function spi_data_frame_delay_set

Function name	spi_data_frame_delay_set
Function prototype	void spi_data_frame_delay_set(uint32_t spi_periph, uint32_t delay_cycle);
Function descriptions	set SPI master data frame delay
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
delay_cycle	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)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1512. Function spi_master_receive_clock_delay_set

Function name	spi_master_receive_clock_delay_set
Function prototype	void spi_master_receive_clock_delay_set(uint32_t spi_periph, uint32_t delay_unit);
Function descriptions	set SPI master mode rx clock delay
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
delay_unit	clock delay unit(0-0x1F)

Output parameter{out}	
-	-
Return value	
-	-

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 3-1513. Function spi_slave_receive_clock_delay_set

Function name	spi_slave_receive_clock_delay_set
Function prototype	void spi_slave_receive_clock_delay_set(uint32_t spi_periph, uint32_t delay_unit);
Function descriptions	set SPI slave mode rx clock delay
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Input parameter{in}	
delay_unit	clock delay unit(0-0x1F)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1514. Function spi_master_receive_clock_delay_clear

Function name	spi_master_receive_clock_delay_clear
Function prototype	void spi_master_receive_clock_delay_clear(uint32_t spi_periph);
Function descriptions	clear SPI master mode rx clock delay
Precondition	-

The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

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 3-1515. Function spi_slave_receive_clock_delay_clear

Function name	spi_slave_receive_clock_delay_clear
Function prototype	void spi_slave_receive_clock_delay_clear(uint32_t spi_periph);
Function descriptions	clear SPI slave mode rx clock delay
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

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 3-1516. Function spi_nss_output_control

Function name	spi_nss_output_control
Function prototype	void spi_nss_output_control(uint32_t spi_periph, uint32_t nss_ctl);
Function descriptions	SPI NSS output control

Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
nss_ctl	nss control bit
<i>SPI_NSS_HOLD_UNTIL_TRANS_END</i>	SPI NSS remains active level until data transfer complete
<i>SPI_NSS_INVALID_PULSE</i>	SPI data frames are interleaved with NSS invalid pulses
Output parameter{out}	
-	-
Return value	
-	-

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 3-1517. Function spi_nss_polarity_set

Function name	spi_nss_polarity_set
Function prototype	void spi_nss_polarity_set(uint32_t spi_periph, uint32_t polarity);
Function descriptions	set SPI NSS active polarity
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
polarity	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1518. Function spi_nss_output_enable

Function name	spi_nss_output_enable
Function prototype	void spi_nss_output_enable(uint32_t spi_periph);
Function descriptions	enable SPI NSS output
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 NSS output */

spi_nss_output_enable(SPI0);
```

spi_nss_output_disable

The description of spi_nss_output_disable is shown as below:

Table 3-1519. Function spi_nss_output_disable

Function name	spi_nss_output_disable
Function prototype	void spi_nss_output_disable(uint32_t spi_periph);
Function descriptions	disable SPI NSS output
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 NSS output */

spi_nss_output_disable(SPI0);
```

spi_nss_internal_high

The description of spi_nss_internal_high is shown as below:

Table 3-1520. Function spi_nss_internal_high

Function name	spi_nss_internal_high
Function prototype	void spi_nss_internal_high(uint32_t spi_periph);
Function descriptions	SPI NSS pin high level in software mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 NSS pin is pulled high level in software mode */

spi_nss_internal_high(SPI0);
```

spi_nss_internal_low

The description of spi_nss_internal_low is shown as below:

Table 3-1521. Function spi_nss_internal_low

Function name	spi_nss_internal_low
Function prototype	void spi_nss_internal_low(uint32_t spi_periph);
Function descriptions	SPI NSS pin low level in software mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 NSS pin is pulled low level in software mode */
```

```
spi_nss_internal_low(SPI0);
```

spi_dma_enable

The description of spi_dma_enable is shown as below:

Table 3-1522. Function spi_dma_enable

Function name	spi_dma_enable
Function prototype	void spi_dma_enable(uint32_t spi_periph, uint8_t spi_dma);
Function descriptions	enable SPI DMA send or receive
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
spi_dma	SPI DMA mode
<i>SPI_DMA_TRANSMIT</i>	SPI transmit data use DMA
<i>SPI_DMA_RECEIVE</i>	SPI receive data use DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 transmit data DMA function */
```

```
spi_dma_enable(SPI0, SPI_DMA_TRANSMIT);
```

spi_dma_disable

The description of spi_dma_disable is shown as below:

Table 3-1523. Function spi_dma_disable

Function name	spi_dma_disable
Function prototype	void spi_dma_disable(uint32_t spi_periph, uint8_t spi_dma);
Function descriptions	disable SPI DMA function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1

Input parameter{in}	
spi_dma	SPI DMA mode
<i>SPI_DMA_TRANSMIT</i>	SPI transmit data use DMA
<i>SPI_DMA_RECEIVE</i>	SPI receive data use DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 transmit data DMA function */
```

```
spi_dma_disable(SPI0, SPI_DMA_TRANSMIT);
```

spi_i2s_data_frame_size_config

The description of spi_i2s_data_frame_size_config is shown as below:

Table 3-1524. Function spi_i2s_data_frame_size_config

Function name	spi_i2s_data_frame_size_config
Function prototype	void spi_i2s_data_frame_size_config(uint32_t spi_periph, uint32_t frame_size);
Function descriptions	configure SPI/I2S data frame size
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
frame_size	SPI frame size
<i>SPI_DATASIZE_xBIT</i>	SPI x-bit data frame size (x=4,5,...32)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1525. Function spi_i2s_data_transmit

Function name	spi_i2s_data_transmit
Function prototype	void spi_i2s_data_transmit(uint32_t spi_periph, uint32_t data);
Function descriptions	SPI transmit data
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Input parameter{in}	
data	32-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 transmit data */
```

```
spi_i2s_data_transmit(SPI0, spi0_send_array[send_n]);
```

spi_i2s_data_receive

The description of spi_i2s_data_receive is shown as below:

Table 3-1526. Function spi_i2s_data_receive

Function name	spi_i2s_data_receive
Function prototype	Uint32_t spi_i2s_data_receive(uint32_t spi_periph);
Function descriptions	SPI receive data
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit data

Example:

```
/* SPI0 receive data */
```

```
spi0_receive_array[receive_n] = spi_i2s_data_receive(SPI0);
```

spi_bidirectional_transfer_config

The description of spi_bidirectional_transfer_config is shown as below:

Table 3-1527. Function spi_bidirectional_transfer_config

Function name	spi_bidirectional_transfer_config
Function prototype	void spi_bidirectional_transfer_config(uint32_t spi_periph, uint32_t transfer_direction);
Function descriptions	configure SPI bidirectional transfer direction
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
transfer_direction	SPI transfer direction
<i>SPI_BIDIRECTIONAL_TRANSMIT</i>	SPI work in transmit-only mode
<i>SPI_BIDIRECTIONAL_RECEIVE</i>	SPI work in receive-only mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 works in transmit-only mode */
```

```
spi_bidirectional_transfer_config(SPI0, SPI_BIDIRECTIONAL_TRANSMIT);
```

spi_master_transfer_start

The description of spi_master_transfer_start is shown as below:

Table 3-1528. Function spi_master_transfer_start

Function name	spi_master_transfer_start
Function prototype	void spi_master_transfer_start(uint32_t spi_periph, uint32_t transfer_start);
Function descriptions	SPI/I2S master start transfer
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
transfer_start	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1529. 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
<i>SPIx</i>	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 3-1530. 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
<i>SPIx</i>	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 3-1531. Function spi_crc_polynomial_set

Function name	spi_crc_polynomial_set
Function prototype	void spi_crc_polynomial_set(uint32_t spi_periph, uint32_t crc_poly);
Function descriptions	set SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
crc_poly	CRC polynomial value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SPI0 CRC polynomial */
spi_crc_polynomial_set(SPI0,CRC_VALUE);
```

spi_crc_polynomial_get

The description of spi_crc_polynomial_get is shown as below:

Table 3-1532. Function spi_crc_polynomial_get

Function name	spi_crc_polynomial_get
Function prototype	uint32_t spi_crc_polynomial_get(uint32_t spi_periph);
Function descriptions	get SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
uint16_t	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 3-1533. Function spi_crc_length_config

Function name	spi_crc_length_config
Function prototype	void spi_crc_length_config(uint32_t spi_periph, uint32_t crc_size);
Function descriptions	configure SPI CRC length
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
crc_size	crc size
<i>SPI_CRCSIZE_xBIT</i>	SPI x-bit crc size (x = 4,5,6...32)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1534. Function spi_crc_on

Function name	spi_crc_on
Function prototype	void spi_crc_on(uint32_t spi_periph);
Function descriptions	turn on CRC function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn on SPI0 CRC function */
spi_crc_on(SPI0);
```

spi_crc_off

The description of spi_crc_off is shown as below:

Table 3-1535. 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
<i>SPIx</i>	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 3-1536. 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
<i>SPIx</i>	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
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1537. Function spi_crc_full_size_enable

Function name	spi_crc_full_size_enable
Function prototype	void spi_crc_full_size_enable(uint32_t spi_periph);
Function descriptions	enable SPI CRC full size(33 bit or 17 bit) polynomial
Precondition	-
The called functions	-
Input parameter{in}	

spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	

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 3-1538. Function spi_crc_full_size_disable

Function name	spi_crc_full_size_disable
Function prototype	void spi_crc_full_size_disable(uint32_t spi_periph);
Function descriptions	disable SPI CRC full size(33 bit or 17 bit) polynomial
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	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_tcr_init_pattern

The description of spi_tcr_init_pattern is shown as below:

Table 3-1539. Function spi_tcr_init_pattern

Function name	spi_tcr_init_pattern
Function prototype	void spi_tcr_init_pattern(uint32_t spi_periph, uint32_t init_pattern);
Function descriptions	configure SPI TCRC init pattern
Precondition	-
The called functions	-

Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
init_pattern	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_tcrc_init_pattern(SPI0, SPI_TCRC_INIT_1);
```

spi_rcrc_init_pattern

The description of spi_rcrc_init_pattern is shown as below:

Table 3-1540. Function spi_rcrc_init_pattern

Function name	spi_rcrc_init_pattern
Function prototype	void spi_rcrc_init_pattern(uint32_t spi_periph, uint32_t init_pattern);
Function descriptions	configure SPI RCRC init pattern
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
init_pattern	SPI crc value
<i>SPI_RCRC_INIT_1</i>	use all 1 pattern
<i>SPI_RCRC_INIT_0</i>	use all 0 pattern
Output parameter{out}	
-	-
Return value	
-	-

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 3-1541. Function spi_ti_mode_enable

Function name	spi_ti_mode_enable
Function prototype	void spi_ti_mode_enable(uint32_t spi_periph);
Function descriptions	enable SPI TI mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	

Example:

```
/* enable SPI0 TI mode */
spi_ti_mode_enable(SPI0);
```

spi_ti_mode_disable

The description of spi_ti_mode_disable is shown as below:

Table 3-1542. Function spi_ti_mode_disable

Function name	spi_ti_mode_disable
Function prototype	void spi_ti_mode_disable(uint32_t spi_periph);
Function descriptions	disable SPI TI mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	

Example:

```
/* disable SPI0 TI mode */
spi_ti_mode_disable(SPI0);
```

spi_quad_enable

The description of spi_quad_enable is shown as below:

Table 3-1543. Function spi_quad_enable

Function name	spi_quad_enable
Function prototype	void spi_quad_enable (uint32_t spi_periph);
Function descriptions	enable quad wire SPI
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI3 quad wire mode */
```

```
spi_quad_enable(SPI3);
```

spi_quad_disable

The description of spi_quad_disable is shown as below:

Table 3-1544. Function spi_quad_disable

Function name	spi_quad_disable
Function prototype	void spi_quad_disable (uint32_t spi_periph);
Function descriptions	disable quad wire SPI
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

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 3-1545. Function spi_quad_write_enable

Function name	spi_quad_write_enable
Function prototype	void spi_quad_write_enable (uint32_t spi_periph);
Function descriptions	enable quad wire SPI write
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

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 3-1546. Function spi_quad_read_enable

Function name	spi_quad_read_enable
Function prototype	void spi_quad_read_enable (uint32_t spi_periph);
Function descriptions	enable quad wire SPI read
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

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 3-1547. Function spi_quad_io23_output_enable

Function name	spi_quad_io23_output_enable
Function prototype	void spi_quad_io23_output_enable(uint32_t spi_periph);
Function descriptions	enable SPI_IO2 and SPI_IO3 pin output
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

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 3-1548. Function spi_quad_io23_output_disable

Function name	spi_quad_io23_output_disable
Function prototype	void spi_quad_io23_output_disable(uint32_t spi_periph);
Function descriptions	disable SPI_IO2 and SPI_IO3 pin output
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

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 3-1549. Function spi_underrun_operation

Function name	spi_underrun_operation
Function prototype	void spi_underrun_operation(uint32_t spi_periph, uint32_t ur_ope);
Function descriptions	slave transmitter underrun detected operation
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
ur_ope	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
Output parameter{out}	
-	-
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 3-1550. Function spi_underrun_config

Function name	spi_underrun_config
Function prototype	void spi_underrun_config(uint32_t spi_periph, uint32_t ur_cfg);
Function descriptions	configure slave transmitter underrun detected
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	

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

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 3-1551. Function spi_underrun_data_config

Function name	spi_underrun_data_config
Function prototype	void spi_underrun_data_config(uint32_t spi_periph, uint32_t udata);
Function descriptions	configure underrun data at slave mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
udata	underrun data (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1552. Function spi_suspend_mode_config

Function name	spi_suspend_mode_config
Function prototype	void spi_suspend_mode_config(uint32_t spi_periph, uint32_t sus_mode);
Function descriptions	configure SPI suspend in receive mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
sus_mode	suspend mode
<i>SPI_AUTO_SUSPEND</i>	until the overrun condition is reached, the SPI stream is suspended in the full RxFIFO state
<i>SPI_CONTINUOUS</i>	SPI stream/clock generation is continuous whether or not an overrun occurs
Output parameter{out}	
-	-
Return value	
-	-

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 3-1553. Function spi_suspend_request

Function name	spi_suspend_request
Function prototype	void spi_suspend_request(uint32_t spi_periph);
Function descriptions	SPI master mode suspend request
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

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 3-1554. Function spi_related_ios_af_enable

Function name	spi_related_ios_af_enable
Function prototype	void spi_related_ios_af_enable(uint32_t spi_periph);
Function descriptions	enable SPI related IOs AF
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	

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 3-1555. Function spi_related_ios_af_disable

Function name	spi_related_ios_af_disable
Function prototype	void spi_related_ios_af_disable(uint32_t spi_periph);
Function descriptions	disable SPI related IOs AF
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	

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 3-1556. Function spi_af_gpio_control

Function name	spi_af_gpio_control
Function prototype	void spi_af_gpio_control(uint32_t spi_periph, uint32_t ctl);
Function descriptions	SPI af gpio control
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
ctl	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1557. Function spi_i2s_interrupt_enable

Function name	spi_i2s_interrupt_enable
Function prototype	void spi_i2s_interrupt_enable(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	enable SPI and I2S interrupt
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
interrupt	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_CONFE</i>	mode error interrupt
<i>SPI_I2S_INT_TXSERF</i>	TXSER reload interrupt
Output parameter{out}	
-	-
Return value	
-	-

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 3-1558. Function spi_i2s_interrupt_disable

Function name	spi_i2s_interrupt_disable
Function prototype	void spi_i2s_interrupt_disable(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	disable SPI and I2S interrupt
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
interrupt	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_CONFE</i>	mode error interrupt
<i>SPI_I2S_INT_TXSERF</i>	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 3-1559. 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
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
interrupt	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_XF</i>	transmission filled interrupt flag
<i>SPI_I2S_INT_FLAG_XURERR</i>	underrun error interrupt flag
<i>SPI_I2S_INT_FLAG_XORERR</i>	overrun error interrupt flag
<i>SPI_I2S_INT_FLAG_RCERR</i>	CRC error interrupt flag
<i>SPI_I2S_INT_FLAG_F</i>	TI frame error interrupt flag

<i>ERR</i>	
<i>SPI_I2S_INT_FLAG_C</i> <i>ONFERR</i>	mode error interrupt flag
<i>SPI_I2S_INT_FLAG_T</i> <i>XSERF</i>	TXSER reload interrupt flag
<i>SPI_I2S_INT_FLAG_S</i> <i>PD</i>	suspend interrupt flag
<i>SPI_I2S_INT_FLAG_T</i> <i>C</i>	TxFIFO clear interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-1560. Function spi_i2s_flag_get

Function name	spi_i2s_flag_get
Function prototype	FlagStatus spi_i2s_flag_get(uint32_t spi_periph, uint32_t flag);
Function descriptions	get SPI and I2S flag status
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
flag	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
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-1561. Function spi_i2s_flag_clear

Function name	spi_i2s_flag_clear
Function prototype	void spi_i2s_flag_clear(uint32_t spi_periph, uint32_t flag);
Function descriptions	clear SPI and I2S flag
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
flag	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</i> <i>RC</i>	clear the transmission underrun error flag
<i>SPI_STATC_RXORER</i> <i>RC</i>	clear the reception overrun error flag
<i>SPI_STATC_CRCERR</i> <i>C</i>	clear the CRC error flag
<i>SPI_STATC_FERRC</i>	clear the SPI TI format error flag
<i>SPI_STATC_CONFER</i> <i>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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1562. Function spi_i2s_rxfifo_plevel_get

Function name	spi_i2s_rxfifo_plevel_get
Function prototype	uint32_t spi_i2s_rxfifo_plevel_get(uint32_t spi_periph);
Function descriptions	get SPI and I2S RXFIFO packing level
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
uint32_t	2-bit RXFIFO packing data number

Example:

```
/* get SPI0 RxFIFO 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 3-1563. Function spi_i2s_remain_data_num_get

Function name	spi_i2s_remain_data_num_get
Function prototype	uint32_t spi_i2s_remain_data_num_get(uint32_t spi_periph);
Function descriptions	get SPI and I2S remaining data frames number in the TXSIZE session
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1564. Function spi_fifo_threshold_level_set

Function name	spi_fifo_threshold_level_set
Function prototype	void spi_fifo_threshold_level_set(uint32_t spi_periph, uint32_t fifo_thl);
Function descriptions	set SPI FIFO threshold level
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
fifo_thl	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)
Output parameter{out}	

-	-
Return value	
-	-

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 3-1565. Function spi_word_access_enable

Function name	spi_word_access_enable
Function prototype	void spi_word_access_enable(uint32_t spi_periph);
Function descriptions	enable SPI word access
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

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 3-1566. Function spi_word_access_disable

Function name	spi_word_access_disable
Function prototype	void spi_word_access_disable(uint32_t spi_periph);
Function descriptions	disable SPI word access
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5

Output parameter{out}	
-	-
Return value	
-	-

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 3-1567. Function spi_byte_access_enable

Function name	spi_byte_access_enable
Function prototype	void spi_byte_access_enable(uint32_t spi_periph);
Function descriptions	enable SPI byte access
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

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 3-1568. Function spi_byte_access_disable

Function name	spi_byte_access_disable
Function prototype	void spi_byte_access_disable(uint32_t spi_periph);
Function descriptions	disable SPI byte access
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral

<i>SPIx</i>	<i>x=0,1...5</i>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 byte access */
spi_byte_access_disable(SPI0);
```

3.42. SYSCFG

The SYSCFG registers are listed in chapter [3.42.1](#), and the SYSCFG firmware functions are introduced in chapter [3.42.2](#).

3.42.1. Descriptions of Peripheral registers

SYSCFG registers are listed in the table shown as below:

Table 3-1569. 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_CPSCTL	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
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

3.42.2. Descriptions of Peripheral functions

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

Table 3-1570. 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
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 3-1571. Enum timer_channel_input_enum

enum name	Function description
TIMER7_CIO_INPUT_TIMER7_CH0	select CMP1 output as TIMER7 CIO
TIMER7_CIO_INPUT_C	select TIMER7 CH0 as TIMER7 CIO

enum name	Function description
MP1_OUT	
TIMER7_CI1_INPUT_T IMER7_CH1	select TIMER7 CH1 as TIMER7 CI1
TIMER7_CI2_INPUT_T IMER7_CH2	select TIMER7 CH2 as TIMER7 CI2
TIMER7_CI3_INPUT_T IMER7_CH3	select TIMER7 CH3 as TIMER7 CI3
TIMER0_CIO_INPUT_T IMER0_CH0	select CMP0 output as TIMER0 CIO
TIMER0_CIO_INPUT_C MP0_OUT	select TIMER0 CH0 as TIMER0 CIO
TIMER0_CI1_INPUT_T IMER0_CH1	select TIMER0 CH1 as TIMER0 CI1
TIMER0_CI2_INPUT_T IMER0_CH2	select TIMER0 CH2 as TIMER0 CI2
TIMER0_CI3_INPUT_T IMER0_CH3	select TIMER0 CH3 as TIMER0 CI3
TIMER2_CIO_INPUT_T IMER2_CH0	select TIMER2 CH0 as TIMER2 CIO
TIMER2_CIO_INPUT_C MP0_OUT	select CMP0 as TIMER2 CIO
TIMER2_CIO_INPUT_C MP1_OUT	select CMP1 as TIMER2 CIO
TIMER2_CIO_INPUT_C MP0_OR_CMP1_OUT	select CMP0 or CMP1 as TIMER2 CIO
TIMER2_CI1_INPUT_T IMER2_CH1	select TIMER2 CH1 as TIMER2 CI1
TIMER2_CI2_INPUT_T IMER2_CH2	select TIMER2 CH2 as TIMER2 CI2
TIMER2_CI3_INPUT_T IMER2_CH3	select TIMER2 CH3 as TIMER2 CI3
TIMER1_CIO_INPUT_T IMER1_CH0	select TIMER1 CH0 as TIMER1 CIO
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	select CMP1 output as TIMER1 CI3

enum name	Function description
MP1_OUT	
TIMER1_CI3_INPUT_C MP0_OR_CMP1_OUT	select CMP0 or CMP1 output as TIMER1 CI3
TIMER4_CI0_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_CI0_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_CI0_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_CI0_INPUT_ TIMER22_CH0	select TIMER22 CH0 as TIMER22 CI0
TIMER22_CI1_INPUT_ TIMER22_CH1	select TIMER22 CH1 as TIMER22 CI1
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

enum name	Function description
TIMER31_CIO_INPUT_ TIMER31_CH0	select TIMER31 CH0 as TIMER31 CIO
TIMER31_CIO_INPUT_ CMP0_OUT	select CMP0 output as TIMER31 CIO
TIMER31_CIO_INPUT_ CMP1_OUT	select CMP1 output as TIMER31 CIO
TIMER31_CIO_INPUT_ CMP0_OR_CMP1_OUT	select CMP0 or CMP1 output as TIMER31 CIO
TIMER31_CIO_INPUT_ TIMER31_CH1	select TIMER31 CH1 as TIMER31 CIO
TIMER31_CIO_INPUT_ TIMER31_CH2	select TIMER31 CH2 as TIMER31 CIO
TIMER31_CIO_INPUT_ TIMER31_CH3	select TIMER31 CH3 as TIMER31 CIO
TIMER30_CIO_INPUT_ TIMER30_CH0	select TIMER30 CH0 as TIMER30 CIO
TIMER30_CIO_INPUT_ CMP0_OUT	select CMP0 output as TIMER30 CIO
TIMER30_CIO_INPUT_ CMP1_OUT	select CMP1 output as TIMER30 CIO
TIMER30_CIO_INPUT_ CMP0_OR_CMP1_OUT	select CMP0 or CMP1 output as TIMER30 CIO
TIMER30_CIO_INPUT_ TIMER30_CH1	select TIMER30 CH1 as TIMER30 CIO
TIMER30_CIO_INPUT_ TIMER30_CH2	select TIMER30 CH2 as TIMER30 CIO
TIMER30_CIO_INPUT_ TIMER30_CH3	select TIMER30 CH3 as TIMER30 CIO
TIMER14_CIO_INPUT_ TIMER14_CH0	select TIMER14 CH0 as TIMER14 CIO
TIMER14_CIO_INPUT_ TIMER1_CH0	select TIMER1 CH0 as TIMER14 CIO
TIMER14_CIO_INPUT_ TIMER2_CH0	select TIMER2 CH0 as TIMER14 CIO
TIMER14_CIO_INPUT_ TIMER3_CH0	select TIMER3 CH0 as TIMER14 CIO
TIMER14_CIO_INPUT_ LXTAL	select LXTAL as TIMER14 CIO
TIMER14_CIO_INPUT_ LPIRC4M	select LPIRC4M as TIMER14 CIO

enum name	Function description
TIMER14_CIO_INPUT_ CKOUT1	select CKOUT1 as TIMER14 CIO
TIMER14_C11_INPUT_ TIMER14_CH1	select TIMER14 CH1 as TIMER14 C11
TIMER14_C11_INPUT_ TIMER1_CH1	select TIMER1 CH1 as TIMER14 C11
TIMER14_C11_INPUT_ TIMER2_CH1	select TIMER2 CH1 as TIMER14 C11
TIMER14_C11_INPUT_ TIMER3_CH1	select TIMER3 CH1 as TIMER14 C11
TIMER40_CIO_INPUT_ TIMER40_CH0	select TIMER40 CH0 as TIMER40 CIO
TIMER40_CIO_INPUT_ TIMER2_CH0	select TIMER2 CH0 as TIMER40 CIO
TIMER40_CIO_INPUT_ TIMER3_CH0	select TIMER3 CH0 as TIMER40 CIO
TIMER40_CIO_INPUT_ TIMER4_CH0	select TIMER4 CH0 as TIMER40 CIO
TIMER40_CIO_INPUT_ LXTAL	select LXTAL as TIMER40 CIO
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

enum name	Function description
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
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_IRC32K	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

enum name	Function description
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_ TIMER30_CH0	select TIMER30 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_CIO_INPUT_ TIMER43_CH1	select TIMER43 CH1 as TIMER43 CIO
TIMER43_CIO_INPUT_ TIMER22_CH1	select TIMER22 CH1 as TIMER43 CIO
TIMER43_CIO_INPUT_ TIMER23_CH1	select TIMER23 CH1 as TIMER43 CIO
TIMER43_CIO_INPUT_ TIMER30_CH1	select TIMER30 CH1 as TIMER43 CIO
TIMER44_CIO_INPUT_ TIMER44_CH0	select TIMER44 CH0 as TIMER44 CIO
TIMER44_CIO_INPUT_ TIMER23_CH0	select TIMER23 CH0 as TIMER44 CIO
TIMER44_CIO_INPUT_ TIMER30_CH0	select TIMER30 CH0 as TIMER44 CIO
TIMER44_CIO_INPUT_ TIMER31_CH0	select TIMER31 CH0 as TIMER44 CIO
TIMER44_CIO_INPUT_ LXTAL	select LXTAL as TIMER44 CIO
TIMER44_CIO_INPUT_ LPIRC4M	select LPIRC4M as TIMER44 CIO
TIMER44_CIO_INPUT_ CKOUT1	select CKOUT1 as TIMER44 CIO

enum name	Function description
TIMER44_C11_INPUT_ TIMER44_CH1	select TIMER44 CH1 as TIMER44 CI1
TIMER44_C11_INPUT_ TIMER23_CH1	select TIMER23 CH1 as TIMER44 CI1
TIMER44_C11_INPUT_ TIMER30_CH1	select TIMER30 CH1 as TIMER44 CI1
TIMER44_C11_INPUT_ TIMER31_CH1	select TIMER31 CH1 as TIMER44 CI1

syscfg_deinit

The description of syscfg_deinit is shown as below:

Table 3-1572. Function syscfg_deinit

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

Example:

```
/* reset SYSCFG registers */
syscfg_deinit();
```

syscfg_i2c_fast_mode_plus_enable

The description of syscfg_i2c_fast_mode_plus_enable is shown as below:

Table 3-1573. Function syscfg_i2c_fast_mode_plus_enable

Function name	syscfg_i2c_fast_mode_plus_enable
Function prototype	void syscfg_i2c_fast_mode_plus_enable(uint32_t i2c_fmp);
Function descriptions	enable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
Precondition	-
The called functions	-
Input parameter{in}	
i2c_fmp	I2C fast mode plus function
<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:

```
/* 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 3-1574. Function `syscfg_i2c_fast_mode_plus_disable`

Function name	<code>syscfg_i2c_fast_mode_plus_disable</code>
Function prototype	<code>void syscfg_i2c_fast_mode_plus_disable(uint32_t i2c_fmp);</code>
Function descriptions	disable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
Precondition	-
The called functions	-
Input parameter{in}	
i2c_fmp	I2C fast mode plus function
<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

SYSCFG_I2C_FMP_P B9	I2C fast mode plus on PB9 pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C0 fast mode plus function */
syscfg_i2c_fast_mode_plus_disable(SYSCFG_I2C0_FMP);
```

syscfg_analog_switch_enable

The description of syscfg_analog_switch_enable is shown as below:

Table 3-1575. Function syscfg_analog_switch_enable

Function name	syscfg_analog_switch_enable
Function prototype	void syscfg_analog_switch_enable(uint32_t gpio_answ);
Function descriptions	open analog switch
Precondition	-
The called functions	-
Input parameter{in}	
gpio_answ	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
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 3-1576. Function syscfg_analog_switch_disable

Function name	syscfg_analog_switch_disable
Function prototype	void syscfg_analog_switch_disable(uint32_t gpio_answ);
Function descriptions	close analog switch
Precondition	-
The called functions	-
Input parameter{in}	
gpio_answ	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:

```
/* 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 3-1577. Function syscfg_enet_phy_interface_config

Function name	syscfg_enet_phy_interface_config
Function prototype	void syscfg_enet_phy_interface_config(uint32_t ethernet, uint32_t phy_interface);
Function descriptions	configure the PHY interface for the ethernet MAC
Precondition	-
The called functions	-
Input parameter{in}	
ethernet	Ethernet
<i>ENET0</i>	Ethernet 0
<i>ENET1</i>	Ethernet 1
Input parameter{in}	
phy_interface	specifies the media interface mode
<i>SYSCFG_ENET_PHY_MII</i>	MII mode is selected

<i>MII</i>	
<i>SYSCFG_ENET_PHY_</i> <i>RMII</i>	RMII mode is selected
Output parameter{out}	
-	-
Return value	
-	-

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 3-1578. Function syscfg_exti_line_config

Function name	syscfg_exti_line_config
Function prototype	void syscfg_exti_line_config(uint8_t exti_port, uint8_t exti_pin);
Function descriptions	configure the GPIO pin as EXTI Line
Precondition	-
The called functions	-
Input parameter{in}	
exti_port	specify the GPIO port used in EXTI
<i>EXTI_SOURCE_GPIOx</i>	x = A,B,C,D,E,F,G,H,J,K
Input parameter{in}	
exti_pin	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1579. Function syscfg_lockup_enable

Function name	syscfg_lockup_enable
Function prototype	void syscfg_lockup_enable(uint32_t lockup);
Function descriptions	enable module lockup function
Precondition	-
The called functions	-
Input parameter{in}	
lockup	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1580. Function syscfg_lockup_disable

Function name	syscfg_lockup_disable
Function prototype	void syscfg_lockup_disable(uint32_t lockup);
Function descriptions	disable module lockup function
Precondition	-
The called functions	-

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

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 3-1581. Function syscfg_timer_input_source_select

Function name	syscfg_timer_input_source_select
Function prototype	void syscfg_timer_input_source_select(timer_channel_input_enum timer_input);
Function descriptions	select timer channel input source
Precondition	-
The called functions	-
Input parameter{in}	
timer_input	TIMER channel input select, refer to Table 3-1571. Enum timer_channel_input_enum
Output parameter{out}	
-	-

Return value	
-	-

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 3-1582. Function syscfg_compensation_config

Function name	syscfg_compensation_config
Function prototype	void syscfg_compensation_config(uint32_t syscfg_compensation);
Function descriptions	configure the I/O compensation cell
Precondition	-
The called functions	-
Input parameter{in}	
syscfg_compensation	syscfg compensation
SYSCFG_COMPENSATION_ENABLE	I/O compensation cell is enabled
SYSCFG_COMPENSATION_DISABLE	I/O compensation cell is disabled
Output parameter{out}	
-	-
Return value	
-	-

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 3-1583. Function syscfg_io_low_voltage_speed_optimization_enable

Function name	syscfg_io_low_voltage_speed_optimization_enable
Function prototype	void syscfg_io_low_voltage_speed_optimization_enable(void);
Function descriptions	enable I/O speed optimization, high-speed at low-voltage
Precondition	-
The called functions	-
Input parameter{in}	

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

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 3-1584. Function syscfg_io_low_voltage_speed_optimization_disable

Function name	syscfg_io_low_voltage_speed_optimization_disable
Function prototype	void syscfg_io_low_voltage_speed_optimization_disable(void);
Function descriptions	disable I/O speed optimization, high-speed at low-voltage
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1585. Function syscfg_pnmos_compensation_code_set

Function name	syscfg_pnmos_compensation_code_set
Function prototype	void syscfg_pnmos_compensation_code_set(uint32_t mos, uint32_t code);
Function descriptions	set P/N MOS compensation value
Precondition	-
The called functions	-
Input parameter{in}	
mos	P/N MOS

NMOS_COMPENSATION	NMOS
PMOS_COMPENSATION	PMOS
Input parameter{in}	
code	P/N MOS compensation value
Output parameter{out}	
-	-
Return value	
-	-

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 3-1586. Function syscfg_secure_sram_size_set

Function name	syscfg_secure_sram_size_set
Function prototype	void syscfg_secure_sram_size_set(uint32_t size);
Function descriptions	set secure SRAM size
Precondition	-
The called functions	-
Input parameter{in}	
size	secure 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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1587. Function syscfg_secure_sram_size_get

Function name	syscfg_secure_sram_size_get
Function prototype	uint32_t syscfg_secure_sram_size_get(void);
Function descriptions	get secure SRAM size
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1588. Function syscfg_bootmode_get

Function name	syscfg_bootmode_get
Function prototype	uint32_t syscfg_bootmode_get(void);
Function descriptions	get BOOT mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
uint32_t	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 3-1589. Function syscfg_tcm_wait_state_enable

Function name	syscfg_tcm_wait_state_enable
Function prototype	void syscfg_tcm_wait_state_enable(void);
Function descriptions	enable TCM wait state
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1590. Function syscfg_tcm_wait_state_disable

Function name	syscfg_tcm_wait_state_disable
Function prototype	void syscfg_tcm_wait_state_disable(void);
Function descriptions	disable TCM wait state
Precondition	-

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

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 3-1591. Function syscfg_fpu_interrupt_enable

Function name	syscfg_fpu_interrupt_enable
Function prototype	void syscfg_fpu_interrupt_enable(uint32_t fpu_int);
Function descriptions	enable 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:

```
/* enable FPU inexact interrupt */
```

```
syscfg_fpu_interrupt_enable(SYS_CFG_FPUINT_INEXACT);
```

syscfg_fpu_interrupt_disable

The description of syscfg_fpu_interrupt_disable is shown as below:

Table 3-1592. 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
<i>SYS_CFG_FPUINT_INEXACT</i>	inexact interrupt
<i>SYS_CFG_FPUINT_INPUT_ABNORMAL</i>	input abnormal interrupt
<i>SYS_CFG_FPUINT_OVERFLOW</i>	overflow interrupt
<i>SYS_CFG_FPUINT_UNDERFLOW</i>	underflow interrupt
<i>SYS_CFG_FPUINT_DIV0</i>	divide-by-zero interrupt
<i>SYS_CFG_FPUINT_INVALID_OPERATION</i>	invalid operation interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable FPU inexact interrupt */
```

```
syscfg_fpu_interrupt_disable(SYS_CFG_FPUINT_INEXACT);
```

syscfg_compensation_flag_get

The description of syscfg_compensation_flag_get is shown as below:

Table 3-1593. Function syscfg_compensation_flag_get

Function name	syscfg_compensation_flag_get
Function prototype	FlagStatus syscfg_compensation_flag_get(uint32_t cps_flag);
Function descriptions	get compensation cell flags
Precondition	-

The called functions	-
Input parameter{in}	
cps_flag	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
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-1594. Function syscfg_cpu_cache_status_get

Function name	syscfg_cpu_cache_status_get
Function prototype	uint32_t syscfg_cpu_cache_status_get(uint32_t cache, uint32_t status);
Function descriptions	get the ICACHE or DCACHE detection and error information
Precondition	-
The called functions	-
Input parameter{in}	
cache	cache
<i>ICACHE_STATUS</i>	ICACHE status
<i>DCACHE_STATUS</i>	DCACHE status
Input parameter{in}	
status	status
<i>CPU_CACHE_ERROR_DETECTION</i>	select detection information
<i>CPU_CACHE_ERROR_BANK</i>	select error information
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1595. Function syscfg_brownout_reset_threshold_level_get

Function name	syscfg_brownout_reset_threshold_level_get
Function prototype	uint32_t syscfg_brownout_reset_threshold_level_get(void);
Function descriptions	get brownout reset threshold level
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	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();
```

3.43. 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, 30, 31), 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 [3.43.1](#), the TIMER firmware functions are introduced in chapter

[3.43.2.](#)

3.43.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

Table 3-1596. 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
TIMER_CH3COMV_ADD	TIMER channel 3 additional compare value register

Registers	Descriptions
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

3.43.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

Table 3-1597. 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
timer_prescaler_read	read TIMER prescaler value

Function name	Function description
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 3-1598. Structure timer_parameter_struct

Member name	Function description
prescaler	prescaler value(0~65535)
alignedmode	aligned mode(TIMER_COUNTER_EDGE,

Member name	Function description
	TIMER_COUNTER_CENTER_DOWN, TIMER_COUNTER_CENTER_UP, 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=1~6,22,23,30,31)), 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 3-1599. 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 3-1600. 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 3-1601. 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 3-1602. 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 3-1603. 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 3-1604. Function timer_deinit

Function name	timer_deinit
Function prototype	void timer_deinit(uint32_t timer_periph);
Function descriptions	deinit a TIMER
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 22, 23, 30, 31, 40~44, 50, 51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset TIMER0 */
timer_deinit(TIMER0);
```

timer_struct_para_init

The description of timer_struct_para_init is shown as below:

Table 3-1605. Function timer_struct_para_init

Function name	timer_struct_para_init
Function prototype	void timer_struct_para_init(timer_parameter_struct* initpara);

Function descriptions	initialize the parameters of TIMER init parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to Structure timer parameter struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER init parameter struct with a default value */
```

```
timer_parameter_struct timer_initpara;
```

```
timer_struct_para_init(&timer_initpara);
```

timer_init

The description of timer_init is shown as below:

Table 3-1606. Function timer_init

Function name	timer_init
Function prototype	void timer_init(uint32_t timer_periph, timer_parameter_struct* initpara);
Function descriptions	initialize TIMER counter
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 22, 23, 30, 31, 40~44, 50, 51)</i>	TIMER peripheral selection
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to Structure timer parameter struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER0 */
```

```

timer_parameter_struct timer_initpara;

timer_initpara.prescaler      = 107;

timer_initpara.alignedmode    = TIMER_COUNTER_EDGE;

timer_initpara.counterdirection = TIMER_COUNTER_UP;

timer_initpara.period         = 999;

timer_initpara.clockdivision   = TIMER_CKDIV_DIV1;

timer_initpara.repetitioncounter = 1;

timer_init(TIMERO,&timer_initpara);

```

timer_enable

The description of timer_enable is shown as below:

Table 3-1607. Function timer_enable

Function name	timer_enable
Function prototype	void timer_enable(uint32_t timer_periph);
Function descriptions	enable a TIMER
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 22, 23, 30, 31, 40~44, 50, 51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable TIMERO */

timer_enable(TIMERO);

```

timer_disable

The description of timer_disable is shown as below:

Table 3-1608. Function timer_disable

Function name	timer_disable
Function prototype	void timer_disable(uint32_t timer_periph);
Function descriptions	disable a TIMER

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~7, 14~16, 22, 23, 30, 31, 40~44, 50, 51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMERO */
```

```
timer_disable(TIMERO);
```

timer_auto_reload_shadow_enable

The description of timer_auto_reload_shadow_enable is shown as below:

Table 3-1609. Function timer_auto_reload_shadow_enable

Function name	timer_auto_reload_shadow_enable
Function prototype	void timer_auto_reload_shadow_enable(uint32_t timer_periph);
Function descriptions	enable the auto reload shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~7, 14~16, 22, 23, 30, 31, 40~44, 50, 51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TIMERO auto reload shadow function */
```

```
timer_auto_reload_shadow_enable(TIMERO);
```

timer_auto_reload_shadow_disable

The description of timer_auto_reload_shadow_disable is shown as below:

Table 3-1610. 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, 22, 23, 30, 31, 40~44, 50, 51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMER0 auto reload shadow function */
timer_auto_reload_shadow_disable(TIMER0);
```

timer_update_event_enable

The description of timer_update_event_enable is shown as below:

Table 3-1611. Function timer_update_event_enable

Function name	timer_update_event_enable
Function prototype	void timer_update_event_enable(uint32_t timer_periph);
Function descriptions	enable the update event
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~7, 14~16, 22, 23, 30, 31, 40~44, 50, 51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 the update event */
timer_update_event_enable (TIMER0);
```

timer_update_event_disable

The description of timer_update_event_disable is shown as below:

Table 3-1612. Function timer_update_event_disable

Function name	timer_update_event_disable
Function prototype	void timer_update_event_disable(uint32_t timer_periph);
Function descriptions	disable the update event
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 22, 23, 30, 31, 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 3-1613. Function timer_counter_alignment

Function name	timer_counter_alignment
Function prototype	void timer_counter_alignment(uint32_t timer_periph, uint16_t aligned);
Function descriptions	set TIMER counter alignment mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4, 7, 22, 23, 30, 31)	TIMER peripheral selection
Input parameter{in}	
aligned	alignment mode
TIMER_COUNTER_EDGE	edge-aligned mode
TIMER_COUNTER_CENTER_DOWN	center-aligned and counting down assert mode

<i>TIMER_COUNTER_CENTER_UP</i>	center-aligned and counting up assert mode
<i>TIMER_COUNTER_CENTER_BOTH</i>	center-aligned and counting up/down assert mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set TIMER0 counter center-aligned and counting up assert mode */
timer_counter_alignment(TIMER0, TIMER_COUNTER_CENTER_UP);
```

timer_counter_up_direction

The description of timer_counter_up_direction is shown as below:

Table 3-1614. Function timer_counter_up_direction

Function name	timer_counter_up_direction
Function prototype	void timer_counter_up_direction(uint32_t timer_periph);
Function descriptions	set TIMER counter up direction
Precondition	set TIMEx counter no center-aligned mode (edge-aligned mode)
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMEx(x=0~4, 7, 22, 23, 30, 31)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set TIMER0 counter up direction */
timer_counter_up_direction(TIMER0);
```

timer_counter_down_direction

The description of timer_counter_down_direction is shown as below:

Table 3-1615. Function timer_counter_down_direction

Function name	timer_counter_down_direction
Function prototype	void timer_counter_down_direction(uint32_t timer_periph);
Function descriptions	set TIMER counter down direction

Precondition	set TIMEx counter no center-aligned mode (edge-aligned mode)
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMEx(x=0~4,7,22,23,30,31)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set TIMER0 counter down direction */
timer_counter_down_direction(TIMER0);
```

timer_prescaler_config

The description of timer_prescaler_config is shown as below:

Table 3-1616. Function timer_prescaler_config

Function name	timer_prescaler_config
Function prototype	void timer_prescaler_config(uint32_t timer_periph, uint16_t prescaler, uint32_t pscreload);
Function descriptions	configure TIMER prescaler
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMEx(x=0~7,14~16,22,23,30,31,40~44,50,51)</i>	TIMER peripheral selection
Input parameter{in}	
prescaler	prescaler value (0~0xFFFF)
Input parameter{in}	
pscreload	prescaler reload mode
<i>TIMER_PSC_RELOAD_NOW</i>	the prescaler is loaded right now
<i>TIMER_PSC_RELOAD_UPDATE</i>	the prescaler is loaded at the next update event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 prescaler */

timer_prescaler_config(TIMER0, 3000, TIMER_PSC_RELOAD_NOW);
```

timer_repetition_value_config

The description of timer_repetition_value_config is shown as below:

Table 3-1617. Function timer_repetition_value_config

Function name	timer_repetition_value_config
Function prototype	void timer_repetition_value_config(uint32_t timer_periph, uint16_t ccsel, uint32_t repetition);
Function descriptions	configure TIMER repetition register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7,14~16,40~44)	TIMER peripheral selection
Input parameter{in}	
ccsel	repetition register selection
TIMER_CREP0_ENABLE	the update event rate is depended to TIMERx_CREP0 register
TIMER_CREP1_ENABLE	the update event rate is depended to TIMERx_CREP1 register
Input parameter{in}	
repetition	the counter repetition value (0~0xFF, use TIMER_CREP0 register; 0~0xFFFFFFFF, use TIMER_CREP1 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 3-1618. Function timer_runtime_repetition_value_read

Function name	timer_runtime_repetition_value_read
---------------	-------------------------------------

Function prototype	uint32_t timer_runtime_repetition_value_read(uint32_t timer_periph);
Function descriptions	read TIMER runtime repetition register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint32_t	counter repetition value in TIMER_CREP1 register (0~0xFFFFFFFF)

Example:

```
/* read TIMER0 runtime repetition register value */
uint32_t i = 0;
i = timer_runtime_repetition_value_read(TIMER0);
```

timer_autoreload_value_config

The description of timer_autoreload_value_config is shown as below:

Table 3-1619. Function timer_autoreload_value_config

Function name	timer_autoreload_value_config
Function prototype	void timer_autoreload_value_config(uint32_t timer_periph, uint64_t autoreload);
Function descriptions	configure TIMER autoreload register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)</i>	TIMER peripheral selection
Input parameter{in}	
autoreload	the counter auto-reload value 0~0xFFFF:TIMERx(x=0,2,3,7,14~16,30,31,40~44) 0~0xFFFFFFFF, TIMERx(x=1,4,5,6,22,23) 0~0xFFFFFFFFFFFFFFFF, TIMERx(x=50,51)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1620. Function timer_autoreload_value_read

Function name	timer_autoreload_value_read
Function prototype	uint64_t timer_autoreload_value_read(uint32_t timer_periph);
Function descriptions	read TIMER autoreload register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 22, 23, 30, 31, 40~44, 50, 51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint64_t	counter auto reload register value 0~0xFFFF, TIMERx(x=0,2,3,7,14~16,30,31,40~44) 0~0xFFFFFFFF, TIMERx(x=1,4,5,6,22,23) 0~0xFFFFFFFFFFFFFFFF, TIMERx(x=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 3-1621. Function timer_counter_value_config

Function name	timer_counter_value_config
Function prototype	void timer_counter_value_config(uint32_t timer_periph, uint64_t counter);
Function descriptions	configure TIMER counter register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral

<i>TIMERx</i> (<i>x</i> =0~7,14~16,22,23,30,31,40~44,50,51)	TIMER peripheral selection
Input parameter{in}	
counter	the counter value 0~0xFFFF, <i>TIMERx</i> (<i>x</i> =0,2,3,7,14~16,30,31,40~44) 0~0xFFFFFFFF, <i>TIMERx</i> (<i>x</i> =1,4,5,6,22,23) 0~0xFFFFFFFFFFFFFFFF, <i>TIMERx</i> (<i>x</i> =50,51)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 counter register value */
```

```
timer_counter_value_config(TIMER0, 999);
```

timer_counter_read

The description of timer_counter_read is shown as below:

Table 3-1622. Function timer_counter_read

Function name	timer_counter_read
Function prototype	uint64_t timer_counter_read(uint32_t timer_periph);
Function descriptions	read TIMER counter value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~7,14~16,22,23,30,31,40~44,50,51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint64_t	counter value 0~0xFFFF, <i>TIMERx</i> (<i>x</i> =0,2,3,7,14~16,30,31,40~44) 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:

```
/* 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 3-1623. Function timer_prescaler_read

Function name	timer_prescaler_read
Function prototype	uint16_t timer_prescaler_read(uint32_t timer_periph);
Function descriptions	read TIMER prescaler value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint16_t	prescaler register value (0~0xFFFF)

Example:

```
/* read TIMERO prescaler value */
```

```
uint16_t i = 0;
```

```
i = timer_prescaler_read(TIMERO);
```

timer_single_pulse_mode_config

The description of timer_single_pulse_mode_config is shown as below:

Table 3-1624. Function timer_single_pulse_mode_config

Function name	timer_single_pulse_mode_config
Function prototype	void timer_single_pulse_mode_config(uint32_t timer_periph, uint32_t spmode);
Function descriptions	configure TIMER single pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)</i>	TIMER peripheral selection
Input parameter{in}	
spmode	pulse mode
<i>TIMER_SP_MODE_SINGLE</i>	single pulse mode

<i>TIMER_SP_MODE_REPETITIVE</i>	repetitive pulse mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 single pulse mode */
```

```
timer_single_pulse_mode_config(TIMER0, TIMER_SP_MODE_SINGLE);
```

timer_delayable_single_pulse_mode_config

The description of timer_delayable_single_pulse_mode_config is shown as below:

Table 3-1625. Function timer_delayable_single_pulse_mode_config

Function name	timer_delayable_single_pulse_mode_config
Function prototype	void timer_delayable_single_pulse_mode_config(uint32_t timer_periph, uint16_t channel, uint32_t dspmode, uint16_t cnt_dir);
Function descriptions	configure timer delayable single pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,14,22,23,30,31,40~44)	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
<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)
Input parameter{in}	
dspmode	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
Input parameter{in}	

cnt_dir	counter direction selection
<i>TIMER_COUNTER_UP</i>	count up
<i>TIMER_COUNTER_DOWN</i>	count down
Output parameter{out}	
-	-
Return value	
-	-

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 3-1626. Function timer_update_source_config

Function name	timer_update_source_config
Function prototype	void timer_update_source_config(uint32_t timer_periph, uint32_t update);
Function descriptions	configure TIMER update source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 22, 23, 30, 31, 40~44, 50, 51)</i>	TIMER peripheral selection
Input parameter{in}	
update	update source
<i>TIMER_UPDATE_SRC_GLOBAL</i>	any of the following events generate an update interrupt or DMA request: <ul style="list-style-type: none"> – The UPG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event
<i>TIMER_UPDATE_SRC_REGULAR</i>	only counter overflow/underflow generates an update interrupt or DMA request.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER update only by counter overflow/underflow */
```

```
timer_update_source_config(TIMER0, TIMER_UPDATE_SRC_REGULAR);
```

timer_dma_enable

The description of timer_dma_enable is shown as below:

Table 3-1627. 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=0~7,14~16,22,23,30,31,40~44,50,51)	please refer to the following parameters
Input parameter{in}	
dma	timer DMA source enable
TIMER_DMA_UPD	update DMA request, TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)
TIMER_DMA_CH0D	channel 0 DMA request, TIMERx (x=0~4,7,14~16,22,23,30,31,40~44)
TIMER_DMA_CH1D	channel 1 DMA request, TIMERx (x=0~4,7,14,22,23,30,31,40~44)
TIMER_DMA_CH2D	channel 2 DMA request, TIMERx (x=0~4,7,22,23,30,31)
TIMER_DMA_CH3D	channel 3 DMA request, TIMERx (x=0~4,7,22,23,30,31)
TIMER_DMA_CMTD	commutation DMA request, TIMERx (x=0,7,14~16,40~44)
TIMER_DMA_TRGD	trigger DMA request, TIMERx (x=0~4,7,14,22,23,30,31,40~44)
TIMER_DMA_MCH0D	multi mode channel 0 DMA request, TIMERx (x=0,7,14~16,40~44)
TIMER_DMA_MCH1D	multi mode channel 1 DMA request, TIMERx (x=0,7)
TIMER_DMA_MCH2D	multi mode channel 2 DMA request, TIMERx (x=0,7)
TIMER_DMA_MCH3D	multi mode channel 3 DMA request, TIMERx (x=0,7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TIMER0 update DMA */
```

```
timer_dma_enable(TIMER0, TIMER_DMA_UPD);
```

timer_dma_disable

The description of timer_dma_disable is shown as below:

Table 3-1628. Function timer_dma_disable

Function name	timer_dma_disable
---------------	-------------------

Function prototype	void timer_dma_disable (uint32_t timer_periph, uint32_t dma);
Function descriptions	disable the TIMER DMA
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)	please refer to the following parameters
Input parameter{in}	
dma	timer DMA source disable
TIMER_DMA_UPD	update DMA request, TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)
TIMER_DMA_CH0D	channel 0 DMA request, TIMERx (x=0~4,7,14~16,22,23,30,31,40~44)
TIMER_DMA_CH1D	channel 1 DMA request, TIMERx (x=0~4,7,14,22,23,30,31,40~44)
TIMER_DMA_CH2D	channel 2 DMA request, TIMERx (x=0~4,7,22,23,30,31)
TIMER_DMA_CH3D	channel 3 DMA request, TIMERx (x=0~4,7,22,23,30,31)
TIMER_DMA_CMTD	commutation DMA request, TIMERx (x=0,7,14~16,40~44)
TIMER_DMA_TRGD	trigger DMA request, TIMERx (x=0~4,7,14,22,23,30,31,40~44)
TIMER_DMA_MCH0D	multi mode channel 0 DMA request, TIMERx (x=0,7,14~16,40~44)
TIMER_DMA_MCH1D	multi mode channel 1 DMA request, TIMERx (x=0,7)
TIMER_DMA_MCH2D	multi mode channel 2 DMA request, TIMERx (x=0,7)
TIMER_DMA_MCH3D	multi mode channel 3 DMA request, TIMERx (x=0,7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMER0 update DMA */
```

```
timer_dma_disable(TIMER0, TIMER_DMA_UPD);
```

timer_channel_dma_request_source_select

The description of timer_channel_dma_request_source_select is shown as below:

Table 3-1629. Function timer_channel_dma_request_source_select

Function name	timer_channel_dma_request_source_select
Function prototype	void timer_channel_dma_request_source_select(uint32_t timer_periph, uint32_t dma_request);
Function descriptions	channel DMA request source selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral

<i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)	TIMER peripheral selection
Input parameter{in}	
dma_request	channel DMA request source selection
<i>TIMER_DMAREQUEST_CHANNELEVENT</i>	DMA request of channel n is sent when channel n event occurs
<i>TIMER_DMAREQUEST_UPDATEEVENT</i>	DMA request of channel n is sent when update event occurs
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* TIMER0 channel DMA request of channel n is sent when channel 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 3-1630. Function timer_dma_transfer_config

Function name	timer_dma_transfer_config
Function prototype	void timer_dma_transfer_config(uint32_t timer_periph, uint32_t dma_baseaddr, uint32_t dma_lenth);
Function descriptions	configure the TIMER DMA transfer
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
dma_baseaddr	DMA transfer access start address
<i>TIMER_DMACFG_DMA_TA_CTL0</i>	DMA transfer address is TIMER_CTL0, TIMERx(<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA_TA_CTL1</i>	DMA transfer address is TIMER_CTL1, TIMERx(<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA_TA_SMCFG</i>	DMA transfer address is TIMER_SMCFG, TIMERx(<i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA_TA_DMAINTEN</i>	DMA transfer address is TIMER_DMAINTEN, TIMERx(<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)

<i>TIMER_DMACFG_DMA</i> <i>TA_INTF</i>	DMA transfer address is <i>TIMER_INTF</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_SWEVG</i>	DMA transfer address is <i>TIMER_SWEVG</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_CHCTL0</i>	DMA transfer address is <i>TIMER_CHCTL0</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_CHCTL1</i>	DMA transfer address is <i>TIMER_CHCTL1</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_DMACFG_DMA</i> <i>TA_CHCTL2</i>	DMA transfer address is <i>TIMER_CHCTL2</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_CNT</i>	DMA transfer address is <i>TIMER_CNT</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_PSC</i>	DMA transfer address is <i>TIMER_PSC</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_CAR</i>	DMA transfer address is <i>TIMER_CAR</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_CREP0</i>	DMA transfer address is <i>TIMER_CREP0</i> , <i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH0CV</i>	DMA transfer address is <i>TIMER_CH0CV</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH1CV</i>	DMA transfer address is <i>TIMER_CH1CV</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,14,22,23,30,31)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH2CV</i>	DMA transfer address is <i>TIMER_CH2CV</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH3CV</i>	DMA transfer address is <i>TIMER_CH3CV</i> , <i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_DMACFG_DMA</i> <i>TA_CCHP</i>	DMA transfer address is <i>TIMER_CCHP</i> , <i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCHCTL0</i>	DMA transfer address is <i>TIMER_MCHCTL0</i> , <i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCHCTL1</i>	DMA transfer address is <i>TIMER_MCHCTL1</i> , <i>TIMERx</i> (<i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCHCTL2</i>	DMA transfer address is <i>TIMER_MCHCTL2</i> , <i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCH0CV</i>	DMA transfer address is <i>TIMER_MCH0CV</i> , <i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCH1CV</i>	DMA transfer address is <i>TIMER_MCH1CV</i> , <i>TIMERx</i> (<i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCH2CV</i>	DMA transfer address is <i>TIMER_TIMER_MCH2CV</i> , <i>TIMERx</i> (<i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCH3CV</i>	DMA transfer address is <i>TIMER_TIMER_MCH3CV</i> , <i>TIMERx</i> (<i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i>	DMA transfer address is <i>TIMER_CH0COMV_ADD</i> ,

<i>TA_CH0COMV_ADD</i>	TIMERx(x=0~4,7,14,22,23,30,31)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH1COMV_ADD</i>	DMA transfer address is TIMER_CH1COMV_ADD, TIMERx(x=0~4,7,14,22,23,30,31)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH2COMV_ADD</i>	DMA transfer address is TIMER_CH2COMV_ADD, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH3COMV_ADD</i>	DMA transfer address is TIMER_CH3COMV_ADD, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_DMACFG_DMA</i> <i>TA_CTL2</i>	DMA transfer address is TIMER_CTL2, (x=0~4,7,14~16,22,23,30,31,40~44)
<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 TIMER_AFCTL1, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_WDGCNT</i>	DMA transfer address is TIMER_WDGCNT, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_DMACFG_DMA</i> <i>TA_CREP1</i>	DMA transfer address is TIMER_CREP1, TIMERx(x=0,7,14~16,40~44)
Input parameter{in}	
dma_lenth	DMA transfer count
<i>TIMER_DMACFG_DMA</i> <i>TC_xTRANSFER</i>	(x=1~38), DMA transfer x time
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the TIMER0 DMA transfer */
```

```
timer_dma_transfer_config(TIMER0, TIMER_DMACFG_DMATA_CTL0,  
TIMER_DMACFG_DMATC_5TRANSFER);
```

timer_event_software_generate

The description of timer_event_software_generate is shown as below:

Table 3-1631. Function timer_event_software_generate

Function name	timer_event_software_generate
Function prototype	void timer_event_software_generate(uint32_t timer_periph, uint32_t event);
Function descriptions	software generate events
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)	please refer to the following parameters
Input parameter{in}	
event	the timer software event generation sources
TIMER_EVENT_SRC_UPG	update event, TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)
TIMER_EVENT_SRC_CH0G	channel 0 capture or compare event generation, TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)
TIMER_EVENT_SRC_CH1G	channel 1 capture or compare event generation, TIMERx(x=0~4,7,14,22,23,30,31,40~44)
TIMER_EVENT_SRC_CH2G	channel 2 capture or compare event generation, TIMERx(x=0~4,7,22,23,30,31)
TIMER_EVENT_SRC_CH3G	channel 3 capture or compare event generation, TIMERx(x=0~4,7,22,23,30,31)
TIMER_EVENT_SRC_CHMTG	channel commutation event generation, TIMERx(x=0,7,14~16,40~44)
TIMER_EVENT_SRC_TRGG	trigger event generation, TIMERx(x=0~4,7,14,22,23,30,31,40~44)
TIMER_EVENT_SRC_BREAK0G	BREAK0 event generation, TIMERx(x=0,7,14~16,40~44)
TIMER_EVENT_SRC_BREAK1G	BREAK1 event generation, TIMERx(x=0,7)
TIMER_EVENT_SRC_MCH0G	multi mode channel 0 capture or compare event generation, TIMERx(x=0,7,14~16,40~44)
TIMER_EVENT_SRC_MCH1G	multi mode channel 1 capture or compare event generation, TIMERx(x=0,7)
TIMER_EVENT_SRC_MCH2G	multi mode channel 2 capture or compare event generation, TIMERx(x=0,7)
TIMER_EVENT_SRC_MCH3G	multi mode channel 3 capture or compare event generation, TIMERx(x=0,7)
TIMER_EVENT_SRC_CH0COMADDG	channel 0 additional compare event generation, TIMERx(x=0~4,7,14,22,23,30,31)
TIMER_EVENT_SRC_CH1COMADDG	channel 1 additional compare event generation, TIMERx(x=0~4,7,14,22,23,30,31)

<i>TIMER_EVENT_SRC_C</i> <i>H2COMADDG</i>	channel 2 additional compare event generation, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_EVENT_SRC_C</i> <i>H3COMADDG</i>	channel 3 additional compare event generation, TIMERx(x=0~4,7,22,23,30,31)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* software generate update event*/
```

```
timer_event_software_generate(TIMER0, TIMER_EVENT_SRC_UPG);
```

timer_break_struct_para_init

The description of timer_break_struct_para_init is shown as below:

Table 3-1632. Function timer_break_struct_para_init

Function name	timer_break_struct_para_init
Function prototype	void timer_break_struct_para_init(timer_break_parameter_struct* breakpara);
Function descriptions	initialize the parameters of TIMER break parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
breakpara	TIMER break parameter struct, the structure members can refer to <u>Structure timer break parameter struct.</u>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER break parameter struct with a default value */
```

```
timer_break_parameter_struct timer_breakpara;
```

```
timer_break_struct_para_init(&timer_breakpara);
```

timer_break_config

The description of timer_break_config is shown as below:

Table 3-1633. Function timer_break_config

Function name	timer_break_config
Function prototype	void timer_break_config(uint32_t timer_periph, timer_break_parameter_struct* breakpara);
Function descriptions	configure TIMER break function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMEx(x=0,7,14~16,40~44)	TIMER peripheral selection
Input parameter{in}	
breakpara	TIMER break parameter struct, the structure members can refer to Structure timer_break_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

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(TIMERO, &timer_breakpara);
```

timer_break_enable

The description of timer_break_enable is shown as below:

Table 3-1634. 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
<i>TIMERx(x=0,7,14~16,40~44)</i>	TIMER peripheral selection
Input parameter{in}	
break_num	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)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMERO BREAK0 function*/
```

```
timer_break_enable(TIMERO, TIMER_BREAK0);
```

timer_break_disable

The description of timer_break_disable is shown as below:

Table 3-1635. Function timer_break_disable

Function name	timer_break_disable
Function prototype	void timer_break_disable(uint32_t timer_periph, uint16_t break_num);
Function descriptions	disable TIMER break function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	TIMER peripheral selection

~44)	
Input parameter{in}	
break_num	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)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1636. Function timer_automatic_output_enable

Function name	timer_automatic_output_enable
Function prototype	void timer_automatic_output_enable(uint32_t timer_periph);
Function descriptions	enable TIMER output automatic function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 output automatic function */
```

```
timer_automatic_output_enable(TIMER0);
```

timer_automatic_output_disable

The description of timer_automatic_output_disable is shown as below:

Table 3-1637. Function timer_automatic_output_disable

Function name	timer_automatic_output_disable
Function prototype	void timer_automatic_output_disable (uint32_t timer_periph);
Function descriptions	disable TIMER output automatic function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 output automatic function */
```

```
timer_automatic_output_disable(TIMER0);
```

timer_primary_output_config

The description of timer_primary_output_config is shown as below:

Table 3-1638. Function timer_primary_output_config

Function name	timer_primary_output_config
Function prototype	void timer_primary_output_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	configure TIMER primary output function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 primary output function */

timer_primary_output_config(TIMER0, ENABLE);
```

timer_channel_control_shadow_config

The description of timer_channel_control_shadow_config is shown as below:

Table 3-1639. Function timer_channel_control_shadow_config

Function name	timer_channel_control_shadow_config
Function prototype	void timer_channel_control_shadow_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	configure channel commutation control shadow register
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7,14~16,40~44)	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* channel commutation control shadow register enable */

timer_channel_control_shadow_config(TIMER0, ENABLE);
```

timer_channel_control_shadow_update_config

The description of timer_channel_control_shadow_update_config is shown as below:

Table 3-1640. Function timer_channel_control_shadow_update_config

Function name	timer_channel_control_shadow_update_config
Function prototype	void timer_channel_control_shadow_update_config(uint32_t timer_periph, uint32_t ccuctl);
Function descriptions	configure TIMER channel control shadow register update control
Precondition	-
The called functions	-

Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	TIMER peripheral selection
Input parameter{in}	
ccuctl	channel control shadow register update control
<i>TIMER_UPDATECTL_CCU</i>	the shadow registers update by when CMTG bit is set
<i>TIMER_UPDATECTL_CUTRI</i>	the shadow registers update by when CMTG bit is set or an rising edge of TRGI occurs
<i>TIMER_UPDATECTL_CCUOVER</i>	the shadow registers update by when the overflow event occurs
<i>TIMER_UPDATECTL_CCUUNDER</i>	the shadow registers update by when the underflow event occurs
<i>TIMER_UPDATECTL_CCUOVERUNDER</i>	the shadow registers update by when the overflow or underflow event occurs
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel control shadow register update when CMTG bit is set */
timer_channel_control_shadow_update_config(TIMER0, TIMER_UPDATECTL_CCU);
```

timer_channel_output_struct_para_init

The description of timer_channel_output_struct_para_init is shown as below:

Table 3-1641. Function timer_channel_output_struct_para_init

Function name	timer_channel_output_struct_para_init
Function prototype	void timer_channel_output_struct_para_init(timer_oc_parameter_struct* ocpa);
Function descriptions	initialize the parameters of TIMER channel output parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
ocpa	TIMER channel output parameter struct, the structure members can refer to Structure timer oc parameter struct .
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* initialize TIMER channel output parameter struct with a default value */
```

```
timer_oc_parameter_struct timer_ocinitpara;
```

```
timer_channel_output_struct_para_init(&timer_ocinitpara);
```

timer_channel_output_config

The description of timer_channel_output_config is shown as below:

Table 3-1642. Function timer_channel_output_config

Function name	timer_channel_output_config
Function prototype	void timer_channel_output_config(uint32_t timer_periph, uint16_t channel, timer_oc_parameter_struct* ocpa);
Function descriptions	configure TIMER channel output function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx (x=0~4,7,14~16,22,23,30,31,40~44))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_CH_2	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
TIMER_CH_3	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
Input parameter{in}	
ocpara	TIMER channel output parameter struct, the structure members can refer to Structure timer_oc_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output function */
```

```
timer_oc_parameter_struct timer_ocinitpara;
```

```
timer_ocinitpara.outputstate = TIMER_CCX_ENABLE;
```

```
timer_ocinitpara.outputnstate = TIMER_CCXN_ENABLE;
```

```

timer_ocinitpara.ocpolarity   = TIMER_OC_POLARITY_HIGH;

timer_ocinitpara.ocnpolarity  = TIMER_OCN_POLARITY_HIGH;

timer_ocinitpara.ocidlestate  = TIMER_OC_IDLE_STATE_HIGH;

timer_ocinitpara.ocnidlestate = TIMER_OCN_IDLE_STATE_LOW;

timer_channel_output_config(TIMER0, TIMER_CH_0, &timer_ocinitpara);

```

timer_channel_output_mode_config

The description of timer_channel_output_mode_config is shown as below:

Table 3-1643. Function timer_channel_output_mode_config

Function name	timer_channel_output_mode_config
Function prototype	void timer_channel_output_mode_config(uint32_t timer_periph, uint16_t channel, uint32_t ocmode);
Function descriptions	configure TIMER channel output compare mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0(TIMERx(x=0~4,7,14~16,22,23,30,31,40~44))
TIMER_CH_1	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
TIMER_CH_2	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
TIMER_CH_3	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
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}	
ocmode	channel output compare mode
TIMER_OC_MODE_TIMING	timing mode
TIMER_OC_MODE_ACTIVE	set the channel output
TIMER_OC_MODE_INACTIVE	clear the channel output
TIMER_OC_MODE_TOGGLE	toggle on match
TIMER_OC_MODE_LOW	force low mode

<i>W</i>	
<i>TIMER_OC_MODE_HIG</i> <i>H</i>	force high mode
<i>TIMER_OC_MODE_PW</i> <i>M0</i>	PWM mode 0
<i>TIMER_OC_MODE_PW</i> <i>M1</i>	PWM mode 1
<i>TIMER_OC_MODE_DS</i> <i>PM0</i>	delayable SPM mode 0
<i>TIMER_OC_MODE_DS</i> <i>PM1</i>	delayable SPM mode 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel PWM 0 mode */
```

```
timer_channel_output_mode_config(TIMER0, TIMER_CH_0, TIMER_OC_MODE_PWM0);
```

timer_channel_output_pulse_value_config

The description of timer_channel_output_pulse_value_config is shown as below:

Table 3-1644. Function timer_channel_output_pulse_value_config

Function name	timer_channel_output_pulse_value_config
Function prototype	void timer_channel_output_pulse_value_config(uint32_t timer_periph, uint16_t channel, uint32_t pulse);
Function descriptions	configure TIMER channel output pulse value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,14~16, 22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
<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}	
pulse	channel output pulse value(0~65535)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output pulse value */
```

```
timer_channel_output_pulse_value_config(TIMER0, TIMER_CH_0, 399);
```

timer_channel_output_shadow_config

The description of timer_channel_output_shadow_config is shown as below:

Table 3-1645. Function timer_channel_output_shadow_config

Function name	timer_channel_output_shadow_config
Function prototype	void timer_channel_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t ocshadow);
Function descriptions	configure TIMER channel output shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
<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}	
ocshadow	channel output compare shadow
<i>TIMER_OC_SHADOW_ENABLE</i>	channel output compare shadow enable
<i>TIMER_OC_SHADOW_DISABLE</i>	channel output compare shadow disable

<i>DISABLE</i>	
<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 3-1646. Function timer_channel_output_clear_config

Function name	timer_channel_output_clear_config
Function prototype	void timer_channel_output_clear_config(uint32_t timer_periph, uint16_t channel, uint16_t occlear);
Function descriptions	configure TIMER channel output clear function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER periphera
<i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(<i>TIMERx</i> , <i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(<i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
<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)
Input parameter{in}	
occlear	channel output clear function
<i>TIMER_OC_CLEAR_EN</i>	channel output clear function enable

<i>ABLE</i>	
<i>TIMER_OC_CLEAR_DISABLE</i>	channel output clear function disable
<i>TIMER_OMC_CLEAR_ENABLE</i>	multi mode channel output clear function enable
<i>TIMER_OMC_CLEAR_DISABLE</i>	multi mode channel output clear function disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output clear function */
```

```
timer_channel_output_clear_config (TIMER0, TIMER_CH_0,  
TIMER_OC_CLEAR_ENABLE);
```

timer_channel_output_polarity_config

The description of timer_channel_output_polarity_config is shown as below:

Table 3-1647. Function timer_channel_output_polarity_config

Function name	timer_channel_output_polarity_config
Function prototype	void timer_channel_output_polarity_config(uint32_t timer_periph, uint16_t channel, uint16_t ocpolarity);
Function descriptions	configure TIMER channel output polarity
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
<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}	

ocpolarity	channel output polarity
<i>TIMER_OC_POLARITY_HIGH</i>	channel output polarity is high
<i>TIMER_OC_POLARITY_LOW</i>	channel output polarity is low
<i>TIMER_OMC_POLARITY_HIGH</i>	multi mode channel output polarity is high
<i>TIMER_OMC_POLARITY_LOW</i>	multi mode channel output polarity is low
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output polarity */
```

```
timer_channel_output_polarity_config(TIMER0, TIMER_CH_0,  
TIMER_OC_POLARITY_HIGH);
```

timer_channel_complementary_output_polarity_config

The description of timer_channel_complementary_output_polarity_config is shown as below:

Table 3-1648. Function timer_channel_complementary_output_polarity_config

Function name	timer_channel_complementary_output_polarity_config
Function prototype	void timer_channel_complementary_output_polarity_config(uint32_t timer_periph, uint16_t channel, uint16_t ocnpolarity);
Function descriptions	configure TIMER channel complementary output polarity
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (x=0~4,7,14~16,22,23,30,31,40~44)	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
Input parameter{in}	
ocpolarity	channel complementary output polarity
<i>TIMER_OCN_POLARITY</i>	channel complementary output polarity is high

<i>Y_HIGH</i>	
<i>TIMER_OCN_POLARITY_LOW</i>	channel complementary output polarity is low
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 complementary output polarity */
```

```
timer_channel_complementary_output_polarity_config(TIMER0, TIMER_CH_0,  
TIMER_OCN_POLARITY_HIGH);
```

timer_channel_output_state_config

The description of timer_channel_output_state_config is shown as below:

Table 3-1649. Function timer_channel_output_state_config

Function name	timer_channel_output_state_config
Function prototype	void timer_channel_output_state_config(uint32_t timer_periph, uint16_t channel, uint32_t state);
Function descriptions	configure TIMER channel enable state
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, <i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, <i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, <i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, <i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, <i>x</i> =0,7,14~16,40~44)
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx, <i>x</i> =0,7)
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx, <i>x</i> =0,7)
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx, <i>x</i> =0,7)
Input parameter{in}	
state	TIMER channel enable state
<i>TIMER_CCX_ENABLE</i>	channel enable
<i>TIMER_CCX_DISABLE</i>	channel disable
<i>TIMER_MCCX_ENABLE</i>	multi mode channel enable

<i>TIMER_MCCX_DISABL</i> <i>E</i>	multi mode channel disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 enable state */
```

```
timer_channel_output_state_config(TIMER0, TIMER_CH_0, TIMER_CCX_ENABLE);
```

timer_channel_complementary_output_state_config

The description of timer_channel_complementary_output_state_config is shown as below:

Table 3-1650. Function timer_channel_complementary_output_state_config

Function name	timer_channel_complementary_output_state_config
Function prototype	void timer_channel_complementary_output_state_config(uint32_t timer_periph, uint16_t channel, uint16_t ocnstate);
Function descriptions	configure TIMER channel complementary output enable state
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
Input parameter{in}	
channel	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)
Input parameter{in}	
state	TIMER channel complementary output enable state
<i>TIMER_CCXN_ENABLE</i>	channel complementary enable
<i>TIMER_CCXN_DISABL</i> <i>E</i>	channel complementary disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 complementary output enable state */
```

```
timer_channel_complementary_output_state_config(TIMER0, TIMER_CH_0,
TIMER_CCXN_ENABLE);
```

timer_channel_input_struct_para_init

The description of timer_channel_input_struct_para_init is shown as below:

Table 3-1651. Function timer_channel_input_struct_para_init

Function name	timer_channel_input_struct_para_init
Function prototype	void timer_channel_input_struct_para_init(timer_ic_parameter_struct* icpara);
Function descriptions	initialize the parameters of TIMER channel input parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
icpara	TIMER channel input parameter struct, the structure members can refer to Structure timer_ic_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER channel input parameter struct with a default value */
```

```
timer_ic_parameter_struct timer_icinitpara;
```

```
timer_channel_input_struct_para_init(&timer_icinitpara);
```

timer_input_capture_config

The description of timer_input_capture_config is shown as below:

Table 3-1652. 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,	please refer to the following parameters

22,23,30,31,40~44)	
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
<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}	
icpara	TIMER channel input parameter struct, the structure members can refer to Structure timer_ic_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TIMER0 input capture parameter */

timer_ic_parameter_struct timer_icinitpara;

timer_icinitpara.icpolarity = TIMER_IC_POLARITY_RISING;

timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTTI;

timer_icinitpara.icprescaler = TIMER_IC_PSC_DIV1;

timer_icinitpara.icfilter = 0x0;

timer_input_capture_config(TIMER0, TIMER_CH_0, &timer_icinitpara);

```

timer_channel_input_capture_prescaler_config

The description of timer_channel_input_capture_prescaler_config is shown as below:

Table 3-1653. Function timer_channel_input_capture_prescaler_config

Function name	timer_channel_input_capture_prescaler_config
Function prototype	void timer_channel_input_capture_prescaler_config(uint32_t timer_periph, uint16_t channel, uint16_t prescaler);
Function descriptions	configure TIMER channel input capture prescaler value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral

<i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(<i>TIMERx</i> , <i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(<i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
<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)
Input parameter{in}	
prescaler	channel input capture prescaler value
<i>TIMER_IC_PSC_DIV1</i>	no prescaler
<i>TIMER_IC_PSC_DIV2</i>	divided by 2
<i>TIMER_IC_PSC_DIV4</i>	divided by 4
<i>TIMER_IC_PSC_DIV8</i>	divided by 8
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 input capture prescaler value */
```

```
timer_channel_input_capture_prescaler_config(TIMER0, TIMER_CH_0,  
TIMER_IC_PSC_DIV2);
```

timer_channel_capture_value_register_read

The description of timer_channel_capture_value_register_read is shown as below:

Table 3-1654. Function timer_channel_capture_value_register_read

Function name	timer_channel_capture_value_register_read
Function prototype	uint32_t timer_channel_capture_value_register_read(uint32_t timer_periph, uint16_t channel);
Function descriptions	read TIMER channel capture compare register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)	please refer to the following parameters

Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
<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)
Output parameter{out}	
-	-
Return value	
uint32_t	channel capture compare register value (0~0xFFFFFFFF)

Example:

```
/* read TIMER0 channel 0 capture compare register value */
```

```
uint32_t ch0_value = 0;
```

```
ch0_value = timer_channel_capture_value_register_read(TIMER0, TIMER_CH_0);
```

timer_input_pwm_capture_config

The description of timer_input_pwm_capture_config is shown as below:

Table 3-1655. Function timer_input_pwm_capture_config

Function name	timer_input_pwm_capture_config
Function prototype	void timer_input_pwm_capture_config(uint32_t timer_periph, uint16_t channel, timer_ic_parameter_struct* icpwm);
Function descriptions	configure TIMER input pwm capture function
Precondition	-
The called functions	timer_channel_input_capture_prescaler_config
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)</i>	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
Input parameter{in}	
icpwm	TIMER channel input PWM parameter struct, the structure members can refer to Structure timer_ic_parameter_struct .

Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TIMER0 input pwm capture parameter */

timer_ic_parameter_struct timer_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 3-1656. Function timer_hall_mode_config

Function name	timer_hall_mode_config
Function prototype	void timer_hall_mode_config(uint32_t timer_periph, uint32_t hallmode);
Function descriptions	configure TIMER hall sensor mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4, 7, 14~16, 22, 23, 30, 31, 40~44)	TIMER peripheral selection
Input parameter{in}	
hallmode	TIMER hall sensor mode state
TIMER_HALLINTERFA CE_ENABLE	TIMER hall sensor mode enable
TIMER_HALLINTERFA CE_DISABLE	TIMER hall sensor mode disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TIMER0 hall sensor mode */

```

timer_hall_mode_config (TIMER0, TIMER_HALLINTERFACE_ENABLE);

timer_multi_mode_channel_output_parameter_struct_init

The description of timer_multi_mode_channel_output_parameter_struct_init is shown as below:

Table 3-1657. Function timer_multi_mode_channel_output_parameter_struct_init

Function name	timer_multi_mode_channel_output_parameter_struct_init
Function prototype	void timer_multi_mode_channel_output_parameter_struct_init(timer_omc_parameter_struct *omcpara);
Function descriptions	initialize TIMER multi mode channel output parameter struct
Precondition	-
The called functions	-
Input parameter{in}	
omcpara	TIMER multi mode channel output parameter struct, the structure members can refer to Structure timer_omc_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER multi mode channel output parameter struct with a default value */
```

```
timer_omc_parameter_struct timer_omcinitpara;
```

```
timer_multi_mode_channel_output_parameter_struct_init(&timer_omcinitpara);
```

timer_multi_mode_channel_output_config

The description of timer_multi_mode_channel_output_config is shown as below:

Table 3-1658. Function timer_multi_mode_channel_output_config

Function name	timer_multi_mode_channel_output_config
Function prototype	void timer_multi_mode_channel_output_config(uint32_t timer_periph, uint16_t channel, timer_omc_parameter_struct *omcpara);
Function descriptions	configure TIMER multi mode channel output function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	please refer to the following parameters
Input parameter{in}	

channel	TIMER channel
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
<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}	
omcpara	TIMER multi mode channel output parameter struct, the structure members can refer to Structure timer omc parameter struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 multi mode channel 0 output function */
```

```
timer_omc_parameter_struct timer_omcinitpara;
```

```
omcpara->outputmode = TIMER_MCH_MODE_INDEPENDENTLY;
```

```
omcpara->outputstate = TIMER_MCCX_ENABLE;
```

```
omcpara->ocpolarity = TIMER_OMC_POLARITY_HIGH;
```

```
timer_multi_mode_channel_output_parameter_struct_init(TIMER0,          TIMER_MCH_0,
&timer_omcinitpara);
```

timer_multi_mode_channel_mode_config

The description of timer_multi_mode_channel_mode_config is shown as below:

Table 3-1659. Function timer_multi_mode_channel_mode_config

Function name	timer_multi_mode_channel_mode_config
Function prototype	void timer_multi_mode_channel_mode_config(uint32_t timer_periph, uint32_t channel, uint32_t multi_mode_sel);
Function descriptions	multi mode channel mode select
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
<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}	
multi_mode_sel	multi mode channel mode selection
<i>TIMER_MCH_MODE_INDEPENDENTLY</i>	multi mode channel work in independently mode
<i>TIMER_MCH_MODE_COMPLEMENTARY</i>	multi mode channel work in complementary output mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TIMER0 multi mode channel 0 mode */
```

```
timer_multi_mode_channel_mode_config(TIMER0, TIMER_MCH_0,
TIMER_MCH_MODE_INDEPENDENTLY);
```

timer_input_trigger_source_select

The description of timer_input_trigger_source_select is shown as below:

Table 3-1660. Function timer_input_trigger_source_select

Function name	timer_input_trigger_source_select
Function prototype	void timer_input_trigger_source_select(uint32_t timer_periph, uint32_t intrigger);
Function descriptions	select TIMER input trigger source
Precondition	SMC[2:0] = 000
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,30,31,40~44)</i>	please refer to the following parameters
Input parameter{in}	
intrigger	input trigger source
<i>TIMER_SMCFG_TRGS_EL_ITI0</i>	internal trigger input 0 (ITI0, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS_EL_ITI1</i>	internal trigger input 1 (ITI1, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS_EL_ITI2</i>	internal trigger input 2 (ITI2, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS_EL_ITI3</i>	internal trigger input 3 (ITI3, TIMERx(x=0~4,7,14,22,23,30,31,40~44))

<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOF_ED</i>	TIO edge detector (CIOF_ED, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOFE0</i>	filtered channel channel 0 input (CIOFE0, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI1FE1</i>	filtered channel channel 1 input (CI1FE1, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ETIFP</i>	external trigger input filter output(ETIFP, TIMERx(x=0~4,7,22,23,30,31))
<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))
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI4</i>	internal trigger 4 for General-L0 timer(TIMERx(x=1,2,22,23,30,31))
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI5</i>	internal trigger 5 for General-L0 timer(TIMERx(x=1,22,23,30))
<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=0~4,7,23,30,31))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI13</i>	internal trigger input 13(TIMERx(x=0~4,7,22,30,31))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI14</i>	internal trigger input 14(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* select TIMER0 input trigger source */
```

```
timer_input_trigger_source_select (TIMER0, TIMER_SMCFG_TRGSEL_ITI0);
```

timer_master_output0_trigger_source_select

The description of timer_master_output0_trigger_source_select is shown as below:

Table 3-1661. Function timer_master_output0_trigger_source_select

Function name	timer_master_output0_trigger_source_select
Function prototype	void timer_master_output0_trigger_source_select(uint32_t timer_periph, uint32_t outrigger);
Function descriptions	select TIMER master mode output 0 trigger source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7,14,22,23,30,31,50,51)	TIMER peripheral selection
Input parameter{in}	
outrigger	trigger output source
TIMER_TRI_OUT0_SR_C_RESET	the UPG bit as trigger output 0 (TIMERx(x=0~7,14,22,23,30,31,50,51))
TIMER_TRI_OUT0_SR_C_ENABLE	the counter enable signal as trigger output 0(TIMERx(x=0~7,14,22,23,30,31,50,51))
TIMER_TRI_OUT0_SR_C_UPDATE	update event as trigger output 0(TIMERx(x=0~7,14,22,23,30,31,50,51))
TIMER_TRI_OUT0_SR_C_CH0	a capture or a compare match occurred in channel 0 as trigger output 0 (TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_TRI_OUT0_SR_C_O0CPRE	O0CPRE as trigger output 0(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_TRI_OUT0_SR_C_O1CPRE	O1CPRE as trigger output 0(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_TRI_OUT0_SR_C_O2CPRE	O2CPRE as trigger output 0(TIMERx(x=0~4,7,22,23,30,31))
TIMER_TRI_OUT0_SR_C_O3CPRE	O3CPRE as trigger output 0(TIMERx(x=0~4,7,22,23,30,31))
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* select TIMER0 master mode output 0 trigger source */
```

```
timer_master_output0_trigger_source_select (TIMER0, TIMER_TRI_OUT0_SRC_RESET);
```

timer_master_output1_trigger_source_select

The description of timer_master_output1_trigger_source_select is shown as below:

Table 3-1662. Function timer_master_output1_trigger_source_select

Function name	timer_master_output1_trigger_source_select
Function prototype	void timer_master_output1_trigger_source_select(uint32_t timer_periph, uint32_t outrigger);
Function descriptions	select TIMER master mode output 1 trigger source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
outrigger	trigger output source
TIMER_TRI_OUT1_SRC_RESET	the UPG bit as trigger output 1
TIMER_TRI_OUT1_SRC_ENABLE	the counter enable signal as trigger output 1
TIMER_TRI_OUT1_SRC_UPDATE	update event as trigger output 1
TIMER_TRI_OUT1_SRC_CH0	a capture or a compare match occurred in channel 0 as trigger output 1
TIMER_TRI_OUT1_SRC_O0CPRE	O0CPRE as trigger output 1
TIMER_TRI_OUT1_SRC_O1CPRE	O1CPRE as trigger output 1
TIMER_TRI_OUT1_SRC_O2CPRE	O2CPRE as trigger output 1
TIMER_TRI_OUT1_SRC_O3CPRE	O3CPRE as trigger output 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TIMER0 master mode output 1 trigger source */
```

```
timer_master_output1_trigger_source_select (TIMER0, TIMER_TRI_OUT1_SRC_RESET);
```

timer_slave_mode_select

The description of timer_slave_mode_select is shown as below:

Table 3-1663. Function timer_slave_mode_select

Function name	timer_slave_mode_select
Function prototype	void timer_slave_mode_select(uint32_t timer_periph, uint32_t slavemode);
Function descriptions	select TIMER slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14,22,23,30,31,40~44)	TIMER peripheral selection
Input parameter{in}	
slavemode	slave mode
TIMER_SLAVE_MODE_DISABLE	slave mode disable(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_ENCODER_MODE0	encoder mode 0(TIMERx(x=0~4,7,22,23,30,31))
TIMER_ENCODER_MODE1	encoder mode 1(TIMERx(x=0~4,7,22,23,30,31))
TIMER_ENCODER_MODE2	encoder mode 2(TIMERx(x=0~4,7,22,23,30,31))
TIMER_SLAVE_MODE_RESTART	restart mode(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_SLAVE_MODE_PAUSE	pause mode(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_SLAVE_MODE_EVENT	event mode(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_SLAVE_MODE_EXTERNAL0	external clock mode 0(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_SLAVE_MODE_RESTART_EVENT	restart + event mode(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
TIMER_NONQUAD_MODE0	non-quadrature decoder mode 0(TIMERx(x=0~4,7,22,23,30,31))
TIMER_NONQUAD_MODE1	non-quadrature decoder mode 1(TIMERx(x=0~4,7,22,23,30,31))
Output parameter{out}	

-	-
Return value	
-	-

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 3-1664. Function timer_master_slave_mode_config

Function name	timer_master_slave_mode_config
Function prototype	void timer_master_slave_mode_config(uint32_t timer_periph, uint32_t masterslave);
Function descriptions	configure TIMER master slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14,22,23,30,31,40~44)	TIMER peripheral selection
Input parameter{in}	
masterslave	master slave mode state
TIMER_MASTER_SLAVE_MODE_ENABLE	master slave mode enable
TIMER_MASTER_SLAVE_MODE_DISABLE	master slave mode disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 master slave mode */
```

```
timer_master_slave_mode_config(TIMER0, TIMER_MASTER_SLAVE_MODE_ENABLE);
```

timer_external_trigger_config

The description of timer_external_trigger_config is shown as below:

Table 3-1665. 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=0~4,7,22,23,30,31)	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 3-1666. Function timer_quadrature_decoder_mode_config

Function name	timer_quadrature_decoder_mode_config
Function prototype	void timer_quadrature_decoder_mode_config(uint32_t timer_periph, uint32_t decomode, uint16_t ic0polarity, uint16_t ic1polarity);

Function descriptions	configure TIMER quadrature decoder mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)	TIMER peripheral selection
Input parameter{in}	
decomode	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
Input parameter{in}	
ic0polarity	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
Input parameter{in}	
ic1polarity	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1667. Function timer_non_quadrature_decoder_mode_config

Function name	timer_non_quadrature_decoder_mode_config
Function prototype	void timer_non_quadrature_decoder_mode_config(uint32_t timer_periph, uint32_t decompode, uint16_t ic1polarity);
Function descriptions	configure TIMER non-quadrature decoder mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,22,23,30,31)	TIMER peripheral selection
Input parameter{in}	
decompode	quadrature decoder mode
TIMER_NONQUAD_MODE0	non-quadrature decoder mode 0
TIMER_NONQUAD_MODE1	non-quadrature decoder mode 1
Input parameter{in}	
ic1polarity	CI1 input capture polarity
TIMER_IC_POLARITY_RISING	capture rising edge
TIMER_IC_POLARITY_FALLING	capture falling edge
Output parameter{out}	
-	-
Return value	
-	-

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 3-1668. Function timer_internal_clock_config

Function name	timer_internal_clock_config
Function prototype	void timer_internal_clock_config(uint32_t timer_periph);

Function descriptions	configure TIMER internal clock mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,30,31,40~44)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 internal clock mode */
```

```
timer_internal_clock_config (TIMER0);
```

timer_internal_trigger_as_external_clock_config

The description of timer_internal_trigger_as_external_clock_config is shown as below:

Table 3-1669. Function timer_internal_trigger_as_external_clock_config

Function name	timer_internal_trigger_as_external_clock_config
Function prototype	void timer_internal_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t intrigger);
Function descriptions	configure TIMER the internal trigger as external clock input
Precondition	-
The called functions	timer_input_trigger_source_select
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,30,31,40~44)</i>	TIMER peripheral selection
Input parameter{in}	
intrigger	trigger selection
<i>TIMER_SMCFG_TRGS_EL_ITI0</i>	internal trigger input 0 (ITI0, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS_EL_ITI1</i>	internal trigger input 1 (ITI1, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS_EL_ITI2</i>	internal trigger input 2 (ITI2, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS_EL_ITI3</i>	internal trigger input 3 (ITI3, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_L0_SMCFG_TRGS_EL_ITI4</i>	internal trigger 4 for General-L0 timer(TIMERx(x=1,2,22,23,30,31))

<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI5</i>	internal trigger 5 for General-L0 timer(TIMERx(x=1,22,23,30))
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI7</i>	internal trigger 7 for General-L0 timer(TIMERx(x=4))
<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=0~4,7,23,30,31))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI13</i>	internal trigger input 13(TIMERx(x=0~4,7,22,30,31))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI14</i>	internal trigger input 14(TIMERx(x=0~4,7,14,22,23,30,31,40~44))
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 the internal trigger ITI0 as external clock input */
```

```
timer_internal_trigger_as_external_clock_config(TIMER0, TIMER_SMCFG_TRGSEL_ITI0);
```

timer_external_trigger_as_external_clock_config

The description of timer_external_trigger_as_external_clock_config is shown as below:

Table 3-1670. Function timer_external_trigger_as_external_clock_config

Function name	timer_external_trigger_as_external_clock_config
Function prototype	void timer_external_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t extrigger, uint16_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER the external trigger as external clock input
Precondition	-
The called functions	timer_input_trigger_source_select
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,30,31,40~44)</i>	TIMER peripheral selection
Input parameter{in}	
extrigger	external trigger selection
<i>TIMER_SMCFG_TRGS</i>	TI0 edge detector (CI0F_ED, TIMERx(x=0~4,7,14,22,23,30,31,40~44))

<i>EL_CIOF_ED</i>	
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOFE0</i>	filtered channel channel 0 input (CIOFE0, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI1FE1</i>	filtered channel channel 1 input (CI1FE1, TIMERx(x=0~4,7,14,22,23,30,31,40~44))
<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))
Input parameter{in}	
expolarity	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
Input parameter{in}	
extfilter	external trigger filter control(0~15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 the external trigger CIOFE0 as external clock input */
```

```
timer_external_trigger_as_external_clock_config(TIMER0,  
TIMER_SMCFG_TRGSEL_CIOFE0, TIMER_IC_POLARITY_RISING, 0);
```

timer_external_clock_mode0_config

The description of timer_external_clock_mode0_config is shown as below:

Table 3-1671. Function timer_external_clock_mode0_config

Function name	timer_external_clock_mode0_config
----------------------	-----------------------------------

Function prototype	void timer_external_clock_mode0_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER the external clock mode0
Precondition	-
The called functions	timer_external_trigger_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14,22,23,30,31,40~44)	TIMER peripheral selection
Input parameter{in}	
extprescaler	ETI 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	ETI 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	ETI external trigger filter control(0~15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 the external clock mode0 */
```

```
timer_external_clock_mode0_config(TIMER0, TIMER_EXT_TRI_PSC_DIV2,
TIMER_ETP_FALLING, 0);
```

timer_external_clock_mode1_config

The description of timer_external_clock_mode1_config is shown as below:

Table 3-1672. Function timer_external_clock_mode1_config

Function name	timer_external_clock_mode1_config
Function prototype	void timer_external_clock_mode1_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);

Function descriptions	configure TIMER the external clock mode1
Precondition	-
The called functions	timer_external_trigger_config
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,22,23,30,31)</i>	TIMER peripheral selection
Input parameter{in}	
extprescaler	ETI external trigger prescaler
<i>TIMER_EXT_TRI_PSC_OFF</i>	no divided
<i>TIMER_EXT_TRI_PSC_DIV2</i>	divided by 2
<i>TIMER_EXT_TRI_PSC_DIV4</i>	divided by 4
<i>TIMER_EXT_TRI_PSC_DIV8</i>	divided by 8
Input parameter{in}	
expolarity	ETI external trigger polarity
<i>TIMER_ETP_FALLING</i>	active low or falling edge active
<i>TIMER_ETP_RISING</i>	active high or rising edge active
Input parameter{in}	
extfilter	ETI external trigger filter control(0~15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 the external clock mode1 */
```

```
timer_external_clock_mode1_config(TIMER0, TIMER_EXT_TRI_PSC_DIV2,  
TIMER_ETP_FALLING, 0);
```

timer_external_clock_mode1_disable

The description of timer_external_clock_mode1_disable is shown as below:

Table 3-1673. Function timer_external_clock_mode1_disable

Function name	timer_external_clock_mode1_disable
Function prototype	void timer_external_clock_mode1_disable(uint32_t timer_periph);
Function descriptions	disable TIMER the external clock mode1
Precondition	-
The called functions	-

Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 the external clock mode1 */
```

```
timer_external_clock_mode1_disable(TIMER0);
```

timer_write_chxval_register_config

The description of timer_write_chxval_register_config is shown as below:

Table 3-1674. Function timer_write_chxval_register_config

Function name	timer_write_chxval_register_config
Function prototype	void timer_write_chxval_register_config(uint32_t timer_periph, uint16_t ccsel);
Function descriptions	configure TIMER write CHxVAL register selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,14~16,22,23,30,31,40~44)	TIMER peripheral selection
Input parameter{in}	
ccsel	write CHxVAL register selection
<i>TIMER_CHVSEL_DISABLE</i>	no effect
<i>TIMER_CHVSEL_ENABLE</i>	when write the CHxVAL register, if the write value is same as the CHxVAL value, the write access is ignored
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 write CHxVAL register selection */
```

```
timer_write_chxval_register_config(TIMER0, TIMER_CHVSEL_ENABLE);
```

timer_output_value_selection_config

The description of timer_output_value_selection_config is shown as below:

Table 3-1675. Function timer_output_value_selection_config

Function name	timer_output_value_selection_config
Function prototype	void timer_output_value_selection_config(uint32_t timer_periph, uint16_t outsel);
Function descriptions	configure TIMER output value selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMEx(x=0,7,14~16,40~44)	TIMER peripheral selection
Input parameter{in}	
outsel	output value selection
TIMER_OUTSEL_DISABLE	no effect
TIMER_OUTSEL_ENABLE	if POEN and IOS is 0, the output disabled
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER output value selection */
```

```
timer_output_value_selection_config(TIMER0, TIMER_OUTSEL_ENABLE);
```

timer_commutation_control_shadow_register_config

The description of timer_commutation_control_shadow_register_config is shown as below:

Table 3-1676. Function timer_commutation_control_shadow_register_config

Function name	timer_commutation_control_shadow_register_config
Function prototype	void timer_commutation_control_shadow_register_config(uint32_t timer_periph, uint16_t ccssel);
Function descriptions	configure commutation control shadow register update selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMEx(x=0,7,14~16,40	TIMER peripheral selection

~44)	
Input parameter{in}	
csel	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1677. Function timer_output_match_pulse_select

Function name	timer_output_match_pulse_select
Function prototype	void timer_output_match_pulse_select(uint32_t timer_periph, uint32_t channel, uint16_t pulsesel);
Function descriptions	configure TIMER output match pulse selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,14,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(<i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(<i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
Input parameter{in}	
pulsesel	output match pulse selection
<i>TIMER_PULSE_OUTPUT_NORMAL</i>	channel output normal
<i>TIMER_PULSE_OUTPUT_CNT_UP</i>	pulse output only when counting up

<i>TIMER_PULSE_OUTPUT</i> <i>T_CNT_DOWN</i>	pulse output only when counting down
<i>TIMER_PULSE_OUTPUT</i> <i>T_CNT_BOTH</i>	pulse output when counting up or down
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 match pulse selection */
```

```
timer_output_match_pulse_select          (TIMER0,          TIMER_CH_0,
timer_pulse_output_cnt_up);
```

timer_channel_composite_pwm_mode_config

The description of timer_channel_composite_pwm_mode_config is shown as below:

Table 3-1678. Function timer_channel_composite_pwm_mode_config

Function name	timer_channel_composite_pwm_mode_config
Function prototype	void timer_channel_composite_pwm_mode_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
Function descriptions	configure the TIMER composite PWM mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,14,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(<i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(<i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the TIMER composite PWM mode */
```

```
timer_channel_composite_pwm_mode_config(TIMER0, TIMER_CH_0, ENABLE);
```

timer_channel_composite_pwm_mode_output_pulse_value_config

The description of timer_channel_composite_pwm_mode_output_pulse_value_config is shown as below:

Table 3-1679. Function timer_channel_composite_pwm_mode_output_pulse_value_config

Function name	timer_channel_composite_pwm_mode_output_pulse_value_config
Function prototype	void timer_channel_composite_pwm_mode_output_pulse_value_config(uint32_t timer_periph, uint32_t channel, uint32_t pulse, uint32_t add_pulse);
Function descriptions	configure the TIMER composite PWM mode output pulse value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
TIMER_CH_1	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
TIMER_CH_2	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
TIMER_CH_3	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
Input parameter{in}	
pulse	channel compare value(0~0xFFFFFFFF)
Input parameter{in}	
add_pulse	channel additional compare value(0~0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output pulse value */
```

```
timer_channel_output_pulse_value_config(TIMER0, TIMER_CH_0, 399, 3999);
```

timer_channel_additional_compare_value_config

The description of timer_channel_additional_compare_value_config is shown as below:

Table 3-1680. Function timer_channel_additional_compare_value_config

Function name	timer_channel_additional_compare_value_config
Function prototype	void timer_channel_additional_compare_value_config(uint32_t timer_periph, uint16_t channel, uint32_t value);
Function descriptions	configure TIMER channel additional compare value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,30,31,40~44)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
Input parameter{in}	
value	channel additional compare value(0~0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 additional compare value */
```

```
timer_channel_additional_compare_value_config (TIMER0, TIMER_CH_0, 399);
```

timer_channel_additional_output_shadow_config

The description of timer_channel_additional_output_shadow_config is shown as below:

Table 3-1681. Function timer_channel_additional_output_shadow_config

Function name	timer_channel_additional_output_shadow_config
Function prototype	void timer_channel_additional_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t aocshadow);
Function descriptions	configure TIMER channel additional output shadow function
Precondition	-
The called functions	-
Input parameter{in}	

timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,14,22,23,30,31,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(<i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(<i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(<i>TIMERx</i> , <i>x</i> =0~4,7,22,23,30,31)
Input parameter{in}	
aocshadow	channel additional output compare shadow
<i>TIMER_ADD_SHADOW_ENABLE</i>	channel additional output compare shadow enable
<i>TIMER_ADD_SHADOW_DISABLE</i>	channel additional output compare shadow disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*configure TIMER0 channel 0 additional output shadow function */
```

```
timer_channel_additional_output_shadow_config      (TIMER0,          TIMER_CH_0,
TIMER_OC_SHADOW_ENABLE);
```

timer_channel_additional_compare_value_read

The description of timer_channel_additional_compare_value_read is shown as below:

Table 3-1682. 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
<i>TIMERx</i> (<i>x</i> =0~4,7,14,22,23,30,31,40~44)	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(<i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,30,31,40~44)

<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23,30,31)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23,30,31)
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 3-1683. 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
<i>TIMERx</i> (x=0,7,14~16,40~44)	please refer to the following parameters
Input parameter{in}	
break_num	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)
Input parameter{in}	
break_src	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)
<i>TIMER_BRKHPDF</i>	HPDF input enable, TIMERx(x=0,7,14~16,40~44)
Input parameter{in}	
newvalue	control value

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

Example:

```
/* configure the TIMER 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 3-1684. 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);
Function descriptions	configure TIMER break polarity
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)	please refer to the following parameters
Input parameter{in}	
break_num	TIMER BREAKx
<i>TIMER_BREAK0</i>	BREAK0 input signals, <i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)
<i>TIMER_BREAK1</i>	BREAK1 input signals, <i>TIMERx</i> (<i>x</i> =0,7)
Input parameter{in}	
break_src	break source
<i>TIMER_BRKIN0</i>	BRKIN0 alternate function input enable, <i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)
<i>TIMER_BRKIN1</i>	BRKIN1 alternate function input enable, <i>TIMERx</i> (<i>x</i> =0,7)
<i>TIMER_BRKIN2</i>	BRKIN2 alternate function input enable, <i>TIMERx</i> (<i>x</i> =0,7)
<i>TIMER_BRKCOMP0</i>	CMP0 input enable, <i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)
<i>TIMER_BRKCOMP1</i>	CMP1 input enable, <i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)
Input parameter{in}	
bkinpolarity	break polarity
<i>TIMER_BRKIN_POLARITY_LOW</i>	active low
<i>TIMER_BRKIN_POLARITY_HIGH</i>	active high

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

Example:

```
/* configure the TIMER break polarity */
```

```
timer_break_external_polarity_config(TIMER0,    TIMER_BREAK0,    TIMER_BRKIN0,
timer_break_external_polarity_config(TIMER0,    TIMER_BREAK0,    TIMER_BRKIN0,
TIMER_BRKIN_POLARITY_HIGH);
```

timer_break_lock_config

The description of timer_break_lock_config is shown as below:

Table 3-1685. Function timer_break_lock_config

Function name	timer_break_lock_config
Function prototype	void timer_break_lock_config(uint32_t timer_periph, uint16_t break_num, ControlStatus newvalue);
Function descriptions	configure TIMER break lock function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
Input parameter{in}	
break_num	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)
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER break lock function */
```

```
timer_break_lock_config(TIMER0, TIMER_BREAK0, ENABLE);
```


timer_break_lock_release_config

The description of timer_break_lock_release_config is shown as below:

Table 3-1686. Function timer_break_lock_release_config

Function name	timer_break_lock_release_config
Function prototype	void timer_break_lock_release_config(uint32_t timer_periph, uint16_t break_num, ControlStatus newvalue);
Function descriptions	release the TIMER break lock function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	TIMER peripheral selection
Input parameter{in}	
break_num	TIMER BREAKx
<i>TIMER_BREAK0</i>	release the BREAK0 lock function, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_BREAK1</i>	release the BREAK1 lock function, TIMERx(x=0,7)
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* release the TIMER break lock function */
```

```
timer_break_lock_release_config (TIMER0, TIMER_BREAK0, ENABLE);
```

timer_channel_break_control_config

The description of timer_channel_break_control_config is shown as below:

Table 3-1687. Function timer_channel_break_control_config

Function name	timer_channel_break_control_config
Function prototype	void timer_channel_break_control_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
Function descriptions	configure the TIMER channel break function
Precondition	-
The called functions	-
Input parameter{in}	

timer_periph	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
<i>TIMER_CH_2</i>	TIMER channel 2
<i>TIMER_CH_3</i>	TIMER channel 3
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 break function */
```

```
timer_channel_break_control_config (TIMER0, TIMER_CH_0, ENABLE);
```

timer_channel_dead_time_config

The description of timer_channel_dead_time_config is shown as below:

Table 3-1688. Function timer_channel_dead_time_config

Function name	timer_channel_dead_time_config
Function prototype	void timer_channel_dead_time_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
Function descriptions	configure the TIMER channel free dead time function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
<i>TIMER_CH_2</i>	TIMER channel 2
<i>TIMER_CH_3</i>	TIMER channel 3
Input parameter{in}	
newvalue	control value

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

Example:

```
/* configure TIMER0 channel 0 free dead time function */
```

```
timer_channel_dead_time_config (TIMER0, TIMER_CH_0, ENABLE);
```

timer_free_complementary_struct_para_init

The description of timer_free_complementary_struct_para_init is shown as below:

Table 3-1689. Function timer_free_complementary_struct_para_init

Function name	timer_free_complementary_struct_para_init
Function prototype	void timer_free_complementary_struct_para_init(timer_free_complementary_parameter_struct *freecompara);
Function descriptions	initialize TIMER channel free complementary parameter struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
freecompara	TIMER channel free complementary parameter struct, the structure members can refer to Structure timer free complementary parameter struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER channel free complementary parameter struct with a default value */
```

```
timer_free_complementary_parameter_struct timer_freecompara;
```

```
timer_free_complementary_struct_para_init (&timer_freecompara);
```

timer_channel_free_complementary_config

The description of timer_channel_free_complementary_config is shown as below:

Table 3-1690. 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 Structure timer free complementary parameter struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize TIMER break parameter struct with a default value */

timer_free_complementary_parameter_struct timer_freecompara;

timer_freecompara.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 3-1691. Function timer_watchdog_value_config

Function name	timer_watchdog_value_config
Function prototype	void timer_watchdog_value_config(uint32_t timer_periph, uint32_t value);

Function descriptions	configure TIMER autoreload register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)	TIMER peripheral selection
Input parameter{in}	
value	watchdog counter period value, 0~0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
-	-

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 3-1692. Function timer_watchdog_value_read

Function name	timer_watchdog_value_read
Function prototype	uint32_t timer_watchdog_value_read(uint32_t timer_periph);
Function descriptions	read quadrature decoder signal disconnection detection watchdog value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1693. Function timer_decoder_disconnection_detection_config

Function name	timer_decoder_disconnection_detection_config
Function prototype	void timer_decoder_disconnection_detection_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	configure quadrature decoder signal disconnection detection function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,22,23,30,31)</i>	please refer to the following parameters
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

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 3-1694. Function timer_decoder_jump_detection_config

Function name	timer_decoder_jump_detection_config
Function prototype	void timer_decoder_jump_detection_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	configure quadrature decoder signal jump detection function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~4,7,22,23,30,31)</i>	please refer to the following parameters
Input parameter{in}	

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

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 3-1695. Function timer_upif_backup_config

Function name	timer_upif_backup_config
Function prototype	void timer_upif_backup_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	configure the UPIF bit backup function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~7,14~16,22,23,30,31,40~44,50,51)	please refer to the following parameters
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

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 3-1696. Function timer_upifbu_bit_get

Function name	timer_upifbu_bit_get
Function prototype	UPIFBUSStatus timer_upifbu_bit_get(uint32_t timer_periph);
Function descriptions	get the UPIFBU bit in the TIMEx_CNT register
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMEx(x=0~7,14~16,22,23,30,31,40~44,50,51)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
UPIFBUSStatus	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 3-1697. Function timer_flag_get

Function name	timer_flag_get
Function prototype	FlagStatus timer_flag_get(uint32_t timer_periph, uint32_t flag);
Function descriptions	get TIMER flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMEx(x=0~7,14~16,22,23,30,31,40~44,50,51)</i>	please refer to the following parameters
Input parameter{in}	
flag	the TIMER flags
<i>TIMER_FLAG_UP</i>	update flag, TIMEx(x=0~7,14~16,22,23,30,31,40~44,50,51)
<i>TIMER_FLAG_CH0</i>	channel 0 capture or compare flag,

	TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_FLAG_CH1</i>	channel 1 capture or compare flag, TIMERx(x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_FLAG_CH2</i>	channel 2 capture or compare flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_FLAG_CH3</i>	channel 3 capture or compare flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_FLAG_CMT</i>	channel commutation flag, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_FLAG_TRG</i>	trigger flag, TIMERx(x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_FLAG_BRK0</i>	BREAK0 flag, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_FLAG_BRK1</i>	BREAK1 flag, TIMERx(x=0,7)
<i>TIMER_FLAG_CH0O</i>	channel 0 overcapture flag, TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_FLAG_CH1O</i>	channel 1 overcapture flag, TIMERx(x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_FLAG_CH2O</i>	channel 2 overcapture flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_FLAG_CH3O</i>	channel 3 overcapture flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_FLAG_SYSB</i>	system source break flag, TIMERx(x=0,7)
<i>TIMER_FLAG_DECJ</i>	quadrature decoder signal jump flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_FLAG_DECDIS</i>	quadrature decoder signal disconnection flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_FLAG_MCH0</i>	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=0~4,7,14,22,23,30,31)
<i>TIMER_FLAG_CH1CO MADD</i>	channel 1 additional compare flag, TIMERx(x=0~4,7,14,22,23,30,31)
<i>TIMER_FLAG_CH2CO MADD</i>	channel 2 additional compare flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_FLAG_CH3CO MADD</i>	channel 3 additional compare flag, TIMERx(x=0~4,7,22,23,30,31)
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TIMERO 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 3-1698. Function timer_flag_clear

Function name	timer_flag_clear
Function prototype	void timer_flag_clear(uint32_t timer_periph, uint32_t flag);
Function descriptions	clear TIMER flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 22, 23, 30, 31, 40~44, 50, 51)</i>	please refer to the following parameters
Input parameter{in}	
flag	the timer interrupt flags
<i>TIMER_FLAG_UP</i>	update flag, <i>TIMERx(x=0~7, 14~16, 22, 23, 30, 31, 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, 30, 31, 40~44)</i>
<i>TIMER_FLAG_CH1</i>	channel 1 capture or compare flag, <i>TIMERx(x=0~4, 7, 14, 22, 23, 30, 31, 40~44)</i>
<i>TIMER_FLAG_CH2</i>	channel 2 capture or compare flag, <i>TIMERx(x=0~4, 7, 22, 23, 30, 31)</i>
<i>TIMER_FLAG_CH3</i>	channel 3 capture or compare flag, <i>TIMERx(x=0~4, 7, 22, 23, 30, 31)</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, 30, 31, 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, 30, 31, 40~44)</i>
<i>TIMER_FLAG_CH1O</i>	channel 1 overcapture flag, <i>TIMERx(x=0~4, 7, 14, 22, 23, 30, 31, 40~44)</i>
<i>TIMER_FLAG_CH2O</i>	channel 2 overcapture flag, <i>TIMERx(x=0~4, 7, 22, 23, 30, 31)</i>
<i>TIMER_FLAG_CH3O</i>	channel 3 overcapture flag, <i>TIMERx(x=0~4, 7, 22, 23, 30, 31)</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, 30, 31)</i>
<i>TIMER_FLAG_DECDIS</i>	quadrature decoder signal disconnection flag, <i>TIMERx(x=0~4, 7, 22, 23, 30, 31)</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</i> (<i>x</i> =0,7)
<i>TIMER_FLAG_MCH2O</i>	multi mode channel 2 overcapture flag, <i>TIMERx</i> (<i>x</i> =0,7)
<i>TIMER_FLAG_MCH3O</i>	multi mode channel 3 overcapture flag, <i>TIMERx</i> (<i>x</i> =0,7)
<i>TIMER_FLAG_CH0CO</i> <i>MADD</i>	channel 0 additional compare flag, <i>TIMERx</i> (<i>x</i> =0~4,7,14,22,23,30,31)
<i>TIMER_FLAG_CH1CO</i> <i>MADD</i>	channel 1 additional compare flag, <i>TIMERx</i> (<i>x</i> =0~4,7,14,22,23,30,31)
<i>TIMER_FLAG_CH2CO</i> <i>MADD</i>	channel 2 additional compare flag, <i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)
<i>TIMER_FLAG_CH3CO</i> <i>MADD</i>	channel 3 additional compare flag, <i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear TIMER0 update flags */
```

```
timer_flag_clear(TIMER0, TIMER_FLAG_UP);
```

timer_interrupt_enable

The description of `timer_interrupt_enable` is shown as below:

Table 3-1699. Function `timer_interrupt_enable`

Function name	<code>timer_interrupt_enable</code>
Function prototype	<code>void timer_interrupt_enable(uint32_t timer_periph, uint32_t interrupt);</code>
Function descriptions	enable the TIMER interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0~7,14~16,22,23,30,31,40~44,50,51)	please refer to the following parameters
Input parameter{in}	
interrupt	timer interrupt enable source
<i>TIMER_INT_UP</i>	update interrupt enable, <i>TIMERx</i> (<i>x</i> =0~7,14~16,22,23,30,31,40~44,50,51)
<i>TIMER_INT_CH0</i>	channel 0 capture or compare interrupt enable, <i>TIMERx</i> (<i>x</i> =0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_INT_CH1</i>	channel 1 capture or compare interrupt enable, <i>TIMERx</i> (<i>x</i> =0~4,7,14,22,23,30,31,40~44)
<i>TIMER_INT_CH2</i>	channel 2 capture or compare interrupt enable, <i>TIMERx</i> (<i>x</i> =0~4,7,22,23,30,31)

<i>TIMER_INT_CH3</i>	channel 3 capture or compare interrupt enable , TIMERx(x=0~4,7,22,23,30,31)
<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=0~4,7,14,22,23,30,31,40~44)
<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=0~4,7,22,23,30,31)
<i>TIMER_INT_DECDIS</i>	quadrature decoder signal disconnection interrupt, TIMERx(x=0~4,7,22,23,30,31)
<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=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_INT_CH1COMA DD</i>	channel 1 additional compare interrupt, TIMERx(x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_INT_CH2COMA DD</i>	channel 2 additional compare interrupt, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_INT_CH3COMA DD</i>	channel 3 additional compare interrupt, TIMERx(x=0~4,7,22,23,30,31)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TIMER0 update interrupt */
```

```
timer_interrupt_enable (TIMER0, TIMER_INT_UP);
```

timer_interrupt_disable

The description of timer_interrupt_disable is shown as below:

Table 3-1700. Function timer_interrupt_disable

Function name	timer_interrupt_disable
Function prototype	void timer_interrupt_disable (uint32_t timer_periph, uint32_t interrupt);
Function descriptions	disable the TIMER interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~7,14~16,2</i>	please refer to the following parameters

2,23,30,31,40~44,50,51)	
Input parameter{in}	
interrupt	timer interrupt disable source
<i>TIMER_INT_UP</i>	update interrupt enable, $TIMERx(x=0\sim7,14\sim16,22,23,30,31,40\sim44,50,51)$
<i>TIMER_INT_CH0</i>	channel 0 capture or compare interrupt enable, $TIMERx(x=0\sim4,7,14\sim16,22,23,30,31,40\sim44)$
<i>TIMER_INT_CH1</i>	channel 1 capture or compare interrupt enable, $TIMERx(x=0\sim4,7,14,22,23,30,31,40\sim44)$
<i>TIMER_INT_CH2</i>	channel 2 capture or compare interrupt enable, $TIMERx(x=0\sim4,7,22,23,30,31)$
<i>TIMER_INT_CH3</i>	channel 3 capture or compare interrupt enable , $TIMERx(x=0\sim4,7,22,23,30,31)$
<i>TIMER_INT_CMT</i>	commutation interrupt enable, $TIMERx(x=0,7,14\sim16,40\sim44)$
<i>TIMER_INT_TRG</i>	trigger interrupt enable, $TIMERx(x=0\sim4,7,14,22,23,30,31,40\sim44)$
<i>TIMER_INT_BRK</i>	break interrupt enable, $TIMERx(x=0,7,14\sim16,40\sim44)$
<i>TIMER_INT_DECJ</i>	quadrature decoder signal jump interrupt, $TIMERx(x=0\sim4,7,22,23,30,31)$
<i>TIMER_INT_DECDIS</i>	quadrature decoder signal disconnection interrupt, $TIMERx(x=0\sim4,7,22,23,30,31)$
<i>TIMER_INT_MCH0</i>	multi mode channel 0 capture or compare interrupt, $TIMERx(x=0,7,14\sim16,40\sim44)$
<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</i> <i>DD</i>	channel 0 additional compare interrupt, $TIMERx(x=0\sim4,7,14,22,23,30,31,40\sim44)$
<i>TIMER_INT_CH1COMA</i> <i>DD</i>	channel 1 additional compare interrupt, $TIMERx(x=0\sim4,7,14,22,23,30,31,40\sim44)$
<i>TIMER_INT_CH2COMA</i> <i>DD</i>	channel 2 additional compare interrupt, $TIMERx(x=0\sim4,7,22,23,30,31)$
<i>TIMER_INT_CH3COMA</i> <i>DD</i>	channel 3 additional compare interrupt, $TIMERx(x=0\sim4,7,22,23,30,31)$
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 3-1701. Function timer_interrupt_flag_get

Function name	timer_interrupt_flag_get
Function prototype	FlagStatus timer_interrupt_flag_get(uint32_t timer_periph, uint32_t int_flag);
Function descriptions	get timer interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)</i>	please refer to the following parameters
Input parameter{in}	
int_flag	the timer interrupt bits
<i>TIMER_INT_FLAG_UP</i>	update interrupt flag, <i>TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)</i>
<i>TIMER_INT_FLAG_CH0</i>	channel 0 capture or compare interrupt flag, <i>TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)</i>
<i>TIMER_INT_FLAG_CH1</i>	channel 1 capture or compare interrupt flag, <i>TIMERx(x=0~4,7,14,22,23,30,31,40~44)</i>
<i>TIMER_INT_FLAG_CH2</i>	channel 2 capture or compare interrupt flag, <i>TIMERx(x=0~4,7,22,23,30,31)</i>
<i>TIMER_INT_FLAG_CH3</i>	channel 3 capture or compare interrupt flag, <i>TIMERx(x=0~4,7,22,23,30,31)</i>
<i>TIMER_INT_FLAG_CMT</i>	channel commutation interrupt flag, <i>TIMERx(x=0,7,14~16,40~44)</i>
<i>TIMER_INT_FLAG_TRG</i>	trigger interrupt flag, <i>TIMERx(x=0~4,7,14,22,23,30,31,40~44)</i>
<i>TIMER_INT_FLAG_BRK0</i>	BREAK0 interrupt flag, <i>TIMERx(x=0,7,14~16,40~44)</i>
<i>TIMER_INT_FLAG_BRK1</i>	BREAK1 interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_SYSB</i>	system source break interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_DECJ</i>	quadrature decoder signal jump interrupt flag, <i>TIMERx(x=0~4,7,22,23,30,31)</i>
<i>TIMER_INT_FLAG_DECDIS</i>	quadrature decoder signal disconnection interrupt flag, <i>TIMERx(x=0~4,7,22,23,30,31)</i>
<i>TIMER_INT_FLAG_MCH0</i>	multi mode channel 0 capture or compare interrupt flag, <i>TIMERx(x=0,7,14~16,40~44)</i>
<i>TIMER_INT_FLAG_MCH1</i>	multi mode channel 1 capture or compare interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_MCH2</i>	multi mode channel 2 capture or compare interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_MCH3</i>	multi mode channel 3 capture or compare interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_CH0COMADD</i>	channel 0 additional compare interrupt flag, <i>TIMERx(x=0~4,7,14,22,23,30,31)</i>

<i>TIMER_INT_FLAG_CH1</i> <i>COMADD</i>	channel 1 additional compare interrupt flag, TIMERx(x=0~4,7,14,22,23,30,31)
<i>TIMER_INT_FLAG_CH2</i> <i>COMADD</i>	channel 2 additional compare interrupt flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_INT_FLAG_CH3</i> <i>COMADD</i>	channel 3 additional compare interrupt flag, TIMERx(x=0~4,7,22,23,30,31)
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TIMERO update interrupt flag */
```

```
FlagStatus Flag_interrupt = RESET;
```

```
Flag_interrupt = timer_interrupt_flag_get(TIMERO, TIMER_INT_FLAG_UP);
```

timer_interrupt_flag_clear

The description of timer_interrupt_flag_clear is shown as below:

Table 3-1702. Function timer_interrupt_flag_clear

Function name	timer_interrupt_flag_clear
Function prototype	void timer_interrupt_flag_clear(uint32_t timer_periph, uint32_t int_flag);
Function descriptions	clear TIMER interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)</i>	please refer to the following parameters
Input parameter{in}	
int_flag	the timer interrupt bits
<i>TIMER_INT_FLAG_UP</i>	update interrupt flag, TIMERx(x=0~7,14~16,22,23,30,31,40~44,50,51)
<i>TIMER_INT_FLAG_CH0</i>	channel 0 capture or compare interrupt flag, TIMERx(x=0~4,7,14~16,22,23,30,31,40~44)
<i>TIMER_INT_FLAG_CH1</i>	channel 1 capture or compare interrupt flag, TIMERx(x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_INT_FLAG_CH2</i>	channel 2 capture or compare interrupt flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_INT_FLAG_CH3</i>	channel 3 capture or compare interrupt flag, TIMERx(x=0~4,7,22,23,30,31)
<i>TIMER_INT_FLAG_CMT</i>	channel commutation interrupt flag, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_INT_FLAG_TRG</i>	trigger interrupt flag, TIMERx(x=0~4,7,14,22,23,30,31,40~44)
<i>TIMER_INT_FLAG_BRK</i>	BREAK0 interrupt flag, TIMERx(x=0,7,14~16,40~44)

0	
<i>TIMER_INT_FLAG_BRK</i> 1	BREAK1 interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_SYS</i> B	system source break interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_DEC</i> J	quadrature decoder signal jump interrupt flag, $TIMERx(x=0\sim4,7,22,23,30,31)$
<i>TIMER_INT_FLAG_DEC</i> DIS	quadrature decoder signal disconnection interrupt flag, $TIMERx(x=0\sim4,7,22,23,30,31)$
<i>TIMER_INT_FLAG_MC</i> H0	multi mode channel 0 capture or compare interrupt flag, $TIMERx(x=0,7,14\sim16,40\sim44)$
<i>TIMER_INT_FLAG_MC</i> H1	multi mode channel 1 capture or compare interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_MC</i> H2	multi mode channel 2 capture or compare interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_MC</i> H3	multi mode channel 3 capture or compare interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_CH0</i> COMADD	channel 0 additional compare interrupt flag, $TIMERx(x=0\sim4,7,14,22,23,30,31)$
<i>TIMER_INT_FLAG_CH1</i> COMADD	channel 1 additional compare interrupt flag, $TIMERx(x=0\sim4,7,14,22,23,30,31)$
<i>TIMER_INT_FLAG_CH2</i> COMADD	channel 2 additional compare interrupt flag, $TIMERx(x=0\sim4,7,22,23,30,31)$
<i>TIMER_INT_FLAG_CH3</i> COMADD	channel 3 additional compare interrupt flag, $TIMERx(x=0\sim4,7,22,23,30,31)$
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear TIMER0 update interrupt flag */
```

```
timer_interrupt_flag_clear(TIMER0, TIMER_INT_FLAG_UP);
```

3.44. TLI

The TLI (TFT-LCD Interface) module handles the synchronous LCD interface and provides pixel data, clock and timing signals for passive LCD display. The TLI registers are listed in chapter [3.44.1](#), the TLI firmware functions are introduced in chapter [3.44.2](#).

3.44.1. Descriptions of Peripheral registers

TLI registers are listed in the table shown as below:

Table 3-1703. TLI Registers

Registers	Descriptions
TLI_SPSZ	TLI synchronous pulse size register
TLI_BPSZ	TLI back-porch size register
TLI_ASZ	TLI active size register
TLI_TSZ	TLI total size register
TLI_CTL	TLI control register
TLI_RL	TLI reload Layer register
TLI_BGC	TLI background color register
TLI_INTEN	TLI interrupt enable register
TLI_INTF	TLI interrupt flag register
TLI_INTC	TLI interrupt flag clear register
TLI_LM	TLI line mark register
TLI_CPPOS	TLI current pixel position register
TLI_STAT	TLI status register
TLI_LxCTL	TLI layer x control register
TLI_LxHPOS	TLI layer x horizontal position parameters register
TLI_LxVPOS	TLI layer x vertical position parameters register
TLI_LxCKEY	TLI layer x color key register
TLI_LxPPF	TLI layer x packeted pixel format register
TLI_LxSA	TLI layer x specified alpha register
TLI_LxDC	TLI layer x default color register
TLI_LxBLEND	TLI layer x blending register
TLI_LxFBADDR	TLI layer x frame base address register
TLI_LxFLEN	TLI layer x frame line length register
TLI_LxFTLN	TLI layer x frame total line number register
TLI_LxLUT	TLI layer x Look Up Table register

3.44.2. Descriptions of Peripheral functions

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

Table 3-1704. TLI firmware function

Function name	Function description
tli_deinit	deinitialize TLI registers
tli_struct_para_init	initialize the parameters of TLI parameter structure with the default values, it is suggested that call this function after a tli_parameter_struct structure is defined
tli_init	initialize TLI

Function name	Function description
tli_dither_config	configure TLI dither function
tli_enable	enable TLI
tli_disable	disable TLI
tli_reload_config	configure TLI reload mode
tli_layer_struct_para_init	initialize the parameters of TLI layer structure with the default values, it is suggested that call this function after a tli_layer_parameter_struct structure is defined
tli_layer_init	initialize TLI layer
tli_layer_window_offset_modify	reconfigure window position
tli_lut_struct_para_init	initialize the parameters of TLI layer LUT structure with the default values, it is suggested that call this function after a tli_layer_lut_parameter_struct structure is defined
tli_lut_init	initialize TLI layer LUT
tli_color_key_init	initialize TLI layer color key
tli_layer_enable	enable TLI layer
tli_layer_disable	disable TLI layer
tli_color_key_enable	enable TLI layer color keying
tli_color_key_disable	disable TLI layer color keying
tli_lut_enable	enable TLI layer LUT
tli_lut_disable	disable TLI layer LUT
tli_line_mark_set	set line mark value
tli_current_pos_get	get current displayed position
tli_interrupt_enable	enable TLI interrupt
tli_interrupt_disable	disable TLI interrupt
tli_interrupt_flag_get	get TLI interrupt flag
tli_interrupt_flag_clear	clear TLI interrupt flag
tli_flag_get	get TLI flag or state in TLI_INTF register or TLI_STAT register

Structure tli_parameter_struct

Table3-1705. Structure tli_parameter_struct

Member name	Function description
synpsz_vpsz	size of the vertical synchronous pulse
synpsz_hpsz	size of the horizontal synchronous pulse
backpsz_vbpsz	size of the vertical back porch plus synchronous pulse
backpsz_hbpsz	size of the horizontal back porch plus synchronous pulse
activesz_vasz	size of the vertical active area width plus back porch and synchronous pulse
activesz_hasz	size of the horizontal active area width plus back porch and synchronous pulse
totalsz_vtsz	vertical total size of the display
totalsz_htsz	horizontal total size of the display

Member name	Function description
backcolor_red	background value red
backcolor_green	background value green
backcolor_blue	background value blue
signalpolarity_hs	horizontal pulse polarity selection
signalpolarity_vs	vertical pulse polarity selection
signalpolarity_de	data enable polarity selection
signalpolarity_pixelck	pixel clock polarity selection

Structure tli_layer_parameter_struct

Table3-1706. Structure tli_layer_parameter_struct

Member name	Function description
layer_window_rightpos	window right position
layer_window_leftpos	window left position
layer_window_bottompos	window bottom position
layer_window_toppos	window top position
layer_ppf	packeted pixel format
layer_sa	specified alpha
layer_default_alpha	the default color alpha
layer_default_red	the default color red
layer_default_green	the default color green
layer_default_blue	the default color blue
layer_acf1	alpha calculation factor 1 of blending method
layer_acf2	alpha calculation factor 2 of blending method
layer_frame_bufaddr	frame buffer base address
layer_frame_buf_stride_offset	frame buffer stride offset
layer_frame_line_length	frame line length
layer_frame_total_line_number	frame total line number

Structure tli_layer_lut_parameter_struct

Table3-1707. Structure tli_layer_lut_parameter_struct

Member name	Function description
layer_table_addr	look up table write address
layer_lut_channel_red	red channel of a LUT entry
layer_lut_channel_green	green channel of a LUT entry
layer_lut_channel_blue	blue channel of a LUT entry

tli_deinit

The description of tli_deinit is shown as below:

Table 3-1708. Function tli_deinit

Function name	tli_deinit
Function prototype	void tli_deinit(void);
Function descriptions	deinitialize TLI registers
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize TLI */
```

```
tli_deinit();
```

tli_struct_para_init

The description of tli_struct_para_init is shown as below:

Table 3-1709. Function tli_struct_para_init

Function name	tli_struct_para_init
Function prototype	void tli_struct_para_init(tli_parameter_struct *tli_struct);
Function descriptions	initialize the parameters of TLI parameter structure with the default values, it is suggested that call this function after a tli_parameter_struct structure is defined
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
*tli_struct	a pointer to tli_parameter_struct
Return value	
-	-

Example:

```
tli_parameter_struct tli_init_struct;
```

```
/* initialize the parameters of TLI parameter structure with the default values */
```

```
tli_struct_para_init(&tli_init_struct);
```

tli_init

The description of tli_init is shown as below:

Table 3-1710. Function tli_init

Function name	tli_init
Function prototype	void tli_init(tli_parameter_struct *tli_struct);
Function descriptions	initialize TLI display timing parameters
Precondition	-
The called functions	-
Input parameter{in}	
*tli_struct	a pointer to tli_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

tli_parameter_struct  tli_init_struct;

/* initialize the parameters of TLI parameter structure with the default values */
tli_struct_para_init(&tli_init_struct);

/* configure TLI parameter struct */
tli_init_struct.signalpolarity_hs = TLI_HSYN_ACTLIVE_LOW;
tli_init_struct.signalpolarity_vs = TLI_VSYN_ACTLIVE_LOW;
tli_init_struct.signalpolarity_de = TLI_DE_ACTLIVE_LOW;
tli_init_struct.signalpolarity_pixelclk = TLI_PIXEL_CLOCK_TLI;

/* LCD display timing configuration */
tli_init_struct.synpsz_hpsz = 40;
tli_init_struct.synpsz_vpsz = 9;
tli_init_struct.backpsz_hbpsz = 42;
tli_init_struct.backpsz_vbpsz = 11;
tli_init_struct.activesz_hasz = 522;
tli_init_struct.activesz_vasz = 283;
tli_init_struct.totalsz_htsz = 524;
tli_init_struct.totalsz_vtsz = 285;

```

```
/* configure LCD background R,G,B values */
```

```
tli_init_struct.backcolor_red = 0xFF;
```

```
tli_init_struct.backcolor_green = 0xFF;
```

```
tli_init_struct.backcolor_blue = 0xFF;
```

```
tli_init(&tli_init_struct);
```

tli_dither_config

The description of tli_dither_config is shown as below:

Table 3-1711. Function tli_dither_config

Function name	tli_dither_config
Function prototype	void tli_dither_config(uint8_t dither_stat);
Function descriptions	configure TLI dither function
Precondition	-
The called functions	-
Input parameter{in}	
dither_stat	dither function
TLI_DITHER_ENABLE	enable TLI dither function
TLI_DITHER_DISABLE	disable TLI dither function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TLI dither function */
```

```
tli_dither_config(TLI_DITHER_ENABLE);
```

tli_enable

The description of tli_enable is shown as below:

Table 3-1712. Function tli_enable

Function name	tli_enable
Function prototype	void tli_enable(void);

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

Example:

```
/* enabled TLI */
```

```
tli_enable();
```

tli_disable

The description of tli_disable is shown as below:

Table 3-1713. Function tli_disable

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

Example:

```
/* disable TLI */
```

```
tli_disable();
```

tli_reload_config

The description of tli_reload_config is shown as below:

Table 3-1714. Function tli_reload_config

Function name	tli_reload_config
Function prototype	void tli_reload_config(uint8_t reload_mod);
Function descriptions	configure TLI reload mode

Precondition	-
The called functions	-
Input parameter{in}	
reload_mod	TLI reload mode
<i>TLI_FRAME_BLANK_RELOAD_EN</i>	the layer configuration will be reloaded at frame blank
<i>TLI_REQUEST_RELOAD_EN</i>	the layer configuration will be reloaded after this bit sets
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TLI reload mode */
tli_reload_config(TLI_REQUEST_RELOAD_EN);
```

tli_layer_struct_para_init

The description of tli_layer_struct_para_init is shown as below:

Table 3-1715. Function tli_layer_struct_para_init

Function name	tli_layer_struct_para_init
Function prototype	void tli_layer_struct_para_init(tli_layer_parameter_struct *layer_struct);
Function descriptions	initialize the parameters of TLI layer structure with the default values, it is suggested that call this function after a tli_layer_parameter_struct structure is defined
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
*layer_struct	a pointer to tli_layer_parameter_struct
Return value	
-	-

Example:

```
tli_layer_parameter_struct tli_layer_init_struct;

/* initialize the parameters of TLI layer structure with the default values */
tli_layer_struct_para_init(&tli_layer_init_struct);
```


tli_layer_init

The description of tli_layer_init is shown as below:

Table 3-1716. Function tli_layer_init

Function name	tli_layer_init
Function prototype	void tli_layer_init(uint32_t layerx,tli_layer_parameter_struct *layer_struct);
Function descriptions	initialize TLI layer
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
<i>LAYERx</i>	x=0, 1
Input parameter{in}	
*layer_struct	a pointer to tli_layer_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

tli_layer_parameter_struct tli_layer_init_struct;

tli_layer_struct_para_init(&tli_layer_init_struct);

/* TLI layer0 configuration */

tli_layer_init_struct.layer_window_leftpos = 20 + 43;

tli_layer_init_struct.layer_window_rightpos = (20 + 440 + 43 - 1);

tli_layer_init_struct.layer_window_toppos = 40 + 12;

tli_layer_init_struct.layer_window_bottompos = (40 + 182 + 12 - 1);

tli_layer_init_struct.layer_ppf = LAYER_PPF_RGB565;

/* TLI window specified alpha configuration */

tli_layer_init_struct.layer_sa = 255;

/* TLI layer default alpha R,G,B value configuration */

tli_layer_init_struct.layer_default_blue = 0xFF;

tli_layer_init_struct.layer_default_green = 0xFF;

tli_layer_init_struct.layer_default_red = 0xFF;

tli_layer_init_struct.layer_default_alpha = 0xFF;

```

```

/* TLI window blend configuration */

tli_layer_init_struct.layer_acf1 = LAYER_ACF1_PASA;

tli_layer_init_struct.layer_acf2 = LAYER_ACF2_PASA;

/* TLI layer frame buffer base address configuration */

tli_layer_init_struct.layer_frame_bufaddr = (uint32_t)&gBackground;

tli_layer_init_struct.layer_frame_line_length = ((440 * 2) + 3);

tli_layer_init_struct.layer_frame_buf_stride_offset = (440 * 2);

tli_layer_init_struct.layer_frame_total_line_number = 182;

tli_layer_init(LAYER0, &tli_layer_init_struct);

```

tli_layer_window_offset_modify

The description of tli_layer_window_offset_modify is shown as below:

Table 3-1717. Function tli_layer_window_offset_modify

Function name	tli_layer_window_offset_modify
Function prototype	void tli_layer_window_offset_modify(uint32_t layerx,uint16_t offset_x,uint16_t offset_y);
Function descriptions	reconfigure window position
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
LAYERx	x=0, 1
Input parameter{in}	
offset_x	new horizontal offset
Input parameter{in}	
offset_y	new vertical offset
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* reconfigure LAYER1 window position */

tli_layer_window_offset_modify(LAYER1, 20, 20);

```

tli_lut_struct_para_init

The description of tli_lut_struct_para_init is shown as below:

Table 3-1718. Function tli_lut_struct_para_init

Function name	tli_lut_struct_para_init
Function prototype	void tli_lut_struct_para_init(tli_layer_lut_parameter_struct *lut_struct);
Function descriptions	initialize the parameters of TLI layer LUT structure with the default values, it is suggested that call this function after a tli_layer_lut_parameter_struct structure is defined
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
*lut_struct	a pointer to tli_layer_lut_parameter_struct
Return value	
-	-

Example:

```
tli_layer_lut_parameter_struct  tli_lut_struct;
```

```
/* initialize the parameters of TLI layer LUT structure with the default values */
```

```
tli_lut_struct_para_init(&tli_lut_struct);
```

tli_lut_init

The description of tli_lut_init is shown as below:

Table 3-1719. Function tli_lut_init

Function name	tli_lut_init
Function prototype	void tli_lut_init(uint32_t layerx,tli_layer_lut_parameter_struct *lut_struct);
Function descriptions	initialize TLI layer LUT
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
LAYERx	x=0, 1
Input parameter{in}	
*lut_struct	a pointer to tli_layer_lut_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
tli_layer_lut_parameter_struct  tli_lut_struct;
```

```
tli_lut_struct_para_init(&tli_lut_struct);

/* initialize TLI layer0 LUT */

tli_lut_struct.layer_table_addr = 0x20003000;

tli_lut_struct.layer_lut_channel_red = 0x20;

tli_lut_struct.layer_lut_channel_green = 0x30;

tli_lut_struct.layer_lut_channel_blue = 0xFF;

tli_lut_init(LAYER0, &tli_lut_struct);
```

tli_color_key_init

The description of tli_color_key_init is shown as below:

Table 3-1720. Function tli_color_key_init

Function name	tli_color_key_init
Function prototype	void tli_color_key_init(uint32_t layerx,uint32_t redkey,uint32_t greenkey,uint32_t bluekey);
Function descriptions	initialize TLI layer color key
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
LAYERx	x=0, 1
Input parameter{in}	
redkey	color key red
Input parameter{in}	
greenkey	color key green
Input parameter{in}	
bluekey	color key blue
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TLI layer0 color key */

tli_color_key_init(LAYER0, 0xAA, 0xFF, 0x00);
```

tli_layer_enable

The description of tli_layer_enable is shown as below:

Table 3-1721. Function tli_layer_enable

Function name	tli_layer_enable
Function prototype	void tli_layer_enable(uint32_t layerx);
Function descriptions	enable TLI layer
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
LAYERx	x=0, 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TLI layer0 */
tli_layer_enable(LAYER0);
```

tli_layer_disable

The description of tli_layer_disable is shown as below:

Table 3-1722. Function tli_layer_disable

Function name	tli_layer_disable
Function prototype	void tli_layer_disable(uint32_t layerx);
Function descriptions	disable TLI layer
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
LAYERx	x=0, 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TLI layer0 */
tli_layer_disable(LAYER0);
```

tli_color_key_enable

The description of tli_color_key_enable is shown as below:

Table 3-1723. Function tli_color_key_enable

Function name	tli_color_key_enable
Function prototype	void tli_color_key_enable(uint32_t layerx);
Function descriptions	enable TLI layer color keying
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
<i>LAYERx</i>	x=0, 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TLI layer0 color keying */
tli_color_key_enable(LAYER0);
```

tli_color_key_disable

The description of tli_color_key_disable is shown as below:

Table 3-1724. Function tli_color_key_disable

Function name	tli_color_key_disable
Function prototype	void tli_color_key_disable(uint32_t layerx);
Function descriptions	disable TLI layer color keying
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
<i>LAYERx</i>	x=0, 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TLI layer0 color keying */
tli_color_key_disable(LAYER0);
```

tli_lut_enable

The description of tli_lut_enable is shown as below:

Table 3-1725. Function tli_lut_enable

Function name	tli_lut_enable
Function prototype	void tli_lut_enable(uint32_t layerx);
Function descriptions	enable TLI layer LUT
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
LAYERx	x=0, 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TLI layer0 LUT */
tli_lut_enable(LAYER0);
```

tli_lut_disable

The description of tli_lut_disable is shown as below:

Table 3-1726. Function tli_lut_disable

Function name	tli_lut_disable
Function prototype	void tli_lut_disable(uint32_t layerx);
Function descriptions	disable TLI layer0 LUT
Precondition	-
The called functions	-
Input parameter{in}	
layerx	layer
LAYERx	x=0, 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TLI layer0 LUT */
tli_lut_disable(LAYER0);
```

tli_line_mark_set

The description of tli_line_mark_set is shown as below:

Table 3-1727. Function tli_line_mark_set

Function name	tli_line_mark_set
Function prototype	void tli_line_mark_set(uint32_t line_num);
Function descriptions	set line mark value
Precondition	-
The called functions	-
Input parameter{in}	
line_num	line number
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set line mark value */
tli_line_mark_set(0x20);
```

tli_current_pos_get

The description of tli_current_pos_get is shown as below:

Table 3-1728. Function tli_current_pos_get

Function name	tli_current_pos_get
Function prototype	uint32_t tli_current_pos_get(void);
Function descriptions	get current displayed position
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	position of current pixel

Example:

```
uint32_t pos;
/* get current pixel position */
pos = tli_current_pos_get();
```

tli_interrupt_enable

The description of tli_interrupt_enable is shown as below:

Table 3-1729. Function tli_interrupt_enable

Function name	tli_interrupt_enable
Function prototype	void tli_interrupt_enable(uint32_t int_flag);
Function descriptions	enable TLI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	TLI interrupt flags
<i>TLI_INT_LM</i>	line mark interrupt
<i>TLI_INT_FE</i>	FIFO error interrupt
<i>TLI_INT_TE</i>	transaction error interrupt
<i>TLI_INT_LCR</i>	layer configuration reloaded interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TLI line mark interrupt */
tli_interrupt_enable(TLI_INT_LM);
```

tli_interrupt_disable

The description of tli_interrupt_disable is shown as below:

Table 3-1730. Function tli_interrupt_disable

Function name	tli_interrupt_disable
Function prototype	void tli_interrupt_disable(uint32_t int_flag);
Function descriptions	disable TLI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	TLI interrupt flags
<i>TLI_INT_LM</i>	line mark interrupt
<i>TLI_INT_FE</i>	FIFO error interrupt
<i>TLI_INT_TE</i>	transaction error interrupt
<i>TLI_INT_LCR</i>	layer configuration reloaded interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TLI line mark interrupt */
```

```
tli_interrupt_disable(TLI_INT_LM);
```

tli_interrupt_flag_get

The description of tli_interrupt_flag_get is shown as below:

Table 3-1731. Function tli_interrupt_flag_get

Function name	tli_interrupt_flag_get
Function prototype	FlagStatus tli_interrupt_flag_get(uint32_t int_flag);
Function descriptions	get TLI interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	TLI interrupt flags
<i>TLI_INT_FLAG_LM</i>	line mark interrupt flag
<i>TLI_INT_FLAG_FE</i>	FIFO error interrupt flag
<i>TLI_INT_FLAG_TE</i>	transaction error interrupt flag
<i>TLI_INT_FLAG_LCR</i>	layer configuration reloaded interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TLI interrupt flag */

if(SET == tli_interrupt_flag_get(TLI_INT_FLAG_LM)){

    tli_interrupt_flag_clear(TLI_INT_FLAG_LM);

}
```

tli_interrupt_flag_clear

The description of tli_interrupt_flag_clear is shown as below:

Table 3-1732. Function tli_interrupt_flag_clear

Function name	tli_interrupt_flag_clear
Function prototype	void tli_interrupt_flag_clear(uint32_t int_flag);
Function descriptions	clear TLI interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	TLI interrupt flags

<i>TLI_INT_FLAG_LM</i>	line mark interrupt flag
<i>TLI_INT_FLAG_FE</i>	FIFO error interrupt flag
<i>TLI_INT_FLAG_TE</i>	transaction error interrupt flag
<i>TLI_INT_FLAG_LCR</i>	layer configuration reloaded interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
if(SET == tli_interrupt_flag_get(TLI_INT_FLAG_LM)){
    /* clear TLI interrupt flag */
    tli_interrupt_flag_clear(TLI_INT_FLAG_LM);
}
```

tli_flag_get

The description of tli_flag_get is shown as below:

Table 3-1733. Function tli_flag_get

Function name	tli_flag_get
Function prototype	FlagStatus tli_flag_get(uint32_t flag);
Function descriptions	get TLI flag or state in TLI_INTF register or TLI_STAT register
Precondition	-
The called functions	-
Input parameter{in}	
flag	TLI flags or states
<i>TLI_FLAG_VDE</i>	current VDE state
<i>TLI_FLAG_HDE</i>	current HDE state
<i>TLI_FLAG_VS</i>	current VS status of the TLI
<i>TLI_FLAG_HS</i>	current HS status of the TLI
<i>TLI_FLAG_LM</i>	line mark interrupt flag
<i>TLI_FLAG_FE</i>	FIFO error interrupt flag
<i>TLI_FLAG_TE</i>	transaction error interrupt flag
<i>TLI_FLAG_LCR</i>	layer configuration reloaded interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* wait the TLI_FLAG_LM flag set */
```

```
while(RESET == tli_flag_get(TLI_FLAG_LM)){
}
```

3.45. 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 [3.45.1](#), the TMU firmware functions are introduced in chapter [3.45.2](#).

3.45.1. Descriptions of Peripheral registers

TMU registers are listed in the table shown as below:

Table 3-1734. TMU Registers

Registers	Descriptions
TMU_CS	TMU control and status register
TMU_IDATA	TMU input data register
TMU_ODATA	TMU output data register

3.45.2. Descriptions of Peripheral functions

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

Table 3-1735. 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 3-1736. 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)
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 3-1737. 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 3-1738. 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 Structure tmu_parameter_struct .
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 3-1739. Function tmu_init

Function name	tmu_init
Function prototype	void tmu_init(tmu_parameter_struct* init_struct);
Function descriptions	initialize TMU
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	pointer to TMU init parameter struct, the structure members can refer to Structure tmu_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

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 3-1740. Function tmu_read_interrupt_enable

Function name	tmu_read_interrupt_enable
Function prototype	void tmu_read_interrupt_enable(void);
Function descriptions	enable TMU read interrupt
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1741. Function tmu_read_interrupt_disable

Function name	tmu_read_interrupt_disable
----------------------	----------------------------

Function prototype	void tmu_read_interrupt_disable(void);
Function descriptions	disable TMU read interrupt
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1742. Function tmu_dma_read_enable

Function name	tmu_dma_read_enable
Function prototype	void tmu_dma_read_enable(void);
Function descriptions	enable TMU read interrupt
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TMU DMA read request */
tmu_dma_read_enable();
```

tmu_dma_read_disable

The description of tmu_dma_read_disable is shown as below:

Table 3-1743. Function tmu_dma_read_disable

Function name	tmu_dma_read_disable
Function prototype	void tmu_dma_read_disable(void);

Function descriptions	disable TMU read interrupt
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TMU DMA read request */
```

```
tmu_dma_read_disable();
```

tmu_dma_write_enable

The description of tmu_dma_write_enable is shown as below:

Table 3-1744. Function tmu_dma_write_enable

Function name	tmu_dma_write_enable
Function prototype	void tmu_dma_write_enable(void);
Function descriptions	enable TMU write interrupt
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TMU DMA write request */
```

```
tmu_dma_write_enable();
```

tmu_dma_write_disable

The description of tmu_dma_write_disable is shown as below:

Table 3-1745. Function tmu_dma_write_disable

Function name	tmu_dma_write_disable
Function prototype	void tmu_dma_write_disable(void);
Function descriptions	disable TMU write interrupt

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

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 3-1746. Function tmu_one_q31_write

Function name	tmu_one_q31_write
Function prototype	void tmu_one_q31_write(uint32_t data);
Function descriptions	write one data in q1.31 format
Precondition	-
The called functions	-
Input parameter{in}	
data	the input data in q1.31 format (0x00000000~0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1747. Function tmu_two_q31_write

Function name	tmu_two_q31_write
Function prototype	void tmu_two_q31_write(uint32_t data1, uint32_t data2);
Function descriptions	write two data in q1.31 format

Precondition	-
The called functions	-
Input parameter{in}	
data1	the first input data in q1.31 format (0x00000000~0xFFFFFFFF)
Input parameter{in}	
data2	the second input data in q1.31 format (0x00000000~0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write two data in q1.31 format */
int32_t in1 = -2000;
int32_t in2 = 3000;
tmu_two_q31_write((uint32_t)in1, (uint32_t)in2);
```

tmu_two_q15_write

The description of tmu_two_q15_write is shown as below:

Table 3-1748. Function tmu_two_q15_write

Function name	tmu_two_q15_write
Function prototype	void tmu_two_q15_write(uint16_t data1, uint16_t data2);
Function descriptions	write two data in q1.15 format
Precondition	-
The called functions	-
Input parameter{in}	
data1	the first input data in q1.15 format (0x0000~0xFFFF)
Input parameter{in}	
data2	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 3-1749. Function tmu_one_q31_read

Function name	tmu_one_q31_read
Function prototype	void tmu_one_q31_read(uint32_t* p);
Function descriptions	read one data in q1.31 format
Precondition	-
The called functions	-
Input parameter{in}	
p	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 3-1750. Function tmu_two_q31_read

Function name	tmu_two_q31_read
Function prototype	void tmu_two_q31_read(uint32_t* p1, uint32_t* p2);
Function descriptions	read two data in q1.31 format
Precondition	-
The called functions	-
Input parameter{in}	
p1	A pointer to the first output data(q1.31 format)
Input parameter{in}	
p2	A pointer to the second output data(q1.31 format)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* read two data in q1.31 format */
```

```
uint32_t out1 = 0;

uint32_t out2 = 0;

tmu_two_q31_read(&out1, &out2);
```

tmu_two_q15_read

The description of tmu_two_q15_read is shown as below:

Table 3-1751. Function tmu_two_q15_read

Function name	tmu_two_q15_read
Function prototype	void tmu_two_q15_read(uint16_t* p1, uint16_t* p2);
Function descriptions	read two data in q1.15 format
Precondition	-
The called functions	-
Input parameter{in}	
p1	A pointer to the first output data(q1.15 format)
Input parameter{in}	
p2	A pointer to the second output data(q1.15 format)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* read two data in q1.15 format */

uint16_t out1 = 0;

uint16_t out2 = 0;

tmu_two_q15_read(&out1, &out2);
```

3.46. 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 [3.46.1](#), the TRIGSEL firmware functions are introduced in chapter [3.46.2](#).

3.46.1. Descriptions of Peripheral registers

TRGSEL registers are listed in the table shown as below:

Table 3-1752. 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_LPDTS	TRIGSEL trigger selection for LPDTS register
TRIGSEL_TIMER0ETI	TRIGSEL trigger selection for TIMER0_ETI register
TRIGSEL_TIMER1ETI	TRIGSEL trigger selection for TIMER1_ETI register
TRIGSEL_TIMER2ETI	TRIGSEL trigger selection for TIMER2_ETI register
TRIGSEL_TIMER3ETI	TRIGSEL trigger selection for TIMER3_ETI register
TRIGSEL_TIMER4ETI	TRIGSEL trigger selection for TIMER4_ETI register
TRIGSEL_TIMER7ETI	TRIGSEL trigger selection for TIMER7_ETI register
TRIGSEL_TIMER22ETI	TRIGSEL trigger selection for TIMER22_ETI register
TRIGSEL_TIMER23ETI	TRIGSEL trigger selection for TIMER23_ETI register
TRIGSEL_TIMER30ETI	TRIGSEL trigger selection for TIMER30_ETI register
TRIGSEL_TIMER31ETI	TRIGSEL trigger selection for TIMER31_ETI register
TRIGSEL_EDOUT	TRIGSEL trigger selection for EDOUT register
TRIGSEL_HPDF	TRIGSEL trigger selection for HPDF register
TRIGSEL_TIMER0ITI14	TRIGSEL trigger selection for TIMER0_ITI14register
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

Registers	Descriptions
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_TIMER30ITI14	TRIGSEL trigger selection for TIMER30_ITI14 register
TRIGSEL_TIMER31ITI14	TRIGSEL trigger selection for TIMER31_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

3.46.2. Descriptions of Peripheral functions

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

Table 3-1753. 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 3-1754. Enum trigsel_source_enum

Member name	Function description
TRIGSEL_INPUT_0	Trigger input source 0
TRIGSEL_INPUT_1	Trigger input source 1
TRIGSEL_INPUT_TRIGSEL_IN0	Trigger input source TRIGSEL_IN0 pin
TRIGSEL_INPUT_TRIGSEL_IN1	Trigger input source TRIGSEL_IN1 pin
TRIGSEL_INPUT_TRIGSEL_IN2	Trigger input source TRIGSEL_IN2 pin
TRIGSEL_INPUT_TRIGSEL_IN3	Trigger input source TRIGSEL_IN3 pin
TRIGSEL_INPUT_TRIGSEL_IN4	Trigger input source TRIGSEL_IN4 pin
TRIGSEL_INPUT_TRIGSEL_IN5	Trigger input source TRIGSEL_IN5 pin
TRIGSEL_INPUT_TRIGSEL_IN6	Trigger input source TRIGSEL_IN6 pin
TRIGSEL_INPUT_TRIGSEL_IN7	Trigger input source TRIGSEL_IN7 pin
TRIGSEL_INPUT_TRIGSEL_IN8	Trigger input source TRIGSEL_IN8 pin
TRIGSEL_INPUT_TRIGSEL_IN9	Trigger input source TRIGSEL_IN9 pin
TRIGSEL_INPUT_TRIGSEL_IN10	Trigger input source TRIGSEL_IN10 pin
TRIGSEL_INPUT_TRIGSEL_IN11	Trigger input source TRIGSEL_IN11 pin

Member name	Function description
TRIGSEL_INPUT_TRIGSEL_IN12	Trigger input source TRIGSEL_IN12 pin
TRIGSEL_INPUT_TRIGSEL_IN13	Trigger input source TRIGSEL_IN13 pin
TRIGSEL_INPUT_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
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

Member name	Function description
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
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_TIMER30_TRGO0	Trigger input source TIMER30 TRGO0
TRIGSEL_INPUT_TIMER30_CH0	Trigger input source TIMER30 CH0

Member name	Function description
TRIGSEL_INPUT_TIMER30_CH1	Trigger input source TIMER30 CH1
TRIGSEL_INPUT_TIMER30_CH2	Trigger input source TIMER30 CH2
TRIGSEL_INPUT_TIMER30_CH3	Trigger input source TIMER30 CH3
TRIGSEL_INPUT_TIMER30_ETI	Trigger input source TIMER30 ETI
TRIGSEL_INPUT_TIMER31_TRGO0	Trigger input source TIMER31 TRGO0
TRIGSEL_INPUT_TIMER31_CH0	Trigger input source TIMER31 CH0
TRIGSEL_INPUT_TIMER31_CH1	Trigger input source TIMER31 CH1
TRIGSEL_INPUT_TIMER31_CH2	Trigger input source TIMER31 CH2
TRIGSEL_INPUT_TIMER31_CH3	Trigger input source TIMER31 CH3
TRIGSEL_INPUT_TIMER31_ETI	Trigger input source TIMER31 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

Member name	Function description
TRIGSEL_INPUT_ADC0_WD2_OUT	Trigger input source ADC0 watchdog2 output
TRIGSEL_INPUT_ADC1_WD0_OUT	Trigger input source ADC1 watchdog0 output
TRIGSEL_INPUT_ADC1_WD1_OUT	Trigger input source ADC1 watchdog1 output
TRIGSEL_INPUT_ADC1_WD2_OUT	Trigger input source ADC1 watchdog2 output
TRIGSEL_INPUT_ADC2_WD0_OUT	Trigger input source ADC2 watchdog0 output
TRIGSEL_INPUT_ADC2_WD1_OUT	Trigger input source ADC2 watchdog1 output
TRIGSEL_INPUT_ADC2_WD2_OUT	Trigger input source ADC2 watchdog2 output
TRIGSEL_INPUT_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 3-1755. 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

Member name	Function description
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
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_LPDS_TRG	Output target peripheral LPDS_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_TIMER30_ETI	Output target peripheral TIMER30_ETI
TRIGSEL_OUTPUT_TIMER31_ETI	Output target peripheral TIMER31_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_TIMER30_ITI14	Output target peripheral TIMER30_ITI14
TRIGSEL_OUTPUT_TIMER31_ITI14	Output target peripheral TIMER31_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 3-1756. Function trigsel_deinit

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

Example:

```
/* deinitialize TRIGSEL */
```

```
trigsel_deinit();
```

trigsel_init

The description of trigsel_init is shown as below:

Table 3-1757. Function trigsel_init

Function name	trigsel_init
Function prototype	void trigsel_init(trigsel_periph_enum target_periph, trigsel_source_enum trigger_source);
Function descriptions	Set the trigger input signal for target peripheral
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	Target peripheral value, refer to Table 3-1755. Enum trigsel_periph_enum
Input parameter{in}	
trigger_source	Trigger source value, refer to Table 3-1754. Enum trigsel_source_enum
Output parameter{out}	
-	-
Return value	
-	-

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 3-1758. Function trigsel_trigger_source_get

Function name	trigsel_trigger_source_get
Function prototype	trigsel_source_enum trigsel_trigger_source_get(trigsel_periph_enum target_periph);
Function descriptions	Get the trigger input signal for target peripheral
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	Target peripheral value, refer to Table 3-1755. Enum trigsel_periph_enum
Output parameter{out}	
-	-
Return value	
trigger_source	Trigger source value, refer to Table 3-1754. Enum trigsel_source_enum

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 3-1759. Function trigsel_register_lock_set

Function name	trigsel_register_lock_set
Function prototype	void trigsel_register_lock_set(trigsel_periph_enum target_periph);
Function descriptions	Lock the trigger register
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	Target peripheral value, refer to Table 3-1755. Enum trigsel_periph_enum
Output parameter{out}	
-	-
Return value	
-	-

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 3-1760. Function trigsel_register_lock_get

Function name	trigsel_register_lock_get
Function prototype	FlagStatus trigsel_register_lock_get(trigsel_periph_enum target_periph);
Function descriptions	Get the trigger register lock status
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	Target peripheral value, refer to Table 3-1755. Enum trigsel_periph_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the trigger register lock status of ADC0 */
FlagStatus status;
status = trigsel_register_lock_get(TRIGSEL_OUTPUT_ADC0_REGTRG);
```

3.47. 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 [3.47.1](#). the TRNG firmware functions are introduced in chapter [3.47.2](#).

3.47.1. Descriptions of Peripheral registers

TRNG registers are listed in the table shown as below:

Table 3-1761. TRNG Registers

Registers	Descriptions
TRNG_CTL	control register
TRNG_STAT	status register
TRNG_DATA	data register

Registers	Descriptions
TRNG_HTCFG	health tests configure register

3.47.2. Descriptions of Peripheral functions

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

Table 3-1762. TRNG firmware function

Function name	Function description
trng_deinit	deinitialize the TRNG
trng_enable	enable the TRNG interface
trng_disable	disable the TRNG interface
trng_lock	lock the TRNG control bits
trng_mode_config	Configure TRNG working mode
trng_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 3-1763. 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 3-1764. 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 3-1765. 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 3-1766. 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 3-1767. 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 3-1768. 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 3-1769. Function trng_enable

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

Example:

```
/* enable the TRNG interface */
```

```
trng_enable( );
```

trng_disable

The description of trng_disable is shown as below:

Table 3-1770. Function trng_disable

Function name	trng_disable
Function prototype	void trng_disable(void);
Function descriptions	disable the TRNG interface
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* disable the TRNG interface */
```

```
trng_disable( );
```

trng_lock

The description of trng_lock is shown as below:

Table 3-1771. Function trng_lock

Function name	trng_lock
Function prototype	void trng_lock (void);
Function descriptions	lock the TRNG control bits
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the TRNG control bits */
```

```
trng_lock( );
```

trng_mode_config

The description of trng_mode_config is shown as below:

Table 3-1772. Function trng_mode_config

Function name	trng_mode_config
Function prototype	void trng_mode_config(trng_modsel_enum mode_select);
Function descriptions	configure TRNG working mode
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
mode_select	-
TRNG_MODSEL_LFSR	TRNG working in LFSR mode
TRNG_MODSEL_NIST	TRNG working in NIST mode
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_postprocessing_enable

The description of trng_postprocessing_enable is shown as below:

Table 3-1773. Function trng_postprocessing_enable

Function name	trng_postprocessing_enable
Function prototype	void trng_postprocessing_enable(void);
Function descriptions	enable the TRNG post-processing 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_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 3-1774. Function trng_postprocessing_disable

Function name	trng_postprocessing_disable
Function prototype	void trng_postprocessing_disable(void);
Function descriptions	disable the TRNG post-processing 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_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 3-1775. Function trng_conditioning_enable

Function name	trng_conditioning_enable
Function prototype	void trng_conditioning_enable(void);
Function descriptions	enable 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_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 3-1776. 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 3-1777. Function trng_disable

Function name	trng_conditioning_input_bitwidth
Function prototype	void trng_conditioning_input_bitwidth(trng_inmod_enum input_bitwidth);
Function descriptions	configure TRNG conditioning module input bitwidth
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
input_bitwidth	refer to enum Table 3-1763. Enum trng_inmod_enum
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_output_bitwidth

The description of trng_conditioning_output_bitwidth is shown as below:

Table 3-1778. Function trng_conditioning_output_bitwidth

Function name	trng_conditioning_output_bitwidth
Function prototype	void trng_conditioning_output_bitwidth(trng_outmod_enum output_bitwidth);
Function descriptions	configure TRNG conditioning module output bitwidth
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
output_bitwidth	refer to enum Table 3-1764. Enum trng_outmod_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

```



```

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

trng_replace_test_enable

The description of trng_replace_test_enable is shown as below:

Table 3-1779. Function trng_replace_test_enable

Function name	trng_replace_test_enable
Function prototype	void trng_replace_test_enable(void);
Function descriptions	enable TRNG replace test
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable TRNG replace test */

trng_deinit();

trng_conditioning_reset_enable();

trng_replace_test_enable();

trng_enable();

trng_conditioning_reset_disable();

```

trng_replace_test_disable

The description of trng_replace_test_disable is shown as below:

Table 3-1780. Function trng_replace_test_disable

Function name	trng_replace_test_disable
----------------------	---------------------------

Function prototype	void trng_replace_test_disable(void);
Function descriptions	disable TRNG replace test
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

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 3-1781. Function trng_hash_init_enable

Function name	trng_hash_init_enable
Function prototype	void trng_hash_init_enable(void);
Function descriptions	enable hash algorithm init when conditioning module enabled
Precondition	trng_conditioning_reset_enable / trng_conditioning_enable
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* 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 3-1782. Function trng_hash_init_disable

Function name	trng_hash_init_disable
Function prototype	void trng_hash_init_disable(void);
Function descriptions	disable hash algorithm init when conditioning module enabled
Precondition	trng_conditioning_reset_enable / trng_conditioning_enable
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* disable the TRNG interface */

trng_deinit();

trng_conditioning_reset_enable();

trng_conditioning_enable();

trng_hash_init_disable();

trng_enable();

trng_conditioning_reset_disable();

```

trng_powermode_config

The description of trng_powermode_config is shown as below:

Table 3-1783. Function trng_powermode_config

Function name	trng_powermode_config
Function prototype	void trng_powermode_config(uint32_t powermode);
Function descriptions	configure TRNG analog power mode
Precondition	trng_conditioning_reset_enable

The called functions	-
Input parameter{in}	
powermode	the power mode selection
<i>TRNG_NR_ULATRL0W</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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1784. Function trng_disable

Function name	trng_clockdiv_config
Function prototype	void trng_clockdiv_config(uint32_t clkdiv);
Function descriptions	configure TRNG clock divider
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
clkdiv	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1785. Function trng_clockerror_detection_enable

Function name	trng_clockerror_detection_enable
Function prototype	void trng_clockerror_detection_enable(void);
Function descriptions	enable the TRNG clock error detection
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
-	-
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 3-1786. 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 3-1787. Function trng_get_true_random_data

Function name	trng_get_true_random_data
Function prototype	uint32_t trng_get_true_random_data(void);
Function descriptions	get the true random data
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	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 3-1788. Function trng_conditioning_reset_enable

Function name	trng_conditioning_reset_enable
Function prototype	void trng_conditioning_reset_enable (void)
Function descriptions	enable the conditioning logic reset
Precondition	-
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_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 3-1789. Function trng_conditioning_reset_disable

Function name	trng_conditioning_reset_disable
Function prototype	void trng_conditioning_reset_disable(void)
Function descriptions	disable the conditioning logic reset
Precondition	-
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_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 3-1790. Function trng_conditioning_algo_config

Function name	trng_conditioning_algo_config
Function prototype	void trng_conditioning_algo_config(uint32_t module_algo)
Function descriptions	configure the conditioning module hash algorithm
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
module_algo	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
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_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 3-1791. Function trng_health_tests_config

Function name	trng_health_tests_config
Function prototype	void trng_health_tests_config(uint32_t adpo_threshold, uint8_t rep_threshold)
Function descriptions	configure health tests default value
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
adpo_threshold	adaptive proportion test threshold value
Input parameter{in}	
rep_threshold	repetitive (00/11) test threshold value
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_health_tests_config(700, 50);

trng_enable();

```

trng_flag_get

The description of trng_flag_get is shown as below:

Table 3-1792. Function trng_flag_get

Function name	trng_flag_get
Function prototype	FlagStatus trng_flag_get(trng_flag_enum flag);
Function descriptions	get the trng status flags
Precondition	-
The called functions	-
Input parameter{in}	
flag	trng status flag, refer to Table 3-1766. Enum trng_flag_enum
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 3-1793. 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 3-1794. 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 3-1795. Function trng_interrupt_flag_get

Function name	trng_interrupt_flag_get
Function prototype	FlagStatus trng_interrupt_flag_get(trng_int_flag_enum int_flag)

Function descriptions	get the trng interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	trng interrupt flag, refer to Table 3-1767. Enum trng_int_flag_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-1796. Function trng_interrupt_flag_clear

Function name	trng_interrupt_flag_clear
Function prototype	void trng_interrupt_flag_clear(trng_int_flag_enum int_flag)
Function descriptions	clear the trng interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	trng interrupt flag, refer to Table 3-1767. Enum trng_int_flag_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*clear the trng interrupt flag */
```

```
trng_interrupt_flag_clear(TRNG_INT_FLAG_CEIF);
```

3.48. USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter [3.48.1](#), the USART firmware functions are introduced in chapter [3.48.2](#).

3.48.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-1797. 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

3.48.2. Descriptions of Peripheral functions

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

Table 3-1798. USART firmware function

Function name	Function description
usart_deinit	reset USART
usart_baudrate_set	configure USART baud rate value
usart_parity_config	configure USART parity function
usart_word_length_set	configure USART word length
usart_stop_bit_set	configure USART stop bit length
usart_enable	enable USART
usart_disable	disable USART
usart_transmit_config	configure USART transmitter
usart_receive_config	configure USART receiver
usart_data_first_config	data is transmitted/received with the LSB/MSB first
usart_invert_config	configure USART inverted
usart_overrun_enable	enable the USART overrun function
usart_overrun_disable	disable the USART overrun function
usart_oversample_config	configure the USART oversample mode
usart_sample_bit_config	configure sample bit method
usart_receiver_timeout_enable	enable receiver timeout
usart_receiver_timeout_disable	disable receiver timeout

Function name	Function description
usart_receiver_timeout_threshold_config	configure receiver timeout threshold
usart_data_transmit	USART transmit data function
usart_data_receive	USART receive data function
usart_command_enable	enable USART command
usart_address_0_match_mode_enable	enable address 0 match mode
usart_address_0_match_mode_disable	disable address 0 match mode
usart_address_1_match_mode_enable	enable address 1 match mode
usart_address_1_match_mode_disable	disable address 1 match mode
usart_address_0_config	configure address 0 of the USART
usart_address_1_config	configure address 1 of the USART
usart_address_0_detection_mode_config	configure address 0 detection mode
usart_address_1_detection_mode_config	configure address 1 detection mode
usart_mute_mode_enable	enable mute mode
usart_mute_mode_disable	disable mute mode
usart_mute_mode_wakeup_config	configure wakeup method in mute mode
usart_lin_mode_enable	enable LIN mode
usart_lin_mode_disable	disable LIN mode
usart_lin_break_detection_length_config	LIN break detection length
usart_halfduplex_enable	enable half-duplex mode
usart_halfduplex_disable	disable half-duplex mode
usart_clock_enable	enable clock
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

Function name	Function description
usart_block_length_config	configure block length
usart_irda_mode_enable	enable IrDA mode
usart_irda_mode_disable	disable IrDA mode
usart_prescaler_config	configure the peripheral clock prescaler in USART IrDA low-power or smartcard mode
usart_irda_lowpower_config	configure IrDA low-power
usart_hardware_flow_rts_config	configure hardware flow control RTS
usart_hardware_flow_cts_config	configure hardware flow control CTS
usart_hardware_flow_coherence_config	configure hardware flow control coherence mode
usart_rs485_driver_enable	enable RS485 driver
usart_rs485_driver_disable	disable RS485 driver
usart_driver_asserttime_config	configure driver enable assertion time
usart_driver_deasserttime_config	configure driver enable de-assertion time
usart_depolarity_config	configure driver enable polarity mode
usart_dma_receive_config	configure USART DMA for reception
usart_dma_transmit_config	configure USART DMA for transmission
usart_reception_error_dma_disable	disable DMA on reception error
usart_reception_error_dma_enable	enable DMA on reception error
usart_wakeup_enable	enable USART to wakeup the mcu from deep-sleep mode
usart_wakeup_disable	disable USART to wakeup the mcu from deep-sleep mode
usart_wakeup_mode_config	configure the USART wakeup mode from deep-sleep mode
usart_fifo_enable	enable FIFO
usart_fifo_disable	disable FIFO
usart_transmit_fifo_threshold_config	configure transmit FIFO threshold
usart_receive_fifo_threshold_config	configure receive FIFO threshold
usart_receive_fifo_counter_number	read receive FIFO counter number
usart_flag_get	get USART status
usart_flag_clear	clear USART status
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 3-1799. 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

Member name	Function description
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 3-1800. 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	read data buffer not empty interrupt and overrun error flag

Member name	Function description
RR	
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_ORERR	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 3-1801. 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
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 3-1802. 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 3-1803. Function usart_deinit

Function name	usart_deinit
Function prototype	void usart_deinit(uint32_t usart_periph);
Function descriptions	reset USART
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset USART0 */
usart_deinit(USART0);
```

usart_baudrate_set

The description of usart_baudrate_set is shown as below:

Table 3-1804. Function usart_baudrate_set

Function name	usart_baudrate_set
Function prototype	void usart_baudrate_set(uint32_t usart_periph, uint32_t baudval);
Function descriptions	configure USART baud rate value

Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
baudval	baud rate value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 baud rate value */
usart_baudrate_set(USART0, 115200);
```

usart_parity_config

The description of usart_parity_config is shown as below:

Table 3-1805. Function usart_parity_config

Function name	usart_parity_config
Function prototype	void usart_parity_config(uint32_t usart_periph, uint32_t paritycfg);
Function descriptions	configure USART parity
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
paritycfg	configure USART parity
<i>USART_PM_NONE</i>	no parity
<i>USART_PM_ODD</i>	odd parity
<i>USART_PM_EVEN</i>	even parity
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 parity */
```

```
usart_parity_config(USART0, USART_PM_EVEN);
```

usart_word_length_set

The description of usart_word_length_set is shown as below:

Table 3-1806. Function usart_word_length_set

Function name	usart_word_length_set
Function prototype	void usart_word_length_set(uint32_t usart_periph, uint32_t wlen);
Function descriptions	configure USART word length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Input parameter{in}	
wlen	USART word length configure
USART_WL_8BIT	8 bits
USART_WL_9BIT	9 bits
USART_WL_7BIT	7 bits
USART_WL_10BIT	10 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 word length */
```

```
usart_word_length_set(USART0, USART_WL_9BIT);
```

usart_stop_bit_set

The description of usart_stop_bit_set is shown as below:

Table 3-1807. 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

<i>UARTx</i>	<i>x</i> = 3, 4, 6, 7
Input parameter{in}	
stblen	USART stop bit configure
<i>USART_STB_1BIT</i>	1 bit
<i>USART_STB_0_5BIT</i>	0.5 bit
<i>USART_STB_2BIT</i>	2 bits
<i>USART_STB_1_5BIT</i>	1.5 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 stop bit length */
usart_stop_bit_set(USART0, USART_STB_1_5BIT);
```

usart_enable

The description of usart_enable is shown as below:

Table 3-1808. Function usart_enable

Function name	usart_enable
Function prototype	void usart_enable(uint32_t usart_periph);
Function descriptions	enable USART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	<i>x</i> = 0, 1, 2, 5
<i>UARTx</i>	<i>x</i> = 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 3-1809. Function `usart_disable`

Function name	<code>usart_disable</code>
Function prototype	<code>void usart_disable(uint32_t usart_periph);</code>
Function descriptions	disable USART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 */
```

```
usart_disable(USART0);
```

usart_transmit_config

The description of `usart_transmit_config` is shown as below:

Table 3-1810. Function `usart_transmit_config`

Function name	<code>usart_transmit_config</code>
Function prototype	<code>void usart_transmit_config(uint32_t usart_periph, uint32_t txconfig);</code>
Function descriptions	configure USART transmitter
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
txconfig	enable or disable USART transmitter
<i>USART_TRANSMIT_ENABLE</i>	enable USART transmission
<i>USART_TRANSMIT_DISABLE</i>	disable USART transmission
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 transmitter */

usart_transmit_config(USART0,USART_TRANSMIT_ENABLE);
```

usart_receive_config

The description of usart_receive_config is shown as below:

Table 3-1811. Function usart_receive_config

Function name	usart_receive_config
Function prototype	void usart_receive_config(uint32_t usart_periph, uint32_t rxconfig);
Function descriptions	configure USART receiver
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4,6, 7
Input parameter{in}	
rxconfig	enable or disable USART receiver
USART_RECEIVE_ENABLE	enable USART reception
USART_RECEIVE_DISABLE	disable USART reception
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 receiver */

usart_receive_config(USART0, USART_RECEIVE_ENABLE);
```

usart_data_first_config

The description of usart_data_first_config is shown as below:

Table 3-1812. Function usart_data_first_config

Function name	usart_data_first_config
Function prototype	void usart_data_first_config(uint32_t usart_periph, uint32_t msbf);
Function descriptions	data is transmitted/received with the LSB/MSB first
Precondition	-
The called functions	-

Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
msbf	LSB/MSB
<i>USART_MSBF_LSB</i>	LSB first
<i>USART_MSBF_MSB</i>	MSB first
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure LSB of data first */
```

```
usart_data_first_config(USART0, USART_MSBF_LSB);
```

usart_invert_config

The description of usart_invert_config is shown as below:

Table 3-1813. Function usart_invert_config

Function name	usart_invert_config
Function prototype	void usart_invert_config(uint32_t usart_periph, usart_invert_enum invertpara);
Function descriptions	configure USART inverted
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
invertpara	refer to Table 3-1802. Enum usart invert enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 inversion */
```

```
usart_invert_config(USART0, USART_DINV_ENABLE);
```


usart_oversize_enable

The description of usart_oversize_enable is shown as below:

Table 3-1814. Function usart_oversize_enable

Function name	usart_oversize_enable
Function prototype	void usart_oversize_enable(uint32_t usart_periph);
Function descriptions	enable the USART oversize function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 oversize */
usart_oversize_enable(USART0);
```

usart_oversize_disable

The description of usart_oversize_disable is shown as below:

Table 3-1815. Function usart_oversize_disable

Function name	usart_oversize_disable
Function prototype	void usart_oversize_disable(uint32_t usart_periph);
Function descriptions	disable the USART oversize function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 oversize */
```

```
usart_oversample_disable(USART0);
```

usart_oversample_config

The description of usart_oversample_config is shown as below:

Table 3-1816. Function usart_oversample_config

Function name	usart_oversample_config
Function prototype	void usart_oversample_config(uint32_t usart_periph, uint32_t oversamp);
Function descriptions	configure the USART oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Input parameter{in}	
oversamp	oversample value
USART_OVSMOD_8	oversampling by 8
USART_OVSMOD_16	oversampling by 16
Output parameter{out}	
-	-
Return value	
-	-

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 3-1817. Function usart_sample_bit_config

Function name	usart_sample_bit_config
Function prototype	void usart_sample_bit_config(uint32_t usart_periph, uint32_t osb);
Function descriptions	configure the sample bit method
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Input parameter{in}	

osb	sample bit
<i>USART_OSB_1BIT</i>	1 bit
<i>USART_OSB_3BIT</i>	3 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config USART0 1 bit sample mode */
usart_sample_bit_config(USART0, USART_OSB_1BIT);
```

usart_receiver_timeout_enable

The description of usart_receiver_timeout_enable is shown as below:

Table 3-1818. Function usart_receiver_timeout_enable

Function name	usart_receiver_timeout_enable
Function prototype	void usart_receiver_timeout_enable(uint32_t usart_periph);
Function descriptions	enable receiver timeout
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 receiver timeout */
usart_receiver_timeout_enable(USART0);
```

usart_receiver_timeout_disable

The description of usart_receiver_timeout_disable is shown as below:

Table 3-1819. Function usart_receiver_timeout_disable

Function name	usart_receiver_timeout_disable
Function prototype	void usart_receiver_timeout_disable(uint32_t usart_periph);
Function descriptions	disable receiver timeout
Precondition	-

The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 receiver timeout */
```

```
usart_receiver_timeout_disable(USART0);
```

usart_receiver_timeout_threshold_config

The description of usart_receiver_timeout_threshold_config is shown as below:

Table 3-1820. Function usart_receiver_timeout_threshold_config

Function name	usart_receiver_timeout_threshold_config
Function prototype	void usart_receiver_timeout_threshold_config(uint32_t usart_periph, uint32_t rtimeout);
Function descriptions	configure receiver timeout threshold
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
Input parameter{in}	
rtimeout	receiver timeout (0x00000000-0x00FFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the receiver timeout threshold of USART0 */
```

```
usart_receiver_timeout_threshold_config(USART0, 115200*3);
```

usart_data_transmit

The description of usart_data_transmit is shown as below:

Table 3-1821. Function usart_data_transmit

Function name	usart_data_transmit
Function prototype	void usart_data_transmit(uint32_t usart_periph, uint16_t data);
Function descriptions	USART transmit data function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Input parameter{in}	
data	data of transmission (0x0000-0x01FF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 transmit data */
usart_data_transmit(USART0, 0xAA);
```

usart_data_receive

The description of usart_data_receive is shown as below:

Table 3-1822. Function usart_data_receive

Function name	usart_data_receive
Function prototype	uint16_t usart_data_receive(uint32_t usart_periph);
Function descriptions	USART receive data function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
uint16_t	data of received (0x0000-0x01FF)

Example:

```
/* USART0 receive data */
uint16_t temp;
```

```
temp = usart_data_receive(USART0);
```

usart_command_enable

The description of usart_command_enable is shown as below:

Table 3-1823. Function usart_command_enable

Function name	usart_command_enable
Function prototype	void usart_command_enable(uint32_t usart_periph, uint32_t cmdtype);
Function descriptions	enable USART command
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
cmdtype	command type
<i>USART_CMD_SBKCMD</i> <i>D</i>	send break command
<i>USART_CMD_MMCMMD</i>	mute mode command
<i>USART_CMD_RXFCM</i> <i>D</i>	receive data flush command
<i>USART_CMD_TXFCM</i> <i>D</i>	transmit data flush request
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 command */
```

```
usart_command_enable(USART0, USART_CMD_SBKCMD);
```

usart_address_0_match_mode_enable

The description of usart_address_0_match_mode_enable is shown as below:

Table 3-1824. Function usart_address_0_match_mode_enable

Function name	usart_address_0_match_mode_enable
Function prototype	void usart_address_0_match_mode_enable(uint32_t usart_periph);
Function descriptions	enable address 0 match mode
Precondition	-
The called functions	-

Input parameter{in}	
usart_periph	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 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 3-1825. Function usart_address_0_match_mode_disable

Function name	usart_address_0_match_mode_disable
Function prototype	void usart_address_0_match_mode_disable(uint32_t usart_periph);
Function descriptions	disable address 0 match mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

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 3-1826. Function usart_address_1_match_mode_enable

Function name	usart_address_1_match_mode_enable
Function prototype	void usart_address_1_match_mode_enable(uint32_t usart_periph);

Function descriptions	enable address 1 match mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	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 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 3-1827. 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
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	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 3-1828. Function `usart_address_0_config`

Function name	<code>usart_address_0_config</code>
Function prototype	<code>void usart_address_0_config(uint32_t usart_periph, uint8_t addr);</code>
Function descriptions	configure address 0 of the USART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
addr	address of USART (0x00-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address 0 of the USART0 */
usart_address_0_config(USART0, 0x00);
```

`usart_address_1_config`

The description of `usart_address_1_config` is shown as below:

Table 3-1829. Function `usart_address_1_config`

Function name	<code>usart_address_1_config</code>
Function prototype	<code>void usart_address_1_config(uint32_t usart_periph, uint8_t addr);</code>
Function descriptions	configure address 1 of the USART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
addr	address of USART (0x00-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address 1 of the USART0 */
```

```
usart_address_1_config(USART0, 0x00);
```

usart_address_0_detection_mode_config

The description of usart_address_0_detection_mode_config is shown as below:

Table 3-1830. Function usart_address_0_detection_mode_config

Function name	usart_address_0_detection_mode_config
Function prototype	void usart_address_0_detection_mode_config(uint32_t usart_periph, uint32_t addmod);
Function descriptions	configure address 0 detection mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Input parameter{in}	
addmod	address detection mode
USART_ADDM_4BIT	4 bits
USART_ADDM_FULLBIT	full bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*configure address 0 detection mode */
```

```
usart_address_0_detection_mode_config(USART0, USART_ADDM_4BIT);
```

usart_address_1_detection_mode_config

The description of usart_address_1_detection_mode_config is shown as below:

Table 3-1831. Function usart_address_1_detection_mode_config

Function name	usart_address_1_detection_mode_config
Function prototype	void usart_address_1_detection_mode_config(uint32_t usart_periph, uint32_t addmod);
Function descriptions	configure address 1 detection mode
Precondition	-
The called functions	-

Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
addmod	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 3-1832. Function usart_mute_mode_enable

Function name	usart_mute_mode_enable
Function prototype	void usart_mute_mode_enable(uint32_t usart_periph);
Function descriptions	enable mute mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 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 3-1833. Function usart_mute_mode_disable

Function name	usart_mute_mode_disable
Function prototype	void usart_mute_mode_disable(uint32_t usart_periph);
Function descriptions	disable mute mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 mute mode */
```

```
usart_mute_mode_disable(USART0);
```

usart_mute_mode_wakeup_config

The description of usart_mute_mode_wakeup_config is shown as below:

Table 3-1834. Function usart_mute_mode_wakeup_config

Function name	usart_mute_mode_wakeup_config
Function prototype	void usart_mute_mode_wakeup_config(uint32_t usart_periph, uint32_t wmethod);
Function descriptions	configure wakeup method in mute mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
wmethod	two methods be used to enter or exit the mute mode
<i>USART_WM_IDLE</i>	idle line
<i>USART_WM_ADDR</i>	address mask
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* configure USART0 wakeup method in mute mode */
```

```
usart_mute_mode_wakeup_config(USART0, USART_WM_IDLE);
```

usart_lin_mode_enable

The description of usart_lin_mode_enable is shown as below:

Table 3-1835. Function usart_lin_mode_enable

Function name	usart_lin_mode_enable
Function prototype	void usart_lin_mode_enable(uint32_t usart_periph);
Function descriptions	enable LIN mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

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 3-1836. Function usart_lin_mode_disable

Function name	usart_lin_mode_disable
Function prototype	void usart_lin_mode_disable(uint32_t usart_periph);
Function descriptions	disable LIN mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable USART0 LIN mode */
```

```
usart_lin_mode_disable(USART0);
```

usart_lin_break_dection_length_config

The description of usart_lin_break_dection_length_config is shown as below:

Table 3-1837. Function usart_lin_break_dection_length_config

Function name	usart_lin_break_dection_length_config
Function prototype	void usart_lin_break_dection_length_config(uint32_t usart_periph, uint32_t lblen);
Function descriptions	LIN break detection length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Input parameter{in}	
lblen	two methods be used to enter or exit the mute mode
<i>USART_LBLEN_10B</i>	10 bits
<i>USART_LBLEN_11B</i>	11 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure LIN break frame length */
```

```
usart_lin_break_dection_length_config(USART0, USART_LBLEN_10B);
```

usart_halfduplex_enable

The description of usart_halfduplex_enable is shown as below:

Table 3-1838. Function usart_halfduplex_enable

Function name	usart_halfduplex_enable
Function prototype	void usart_halfduplex_enable(uint32_t usart_periph);
Function descriptions	enable half-duplex mode

Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 half duplex mode */
```

```
usart_halfduplex_enable(USART0);
```

usart_halfduplex_disable

The description of usart_halfduplex_disable is shown as below:

Table 3-1839. Function usart_halfduplex_disable

Function name	usart_halfduplex_disable
Function prototype	void usart_halfduplex_disable(uint32_t usart_periph);
Function descriptions	disable half-duplex mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 half duplex mode */
```

```
usart_halfduplex_disable(USART0);
```

usart_clock_enable

The description of usart_clock_enable is shown as below:

Table 3-1840. Function usart_clock_enable

Function name	usart_clock_enable
Function prototype	void usart_clock_enable(uint32_t usart_periph);
Function descriptions	enable clock
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable clock */
usart_clock_enable(USART0);
```

usart_clock_disable

The description of usart_clock_disable is shown as below:

Table 3-1841. 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 3-1842. Function `usart_synchronous_clock_config`

Function name	<code>usart_synchronous_clock_config</code>
Function prototype	<code>void usart_synchronous_clock_config(uint32_t usart_periph, uint32_t clen, uint32_t cph, uint32_t cpl);</code>
Function descriptions	configure USART synchronous mode parameters
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Input parameter{in}	
clen	last bit clock pulse
<i>USART_CLEN_NONE</i>	clock pulse of the last data bit (MSB) is not output to the CK pin
<i>USART_CLEN_EN</i>	clock pulse of the last data bit (MSB) is output to the CK pin
Input parameter{in}	
cph	clock phase
<i>USART_CPH_1CK</i>	first clock transition is the first data capture edge
<i>USART_CPH_2CK</i>	second clock transition is the first data capture edge
Input parameter{in}	
cpl	clock polarity
<i>USART_CPL_LOW</i>	steady low value on CK pin
<i>USART_CPL_HIGH</i>	steady high value on CK pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 synchronous mode parameters */
```

```
usart_synchronous_clock_config(USART0, USART_CLEN_EN, USART_CPH_2CK,
USART_CPL_HIGH);
```

usart_guard_time_config

The description of `usart_guard_time_config` is shown as below:

Table 3-1843. Function `usart_guard_time_config`

Function name	<code>usart_guard_time_config</code>
Function prototype	<code>void usart_guard_time_config(uint32_t usart_periph, uint32_t guat);</code>
Function descriptions	configure guard time value in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Input parameter{in}	
guat	guard time value (0x00000000-0x000000FF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 guard time value in smartcard mode */
```

```
usart_guard_time_config(USART0, 0x00000055);
```

usart_smartcard_mode_enable

The description of usart_smartcard_mode_enable is shown as below:

Table 3-1844. Function usart_smartcard_mode_enable

Function name	usart_smartcard_mode_enable
Function prototype	void usart_smartcard_mode_enable(uint32_t usart_periph);
Function descriptions	enable smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

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 3-1845. Function usart_smartcard_mode_disable

Function name	usart_smartcard_mode_disable
Function prototype	void usart_smartcard_mode_disable(uint32_t usart_periph);
Function descriptions	disable smartcard mode

Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

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 3-1846. Function usart_smartcard_mode_nack_enable

Function name	usart_smartcard_mode_nack_enable
Function prototype	void usart_smartcard_mode_nack_enable(uint32_t usart_periph);
Function descriptions	enable NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 NACK in smartcard mode */
usart_smartcard_mode_nack_enable(USART0);
```

usart_smartcard_mode_nack_disable

The description of usart_smartcard_mode_nack_disable is shown as below:

Table 3-1847. Function usart_smartcard_mode_nack_disable

Function name	usart_smartcard_mode_nack_disable
Function prototype	void usart_smartcard_mode_nack_disable(uint32_t usart_periph);

Function descriptions	disable NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 NACK in smartcard mode */
usart_smartcard_mode_nack_disable(USART0);
```

usart_smartcard_mode_early_nack_enable

The description of usart_smartcard_mode_early_nack_enable is shown as below:

Table 3-1848. Function usart_smartcard_mode_early_nack_enable

Function name	usart_smartcard_mode_early_nack_enable
Function prototype	void usart_smartcard_mode_early_nack_enable(uint32_t usart_periph);
Function descriptions	enable early NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 early NACK in smartcard mode */
usart_smartcard_mode_early_nack_enable(USART0);
```

usart_smartcard_mode_early_nack_disable

The description of usart_smartcard_mode_early_nack_disable is shown as below:

Table 3-1849. Function usart_smartcard_mode_early_nack_disable

Function name	usart_smartcard_mode_early_nack_disable
----------------------	---

Function prototype	void usart_smartcard_mode_early_nack_disable(uint32_t usart_periph);
Function descriptions	disable early NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 early NACK in smartcard mode */
usart_smartcard_mode_early_nack_disable(USART0);
```

usart_smartcard_autoretry_config

The description of usart_smartcard_autoretry_config is shown as below:

Table 3-1850. Function usart_smartcard_autoretry_config

Function name	usart_smartcard_autoretry_config
Function prototype	void usart_smartcard_autoretry_config(uint32_t usart_periph, uint32_t scrtnum);
Function descriptions	configure smartcard auto-retry number
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Input parameter{in}	
scrtnum	smartcard auto-retry number (0x00000000-0x00000007)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure smartcard auto-retry number */
usart_smartcard_autoretry_config(USART0, 0x00000007);
```

usart_block_length_config

The description of usart_block_length_config is shown as below:

Table 3-1851. Function usart_block_length_config

Function name	usart_block_length_config
Function prototype	void usart_block_length_config(uint32_t usart_periph, uint32_t bl);
Function descriptions	configure block length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Input parameter{in}	
bl	block length(0x00000000-0x000000FF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure block length in smartcard T=1 reception */
usart_block_length_config(USART0, 0x000000FF);
```

usart_irda_mode_enable

The description of usart_irda_mode_enable is shown as below:

Table 3-1852. Function usart_irda_mode_enable

Function name	usart_irda_mode_enable
Function prototype	void usart_irda_mode_enable(uint32_t usart_periph);
Function descriptions	enable IrDA mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 IrDA mode */
```

```
usart_irda_mode_enable(USART0);
```

usart_irda_mode_disable

The description of usart_irda_mode_disable is shown as below:

Table 3-1853. Function usart_irda_mode_disable

Function name	usart_irda_mode_disable
Function prototype	void usart_irda_mode_disable(uint32_t usart_periph);
Function descriptions	disable IrDA mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 IrDA mode */
```

```
usart_irda_mode_disable(USART0);
```

usart_prescaler_config

The description of usart_prescaler_config is shown as below:

Table 3-1854. Function usart_prescaler_config

Function name	usart_prescaler_config
Function prototype	void usart_prescaler_config(uint32_t usart_periph, uint8_t psc);
Function descriptions	configure the peripheral clock prescaler in USART IrDA low-power or smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Input parameter{in}	
psc	clock prescaler (0x00-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1855. Function usart_irda_lowpower_config

Function name	usart_irda_lowpower_config
Function prototype	void usart_irda_lowpower_config(uint32_t usart_periph, uint32_t irlp);
Function descriptions	configure IrDA low-power
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
Input parameter{in}	
irlp	IrDA low-power or normal
USART_IRLP_LOW	low-power
USART_IRLP_NORMAL	normal
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 IrDA low-power */
```

```
usart_irda_lowpower_config(USART0, USART_IRLP_LOW);
```

usart_hardware_flow_rts_config

The description of usart_hardware_flow_rts_config is shown as below:

Table 3-1856. Function usart_hardware_flow_rts_config

Function name	usart_hardware_flow_rts_config
Function prototype	void usart_hardware_flow_rts_config(uint32_t usart_periph, uint32_t rtsconfig);
Function descriptions	configure hardware flow control RTS
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
rtsconfig	enable or disable RTS
<i>USART_RTS_ENABLE</i>	enable RTS
<i>USART_RTS_DISABLE</i>	disable RTS
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 hardware flow control RTS */
```

```
usart_hardware_flow_rts_config(USART0, USART_RTS_ENABLE);
```

usart_hardware_flow_cts_config

The description of usart_hardware_flow_cts_config is shown as below:

Table 3-1857. Function usart_hardware_flow_cts_config

Function name	usart_hardware_flow_cts_config
Function prototype	void usart_hardware_flow_cts_config(uint32_t usart_periph, uint32_t ctsconfig);
Function descriptions	configure hardware flow control CTS
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
ctsconfig	enable or disable CTS
<i>USART_CTS_ENABLE</i>	enable CTS
<i>USART_CTS_DISABLE</i>	disable CTS
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 hardware flow control CTS */
```

```
usart_hardware_flow_cts_config(USART0, USART_CTS_ENABLE);
```

usart_hardware_flow_coherence_config

The description of usart_hardware_flow_coherence_config is shown as below:

Table 3-1858. Function usart_hardware_flow_coherence_config

Function name	usart_hardware_flow_coherence_config
Function prototype	void usart_hardware_flow_coherence_config(uint32_t usart_periph, uint32_t hcm);
Function descriptions	configure hardware flow control coherence mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Input parameter{in}	
hcm	hardware flow control coherence mode
USART_HCM_NONE	nRTS signal equals to the rxne status register
USART_HCM_EN	nRTS signal is set when the last data bit has been sampled
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure hardware flow control coherence mode */
```

```
usart_hardware_flow_coherence_config(USART0, USART_HCM_NONE);
```

usart_rs485_driver_enable

The description of usart_rs485_driver_enable is shown as below:

Table 3-1859. Function usart_rs485_driver_enable

Function name	usart_rs485_driver_enable
Function prototype	void usart_rs485_driver_enable(uint32_t usart_periph);
Function descriptions	enable USART RS485 driver
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral

USARTx	x = 0, 1, 2, 5
UARTx	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 3-1860. Function usart_rs485_driver_disable

Function name	usart_rs485_driver_disable
Function prototype	void usart_rs485_driver_disable(uint32_t usart_periph);
Function descriptions	disable USART RS485 driver
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 RS485 driver */
```

```
usart_rs485_driver_disable(USART0);
```

usart_driver_asserttime_config

The description of usart_driver_asserttime_config is shown as below:

Table 3-1861. Function usart_driver_asserttime_config

Function name	usart_driver_asserttime_config
Function prototype	void usart_driver_asserttime_config(uint32_t usart_periph, uint32_t deatime);
Function descriptions	configure driver enable assertion time

Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
deatime	driver enable assertion time (0x00000000-0x0000001F)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set USART0 driver asserttime */
```

```
usart_driver_asserttime_config(USART0, 0x0000001F);
```

usart_driver_deasserttime_config

The description of usart_driver_deasserttime_config is shown as below:

Table 3-1862. Function usart_driver_deasserttime_config

Function name	usart_driver_deasserttime_config
Function prototype	void usart_driver_deasserttime_config(uint32_t usart_periph, uint32_t dedtime);
Function descriptions	configure driver enable de-assertion time
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
dedtime	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 3-1863. Function usart_depolarity_config

Function name	usart_depolarity_config
Function prototype	void usart_depolarity_config(uint32_t usart_periph, uint32_t dep);
Function descriptions	configure driver enable polarity mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Input parameter{in}	
dep	DE signal
USART_DEP_HIGH	DE signal is active high
USART_DEP_LOW	DE signal is active low
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure driver enable polarity mode */
```

```
usart_depolarity_config(USART0, USART_DEP_HIGH);
```

usart_dma_receive_config

The description of usart_dma_receive_config is shown as below:

Table 3-1864. Function usart_dma_receive_config

Function name	usart_dma_receive_config
Function prototype	void usart_dma_receive_config(uint32_t usart_periph, uint32_t dmacmd);
Function descriptions	configure USART DMA for reception
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}	
dmacmd	USART DMA mode
USART_RECEIVE_DM	USART enable DMA for reception

<i>A_ENABLE</i>	
<i>USART_RECEIVE_DMA_DISABLE</i>	USART disable DMA for reception
Output parameter{out}	
-	-
Return value	
-	-

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 3-1865. Function usart_dma_transmit_config

Function name	usart_dma_transmit_config
Function prototype	void usart_dma_transmit_config(uint32_t usart_periph, uint32_t dmacmd);
Function descriptions	configure USART DMA transmission
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
dmacmd	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
Output parameter{out}	
-	-
Return value	
-	-

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 3-1866. Function usart_reception_error_dma_disable

Function name	usart_reception_error_dma_disable
Function prototype	void usart_reception_error_dma_disable(uint32_t usart_periph);
Function descriptions	disable DMA on reception error
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA on reception error */
usart_reception_error_dma_disable(USART0);
```

usart_reception_error_dma_enable

The description of usart_reception_error_dma_enable is shown as below:

Table 3-1867. Function usart_reception_error_dma_enable

Function name	usart_reception_error_dma_enable
Function prototype	void usart_reception_error_dma_enable(uint32_t usart_periph);
Function descriptions	enable DMA on reception error
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA on reception error */
```

```
usart_reception_error_dma_enable(USART0);
```

usart_wakeup_enable

The description of usart_wakeup_enable is shown as below:

Table 3-1868. Function usart_wakeup_enable

Function name	usart_wakeup_enable
Function prototype	void usart_wakeup_enable(uint32_t usart_periph);
Function descriptions	enable USART to wakeup the mcu from deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 wake up */
usart_wakeup_enable(USART0);
```

usart_wakeup_disable

The description of usart_wakeup_disable is shown as below:

Table 3-1869. Function usart_wakeup_disable

Function name	usart_wakeup_disable
Function prototype	void usart_wakeup_disable(uint32_t usart_periph);
Function descriptions	disable USART to wakeup the mcu from deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 wake up */
```



```
usart_wakeup_disable(USART0);
```

usart_wakeup_mode_config

The description of usart_wakeup_mode_config is shown as below:

Table 3-1870. Function usart_wakeup_mode_config

Function name	usart_wakeup_mode_config
Function prototype	void usart_wakeup_mode_config(uint32_t usart_periph, uint32_t wum);
Function descriptions	configure the USART wakeup mode from deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
Input parameter{in}	
wum	wakeup mode
USART_WUM_ADDR	WUF active on address match
USART_WUM_START B	WUF active on start bit
USART_WUM_RBNE	WUF active on RBNE
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 wake up mode */
usart_wakeup_mode_config(USART0, USART_WUM_ADDR);
```

usart_fifo_enable

The description of usart_fifo_enable is shown as below:

Table 3-1871. Function usart_fifo_enable

Function name	usart_fifo_enable
Function prototype	void usart_fifo_enable(uint32_t usart_periph);
Function descriptions	enable FIFO
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 FIFO */
usart_fifo_enable (USART0);
```

usart_fifo_disable

The description of usart_fifo_disable is shown as below:

Table 3-1872. Function usart_fifo_disable

Function name	usart_fifo_disable
Function prototype	void usart_fifo_disable(uint32_t usart_periph);
Function descriptions	disable FIFO
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 FIFO */
usart_fifo_disable(USART0);
```

usart_transmit_fifo_threshold_config

The description of usart_transmit_fifo_threshold_config is shown as below:

Table 3-1873. 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}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
txthreshold	transmit FIFO threshold
<i>USART_TFTCFG_THRESHOLD_1_8</i>	transmit FIFO reaches 1/8 of its depth
<i>USART_TFTCFG_THRESHOLD_1_4</i>	transmit FIFO reaches 1/4 of its depth
<i>USART_TFTCFG_THRESHOLD_1_2</i>	transmit FIFO reaches 1/8 of its depth
<i>USART_TFTCFG_THRESHOLD_3_4</i>	transmit FIFO reaches 3/4 of its depth
<i>USART_TFTCFG_THRESHOLD_7_8</i>	transmit FIFO reaches 7/8 of its depth
<i>USART_TFTCFG_THRESHOLD_EMPTY</i>	transmit FIFO becomes empty
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 3-1874. Function usart_receive_fifo_threshold_config

Function name	usart_receive_fifo_threshold_config
Function prototype	void usart_receive_fifo_threshold_config(uint32_t usart_periph, uint32_t rxthreshold);
Function descriptions	configure receive FIFO threshold
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
rxthreshold	receive FIFO threshold
<i>USART_RFTCFG_THRESHOLD_1_8</i>	receive FIFO reaches 1/8 of its depth
<i>USART_RFTCFG_THRESHOLD_1_4</i>	receive FIFO reaches 1/4 of its depth
<i>USART_RFTCFG_THRESHOLD_1_2</i>	receive FIFO reaches 1/2 of its depth
<i>USART_RFTCFG_THRESHOLD_3_4</i>	receive FIFO reaches 3/4 of its depth
<i>USART_RFTCFG_THRESHOLD_7_8</i>	receive FIFO reaches 7/8 of its depth
<i>USART_RFTCFG_THRESHOLD_FULL</i>	receive FIFO becomes full
Output parameter{out}	
-	-
Return value	
-	-

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 3-1875. Function usart_receive_fifo_counter_number

Function name	usart_receive_fifo_counter_number
Function prototype	uint8_t usart_receive_fifo_counter_number(uint32_t usart_periph);
Function descriptions	read receive FIFO counter number
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-

Return value	
uint8_t	receive FIFO counter number

Example:

```
/* read receive FIFO counter number */

uint8_t temp;

temp = usart_receive_fifo_counter_number(USART0);
```

usart_flag_get

The description of usart_flag_get is shown as below:

Table 3-1876. Function usart_flag_get

Function name	usart_flag_get
Function prototype	FlagStatus usart_flag_get(uint32_t usart_periph, usart_flag_enum flag);
Function descriptions	get flag in STAT/CHC/FCS register
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}	
flag	USART flags, refer to Table 3-1799. Enum usart_flag_enum only one among these parameters can be selected
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get flag USART0 state */

FlagStatus status;

status = usart_flag_get(USART0, USART_FLAG_TBE);
```

usart_flag_clear

The description of usart_flag_clear is shown as below:

Table 3-1877. Function usart_flag_clear

Function name	usart_flag_clear
Function prototype	void usart_flag_clear(uint32_t usart_periph, usart_flag_enum flag);
Function descriptions	clear flag in STAT register

Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
flag	USART flags, refer to Table 3-1799. Enum usart flag enum only one among these parameters can be selected
<i>USART_FLAG_PERR</i>	parity error flag
<i>USART_FLAG_FERR</i>	frame error flag
<i>USART_FLAG_NERR</i>	noise detected flag
<i>USART_FLAG_ORER</i> <i>R</i>	overrun error flag
<i>USART_FLAG_IDLE</i>	idle line detected flag
<i>USART_FLAG_TC</i>	transmission complete flag
<i>USART_FLAG_LBD</i>	LIN break detected flag
<i>USART_FLAG_CTSF</i>	CTS change flag
<i>USART_FLAG_RT</i>	receiver timeout flag
<i>USART_FLAG_EB</i>	end of block flag
<i>USART_FLAG_AM0</i>	address 0 match flag
<i>USART_FLAG_AM1</i>	address 1 match flag
<i>USART_FLAG_WU</i>	wakeup from deep-sleep mode flag
<i>USART_FLAG_EPERR</i>	early parity error flag
<i>USART_FLAG_TFE</i>	transmit FIFO empty flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear USART0 flag */
```

```
usart_flag_clear(USART0, USART_FLAG_TC);
```

usart_interrupt_enable

The description of usart_interrupt_enable is shown as below:

Table 3-1878. Function usart_interrupt_enable

Function name	usart_interrupt_enable
Function prototype	void usart_interrupt_enable(uint32_t usart_periph, usart_interrupt_enum interrupt);
Function descriptions	enable USART interrupt
Precondition	-

The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
interrupt	interrupt type, refer to Table 3-1801. Enum usart_interrupt_enum only one among these parameters can be selected
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 TBE interrupt */
```

```
usart_interrupt_enable(USART0, USART_INT_TBE);
```

usart_interrupt_disable

The description of usart_interrupt_disable is shown as below:

Table 3-1879. Function usart_interrupt_disable

Function name	usart_interrupt_disable
Function prototype	void usart_interrupt_disable(uint32_t usart_periph, usart_interrupt_enum interrupt);
Function descriptions	disable USART interrupt
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
interrupt	interrupt type, refer to Table 3-1801. Enum usart_interrupt_enum only one among these parameters can be selected
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 TBE interrupt */
```

```
usart_interrupt_disable(USART0, USART_INT_TBE);
```

usart_interrupt_flag_get

The description of usart_interrupt_flag_get is shown as below:

Table 3-1880. Function usart_interrupt_flag_get

Function name	usart_interrupt_flag_get
Function prototype	FlagStatus usart_interrupt_flag_get(uint32_t usart_periph, usart_interrupt_flag_enum int_flag);
Function descriptions	get USART interrupt and flag status
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
int_flag	USART interrupt flag, refer to Table 3-1800. Enum usart_interrupt_flag_enum , only one among these parameters can be selected
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the USART0 interrupt flag status */
```

```
FlagStatus status;
```

```
status = usart_interrupt_flag_get(USART0, USART_INT_FLAG_RBNE);
```

usart_interrupt_flag_clear

The description of usart_interrupt_flag_clear is shown as below:

Table 3-1881. Function usart_interrupt_flag_clear

Function name	usart_interrupt_flag_clear
Function prototype	void usart_interrupt_flag_clear(uint32_t usart_periph, usart_interrupt_flag_enum int_flag);
Function descriptions	clear USART interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	

int_flag	USART interrupt flag, refer to Table 3-1800. Enum usart_interrupt_flag_enum , only one among these parameters can be selected
USART_INT_FLAG_PERRR	parity error flag
USART_INT_FLAG_ERFERR	frame error flag
USART_INT_FLAG_ER_NERR	noise detected flag
USART_INT_FLAG_RBNE_ORERR	read data buffer not empty interrupt and overrun error flag
USART_INT_FLAG_ER_ORERR	error interrupt and overrun error
USART_INT_FLAG_IDLE	idle line detected flag
USART_INT_FLAG_TC	transmission complete flag
USART_INT_FLAG_LBD	LIN break detected flag
USART_INT_FLAG_CTS	CTS change flag
USART_INT_FLAG_RTO	receiver timeout flag
USART_INT_FLAG_EOB	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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the USART0 interrupt flag */
```

```
usart_interrupt_flag_clear(USART0, USART_INT_FLAG_TC);
```

3.49. VREF

The precision internal reference is used to provide reference voltage for ADC / DAC, or used by off-chip circuit connecting to V_{REFP} pin. The VREF registers are listed in chapter [3.49.1](#), the VREF firmware functions are introduced in chapter [3.49.2](#).

3.49.1. Descriptions of Peripheral registers

VREF registers are listed in the table shown as below:

Table 3-1882. VREF Registers

Registers	Descriptions
VREF_CS	VREF control and status register
VREF_CALIB	VREF calibration register

3.49.2. Descriptions of Peripheral functions

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

Table 3-1883. 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 3-1884. Function vref_deinit

Function name	vref_deinit
Function prototype	void vref_deinit(void);
Function descriptions	Deinitialize the VREF
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* deinitialize the VREF */
```

```
vref_deinit();
```

vref_enable

The description of vref_enable is shown as below:

Table 3-1885. Function vref_enable

Function name	vref_enable
Function prototype	void vref_enable(void);
Function descriptions	Enable VREF
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable VREF */
```

```
vref_enable();
```

vref_disable

The description of vref_disable is shown as below:

Table 3-1886. Function vref_disable

Function name	vref_disable
Function prototype	void vref_disable(void);
Function descriptions	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 3-1887. 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 3-1888. 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 3-1889. 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 3-1890. 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}	
vref_voltage	VREF voltage reference select
VREF_VOLTAGE_SEL_2_5V	VREF voltage reference select 2.5 V
VREF_VOLTAGE_SEL	VREF voltage reference select 2.048 V

<code>_2_048V</code>	
<code>VREF_VOLTAGE_SEL</code> <code>_1_8V</code>	VREF voltage reference select 1.8 V
<code>VREF_VOLTAGE_SEL</code> <code>_1_5V</code>	VREF voltage reference select 1.5 V
Output parameter{out}	
-	-
Return value	
-	-

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 3-1891. Function vref_calib_value_set

Function name	vref_calib_value_set
Function prototype	void vref_calib_value_set(uint8_t value);
Function descriptions	Set the calibration value of VREF
Precondition	-
The called functions	-
Input parameter{in}	
value	Calibration value (0x00 - 0x3F)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1892. Function vref_calib_value_get

Function name	vref_calib_value_get
Function prototype	uint8_t vref_calib_value_get(void);
Function descriptions	Get the calibration value of VREF
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	Calibration value (0x00 - 0x3F)

Example:

```
/* get the calibration value of VREF */
```

```
uint8_t cal_val;
```

```
cal_val = vref_calib_value_get();
```

3.50. WWDGT

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. The WWDGT registers are listed in chapter [3.50.1](#), the WWDGT firmware functions are introduced in chapter [3.50.2](#).

3.50.1. Descriptions of Peripheral registers

WWDGT registers are listed in the table shown as below:

Table 3-1893. WWDGT Registers

Registers	Descriptions
WWDGT_CTL	WWDGT control register
WWDGT_CFG	WWDGT configuration register
WWDGT_STAT	WWDGT status register

3.50.2. Descriptions of Peripheral functions

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

Table 3-1894. WWDGT firmware function

Function name	Function description
wwdgt_deinit	reset the window watchdog timer configuration
wwdgt_enable	start the window watchdog timer counter
wwdgt_counter_update	configure the window watchdog timer counter value
wwdgt_config	configure counter value, window value, and prescaler divider value
wwdgt_interrupt_enable	enable early wakeup interrupt of WWDGT
wwdgt_flag_get	check early wakeup interrupt state of WWDGT
wwdgt_flag_clear	clear early wakeup interrupt state of WWDGT

wwdgt_deinit

The description of wwdgt_deinit is shown as below:

Table 3-1895. 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 3-1896. 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 3-1897. Function wwdgt_counter_update

Function name	wwdgt_counter_update
Function prototype	void wwdgt_counter_update(uint16_t counter_value);
Function descriptions	configure the window watchdog timer counter value
Precondition	-
The called functions	-
Input parameter{in}	
counter_value	counter_value: 0x0000 - 0x007F
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* update WWDGT counter to 0x7F */
```

```
wwdgt_counter_update(127);
```

wwdgt_config

The description of wwdgt_config is shown as below:

Table 3-1898. Function wwdgt_config

Function name	wwdgt_config
Function prototype	void wwdgt_config(uint16_t counter, uint16_t window, uint32_t prescaler);
Function descriptions	configure counter value, window value, and prescaler divider value
Precondition	-
The called functions	-
Input parameter{in}	
counter	counter: 0x0000 - 0x007F
Input parameter{in}	
window	window: 0x0000 - 0x007F
Input parameter{in}	
prescaler	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	the time base of WWDGT counter = (PCLK3/4096)/8

IV8	
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 3-1899. 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 3-1900. 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 3-1901. Function wwdgt_flag_clear

Function name	wwdgt_flag_clear
Function prototype	void wwdgt_flag_clear(void);
Function descriptions	clear early wakeup interrupt state of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear early wakeup interrupt state of WWDGT */
```

```
wwdgt_flag_clear();
```

4. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	1. Initial Release	2. Mar.31, 2023
1.1	3. Change the function crc_block_data_calculate_byte (in Table 3-233) / crc_block_data_calculate_byte_halfword (in Table 3-234) / crc_block_data_calculate_byte_word (in Table 3-235) to crc_block_data_calculate in Table 3-233 .	Jun.21, 2023
1.2	1. Delete some members of enum usart_flag_enum (in Table 3-1799), add enum to function usart_interrupt_flag_clear. 2. Consistent update for CMP (in chapter 3.5). 3. Consistent update for DAC (in chapter 3.9).	Jan.5, 2024
1.3	1. Modify the description of actual arguments in Table 3-1047 , Table 3-1049 to Table 3-1054 .	Jul.10, 2024
1.4	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 Table 3-1173 . Function pmu_smps_ldo_supply_config .	Jan.24, 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 under the intellectual property laws and treaties of the People's Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as its suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.