

GigaDevice Semiconductor Inc.

GD32F3x0

ARM[®] Cortex[®]-M4 32-bit MCU

**HAL Firmware Library
User Guide**

Revision 1.0

(Sep 2023)

Table of Contents

Table of Contents	2
List of Figures	6
List of Tables	7
1. Introduction.....	24
1.1. Rules of User Manual and HAL Firmware Library.....	24
1.1.1. Peripherals.....	24
1.1.2. Naming rules.....	25
2. HAL Firmware Library Overview.....	26
2.1. Framework of HAL Firmware Library	26
2.2. Features of HAL Firmware Library	27
2.3. Introduction to the code framework of the HAL firmware library	28
2.4. File Structure of Firmware Library.....	33
2.4.1. Examples Folder.....	33
2.4.2. Firmware Folder.....	34
2.4.3. Utilities Folder.....	34
2.5. Code Migration Instructions	35
2.5.1. New project.....	35
2.5.2. Generate Code.....	37
2.5.3. Copy Code.....	38
2.5.4. Build and download	40
2.5.5. Run project	42
2.6. File descriptions of Firmware Library.....	42
2.7. FAQ for HAL firmware library	43
2.7.1. In many modules, the first parameter of the function is the device information structure, and what is the function of the structure?	43
2.7.2. How to use functions in HAL firmware library? Is there any reference routine?.....	44
2.7.3. Why doesn't the device work properly with the code automatically generated by the host computer?.....	44
2.7.4. Can HAL firmware libraries be tailored? what features can be modified?.....	44
2.7.5. What are the differences between a function prefixed with hal_ and a function prefixed with hals_?	46
3. Firmware Library of Standard Peripherals.....	48
3.1. Overview of Firmware Library of Standard Peripherals.....	48

3.2. ADC	48
3.2.1. Descriptions of Peripheral registers	48
3.2.2. Descriptions of Peripheral functions	49
3.3. CEC	80
3.3.1. Descriptions of Peripheral registers	80
3.3.2. Descriptions of Peripheral functions	80
3.4. CMP	90
3.4.1. Descriptions of Peripheral registers	90
3.4.2. Descriptions of Peripheral functions	91
3.5. CRC	101
3.5.1. Descriptions of Peripheral registers	101
3.5.2. Descriptions of Peripheral functions	102
3.6. CTC	107
3.6.1. Descriptions of Peripheral registers	107
3.6.2. Descriptions of Peripheral functions	107
3.7. DAC	116
3.7.1. Descriptions of Peripheral registers	116
3.7.2. Descriptions of Peripheral functions	116
3.8. SYS	128
3.8.1. Descriptions of Peripheral registers	128
3.8.2. Descriptions of Peripheral functions	128
3.9. DMA	135
3.9.1. Descriptions of Peripheral registers	135
3.9.2. Descriptions of Peripheral functions	136
3.10. EXTI	148
3.10.1. Descriptions of Peripheral registers	148
3.10.2. Descriptions of Peripheral functions	148
3.11. FMC	158
3.11.1. Descriptions of Peripheral registers	158
3.11.2. Descriptions of Peripheral functions	158
3.12. FWDGT	175
3.12.1. Descriptions of Peripheral registers	176
3.12.2. Descriptions of Peripheral functions	176
3.13. GPIO	180
3.13.1. Descriptions of Peripheral registers	180
3.13.2. Descriptions of Peripheral functions	180
3.14. I2C	185
3.14.1. Descriptions of Peripheral registers	185

3.14.2. Descriptions of Peripheral functions	186
3.15. SMBUS	217
3.15.1. Descriptions of Peripheral registers.....	217
3.15.2. Descriptions of Peripheral functions	217
3.16. NVIC	229
3.16.1. Descriptions of Peripheral registers.....	230
3.16.2. Descriptions of Peripheral functions	230
3.17. PMU.....	233
3.17.1. Descriptions of Peripheral registers.....	233
3.17.2. Descriptions of Peripheral functions	233
3.18. RCU	243
3.18.1. Descriptions of Peripheral registers.....	243
3.18.2. Descriptions of Peripheral functions	244
3.19. RTC.....	273
3.19.1. Descriptions of Peripheral registers.....	273
3.19.2. Descriptions of Peripheral functions	274
3.20. SPI.....	293
3.20.1. Descriptions of Peripheral registers.....	293
3.20.2. Descriptions of Peripheral functions	294
3.21. I2S	314
3.21.1. Descriptions of Peripheral registers.....	314
3.21.2. Descriptions of Peripheral functions	315
3.22. TIMER.....	330
3.22.1. Descriptions of Peripheral registers.....	330
3.22.2. Descriptions of Peripheral functions	331
3.23. TSI	406
3.23.1. Descriptions of Peripheral registers.....	406
3.23.2. Descriptions of Peripheral functions	407
3.24. UART	417
3.24.1. Descriptions of Peripheral registers.....	418
3.24.2. Descriptions of Peripheral functions	418
3.25. USRT	437
3.25.1. Descriptions of Peripheral registers.....	437
3.25.2. Descriptions of Peripheral functions	438
3.26. IrDA	459
3.26.1. Descriptions of Peripheral registers.....	460
3.26.2. Descriptions of Peripheral functions	460
3.27. SMARTCARD	478

3.27.1.	Descriptions of Peripheral registers.....	478
3.27.2.	Descriptions of Peripheral functions	478
3.28.	WWDGT.....	496
3.28.1.	Descriptions of Peripheral registers.....	496
3.28.2.	Descriptions of Peripheral functions	496
3.29.	USBFS.....	506
3.29.1.	Descriptions of Peripheral registers.....	506
3.29.2.	Descriptions of Peripheral functions	506
4.	Revision history	508

List of Figures

Figure 2-1. HAL firmware library implementation framework of GD32F3X0	26
Figure 2-2. Implementation of callback function.....	30
Figure 2-3. Device interrupt callback function structure of DMA.....	31
Figure 2-4. hal_dma_irq() for DMA	32
Figure 2-5. adc_irq_handle_set function.....	33
Figure 2-6. File structure of GD32F3X0_HAL_firmware library	33
Figure 2-7. New project interface	35
Figure 2-8. Generate code interface	37
Figure 2-9. After generate code interface.....	38
Figure 2-10. Project address search.....	39
Figure 2-11. Copy and rename process.....	40
Figure 2-12. Bulid project.....	41
Figure 2-13. Debug project.....	41
Figure 2-14. Run project.....	42

List of Tables

Table 1-1. Peripherals.....	24
Table 2-1. Peripheral function format of Firmware Library	43
Table 3-1. Peripheral function format of Firmware Library	48
Table 3-2. ADC Registers.....	48
Table 3-3. ADC firmware function.....	49
Table 3-4. Enum hal_adc_struct_type_enum.....	50
Table 3-5. Enum hal_adc_state_enum	51
Table 3-6. Enum hal_adc_error_enum	51
Table 3-7. Enum hal_adc_oversample_shift_enum.....	52
Table 3-8. Enum hal_adc_oversample_ratio_enum.....	52
Table 3-9. Enum hal_adc_routine_sequence_enum.....	53
Table 3-10. Enum hal_adc_inserted_sequence_enum.....	53
Table 3-11. Structure hal_adc_irq_struct	54
Table 3-12. Structure hal_adc_dma_handle_cb_struct.....	54
Table 3-13. Structure hal_adc_dev_struct.....	54
Table 3-14. Structure hal_adc_init_struct.....	54
Table 3-15. Structure hal_adc_routine_rank_config_struct	55
Table 3-16. Structure hal_adc_routine_config_struct.....	55
Table 3-17. Structure hal_adc_inserted_rank_config_struct.....	55
Table 3-18. Structure hal_adc_inserted_config_struct.....	56
Table 3-19. Structure hal_adc_watchdog_config_struct.....	56
Table 3-20. Function hal_adc_struct_init	56
Table 3-21. Function hal_adc_deinit.....	57
Table 3-22. Function hal_adc_init.....	58
Table 3-23. Function hal_adc_calibration_start.....	59
Table 3-24. Function hal_adc_routine_channel_config	59
Table 3-25. Function hal_adc_routine_rank_config	60
Table 3-26. Function hal_adc_start.....	61
Table 3-27. Function hal_adc_stop.....	62
Table 3-28. Function hal_adc_routine_conversion_poll	62
Table 3-29. Function hal_adc_routine_software_trigger_enable.....	63
Table 3-30. Function hal_adc_start_interrupt.....	63
Table 3-31. Function hal_adc_stop_interrupt	64
Table 3-32. Function hal_adc_start_dma	65
Table 3-33. Function hal_adc_stop_dma.....	66
Table 3-34. Function hal_adc_inserted_channel_config.....	66
Table 3-35. Function hal_adc_inserted_rank_config	67
Table 3-36. Function hal_adc_inserted_start.....	68
Table 3-37. Function hal_adc_inserted_stop.....	69

Table 3-38. Function hal_adc_inserted_conversion_poll.....	69
Table 3-39. Function hal_adc_inserted_software_trigger_enable	70
Table 3-40. Function hal_adc_inserted_start_interrupt.....	71
Table 3-41. Function hal_adc_inserted_stop_interrupt	71
Table 3-42. Function hal_adc_watchdog_config.....	72
Table 3-43. Function hal_adc_watchdog_interrupt_enable.....	73
Table 3-44. Function hal_adc_watchdog_interrupt_disable.....	74
Table 3-45. Function hal_adc_watchdog_event_poll.....	74
Table 3-46. Function hal_adc_irq	75
Table 3-47. Function hal_adc_irq_handle_set	76
Table 3-48. Function hal_adc_irq_handle_all_reset.....	77
Table 3-49. Function hal_adc_routine_value_get.....	77
Table 3-50. Function hal_adc_inserted_value_get.....	78
Table 3-51. Function hal_adc_error_get.....	78
Table 3-52. Function hal_adc_state_get.....	79
Table 3-53. CEC Registers.....	80
Table 3-54. CEC firmware function.....	80
Table 3-55. Enum hal_cec_state_enum.....	81
Table 3-56. Enum hal_cec_struct_type_enum	81
Table 3-57. Enum hal_cec_error_enum.....	81
Table 3-58. Structure hal_cec_irq_struct	82
Table 3-59. Structure hal_cec_dev_struct.....	82
Table 3-60. Structure hal_cec_init_struct.....	82
Table 3-61. Function hal_cec_struct_init	83
Table 3-62. Function hal_cec_init.....	84
Table 3-63. Function hal_cec_init.....	85
Table 3-64. Function hal_cec_start.....	85
Table 3-65. Function hal_cec_stop.....	86
Table 3-66. Function hal_cec_transmit_interrupt.....	86
Table 3-67. Function hal_cec_receive_interrupt	87
Table 3-68. Function hal_cec_irq_handle_set.....	88
Table 3-69. Function hal_cec_irq_handle_all_reset	89
Table 3-70. Function hal_cec_irq.....	90
Table 3-71. CMP Registers.....	90
Table 3-72. CMP firmware function.....	91
Table 3-73. Enum hal_cmp_state_enum.....	91
Table 3-74. Enum hal_cmp_struct_type_enum.....	91
Table 3-75. Enum hal_cmp_noninverting_input_enum.....	92
Table 3-76. Enum hal_cmp_exti_type_enum	92
Table 3-77. Structure hal_cmp_init_struct.....	92
Table 3-78. Structure hal_cmp_dev_struct.....	93
Table 3-79. Structure hal_cmp_irq_struct	93
Table 3-80. Function hal_cmp_struct_init	93

Table 3-81. Function hal_cmp_deinit.....	94
Table 3-82. Function hal_cmp_init.....	94
Table 3-83. Function hal_cmp_start.....	95
Table 3-84. Function hal_cmp_stop.....	96
Table 3-85. Function hal_cmp_start_interrupt.....	96
Table 3-86. Function hal_cmp_stop_interrupt.....	97
Table 3-87. Function hal_cmp_irq_handle_set.....	98
Table 3-88. Function hal_cmp_irq_handle_all_reset.....	98
Table 3-89. Function hal_cmp_irq.....	99
Table 3-90. Function hal_cmp_lock.....	100
Table 3-91. Function hal_cmp_output_level_get.....	100
Table 3-92. Function hal_cmp_state_get.....	101
Table 3-93. CRC Registers.....	101
Table 3-94. CRC firmware function.....	102
Table 3-95. Enum hal_crc_state_enum.....	102
Table 3-96. Enum hal_crc_struct_type_enum.....	102
Table 3-97. Structure hal_crc_dev_struct.....	102
Table 3-98. Structure hal_crc_init_struct.....	103
Table 3-99. Function hal_crc_struct_init.....	103
Table 3-100. Function hal_crc_init.....	104
Table 3-101. Function hal_crc_deinit.....	104
Table 3-102. Function hal_crc_single_data_calculate.....	105
Table 3-103. Function hal_crc_block_data_calculate.....	106
Table 3-104. CTC Registers.....	107
Table 3-105. CTC firmware function.....	107
Table 3-106. Enum hal_ctc_struct_type_enum.....	108
Table 3-107. Enum hal_ctc_error_enum.....	108
Table 3-108. Enum hal_ctc_state_enum.....	108
Table 3-109. Structure hal_ctc_irq_struct.....	108
Table 3-110. Structure hal_ctc_init_struct.....	108
Table 3-111. Structure hal_ctc_dev_struct.....	109
Table 3-112. Function hal_ctc_deinit.....	109
Table 3-113. Function hal_ctc_init.....	110
Table 3-114. Function hal_ctc_irc48m_trim_value_config.....	110
Table 3-115. Function hal_ctc_struct_init.....	111
Table 3-116. Function hal_ctc_start.....	112
Table 3-117. Function hal_ctc_stop.....	112
Table 3-118. Function hal_ctc_irq.....	113
Table 3-119. Function hal_ctc_irq_handle_set.....	113
Table 3-120. Function hal_ctc_irq_handle_all_reset.....	114
Table 3-121. Function hal_ctc_start_interrupt.....	114
Table 3-122. Function hal_ctc_stop_interrupt.....	115
Table 3-123. DAC Registers.....	116

Table 3-124. DAC firmware function	116
Table 3-125. Enum hal_dac_state_enum	117
Table 3-126. Enum hal_dac_struct_type_enum	117
Table 3-127. Enum hal_dac_error_enum	117
Table 3-128. Structure hal_dac_irq_struct	118
Table 3-129. Structure hal_dac_dma_handle_cb_struct	118
Table 3-130. Structure hal_dac_dev_struct	118
Table 3-131. Structure hal_dac_init_struct	118
Table 3-132. Function hal_dac_init	119
Table 3-133. Function hal_dac_init	120
Table 3-134. Function hal_dac_struct_init	120
Table 3-135. Function hal_dac_start	121
Table 3-136. Function hal_dac_stop	122
Table 3-137. Function hal_dac_start_interrupt	122
Table 3-138. Function hal_dac_stop_interrupt	123
Table 3-139. Function hal_dac_start_dma	124
Table 3-140. Function hal_dac_stop_dma	125
Table 3-141. Function hal_dac_irq	126
Table 3-142. Function hal_dac_irq_handle_set	126
Table 3-143. Function hal_dac_irq_handle_all_reset	127
Table 3-112. SYS DBG & SysTick Registers	128
Table 3-113. SYS firmware function	128
Table 3-114. Enum hal_sys_debug_cfg_enum	129
Table 3-115. Enum hal_sys_timebase_source_enum	129
Table 3-122. Function hal_sys_init	129
Table 3-121. Function hal_sys_init	130
Table 3-123. Function hal_sys_timesource_init	130
Table 3-125. Function hal_sys_basetick_count_get	131
Table 3-124. Function hal_sys_basetick_timeout_check	131
Table 3-126. Function hal_sys_basetick_delay_ms	132
Table 3-127. Function hal_sys_basetick_suspend	132
Table 3-128. Function hal_sys_basetick_resume	133
Table 3-130. Function hal_sys_basetick_irq	133
Table 3-131. Function hal_sys_basetick_irq_handle_set	134
Table 3-132. Function hal_sys_basetick_irq_handle_all_reset	135
Table 3-144. DMA Registers	135
Table 3-145. DMA firmware function	136
Table 3-146. dma_channel_enum	136
Table 3-147. hal_dma_struct_type_enum	136
Table 3-148. hal_dma_transfer_state_enum	137
Table 3-149. hal_dma_error_enum	137
Table 3-150. hal_dma_state_enum	137
Table 3-151. Structure hal_dma_irq_struct	138

Table 3-152. Structure hal_dma_dev_struct.....	138
Table 3-153. Structure hal_dma_init_struct.....	138
Table 3-154. Function hal_dma_init.....	138
Table 3-155. Function hal_dma_struct_init.....	139
Table 3-156. Function hal_dma_deinit.....	140
Table 3-157. Function hal_dma_start.....	141
Table 3-158. Function hal_dma_stop.....	142
Table 3-159. Function hal_dma_irq.....	142
Table 3-160. Function hal_dma_irq_handle_set.....	143
Table 3-161. Function hal_dma_irq_handle_all_reset.....	143
Table 3-162. Function hal_dma_start_poll.....	144
Table 3-163. Function hal_dma_start_interrupt.....	145
Table 3-164. Function hal_dma_stop_interrupt.....	146
Table 3-165. Function hal_dma_channel_remap_enable.....	147
Table 3-166. Function hal_dma_channel_remap_disable.....	147
Table 3-167. EXTI registers.....	148
Table 3-168. EXTI firmware function.....	148
3-169. hal_exti_type_enum.....	149
Table 3-170. hal_exti_line_enum.....	150
Table 3-171. hal_exti_internal_line_enum.....	151
Table 3-172. hal_exti_irq_index.....	151
Table 3-173. Function hal_exti_gpio_deinit.....	152
Table 3-174. Function hal_exti_internal_deinit.....	153
Table 3-175. Function hal_exti_gpio_init.....	153
Table 3-177. Function hal_exti_internal_init.....	155
Table 3-178. Function hal_exti_gpio_irq_handle_set.....	155
Table 3-179. Function hal_exti_gpio_irq_handle_all_reset.....	156
Table 3-180. Function hal_exti_gpio_irq.....	157
Table 3-195. FWDGT Registers.....	176
Table 3-182. FWDGT firmware function.....	176
3-183. hal_fwdgt_state_enum.....	176
Table 3-184. hal_fwdgt_struct_type_enum.....	177
Table 3-185. hal_fwdgt_dev_struct.....	177
Table 3-186. hal_fwdgt_init_struct.....	177
Table 3-187. Function hal_fwdgt_struct_init.....	177
Table 3-188. Function hal_fwdgt_init.....	178
Table 3-189. Function hal_fwdgt_deinit.....	179
Table 3-204. GPIO Registers.....	180
Table 3-205. GPIO firmware function.....	180
Table 3-206. Enum hal_gpio_mode_enum.....	181
Table 3-207. Enum hal_gpio_pull_enum.....	181
Table 3-208. Enum hal_gpio_ospeed_enum.....	181
Table 3-209. Enum hal_gpio_af_enum.....	181

Table 3-210. Structure hal_gpio_init_struct.....	182
Table 3-211. Function hal_gpio_init.....	182
Table 3-212. Function hal_gpio_bit_set.....	183
Table 3-213. Function hal_gpio_bit_reset.....	183
Table 3-196. Function hal_gpio_struct_init.....	184
Table 3-197. Function hal_gpio_deinit.....	184
Table 3-216. I2C Registers.....	185
Table 3-217. I2C firmware function.....	186
Table 3-218. Enum i2c_flag_enum.....	187
Table 3-219. Enum i2c_interrupt_flag_enum.....	188
Table 3-220. Enum i2c_interrupt_enum.....	188
Table 3-221. Enum hal_i2c_struct_type_enum.....	189
Table 3-222. Enum hal_i2c_run_state_enum.....	189
Table 3-223. Structure i2c_buffer_struct.....	189
Table 3-224. Structure hal_i2c_irq_struct.....	189
Table 3-225. Structure hal_i2c_slave_address_struct.....	190
Table 3-226. Structure hal_i2c_dev_struct.....	190
Table 3-227. Structure hal_i2c_init_struct.....	190
Table 3-228. Function hal_i2c_struct_init.....	191
Table 3-229. Function hal_i2c_deinit.....	191
Table 3-230. Function hal_i2c_init.....	192
Table 3-231. Function hal_i2c_error_irq.....	193
Table 3-232. Function hal_i2c_event_irq.....	194
Table 3-233. Function hal_i2c_start.....	194
Table 3-234. Function hal_i2c_stop.....	195
Table 3-235. Function hal_i2c_irq_handle_set.....	195
Table 3-236. Function hal_i2c_irq_handle_all_reset.....	196
Table 3-237. Function hal_i2c_master_transmit_poll.....	196
Table 3-238. Function hal_i2c_master_receive_poll.....	197
Table 3-239. Function hal_i2c_slave_transmit_poll.....	198
Table 3-240. Function hal_i2c_slave_receive_poll.....	199
Table 3-241. Function hal_i2c_memory_write_poll.....	200
Table 3-242. Function hal_i2c_memory_read_poll.....	201
Table 3-243. Function hal_i2c_master_transmit_interrupt.....	201
Table 3-244. Function hal_i2c_master_receive_interrupt.....	202
Table 3-245. Function hal_i2c_slave_transmit_interrupt.....	203
Table 3-246. Function hal_i2c_slave_receive_interrupt.....	204
Table 3-247. Function hal_i2c_memory_write_interrupt.....	205
Table 3-248. Function hal_i2c_memory_read_interrupt.....	206
Table 3-249. Function hal_i2c_master_transmit_dma.....	207
Table 3-250. Function hal_i2c_master_receive_dma.....	207
Table 3-251. Function hal_i2c_slave_transmit_dma.....	208
Table 3-252. Function hal_i2c_slave_receive_dma.....	209

Table 3-253. Function hal_i2c_memory_write_dma	210
Table 3-254. Function hal_i2c_memory_read_dma	211
Table 3-255. Function hal_i2c_device_ready_check	212
Table 3-256. Function hal_i2c_master_serial_transmit_interrupt.....	212
Table 3-257. Function hal_i2c_master_serial_receive_interrupt.....	213
Table 3-258. Function hal_i2c_slave_serial_transmit_interrupt.....	214
Table 3-259. Function hal_i2c_slave_serial_receive_interrupt.....	215
Table 3-260. Function hal_i2c_address_listen_interrupt_enable.....	216
Table 3-261. Function hal_i2c_address_listen_interrupt_disable.....	216
Table 3-262. I2C Registers	217
Table 3-263. SMBUS firmware function.....	217
Table 3-264. Enum hal_smbus_run_state_enum	218
Table 3-265. Enum hal_smbus_struct_type_enum.....	218
Table 3-266. Structure hal_smbus_irq_struct.....	218
Table 3-267. smbus_buffer_struct.....	219
Table 3-268. Structure hal_smbus_init_struct.....	219
Table 3-269. Structure hal_smbus_dev_struct	219
Table 3-270. Function hal_smbus_struct_init.....	220
Table 3-271. Function hal_smbus_deinit.....	220
Table 3-272. Function hal_smbus_init.....	221
Table 3-273. Function hal_smbus_slave_receive_interrupt.....	222
Table 3-274. Function hal_smbus_event_irq.....	222
Table 3-275. Function hal_smbus_error_irq.....	223
Table 3-276. Function hal_smbus_start.....	223
Table 3-277. Function hal_smbus_stop	224
Table 3-278. Function hal_smbus_irq_handle_set.....	224
Table 3-279. Function hal_smbus_irq_handle_all_reset.....	225
Table 3-280. Function hal_smbus_master_transmit_interrupt.....	226
Table 3-281. Function hal_smbus_master_receive_interrupt.....	227
Table 3-282. Function hal_smbus_slave_transmit_interrupt.....	227
Table 3-283. Function hal_smbus_slave_receive_interrupt.....	228
Table 3-284. Function hal_smbus_disable_alert_interrupt	229
Table 3-285. NVIC Registers.....	230
Table 3-273. NVIC firmware function.....	230
Table 3-272. IRQn_Type.....	231
Table 3-274. Function hal_nvic_irq_priority_group_set	232
Table 3-275. Function hal_nvic_irq_enable	232
Table 3-290. PMU Registers.....	233
Table 3-291. PMU firmware function	233
Table 3-292. Enum hal_pmu_lvd_voltage_enum.....	234
Table 3-293. Enum hal_pmu_struct_type_enum	234
Table 3-294. Enum hal_pmu_error_enum.....	234
Table 3-295. Enum hal_pmu_state_enum.....	234

Table 3-296. Structure hal_pmu_init_struct.....	235
Table 3-297. Structure hal_pmu_lvd_irq_struct.....	235
Table 3-298. Structure hal_pmu_dev_struct.....	235
Table 3-299. Function hal_pmu_deinit.....	235
Table 3-300. Function hal_pmu_struct_init.....	236
Table 3-301. Function hal_pmu_lvd_init.....	236
Table 3-302. Function hal_pmu_wakeup_pin_enable.....	237
Table 3-303. Function hal_pmu_lvd_start.....	238
Table 3-304. Function hal_pmu_lvd_stop.....	238
Table 3-305. Function hal_pmu_wakeup_pin_enable.....	239
Table 3-306. Function hal_pmu_lvd_irq.....	240
Table 3-307. Function hal_pmu_lvd_irq_handle_set.....	240
Table 3-308. Function hal_pmu_lvd_irq_handle_all_reset.....	241
Table 3-309. Function hal_pmu_lvd_start_interrupt.....	241
Table 3-310. Function hal_pmu_lvd_stop_interrupt.....	242
Table 3-311. RCU Registers	243
Table 3-312. RCU firmware function	244
Table 3-313. Enum hal_rcu_periph_enum	244
Table 3-314. Enum hal_rcu_periph_sleep_enum	245
Table 3-315. Enum hal_rcu_periph_reset_enum.....	245
Table 3-316. Enum hal_rcu_flag_enum.....	246
Table 3-317. Enum hal_rcu_int_flag_enum.....	247
Table 3-318. Enum hal_rcu_int_flag_clear_enum	248
Table 3-319. Enum hal_rcu_int_enum.....	248
Table 3-320. Enum hal_rcu_adc_clksrc_enum	249
Table 3-321. Enum hal_rcu_clock_freq_enum.....	249
Table 3-322. Enum hal_rcu_osc_type_enum	249
Table 3-323. Enum hal_rcu_struct_type_enum.....	250
Table 3-324. Enum hal_rcu_rtc_clksrc_enum.....	250
Table 3-325. Enum hal_rcu_usart_clksrc_enum	250
Table 3-326. Enum hal_rcu_usbfs_clksrc_enum.....	251
Table 3-327. Enum hal_rcu_cec_clksrc_enum.....	251
Table 3-328. Enum hal_rcu_sysclk_src_enum.....	251
Table 3-329. Enum hal_rcu_ck48mclk_src_enum	251
Table 3-330. Enum hal_rcu_sysclk_ahbdiv_enum	252
Table 3-331. Enum hal_rcu_ahbclk_apb1div_enum	252
Table 3-332. Enum hal_rcu_ahbclk_apb2div_enum	253
Table 3-333. Enum hal_rcu_osc_state_enum	253
Table 3-334. Enum hal_rcu_pll_src_enum.....	253
Table 3-335. Enum hal_rcu_pll_prediv_enum.....	253
Table 3-336. Enum hal_rcu_pll_mul_enum.....	254
Table 3-337. Enum hal_rcu_pll_presel_enum	256
Table 3-338. Enum hal_rcu_ckout_src_enum.....	256

Table 3-339. Enum hal_rcu_ckout_div_enum	257
Table 3-340. Structure hal_rcu_periphclk_struct.....	257
Table 3-341. Structure hal_rcu_clk_struct.....	258
Table 3-342. Structure hal_rcu_irq_struct.....	258
Table 3-343. Structure hal_rcu_hxtal_struct.....	258
Table 3-344. Structure hal_rcu_lxtal_struct.....	259
Table 3-345. Structure hal_rcu_irc8m_struct.....	259
Table 3-346. Structure hal_rcu_irc28m_struct.....	259
Table 3-347. Structure hal_rcu_irc48m_struct.....	259
Table 3-348. Structure hal_rcu_irc40k_struct.....	259
Table 3-349. Structure hal_rcu_pll_struct	260
Table 3-350. Structure hal_rcu_osci_struct.....	260
Table 3-351. Function hal_rcu_osci_config.....	260
Table 3-352. Function hal_rcu_clock_out_config	262
Table 3-353. Function hal_rcu_deinit.....	262
Table 3-354. Function hal_rcu_struct_init.....	263
Table 3-355. Function hal_rcu_periph_clk_enable.....	263
Table 3-356. Function hal_rcu_periph_clk_disable	264
Table 3-357. Function hal_rcu_hxtal_clock_monitor_enable.....	264
Table 3-358. Function hal_rcu_hxtal_clock_monitor_disable	265
Table 3-359. Function hal_rcu_periph_reset_enable.....	265
Table 3-360. Function hal_rcu_periph_reset_disable.....	266
Table 3-361. Function hal_rcu_periph_clock_config.....	266
Table 3-362. Function hal_rcu_periph_clkfreq_get	267
Table 3-363. Function hal_rcu_osci_config_get	268
Table 3-364. Function hal_rcu_clock_config	269
Table 3-365. Function hal_rcu_clock_config_get.....	270
Table 3-366. Function hal_SystemCoreClockUpdate	271
Table 3-367. Function hal_rcu_irq.....	271
Table 3-368. Function hal_rcu_irq_handle_set.....	272
Table 3-369. Function hal_rcu_irq_handle_all_reset.....	273
Table 3-370. RTC Registers.....	273
Table 3-371. RTC firmware function.....	274
Table 3-372. Enum hal_rtc_state_enum	275
Table 3-373. Enum hal_rtc_error_enum	275
Table 3-374. Enum hal_rtc_struct_type_enum.....	275
Table 3-375. Structure hal_rtc_irq_struct.....	276
Table 3-376. Structure hal_rtc_dev_struct.....	276
Table 3-377. hal_rtc_timestamp_struct.....	276
Table 3-378. hal_rtc_init_struct.....	277
Table 3-379. hal_rtc_alarm_output_config_struct.....	277
Table 3-380. hal_rtc_alarm_config_struct	277
Table 3-381. hal_rtc_tamper_config_struct.....	278

Table 3-382. Function hal_rtc_init	278
Table 3-383. Function hal_rtc_alarm_output_config	279
Table 3-384. Function hal_rtc_alarm_config	280
Table 3-385. Function hal_rtc_tamp_config	281
Table 3-386. Function hal_rtc_timestamp_config	282
Table 3-387. Function hal_rtc_calib_config	283
Table 3-388. Function hal_rtc_refclock_detection_config	284
Table 3-389. Function hal_rtc_struct_init	285
Table 3-390. Function hal_rtc_deinit	285
Table 3-391. Function hal_rtc_interrupt_enable	286
Table 3-392. Function hal_rtc_interrupt_disable	286
Table 3-393. Function hal_rtc_irq	287
Table 3-394. Function hal_rtc_irq_handle_set	288
Table 3-395. Function hal_rtc_irq_handle_all_reset	288
Table 3-396. Function hal_rtc_current_time_get	289
Table 3-397. Function hal_rtc_alarm_get	289
Table 3-398. Function hal_rtc_alarm_event_poll	290
Table 3-399. Function hal_rtc_timestamp_get	290
Table 3-400. Function hal_rtc_timestamp_event_poll	291
Table 3-401. Function hal_rtc_tamper0_event_poll	291
Table 3-402. Function hal_rtc_tamper1_event_poll	292
Table 3-403. Function hal_rtc_daylight_saving_time_adjust	292
Table 3-404. SPI Registers	294
Table 3-405. DAC firmware function	294
Table 3-406. Enum hal_spi_struct_type_enum	295
Table 3-407. Enum hal_spi_run_state_enum	295
Table 3-408. Enum hal_spi_trans_mode_enum	295
Table 3-409. Enum hal_spi_prescaler_enum	296
Table 3-410. hal_spi_irq_struct	296
Table 3-411. hal_spi_buffer_struct	296
Table 3-412. hal_spi_dev_struct	296
Table 3-413. hal_spi_user_callback_struct	297
Table 3-414. hal_spi_init_struct	297
Table 3-415. Function hal_spi_deinit	297
Table 3-416. Function hal_spi_struct_init	298
Table 3-417. Function hal_spi_init	299
Table 3-418. Function hal_spi_transmit_poll	300
Table 3-419. Function hal_spi_receive_poll	301
Table 3-420. Function hal_spi_transmit_receive_poll	302
Table 3-421. Function hal_spi_transmit_interrupt	303
Table 3-422. Function hal_spi_receive_interrupt	304
Table 3-423. Function hal_spi_transmit_receive_interrupt	305
Table 3-424. Function hal_spi_transmit_dma	306

Table 3-425. Function hal_spi_receive_dma.....	307
Table 3-426. Function hal_spi_transmit_receive_dma.....	308
Table 3-427. Function hal_spi_irq.....	309
Table 3-428. Function hal_spi_irq_handle_set	310
Table 3-429. Function hal_spi_irq_handle_all_reset	311
Table 3-430. Function hal_spi_start.....	311
Table 3-431. Function hal_spi_stop.....	312
Table 3-432. Function hal_spi_abort.....	312
Table 3-433. Function hal_spi_dma_pause	313
Table 3-434. Function hal_spi_dma_resume	313
Table 3-435. Function hal_spi_dma_stop.....	314
Table 3-436. SPI/I2S Registers	315
Table 3-437. I2S firmware function	315
Table 3-438. Enum hal_i2s_struct_type_enum	316
Table 3-439. Enum hal_i2s_run_state_enum	316
Table 3-440. Enum hal_i2s_standard_enum.....	316
Table 3-441. Enum hal_i2s_audiosample_enum	316
Table 3-442. hal_i2s_buffer_struct	317
Table 3-443. hal_i2s_irq_struct.....	317
Table 3-444. hal_i2s_dev_struct.....	317
Table 3-445. hal_i2s_user_callback_struct.....	317
Table 3-446. hal_i2s_init_struct.....	318
Table 3-447. Function hal_i2s_deinit.....	318
Table 3-448. Function hal_i2s_struct_init.....	318
Table 3-449. Function hal_i2s_init.....	319
Table 3-450. Function hal_i2s_transmit_poll	320
Table 3-451. Function hal_i2s_receive_poll.....	321
Table 3-452. Function hal_i2s_transmit_interrupt.....	322
Table 3-453. Function hal_i2s_receive_interrupt.....	323
Table 3-454. Function hal_i2s_transmit_dma.....	324
Table 3-455. Function hal_i2s_receive_dma.....	325
Table 3-456. Function hal_i2s_irq.....	326
Table 3-457. Function hal_i2s_irq_handle_set.....	326
Table 3-458. Function hal_i2s_irq_handle_all_reset	327
Table 3-459. Function hal_i2s_start.....	327
Table 3-460. Function hal_i2s_stop.....	328
Table 3-461. Function hal_i2s_dma_pause.....	328
Table 3-462. Function hal_i2s_dma_resume	329
Table 3-463. Function hal_i2s_dma_stop.....	329
Table 3-464. TIMERx Registers.....	330
Table 3-465. TIMERx firmware function.....	331
Table 3-466. Enum hal_timer_state_enum.....	334
Table 3-467. Enum hal_timer_error_enum.....	334

Table 3-468. Enum hal_timer_service_channel_enum.....	334
Table 3-469. Enum hal_timer_struct_type_enum.....	334
Table 3-470. Enum hal_timer_dma_transfer_start_address_enum.....	335
Table 3-471. Enum hal_timer_dma_transfer_length_enum.....	335
Table 3-472. Enum hal_timer_trgo_selection_enum	336
Table 3-473. Enum hal_timer_output_compare_enum	336
Table 3-474. Enum hal_timer_clock_source_enum.....	337
Table 3-475. Enum hal_timer_slave_mode_enum.....	337
Table 3-476. Enum hal_timer_input_trigger_source_enum.....	337
Table 3-477. Enum hal_timer_decoder_mode_enum.....	338
Table 3-478. Structure hal_timer_decoder_dma_config_struct.....	338
Table 3-479. Structure hal_timer_dma_transfer_config_struct	338
Table 3-480. Structure hal_timer_irq_struct	338
Table 3-481. Structure hal_timer_dma_handle_cb_struct.....	338
Table 3-482. Structure hal_timer_dev_struct.....	339
Table 3-483. Structure hal_timer_init_struct.....	339
Table 3-484. Structure hal_timer_input_capture_struct.....	339
Table 3-485. Structure hal_timer_output_compare_struct.....	340
Table 3-486. Structure hal_timer_break_struct.....	340
Table 3-487. Structure hal_timer_clear_source_struct.....	340
Table 3-488. Structure hal_timer_clock_source_struct	341
Table 3-489. Structure hal_timer_slave_mode_struct	341
Table 3-490. Structure hal_timer_decoder_struct.....	341
Table 3-491. Structure hal_timer_hall_sensor_struct.....	341
Table 3-492. Structure hal_timer_single_pulse_struct	342
Table 3-493. Function hal_timer_init.....	342
Table 3-494. Function hal_timer_input_capture_config.....	343
Table 3-495. Function hal_timer_output_compare_config.....	344
Table 3-496. Function hal_timer_break_config.....	345
Table 3-497. Function hal_timer_ocpre_clear_source_config.....	346
Table 3-498. Function hal_timer_ci0_input_select.....	347
Table 3-499. Function hal_timer_single_pulse_mode_config	348
Table 3-500. Function hal_timer_clock_source_config	348
Table 3-501. Function hal_timer_slave_mode_config	349
Table 3-502. Function hal_timer_decoder_config.....	350
Table 3-503. Function hal_timer_hall_sensor_config.....	351
Table 3-504. Function hal_timer_struct_init	352
Table 3-505. Function hal_timer_deinit.....	353
Table 3-506. Function hal_timer_counter_start	353
Table 3-507. Function hal_timer_counter_stop.....	354
Table 3-508. Function hal_timer_counter_start_interrupt.....	354
Table 3-509. Function hal_timer_counter_stop_interrupt.....	355
Table 3-510. Function hal_timer_counter_start_dma	356

Table 3-511. Function hal_timer_counter_stop_dma.....	357
Table 3-512. Function hal_timer_input_capture_start.....	358
Table 3-513. Function hal_timer_input_capture_stop.....	359
Table 3-514. Function hal_timer_input_capture_start_interrupt.....	359
Table 3-515. Function hal_timer_input_capture_stop_interrupt.....	360
Table 3-516. Function hal_timer_input_capture_start_dma.....	361
Table 3-517. Function hal_timer_input_capture_stop_dma.....	363
Table 3-518. Function hal_timer_output_compare_start.....	363
Table 3-519. Function hal_timer_output_compare_stop.....	364
Table 3-520. Function hal_timer_output_compare_start_interrupt.....	365
Table 3-521. Function hal_timer_output_compare_stop_interrupt.....	366
Table 3-522. Function hal_timer_output_compare_start_dma.....	367
Table 3-523. Function hal_timer_output_compare_stop_dma.....	368
Table 3-524. Function hal_timer_output_compare_complementary_channel_start.....	369
Table 3-525. Function hal_timer_output_compare_complementary_channel_stop.....	370
Table 3-526. Function hal_timer_output_compare_complementary_channel_start_interrupt.....	371
Table 3-527. Function hal_timer_output_compare_complementary_channel_stop_interrupt.....	372
Table 3-528. Function hal_timer_output_compare_complementary_channel_start_dma.....	373
Table 3-529. Function hal_timer_output_compare_complementary_channel_stop_dma.....	374
Table 3-530. Function hal_timer_single_pulse_mode_channel_config.....	375
Table 3-531. Function hal_timer_single_pulse_start.....	376
Table 3-532. Function hal_timer_single_pulse_stop.....	377
Table 3-533. Function hal_timer_single_pulse_start_interrupt.....	378
Table 3-534. Function hal_timer_single_pulse_stop_interrupt.....	379
Table 3-535. Function hal_timer_single_pulse_complementary_channel_start.....	379
Table 3-536. Function hal_timer_single_pulse_complementary_channel_stop.....	380
Table 3-537. Function hal_timer_single_pulse_complementary_channel_start_interrupt.....	381
Table 3-538. Function hal_timer_single_pulse_complementary_channel_stop_interrupt.....	382
Table 3-539. Function hal_timer_slave_mode_interrupt_config.....	383
Table 3-540. Function hal_timer_decoder_start.....	384
Table 3-541. Function hal_timer_decoder_stop.....	385
Table 3-542. Function hal_timer_decoder_start_interrupt.....	386
Table 3-543. Function hal_timer_decoder_stop_interrupt.....	387
Table 3-544. Function hal_timer_decoder_start_dma.....	388
Table 3-545. Function hal_timer_decoder_stop_dma.....	389
Table 3-546. Function hal_timer_hall_sensor_start.....	390
Table 3-547. Function hal_timer_hall_sensor_stop.....	391
Table 3-548. Function hal_timer_hall_sensor_start_interrupt.....	391
Table 3-549. Function hal_timer_hall_sensor_stop_interrupt.....	392
Table 3-550. Function hal_timer_hall_sensor_start_dma.....	393
Table 3-551. Function hal_timer_hall_sensor_stop_dma.....	394
Table 3-552. Function hal_timer_dma_transfer_write_start.....	395
Table 3-553. Function hal_timer_dma_transfer_write_stop.....	396

Table 3-554. Function hal_timer_dma_transfer_read_start.....	397
Table 3-555. Function hal_timer_dma_transfer_read_stop.....	399
Table 3-556. Function hal_timer_commutation_event_config.....	400
Table 3-557. Function hal_timer_commutation_event_interrupt_config.....	401
Table 3-558. Function hal_timer_commutation_event_dma_config.....	402
Table 3-559. Function hal_timer_irq_handle_set.....	404
Table 3-560. Function hal_timer_irq_handle_all_reset.....	405
Table 3-561. Function hal_timer_irq.....	405
Table 3-562. TSI Registers.....	406
Table 3-563. TSI firmware function.....	407
Table 3-564. Enum hal_tsi_state_enum.....	407
Table 3-565. Enum hal_tsi_error_enum.....	407
Table 3-566. Enum hal_tsi_struct_type_enum.....	408
Table 3-567. Structure hal_tsi_irq_struct.....	408
Table 3-568. Structure hal_tsi_dev_struct.....	408
Table 3-569. hal_tsi_init_struct.....	408
Table 3-570. Function hal_tsi_init.....	409
Table 3-571. Function hal_tsi_struct_init.....	410
Table 3-572. Function hal_tsi_deinit.....	411
Table 3-573. Function hal_tsi_start.....	411
Table 3-574. Function hal_tsi_stop.....	412
Table 3-575. Function hal_tsi_start_interrupt.....	412
Table 3-576. Function hal_tsi_stop_interrupt.....	413
Table 3-577. Function hal_tsi_irq.....	413
Table 3-578. Function hal_tsi_irq_handle_set.....	414
Table 3-579. Function hal_tsi_irq_handle_all_reset.....	415
Table 3-580. Function hal_tsi_group_cycle_get.....	415
Table 3-581. Function hal_tsi_pins_config.....	416
Table 3-582. Function hal_tsi_poll_transfer.....	417
Table 3-583. USART Registers.....	418
Table 3-584. UART firmware function.....	418
Table 3-585. Enum hal_uart_struct_type_enum.....	419
Table 3-586. Enum hal_uart_work_mode_enum.....	419
Table 3-587. Structure hal_uart_init_struct.....	420
Table 3-588. Structure hal_uart_irq_struct.....	421
Table 3-589. Enum hal_uart_state_enum.....	421
Table 3-590. Structure hal_uart_dev_struct.....	421
Table 3-591. Structure hal_uart_user_callback_struct.....	422
Table 3-592. Function hal_uart_struct_init.....	422
Table 3-593. Function hal_uart_deinit.....	423
Table 3-594. Function hal_uart_init.....	424
Table 3-595. Function hal_uart_irq.....	425
Table 3-596. Function hal_uart_irq_handle_set.....	426

Table 3-597. Function hal_uart_irq_handle_all_reset.....	427
Table 3-598. Function hal_uart_transmit_poll.....	427
Table 3-599. Function hal_uart_receive_poll.....	428
Table 3-600. Function hal_uart_transmit_interrupt.....	429
Table 3-601. Function hal_uart_receive_interrupt.....	431
Table 3-602. Function hal_uart_transmit_dma.....	432
Table 3-603. Function hal_uart_receive_dma.....	433
Table 3-604. Function hal_uart_dma_pause.....	435
Table 3-605. Function hal_uart_dma_resume.....	435
Table 3-606. Function hal_uart_transmit_stop.....	436
Table 3-607. Function hal_uart_receive_stop.....	437
Table 3-608. USART Registers.....	437
Table 3-609. USRT firmware function.....	438
Table 3-610. Enum hal_usrt_struct_type_enum.....	439
Table 3-611. Structure hal_usrt_init_struct.....	439
Table 3-612. Structure hal_usrt_irq_struct.....	440
Table 3-613. Enum hal_usrt_state_enum.....	440
Table 3-614. Structure hal_usrt_dev_struct.....	440
Table 3-615. Structure hal_usrt_user_callback_struct.....	441
Table 3-616. Function hal_usrt_struct_init.....	441
Table 3-617. Function hal_usrt_deinit.....	442
Table 3-618. Function hal_usrt_init.....	443
Table 3-619. Function hal_usrt_irq.....	444
Table 3-620. Function hal_usrt_irq_handle_set.....	445
Table 3-621. Function hal_usrt_irq_handle_all_reset.....	445
Table 3-622. Function hal_usrt_transmit_poll.....	446
Table 3-623. Function hal_usrt_receive_poll.....	447
Table 3-624. Function hal_usrt_transmit_receive_poll.....	448
Table 3-625. Function hal_usrt_transmit_interrupt.....	449
Table 3-626. Function hal_usrt_receive_interrupt.....	450
Table 3-627. Function hal_usrt_transmit_receive_interrupt.....	452
Table 3-628. Function hal_usrt_transmit_dma.....	453
Table 3-629. Function hal_usrt_receive_dma.....	455
Table 3-630. Function hal_usrt_transmit_receive_dma.....	456
Table 3-631. Function hal_usrt_dma_pause.....	457
Table 3-632. Function hal_usrt_dma_resume.....	458
Table 3-633. Function hal_usrt_transfer_stop.....	459
Table 3-634. USART Registers.....	460
Table 3-635. IRDA firmware function.....	460
Table 3-636. Enum hal_irda_struct_type_enum.....	461
Table 3-637. Structure hal_irda_init_struct.....	461
Table 3-638. Structure hal_irda_irq_struct.....	462
Table 3-639. Enum hal_irda_state_enum.....	462

Table 3-640. Structure hal_irda_dev_struct.....	462
Table 3-641. Structure hal_irda_user_callback_struct.....	463
Table 3-642. Function hal_irda_struct_init.....	463
Table 3-643. Function hal_irda_deinit.....	464
Table 3-644. Function hal_irda_init.....	465
Table 3-645. Function hal_irda_irq.....	466
Table 3-646. Function hal_irda_irq_handle_set.....	466
Table 3-647. Function hal_irda_irq_handle_all_reset.....	467
Table 3-648. Function hal_irda_transmit_poll.....	468
Table 3-649. Function hal_irda_receive_poll.....	469
Table 3-650. Function hal_irda_transmit_interrupt.....	470
Table 3-651. Function hal_irda_receive_interrupt.....	471
Table 3-652. Function hal_irda_transmit_dma.....	472
Table 3-653. Function hal_irda_receive_dma.....	474
Table 3-654. Function hal_irda_dma_pause.....	475
Table 3-655. Function hal_irda_dma_resume.....	476
Table 3-656. Function hal_irda_transmit_stop.....	477
Table 3-657. Function hal_irda_receive_stop.....	477
Table 3-658. USART Registers.....	478
Table 3-659. SMARTCARD firmware function.....	479
Table 3-660. Enum hal_smartcard_struct_type_enum.....	479
Table 3-661. Structure hal_smartcard_init_struct.....	479
Table 3-662. Structure hal_smartcard_irq_struct.....	481
Table 3-663. Enum hal_smartcard_state_enum.....	481
Table 3-664. Structure hal_smartcard_dev_struct.....	481
Table 3-665. Structure hal_smartcard_user_callback_struct.....	482
Table 3-666. Function hal_smartcard_struct_init.....	482
Table 3-667. Function hal_smartcard_deinit.....	483
Table 3-668. Function hal_smartcard_init.....	483
Table 3-669. Function hal_smartcard_irq.....	485
Table 3-670. Function hal_smartcard_irq_handle_set.....	486
Table 3-671. Function hal_smartcard_irq_handle_all_reset.....	487
Table 3-672. Function hal_smartcard_transmit_poll.....	487
Table 3-673. Function hal_smartcard_receive_poll.....	488
Table 3-674. Function hal_smartcard_transmit_interrupt.....	489
Table 3-675. Function hal_smartcard_receive_interrupt.....	491
Table 3-676. Function hal_smartcard_transmit_dma.....	492
Table 3-677. Function hal_smartcard_receive_dma.....	493
Table 3-678. Function hal_smartcard_transmit_stop.....	495
Table 3-679. Function hal_smartcard_receive_stop.....	495
Table 3-680. WWDGT Registers.....	496
Table 3-666. WWDGT firmware function.....	496
3-667. hal_wwdgt_state_enum.....	497

Table 3-668. hal_wwdgt_struct_type_enum	497
Table 3-669. hal_wwdgt_dev_struct.....	497
Table 3-670. hal_wwdgt_init_struct	498
Table 3-671. hal_wwdgt_irq_struct.....	498
Table 3-672. Function hal_wwdgt_struct_init.....	498
Table 3-673. Function hal_wwdgt_init.....	499
Table 3-674. Function hal_wwdgt_deinit.....	500
Table 3-675. Function hal_wwdgt_irq	500
Table 3-676. Function hal_wwdgt_irq_handle_set	501
Table 3-677. Function hal_wwdgt_irq_handle_all_reset.....	502
Table 3-677. Function hal_wwdgt_start.....	503
Table 3-677. Function hal_wwdgt_start_interrupt.....	504
Table 3-677. Function hal_wwdgt_reload.....	505
Table 4-1. Revision history	508

1. Introduction

This manual introduces HAL firmware library of GD32F3X0 devices which are 32-bit microcontrollers based on the ARM processor.

The HAL firmware library is a firmware function package, including program data structure and macro definitions, all the performance features of peripherals of GD32F3X0 devices are involved in the package. Each peripheral driving code and HAL firmware library examples on evaluation board are also included in HAL firmware library. Users need not learn each peripherals in details and it's easy to apply a peripheral by using the HAL 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.

The commonly used HAL 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 HAL firmware library user manual is shown as below:

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

1.1. Rules of User Manual and HAL Firmware Library

1.1.1. Peripherals

Table 1-1. Peripherals

Peripherals	Descriptions
ADC	Analog-to-digital converter
CRC	CRC calculation unit
CEC	HDMI-CEC controller
CMP	Comparator
CTC	Clock trim controller
DAC	Digital-to-analog converter

Peripherals	Descriptions
DBG	Debug
DMA	Direct memory access controller
EXTI	Interrupt/event controller
FMC	Flash memory controller
FWDGT	Free watchdog timer
GPIO	General-purpose and alternate-function I/Os
I2C	Inter-integrated circuit interface
NVIC	Nested Vectored Interrupt Controller
PMU	Power management unit
RCU	Reset and clock unit
RTC	Real-time Clock
SPI/I2S	Serial peripheral interface/Inter-IC sound
TIMER	TIMER
TSI	Touch sensing interface
USART	Universal synchronous/asynchronous receiver /transmitter
WWDGT	Window watchdog timer

1.1.2. Naming rules

The HAL 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 “gd32f3x0_hal_”, such as: gd32f3x0_hal_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 uppcase of English letters;
- Registers are handled as constants. The naming of them are written in uppcase of English letters. In most cases, register names are shortened accord with the user manual;
- Variables are written in lowercase, when concluded by several words, underlines should be adapted among words;
- The naming of peripheral functions are started with the peripheral abbreviation added with an underline, when the function name is concluded by several words, underlines should be adapted among words, and all the peripheral functions are written in lowercase.

2. HAL Firmware Library Overview

2.1. Framework of HAL Firmware Library

The realize of HAL firmware library .c file is corresponding to functions and registers descriptions, the HAL firmware library implementation framework is shown as below.

Figure 2-1. HAL firmware library implementation framework of GD32F3X0



According to functions and characteristics, HAL library can be divided into five classes, and they are General, Internal, Analog, I/O device and Ticker.

General class is the fundamental peripherals functions, and would be used to implement any application code. Peripherals included RCU, GPIO, EXTI, DMA and NVIC.

Internal class is the peripherals functions which do not involve external pin. Peripherals included CRC, FMC, WWDGT, FWDGT, PMU and SYSCFG, CTC.

Analog class is the peripherals functions that are related to analog. Peripherals included ADC and CMP, DAC, TSI.

Ticker class is the peripherals functions that are related to counter and timer. Peripherals included RTC, TIMER and SYSTICK.

I/O device is the peripherals functions that are related to communication. Peripherals included I2C, SPI and USART, CEC.

2.2. Features of HAL Firmware Library

The HAL library code resources can cooperate with the host computer software, so to realize the function of configuration code automatic generation. So, The HAL library code style is standardized and abstract.

■ Unified initialization function

The initialization functions of each peripheral function are highly centralized, and if many parameters need to be configured, structure assignment shall be used. But it must be noted: To use a function in the HAL library, call the initialization function of the corresponding module in the HAL library first.

■ Standardized initialization function

The initialization functions of each peripheral function are highly concentrated. When there are many parameters to be configured, the structure value assignment is used in a unified manner.

■ Interrupt callback mechanism

For modules with interrupt mechanism, HAL library uses interrupt callback mechanism to manage interrupt.

Usually, when users use interrupts, they need to add specific execution code in the interrupt handler, and handle the judging and clearing of interrupt flag bits.

When using HAL library, user only need to place the interrupt processing function of the library at the interrupt entrance. This function will correctly handles the corresponding interrupt flag bit and clean the flag bit before exiting the interrupt. User just need to input callback function pointer, when interrupt is arrived, the interrupt handler of HAL library will call the pointer to execute user's code.

■ Standardized delay management mechanism

In Hal library, `hal_basetick.c` implements a standardized delay management mechanism. User can use the delay function to achieve precise delay. Because we use this delay management to realize the function of timeout mechanism, so, before use HAL library code, user must call `hal_basetick_init()` function to initialize delay management.

■ Standardized abstracted input/output functions

For I/O device class, HAL library realizes standardized abstracted input/output functions. In user layer, the method in function calls under different communication modes is standardized, thus, it is convenient for users to quickly understand and use. Each communication peripheral (including USART/SPI/I2C, etc.) provides the transmit/receive functions in three modes: polling, interrupt and DMA.

- Standardized structure initialization function

In the HAL library, each module provides a standardized structure initialization function `hal_xxx_struct_init`. Before call the function `hal_xxx_init`, user could call the function interface to thoroughly complete the assignment of the initial value of the structure.

2.3. Introduction to the code framework of the HAL firmware library

The original intention of HAL library code development is to cooperate with the IDE host computer to implement the "automatic code generation" function, and to provide users with more convenient and highly integrated functional functions. Therefore, it is necessary to ensure the uniformity and regularity of various peripheral module functions called by the IDE host computer software, and the peripheral modules of the same type with different features must have similar initialization and functional function implementation.

The whole HAL library code includes three types of functions, namely: 1. Initialization function. 2. Configuration function. 3. Functional function.

- Initialization function

This function will be called by the "automatic code generation" function of the IDE host computer software, and it is generally used to initialize the peripheral module functions. The IDE host computer automatically generates a set of corresponding init and deinit functions. Therefore, it has the following characteristics: 1. The function name ends with `_init()`. 2. The corresponding `_deinit()` function is implemented. 3. Try to pass the parameters in the form of structure pointers.

- Configuration function

This function will be called by the "automatic code generation" function of the IDE host computer software, and it is generally used to modify the configuration of a specific function of the peripheral module.

- Functional function

The specific functions of the peripherals are abstracted and integrated into this highly integrated functional function, which needs to be called by users themselves. The following function implementations are provided for peripheral modules involving interrupts and DMA, the implementations are reduced accordingly on this basis for peripheral modules not involving interrupts and DMA:

1. Implement the unified interrupt handler used in the interrupt service function. Judge all the interrupt flag bits of the peripheral modules, and jump to different callback functions for interrupt handling according to the actual situation.

2. Implement the registration (`hal_xxx_irq_handle_set`) and unbinding

(hal_xxx_irq_handle_all_reset) of the interrupt callback function.

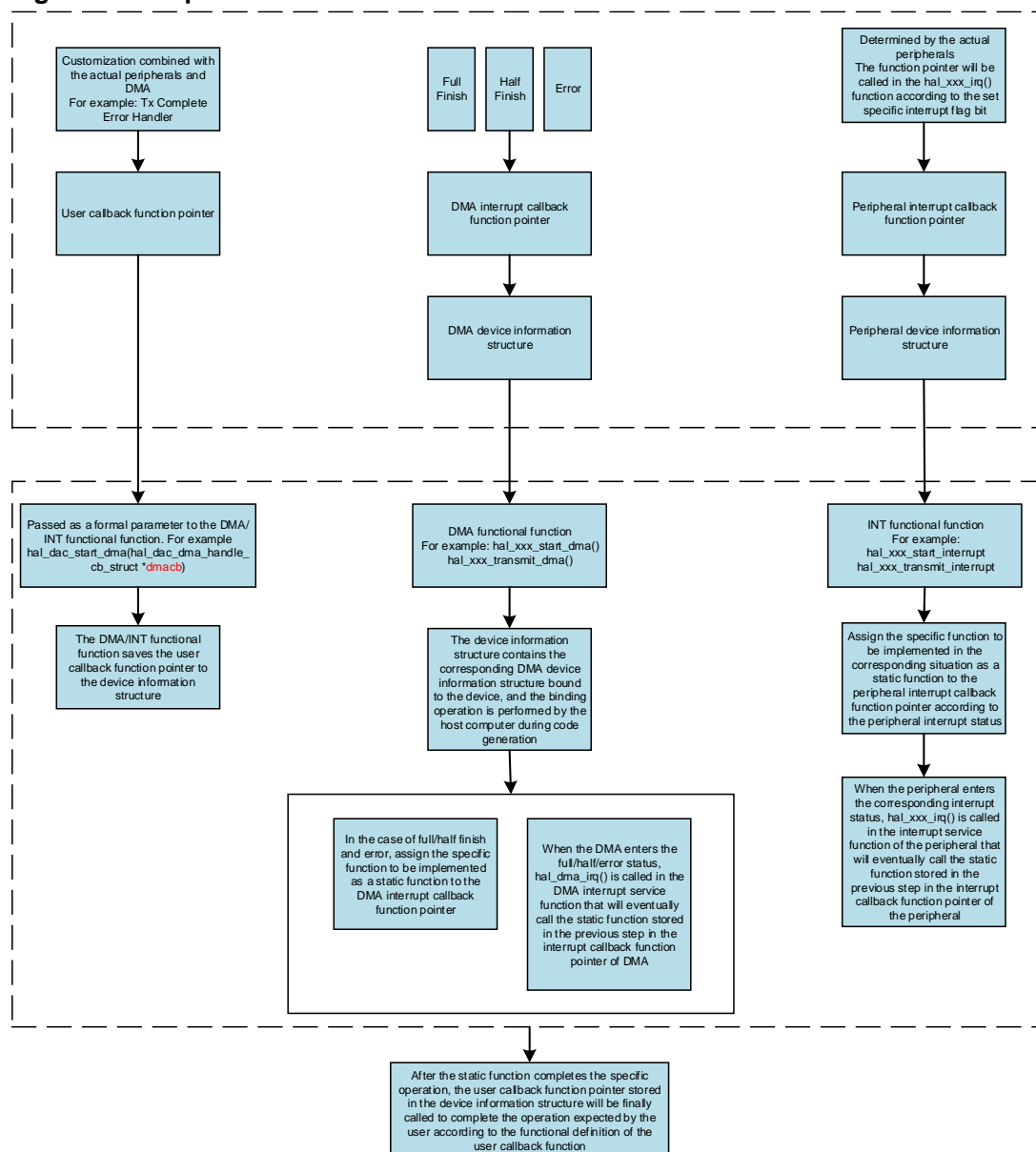
3. Implement the sending and receiving in POLL/DMA/INT mode, or the start and end. At the same time, ensure the reenterability of such functions.

Functional function implementation characteristics

The implementation of DMA/INT functional functions is combined with the interrupt. The peripheral status is guided by the interrupt flag bit and the complex function logic is completed. The formal parameters of the functional functions contain the callback function pointer (if the callback function interfaces in multiple states are provided to users, a structure containing all the callback function interfaces will be defined, and the structure pointer is used to pass the formal parameters), which is provided to users as an interface to guide them to perform desired operations in various functional states.

The callback function is a core implementation method in the implementation of functional functions, and its general design and calling principles are as follows:

Figure 2-2. Implementation of callback function



To implement the above functions, several special structures will be commonly defined in the HAL library for the implementation of the above characteristics:

[Special structure]

Except for some modules of the "general class", the modules maintain several types of special structures. Take USART as an example:

"Device information structure", whose unified naming method is `hal_xxx_dev_struct`, such as `hal_uart_dev_struct` for the UART module

This structure is used to save some important information, which can be easily called by each function. Therefore, the device information structure is defined in conjunction with the specific function implementation of the specific peripheral. It is used to save the base address of the

peripheral, the interrupt callback function (if there is an interrupt), the associated DMA device information structure (if there is a DMA function), etc.

Because the "device information structure" is a global variable and is passed as the first formal parameter of most peripheral functions, the "functional function" uses the "device information structure" to pass and save parameters during the implementation.

In principle, the maintenance of the "device information structure" is completed by the HAL library function, and users only need to create a structure and pass it in to the function as a parameter, without directly assigning a value to the structure. The HAL library function assigns the formal parameters passed in to the corresponding members of the structure.

"Device interrupt callback function structure", whose unified naming method is `hal_xxx_irq_struct`, such as `hal_dma_irq_struct` for the DMA module

Figure 2-3. Device interrupt callback function structure of DMA

```
typedef struct {  
    __IO hal_irq_handle_cb error_handle; .....  
    __IO hal_irq_handle_cb half_finish_handle; .....  
    __IO hal_irq_handle_cb full_finish_handle; .....  
} hal_dma_irq_struct;
```

As a member of the "device information structure", this structure is used to store the pointers of the callback function that users expect to call when the expected interrupt occurs.

"Device interrupt callback function structure" will be used in the unique interrupt service function in the code of each peripheral module, such as `hal_dma_irq()` for DMA.

Figure 2-4. hal_dma_irq() for DMA

```
int32_t hal_dma_irq(hal_dma_dev_struct *dma_dev)
{
    #if (1 == HAL_PARAMETER_CHECK)
        /* check the DMA pointer address and the number length parameter */
        if (NULL == dma_dev) {
            HAL_DEBUGGE("parameter [dma_dev] value is invalid");
            return HAL_ERR_ADDRESS;
        }
    #endif /* 1 == HAL_PARAMETER_CHECK */
    /* full transfer finish interrupt handler */
    if (SET == hals_dma_interrupt_flag_get(dma_dev->channel, DMA_INTF_FTFIF)) {
        hals_dma_interrupt_flag_clear(dma_dev->channel, DMA_INTF_FTFIF);
        if (dma_dev->dma_irq.full_finish_handle != NULL) {
            dma_dev->dma_irq.full_finish_handle(dma_dev);
            /* unlock DMA */
            dma_dev->state = HAL_DMA_STATE_READY;
            HAL_UNLOCK(dma_dev);
        }

        /* half transfer finish interrupt handler */
        if (SET == hals_dma_interrupt_flag_get(dma_dev->channel, DMA_INTF_HTFIF)) {
            /* if DMA not in circular mode */
            if (0 == (DMA_CHCTL(dma_dev->channel) & DMA_CHXCTL_CMEN)) {
                hals_dma_interrupt_disable(dma_dev->channel, DMA_INT_HTF);
            }
            hals_dma_interrupt_flag_clear(dma_dev->channel, DMA_INTF_HTFIF);
            if (dma_dev->dma_irq.half_finish_handle != NULL) {
                dma_dev->dma_irq.half_finish_handle(dma_dev);
            }
        }

        /* error interrupt handler */
        if (SET == hals_dma_interrupt_flag_get(dma_dev->channel, DMA_INTF_ERRIF)) {
            hals_dma_interrupt_flag_clear(dma_dev->channel, DMA_INTF_GIF);
            dma_dev->state = HAL_DMA_STATE_READY;
            dma_dev->error_state = HAL_DMA_ERROR_TRANSFER;
            /* unlock DMA */
            HAL_UNLOCK(dma_dev);
            if (dma_dev->dma_irq.error_handle != NULL) {
                dma_dev->dma_irq.error_handle(dma_dev);
            }
        }

        return HAL_ERR_NONE;
    }
}
```

The `hal_xxx_irq()` function judges all the interrupt flag bits of the peripheral module and clears the interrupts, so that users do not need to concern how to handle various interrupts, and only need to pass the function to be called when the corresponding interrupt occurs in to the "device interrupt callback function structure" in the form of a callback function pointer. When the interrupt occurs, the function judges whether the callback function pointer has been assigned, and performs a callback.

In particular, if users do not need to use the POLL/DMA/INT function, and want to specify a callback function, they can assign a value to this member in the "device information structure" by calling the `xxx_xx_irq_handle_set` function of each module. Take the ADC module as an

example, as shown below.

Figure 2-5. adc_irq_handle_set function

```
void hal_adc_irq_handle_set(hal_adc_dev_struct *adc_dev, hal_adc_irq_struct *p_irq)
{
    /* EOC interrupt handler set */
    if(NULL != p_irq->adc_eoc_handle) {
        adc_dev->adc_irq.adc_eoc_handle = p_irq->adc_eoc_handle;
        hals_adc_interrupt_enable(ADC_INT_EOC);
    } else {
        adc_dev->adc_irq.adc_eoc_handle = NULL;
        hals_adc_interrupt_disable(ADC_INT_EOC);
    }

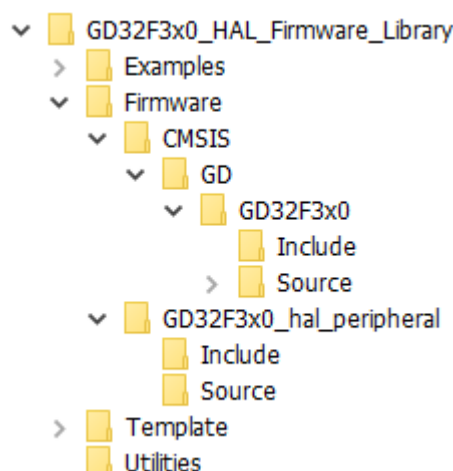
    /* EOIC interrupt handler set */
    if(NULL != p_irq->adc_eoic_handle) {
        adc_dev->adc_irq.adc_eoic_handle = p_irq->adc_eoic_handle;
        hals_adc_interrupt_enable(ADC_INT_EOIC);
    } else {
        adc_dev->adc_irq.adc_eoic_handle = NULL;
        hals_adc_interrupt_disable(ADC_INT_EOIC);
    }

    /* watchdog interrupt handler set */
    if(NULL != p_irq->adc_watchdog_handle) {
        adc_dev->adc_irq.adc_watchdog_handle = p_irq->adc_watchdog_handle;
        hals_adc_interrupt_enable(ADC_INT_WDE);
    } else {
        adc_dev->adc_irq.adc_watchdog_handle = NULL;
        hals_adc_interrupt_disable(ADC_INT_WDE);
    }
}
```

2.4. File Structure of Firmware Library

GD32F3X0_HAL_Firmware_Library, the file structure is shown as below.

Figure 2-6. File structure of GD32F3X0_HAL_firmware library



2.4.1. 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 inc and src two folders and a readme file a gdc file.

readme.txt: the description and using guide of the example;

xxx.gdc: configuration information about Embedded Builder software;

The contents of the inc folder:

- gd32f3x0_libopt.h: The header file consists of different "#include" statements (by default, no modification is required, all peripherals are turned on), and the header file can set the peripherals used by the routine and the special functions of the HAL library (internal flash write and erase working mode, debugging information is printed, error is blocked, etc.);
- gd32f3x0_it.h: the header file include all the prototypes of the interrupt service routines;
- gd32f3x0_hal_init.c: the source file include all the initialization and deinitialization of peripherals;

The contents of the src folder:

- gd32f3x0_it.c: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- gd32f3x0_hal_init.c: the source file include all the initialization and deinitialization of peripherals;
- main.c: example code. Note: all the examples are not influenced by software IDEs.

2.4.2. Firmware Folder

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

- CMSIS subfolder includes the Cortex M4 kernel support files, the startup file based on the Cortex M4 kernel processor, the global header file of GD32F3X0 and system configuration file;
- GD32F3x0_hal_peripheral subfolder:
 - Include subfolder includes all the header files of HAL firmware library, users need not modify this folder;
 - Source subfolder includes all the source files of HAL firmware library, users need not modify this folder;

2.4.3. Utilities Folder

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

- gd32f3x0r_eval.h is related header files of the evaluation board about running the firmware examples;
- gd32f3x0r_eval.c is related source files of the evaluation board about running the firmware examples.

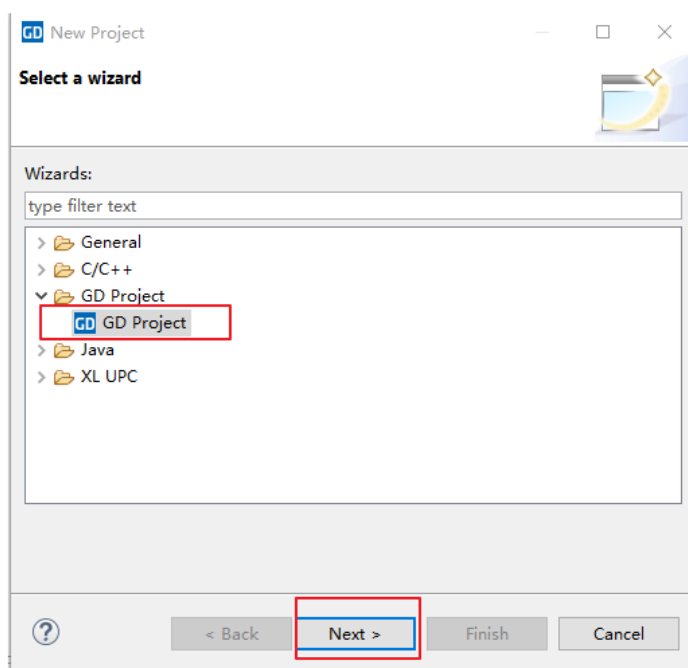
2.5. Code Migration Instructions

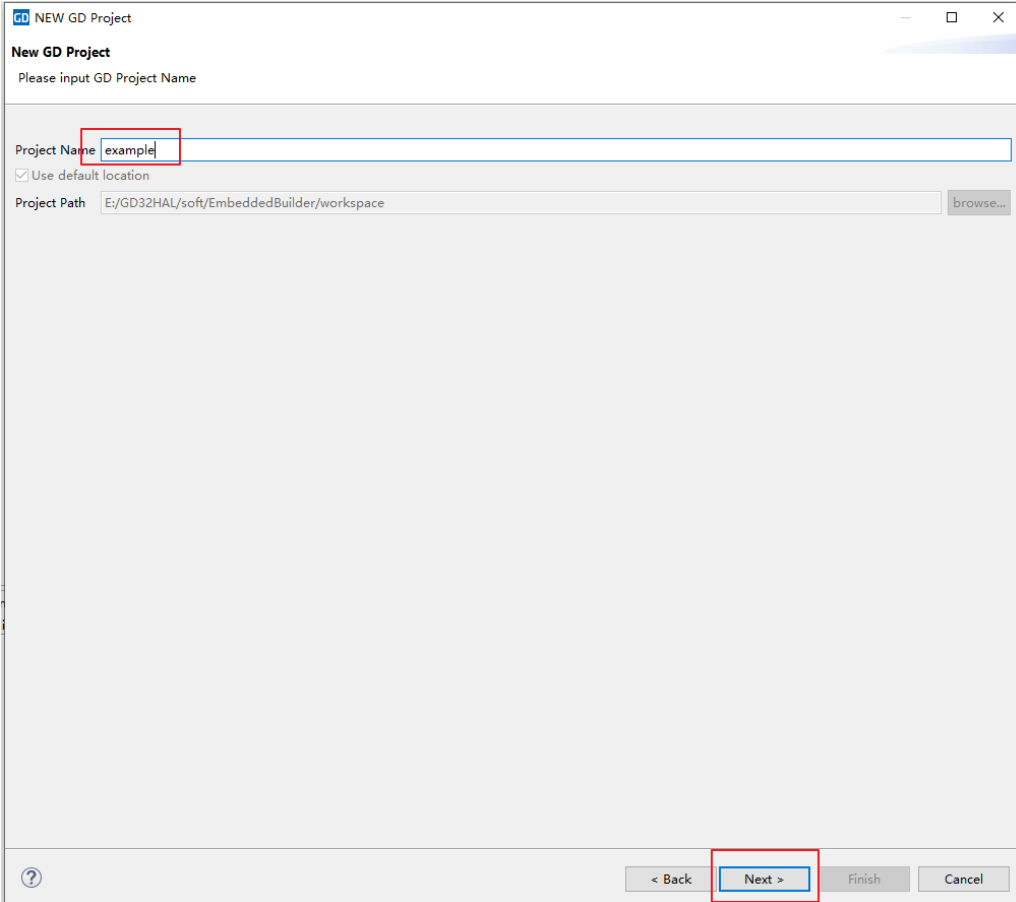
The purpose of code Migration is to guide users to transplant the the example in Examples folder to EmbeddedBuilder software, and can be used correctly.

2.5.1. New project

Open Embedded Builder software and new project, as shown in [Figure 2-7. New project interface](#).

Figure 2-7. New project interface





NEW GD Project

New GD Project

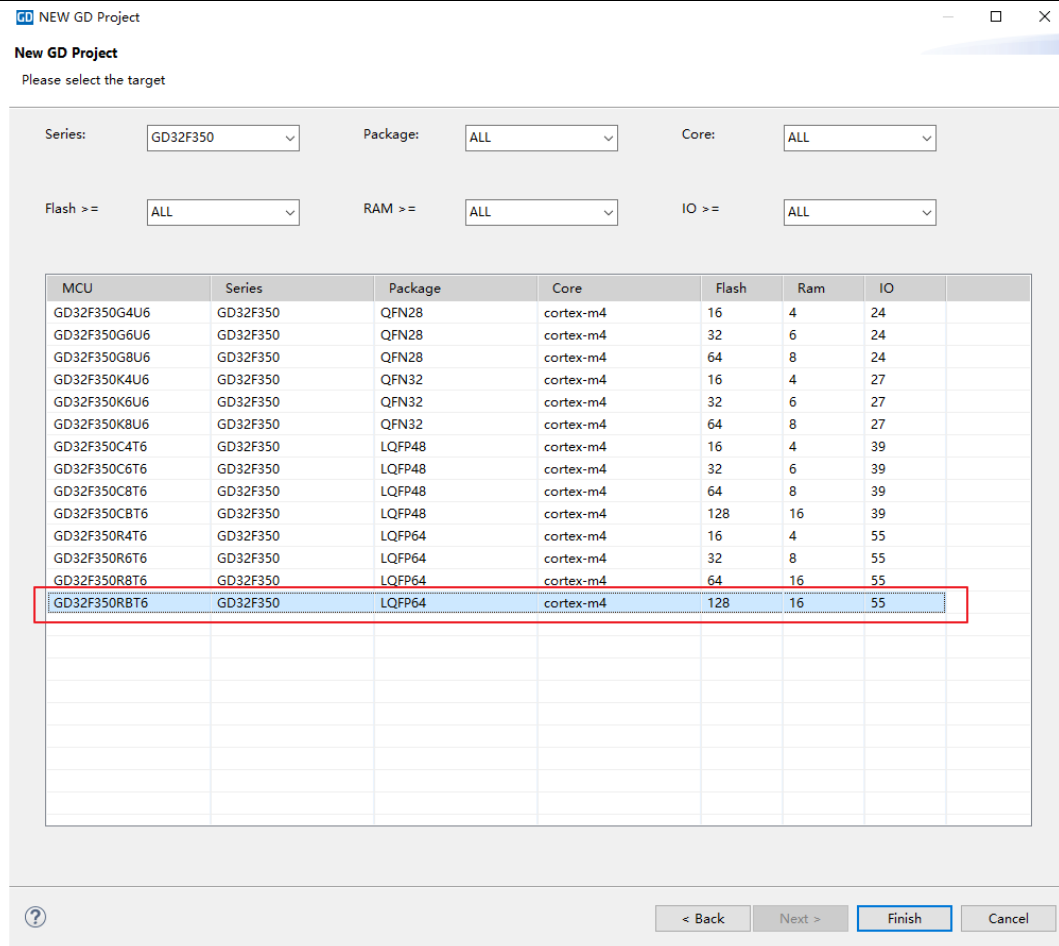
Please input GD Project Name

Project Name

☒ Use default location

Project Path [browse...](#)

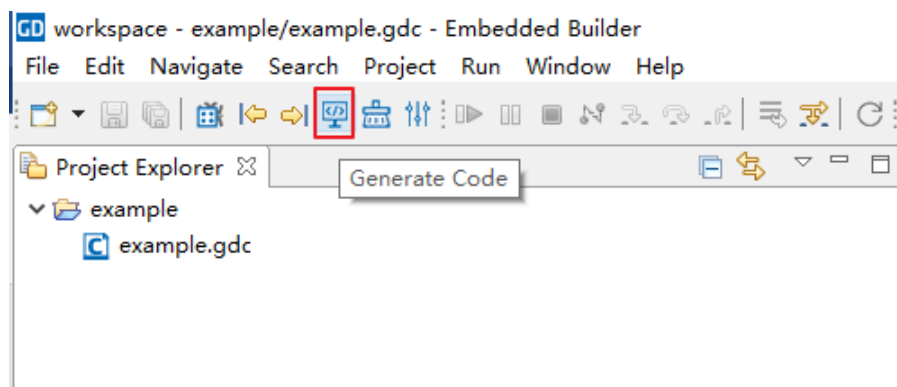
[?<](#) [Next >](#) [Finish](#) [Cancel](#)

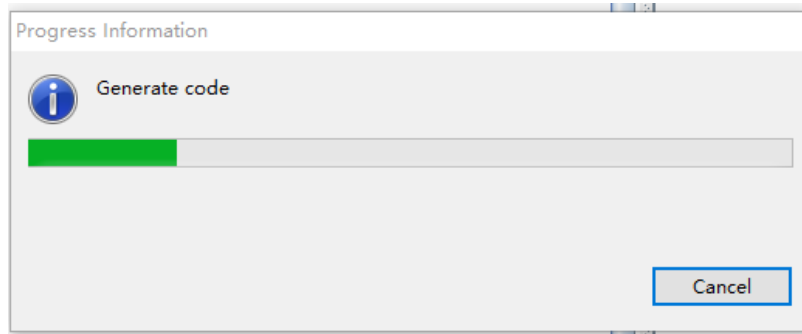


2.5.2. Generate Code

The interface is shown in [Figure 2-8. Generate code interface](#) after completing the new project.

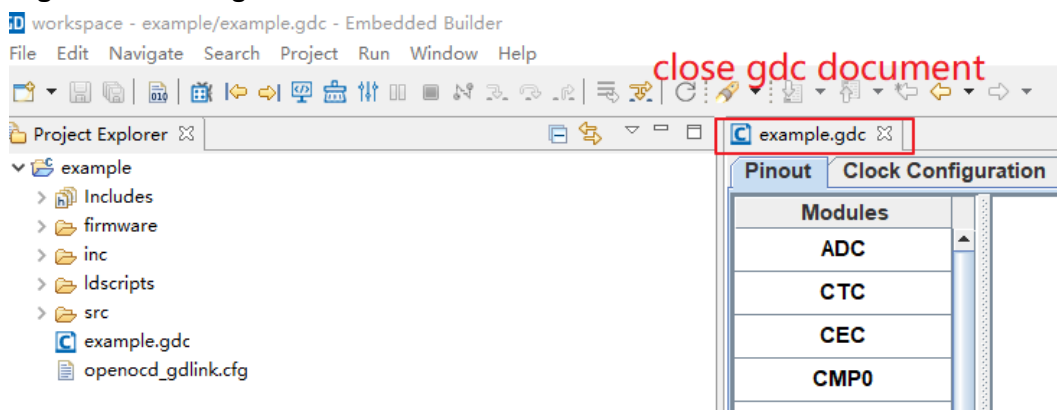
Figure 2-8. Generate code interface





The interface is shown in [Figure 2-9. After generate code interface](#) after code generation, Close the corresponding gdc file at this time (pay special attention to this, otherwise the replacement project cannot be used normally).

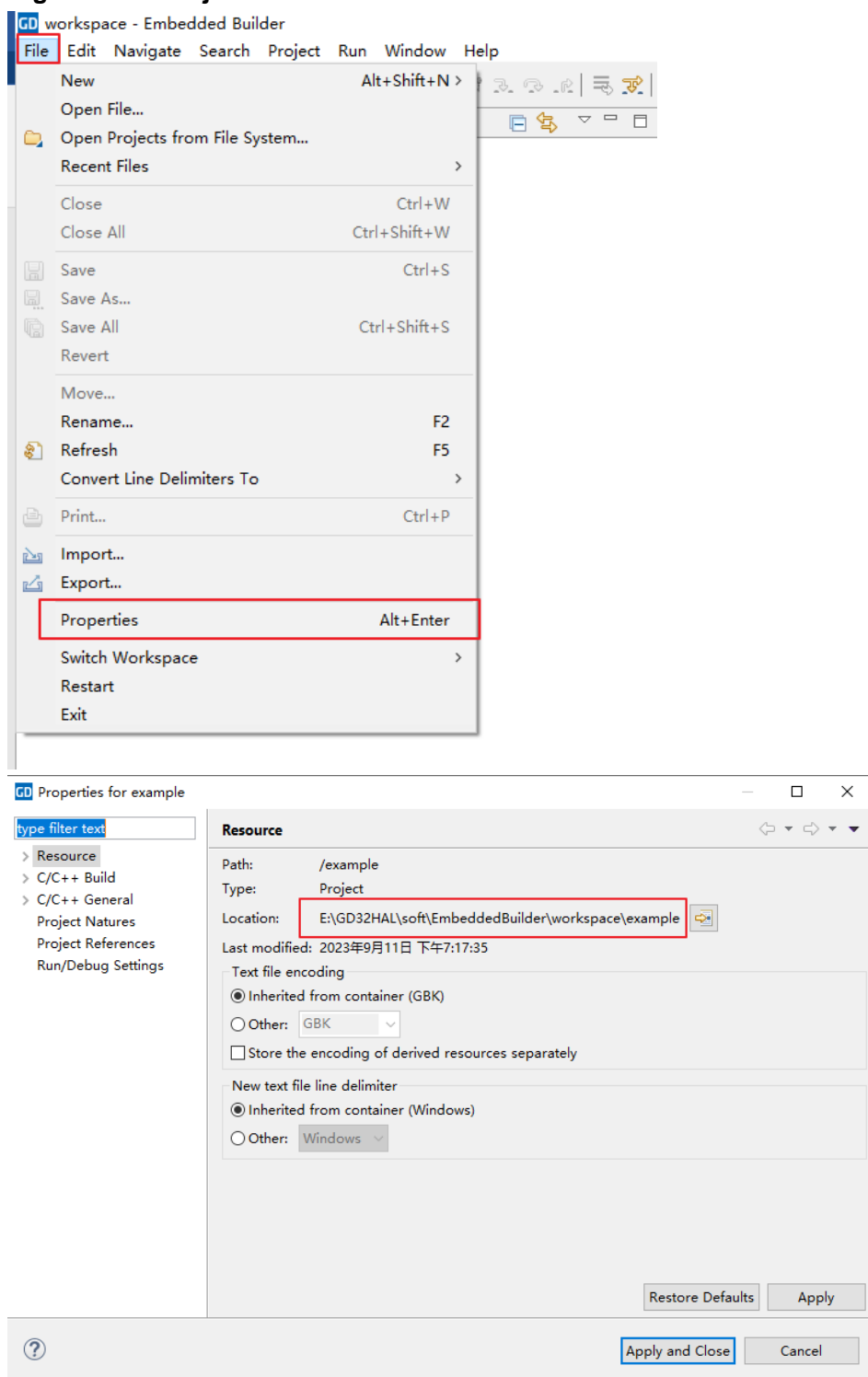
Figure 2-9. After generate code interface



2.5.3. Copy Code

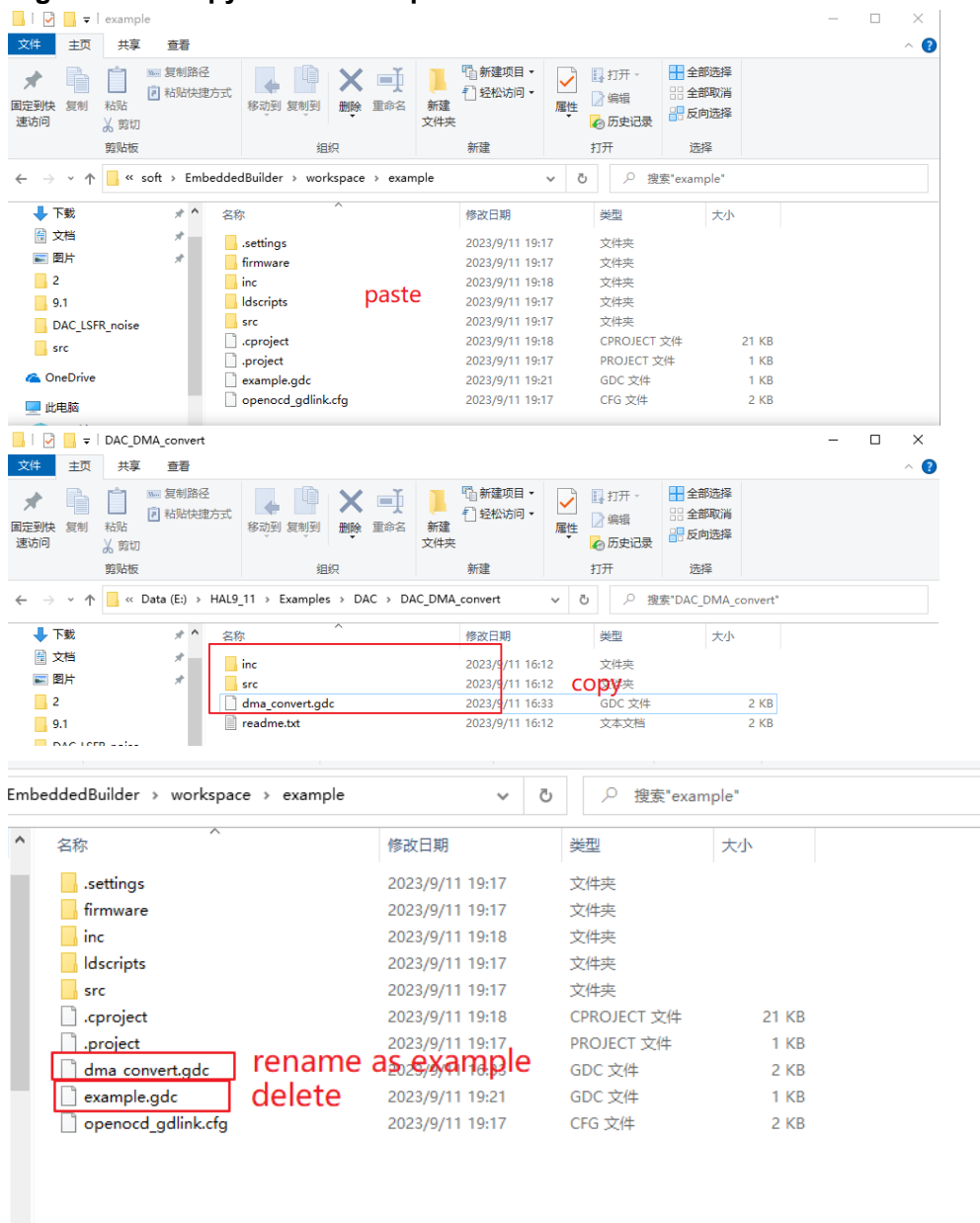
Open the folder address where the code is located, and user can find the corresponding address of the file from the corresponding address, as shown in [Figure 2-10. Project address search](#).

Figure 2-10. Project address search



Copy the file in the Examples folder to the project file, and delete a file and rename a file, as shown in [Figure 2-11. Copy and rename process](#).

Figure 2-11. Copy and rename process



Part of the code will use the files in the Utilities folder, so copy the gd32f3x0r_hal_eval.c and gd32f3x0r_hal_eval.h files to the corresponding inc and src file directories.

2.5.4. Bulid and download

Click Build all to complete the project bulid.

Figure 2-12. Bulid project

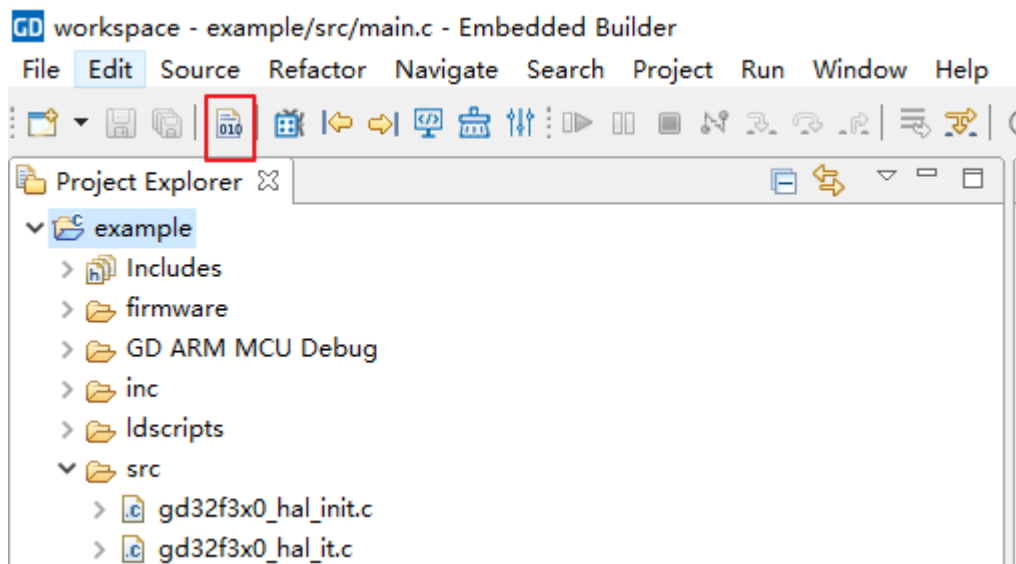
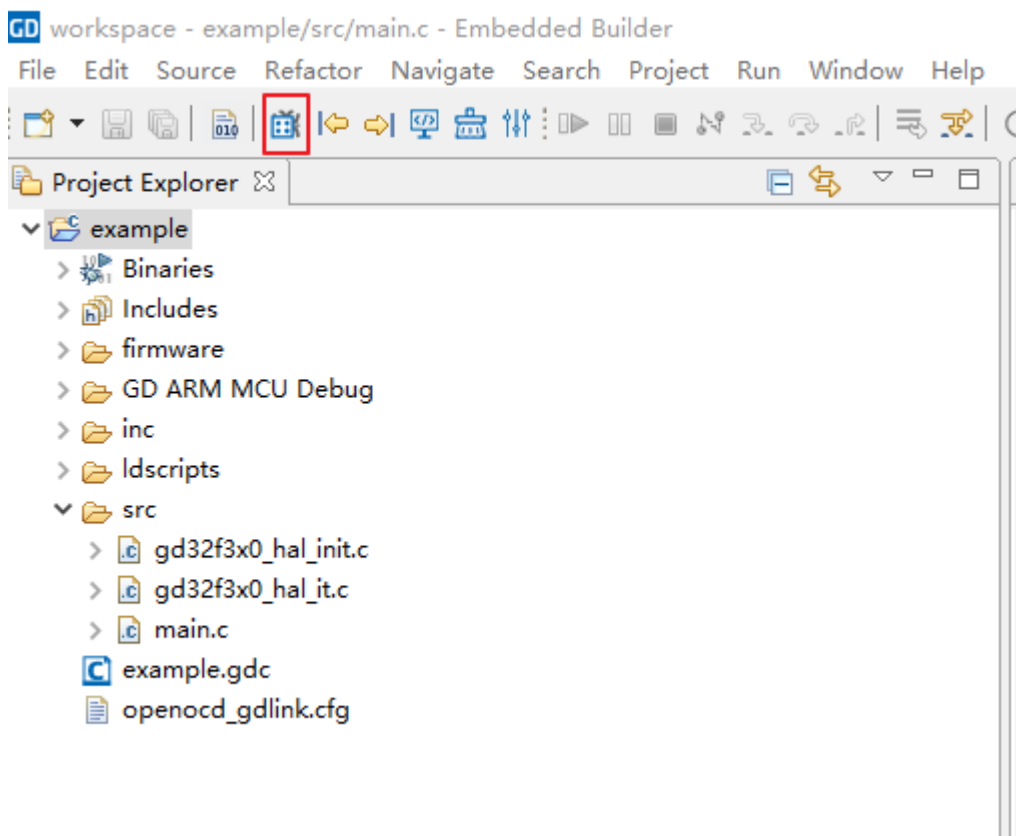
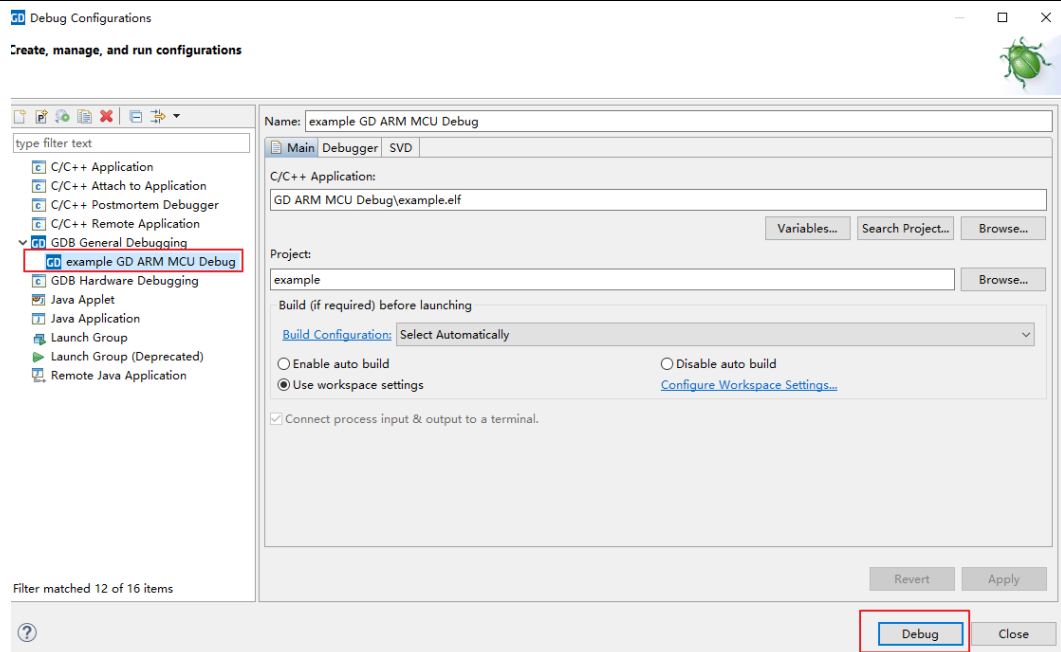


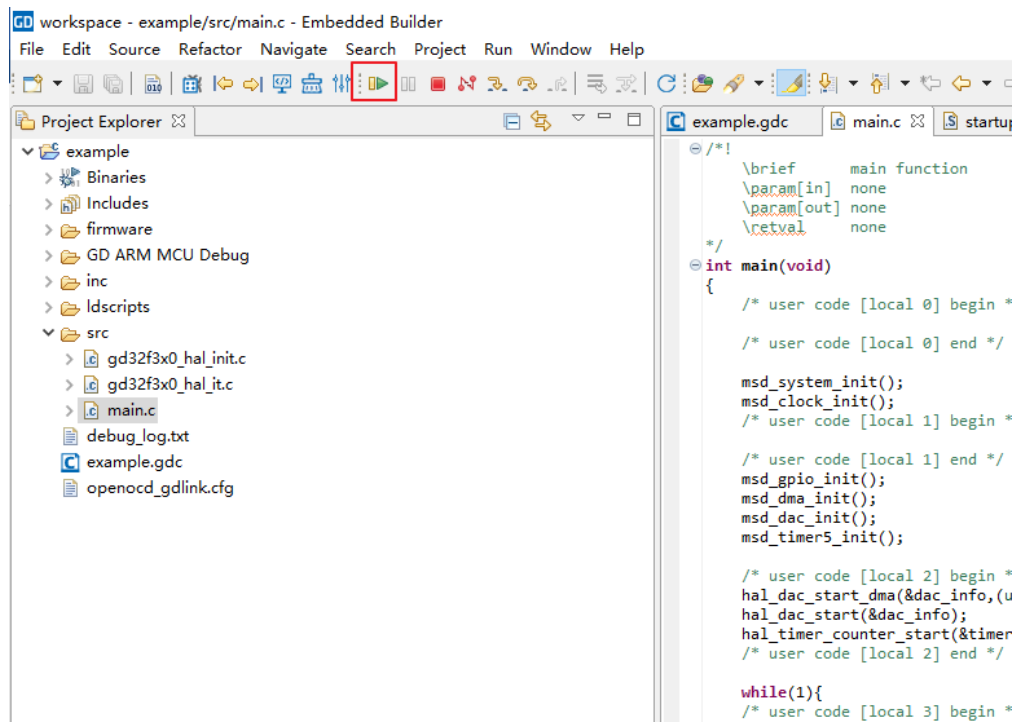
Figure 2-13. Debug project





2.5.5. Run project

Figure 2-14. Run project



2.6. File descriptions of Firmware Library

The major files about the firmware library are listed and described in the table below.

Table 2-1. Peripheral function format of Firmware Library

Files	Descriptions
gd32f3x0_hal_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.
gd32f3x0_hal_it.h	Header file, including all the prototypes of interrupt service routines.
gd32f3x0_hal_it.c	Source files about interrupt service routines of peripherals. User can write its 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 provided in the startup_gd32f3x0.s file.
gd32f3x0_hal_xxx.h	The header file of peripheral PPP, including functions about peripheral PPP, and the variables used for functions.
gd32f3x0_hal_xxx.c	The C source file for driving peripheral PPP.
gd32f3x0_hal_init.c	The source file include all the initialization and deinitialization of peripherals.
gd32f3x0_hal_init.h	The header file include all the prototypes of initialization and deinitialization of peripherals.
readme.txt	Description document about how to configure and how to use the firmware example.

2.7. FAQ for HAL firmware library

2.7.1. In many modules, the first parameter of the function is the device information structure, and what is the function of the structure?

There is a special device information structure named as `hal_xxx_dev_struct` in many modules such as Analog, Ticker, I/O device and General module.

The function of this structure is similar to an implement of peripheral function, which is used to store some important peripheral information which is transferred and called in different functions. In order to realize many complex functions and make different functions work together, `hal_xxx_dev_struct` is needed to save information. At the same time, because the HAL library does not have memory management function, the struct needs to be created by users themselves. Thus, the using of the structure has the following characteristics.

1. The device information structure must be created as a global variable;

2. The device information structure does not need to be modified by users. Users need to create the structure and initialize the structure by calling the corresponding initialization function named as `hal_xxx_struct_init`. Then, they only need to be passed in the function as a parameter;
3. The device information structure corresponds to peripherals. For example, users use both USART0 and USART1 peripherals. Two corresponding device information structures are needed to store their device information separately.

2.7.2. How to use functions in HAL firmware library? Is there any reference routine?

The example folder of HAL firmware library contains abundant routines for users to refer to. The routines basically cover all the functions provided by HAL firmware library. All routines are processed on the basis of code generated automatically by the host computer.

2.7.3. Why doesn't the device work properly with the code automatically generated by the host computer?

Considering that the user may want the module to start working at the right time, the initialization function of each module in the HAL library only completes the module configuration, but does not enable the module. When the code is automatically generated, users need to call function `hal_xxx_start()` or `hal_xxx_transmit_poll()` corresponding to the module to start the module function manually.

2.7.4. Can HAL firmware libraries be tailored? what features can be modified?

Users can tailor HAL firmware library and modify their features by modifying `gd32f3x0_libopt.h` header file.

Feature

The HAL firmware library provides three features that users can choose to turn on or off, as shown in the following figure.

```

/*if set, flash operation (write and eraser) will reserve original data located
...in out of targeted scope*/
#define FLASH_OPER_RESERVE_ORIGINAL_DATA .....0
/*if set, the parameters check will be implemented in function*/
#define HAL_PARAMETER_CHECK .....0
/*if set, print debug message according to level of marco 'HAL_DEBUG_PRINTF_LEVEL'
...and halt code according to level of marco 'HAL_DEBUG_HALT_LEVEL'*/
#define HAL_DEBUG .....0

#if (1 == HAL_DEBUG)
#define HAL_DEBUG_PRINTF .....printf
#define HAL_DEBUG_PRINTF_LEVEL .....HAL_DEBUG_LVL_ALL
#define HAL_DEBUG_HALT_LEVEL .....HAL_DEBUG_LVL_NONE

#define HAL_DEBUG_UART .....USART0
#define HAL_DEBUG_EXTRA_DO
#endif /* 1 == HAL_DEBUG */

```

1. FLASH_OPER_RESERVE_ORIGINAL_DATA

This macro controls flash erase and program mode which affect `hal_fmc_region_write()` / `hal_fmc_region_erase()` function of `gd32f3x0_hal_fmc.c` source file

when the marco equals 0, the function `hal_fmc_region_write()` erases the page in which the area is located before it operates on the incomplete 0xFF area in flash, and then performs the write operation. The data stored in the area not written by the page will be lost. `hal_fmc_region_erase()` erased in one page;

When the macro equals 1, `hal_fmc_region_write()` ensures that data outside the non-operating area of the same page will not be lost before writing. `Hal_fmc_region_erase()` erase in one byte;

2. HAL_PARAMETER_CHECK

This macro controls the performance of the function parameter checking in HAL firmware library

when the marco equals 0, all functions are not checked for parameters

when the marco equals 1, all functions do parameters checking, however, this performance will enlarge the that application size after compiled.

3. HAL_DEBUG

This marco controls the DEBUG function of HAL firmware library.

when the marco equals 0, turn off DEBUG function;

when the marco equals 1, it would print real time information about the running HAL firmware library accroding to the value of marco `HAL_DEBUG_PRINTF_LEVEL` and `HAL_DEBUG_HALT_LEVEL`. The program will be terminated when a fault has been detected. (`HAL_DEBUG_HALT_LEVEL` should equals to `HAL_DEBUG_LVL_FATAL` | `HAL_DEBUG_LVL_ERROR`) When turn on this marco, this performance will enlarge the application size after compiled.

Tailoring

The `gd32f3x0_libopt.h` header file contains all modules by default, as shown in the following figure. If users does not need to use some module, the header file could be uncommented,

and then the compiler will ignore the module in compiling.

```
#include "gd32f3x0_hal_dma.h"      #include "gd32f3x0_hal_dma.h"
#include "gd32f3x0_hal_fmc.h"      #include "gd32f3x0_hal_fmc.h"
#include "gd32f3x0_hal_pmu.h"      #include "gd32f3x0_hal_pmu.h"
#include "gd32f3x0_hal_dac.h"      #include "gd32f3x0_hal_dac.h"
#include "gd32f3x0_hal_gpio.h"     #include "gd32f3x0_hal_gpio.h"
#include "gd32f3x0_hal_rcu.h"      #include "gd32f3x0_hal_rcu.h"
#include "gd32f3x0_hal_exti.h"     #include "gd32f3x0_hal_exti.h"
#include "gd32f3x0_hal_sys.h"      #include "gd32f3x0_hal_sys.h"
#include "gd32f3x0_hal_syscfg.h"   #include "gd32f3x0_hal_syscfg.h"
#include "gd32f3x0_hal_nvic.h"     #include "gd32f3x0_hal_nvic.h"
#include "gd32f3x0_hal_cmp.h"      #include "gd32f3x0_hal_cmp.h"
#include "gd32f3x0_hal_cec.h"      #include "gd32f3x0_hal_cec.h"
#include "gd32f3x0_hal_crc.h"      #include "gd32f3x0_hal_crc.h"
#include "gd32f3x0_hal_adc.h"      #include "gd32f3x0_hal_adc.h"
#include "gd32f3x0_hal_ctc.h"      #include "gd32f3x0_hal_ctc.h"
#include "gd32f3x0_hal_fwdgt.h"    #include "gd32f3x0_hal_fwdgt.h"
#include "gd32f3x0_hal_tsi.h"      #include "gd32f3x0_hal_tsi.h"
#include "gd32f3x0_hal_wwdgt.h"    // #include "gd32f3x0_hal_wwdgt.h"
#include "gd32f3x0_hal_spi_com.h"  // #include "gd32f3x0_hal_spi_com.h"
#include "gd32f3x0_hal_spi.h"      // #include "gd32f3x0_hal_spi.h"
#include "gd32f3x0_hal_i2s.h"      #include "gd32f3x0_hal_i2s.h"
#include "gd32f3x0_hal_usart_com.h" #include "gd32f3x0_hal_usart_com.h"
#include "gd32f3x0_hal_uart.h"     #include "gd32f3x0_hal_uart.h"
#include "gd32f3x0_hal_usrt.h"     #include "gd32f3x0_hal_usrt.h"
#include "gd32f3x0_hal_irda.h"     #include "gd32f3x0_hal_irda.h"
#include "gd32f3x0_hal_smartcard.h" #include "gd32f3x0_hal_smartcard.h"
#include "gd32f3x0_hal_rtc.h"      // #include "gd32f3x0_hal_rtc.h"
#include "gd32f3x0_hal_i2c_com.h"  #include "gd32f3x0_hal_i2c_com.h"
#include "gd32f3x0_hal_i2c.h"      #include "gd32f3x0_hal_i2c.h"
#include "gd32f3x0_hal_smbus.h"    #include "gd32f3x0_hal_smbus.h"
#include "gd32f3x0_hal_timer.h"    #include "gd32f3x0_hal_timer.h"
#endif /* GD32F3X0_LIBOPT_H */    #endif /* GD32F3X0_LIBOPT_H */
```

2.7.5. What are the differences between a function prefixed with hal_ and a function prefixed with hals_?

The code of each peripheral module in the HAL firmware library can be named in two categories: 1. The prefix starts with hal_. 2. The prefix starts with hals_.

Functions prefixed with "hal_" are standard functions in the HAL firmware library. The implementation of each peripheral module has a unified style and characteristics. For the functions and characteristics of the implementation, see the relevant descriptions in Section 2.2/2.3.

Functions prefixed with "hals_" are supplementary simple functional functions, which are not necessary for the function architecture system of the HAL firmware library. The specific function implementation of each peripheral module is very different, and users only use such functions to perform a single quick operation on the specific bit or bit field of the register of the peripheral module.

3. Firmware Library of Standard Peripherals

3.1. Overview of Firmware Library of Standard Peripherals

The description format of firmware functions are shown as below:

Table 3-1. Peripheral function format of Firmware Library

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

3.2. ADC

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

3.2.1. Descriptions of Peripheral registers

ADC registers are listed in the table shown as below:

Table 3-2. ADC Registers

Registers	Descriptions
ADC_STAT	ADC status register
ADC_CTL0	ADC control register 0
ADC_CTL1	ADC control register 1
ADC_SAMPT0	ADC sample time register 0
ADC_SAMPT1	ADC sample time register 1
ADC_IOFFx	ADC inserted channel data offset register x

Registers	Descriptions
ADC_WDHT	ADC watchdog high threshold register
ADC_WDLT	ADC watchdog low threshold register
ADC_RSQ0	ADC routine sequence register 0
ADC_RSQ1	ADC routine sequence register 1
ADC_RSQ2	ADC routine sequence register 2
ADC_ISQ	ADC inserted sequence register
ADC_IDATAx	ADC inserted data register x
ADC_RDATA	ADC routine data register
ADC_OVSAMPCTL	ADC oversampling control register

3.2.2. Descriptions of Peripheral functions

ADC registers are listed in the table shown as below:

Table 3-3. ADC firmware function

Function name	Function description
hal_adc_struct_init	initialize the ADC structure with the default values
hal_adc_deinit	deinitialize ADC
hal_adc_init	initialize ADC
hal_adc_calibration_start	ADC calibration
hal_adc_routine_channel_config	configure adc routine channel
hal_adc_routine_rank_config	configure ADC routine rank
hal_adc_start	enable ADC and start the conversion of routine sequence
hal_adc_stop	stop the conversion of routine sequence and disable ADC
hal_adc_routine_conversion_poll	polling for ADC routine sequence conversion
hal_adc_routine_software_trigger_enable	enable ADC routine sequence software trigger
hal_adc_start_interrupt	start ADC EOC interrupt
hal_adc_stop_interrupt	stop ADC EOC interrupt
hal_adc_start_dma	enable ADC and start the conversion of routine sequence with DMA
hal_adc_stop_dma	stop the conversion of routine sequence, disable ADC DMA mode and disable ADC
hal_adc_inserted_channel_config	configure ADC inserted channel
hal_adc_inserted_rank_config	configure ADC inserted rank
hal_adc_inserted_start	enable ADC and start the conversion of inserted sequence
hal_adc_inserted_stop	stop the conversion of inserted sequence and disable ADC
hal_adc_inserted_conversion_poll	polling for ADC inserted sequence conversion
hal_adc_inserted_software_trigger_enable	enable ADC inserted sequence software trigger
hal_adc_inserted_start_interrupt	start ADC EOIC interrupt

Function name	Function description
hal_adc_inserted_stop_interrupt	stop ADC EOIC interrupt
hal_adc_watchdog_config	configure watchdog
hal_adc_watchdog_interrupt_enable	enable ADC watchdog interrupt
hal_adc_watchdog_interrupt_disable	disable ADC watchdog interrupt
hal_adc_watchdog_event_poll	polling for ADC watchdog event conversion
hal_adc_irq	ADC interrupt handler content function, which is merely used in ADC_CMP_IRQHandler
hal_adc_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_adc_irq_handle_all_reset	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_adc_routine_value_get	get routine sequence conversion result
hal_adc_inserted_value_get	get inserted sequence conversion result
hal_adc_error_get	get ADC error
hal_adc_state_get	get ADC state

Enum hal_adc_struct_type_enum

Table 3-4. Enum hal_adc_struct_type_enum

Member name	Function description
HAL_ADC_INIT_STRUCTURE	ADC initialization structure
HAL_ADC_ROUTINE_RANK_CONFIGURATION_STRUCTURE	ADC routine rank configuration structure
HAL_ADC_ROUTINE_CHANNEL_CONFIGURATION_STRUCTURE	ADC routine channel configuration structure
HAL_ADC_INSERTED_RANK_CONFIGURATION_STRUCTURE	ADC inserted rank configuration structure
HAL_ADC_INSERTED_CHANNEL_CONFIGURATION_STRUCTURE	ADC inserted channel configuration structure
HAL_ADC_IRQ_STRUCTURE	ADC device interrupt callback function pointer structure
HAL_ADC_DMA_HANDLER_CALLBACK_STRUCTURE	ADC DMA callback function pointer structure

HAL_ADC_WATCH DOG_CONFIG_ST RUCT	ADC watchdog configuration structure
HAL_ADC_DEV_ST RUCT	ADC device information structrue

Enum hal_adc_state_enum

Table 3-5. Enum hal_adc_state_enum

Member name	Function description
HAL_ADC_STATE_ RESET	ADC is not initialized or disabled
HAL_ADC_STATE_ READY	ADC is ready
HAL_ADC_STATE_ BUSY_SYSTEM	ADC is busy to internal system (initialization, calibration)
HAL_ADC_STATE_ TIMEOUT	ADC timeout occurs
HAL_ADC_STATE_ ROUTINE_BUSY	a conversion is ongoing on routine sequence
HAL_ADC_STATE_ ROUTINE_EOC	conversion data available on routine sequence
HAL_ADC_STATE_ INSERTED_BUSY	a conversion is ongoing on inserted sequence
HAL_ADC_STATE_ INSERTED_EOC	conversion data available on inserted sequence
HAL_ADC_STATE_ WATCHDOG	analog watchdog

Enum hal_adc_error_enum

Table 3-6. Enum hal_adc_error_enum

Member name	Function description
HAL_ADC_ERROR_ _NONE	no error
HAL_ADC_ERROR_ _SYSTEM	ADC internal error: if problem of clocking, enable/disable, wrong state
HAL_ADC_ERROR_ _DMA	DMA transfer error
HAL_ADC_ERROR_ _CONFIG	configuration error occurs

Enum hal_adc_oversample_shift_enum

Table 3-7. Enum hal_adc_oversample_shift_enum

Member name	Function description
ADC_OVERSAMPL E_SHIFT_NONE	no oversampling shift
ADC_OVERSAMPL E_SHIFT_1B	1-bit oversampling shift
ADC_OVERSAMPL E_SHIFT_2B	2-bit oversampling shift
ADC_OVERSAMPL E_SHIFT_3B	3-bit oversampling shift
ADC_OVERSAMPL E_SHIFT_4B	4-bit oversampling shift
ADC_OVERSAMPL E_SHIFT_5B	5-bit oversampling shift
ADC_OVERSAMPL E_SHIFT_6B	6-bit oversampling shift
ADC_OVERSAMPL E_SHIFT_7B	7-bit oversampling shift
ADC_OVERSAMPL E_SHIFT_8B	8-bit oversampling shift

Enum hal_adc_oversample_ratio_enum

Table 3-8. Enum hal_adc_oversample_ratio_enum

Member name	Function description
ADC_OVERSAMPL E_RATIO_MUL2	oversampling ratio multiple 2
ADC_OVERSAMPL E_RATIO_MUL4	oversampling ratio multiple 4
ADC_OVERSAMPL E_RATIO_MUL8	oversampling ratio multiple 8
ADC_OVERSAMPL E_RATIO_MUL16	oversampling ratio multiple 16
ADC_OVERSAMPL E_RATIO_MUL32	oversampling ratio multiple 32
ADC_OVERSAMPL E_RATIO_MUL64	oversampling ratio multiple 64
ADC_OVERSAMPL E_RATIO_MUL128	oversampling ratio multiple 128
ADC_OVERSAMPL	oversampling ratio multiple 256

E_RATIO_MUL256

Enum hal_adc_routine_sequence_enum

Table 3-9. Enum hal_adc_routine_sequence_enum

Member name	Function description
ADC_ROUTINE_SEQUENCE_0	ADC routine channel sequence 0
ADC_ROUTINE_SEQUENCE_1	ADC routine channel sequence 1
ADC_ROUTINE_SEQUENCE_2	ADC routine channel sequence 2
ADC_ROUTINE_SEQUENCE_3	ADC routine channel sequence 3
ADC_ROUTINE_SEQUENCE_4	ADC routine channel sequence 4
ADC_ROUTINE_SEQUENCE_5	ADC routine channel sequence 5
ADC_ROUTINE_SEQUENCE_6	ADC routine channel sequence 6
ADC_ROUTINE_SEQUENCE_7	ADC routine channel sequence 7
ADC_ROUTINE_SEQUENCE_8	ADC routine channel sequence 8
ADC_ROUTINE_SEQUENCE_9	ADC routine channel sequence 9
ADC_ROUTINE_SEQUENCE_10	ADC routine channel sequence 10
ADC_ROUTINE_SEQUENCE_11	ADC routine channel sequence 11
ADC_ROUTINE_SEQUENCE_12	ADC routine channel sequence 12
ADC_ROUTINE_SEQUENCE_13	ADC routine channel sequence 13
ADC_ROUTINE_SEQUENCE_14	ADC routine channel sequence 14
ADC_ROUTINE_SEQUENCE_15	ADC routine channel sequence 15

Enum hal_adc_inserted_sequence_enum

Table 3-10. Enum hal_adc_inserted_sequence_enum

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

ADC_INSERTED_SEQUENCE_0	ADC inserted channel sequence 0
ADC_INSERTED_SEQUENCE_1	ADC inserted channel sequence 1
ADC_INSERTED_SEQUENCE_2	ADC inserted channel sequence 2
ADC_INSERTED_SEQUENCE_3	ADC inserted channel sequence 3

Structure hal_adc_irq_struct

Table 3-11. Structure hal_adc_irq_struct

Member name	Function description
adc_eoc_handle	EOC interrupt handler function
adc_eoic_handle	EOIC interrupt handler function
adc_watchdog_handle	watchdog event interrupt handler function

Structure hal_adc_dma_handle_cb_struct

Table 3-12. Structure hal_adc_dma_handle_cb_struct

Member name	Function description
full_transcom_handler	ADC DMA transfer complete interrupt handler function
half_transcom_handler	ADC DMA half transfer complete interrupt handler function
error_handle	ADC DMA error underflow handler function

Structure hal_adc_dev_struct

Table 3-13. Structure hal_adc_dev_struct

Member name	Function description
adc_irq	ADC device interrupt callback function pointer structure
*p_dma_adc	DMA device information structrue
adc_dma	DMA callback function pointer structure
error_state	ADC error state
mutex	ADC mutex set
state	ADC state

Structure hal_adc_init_struct

Table 3-14. Structure hal_adc_init_struct

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

data_alignment	ADC data alignment
resolution	ADC data resolution select
scan_mode	scan mode
hardware_oversampling	oversampling enable
oversample_trigger_mode	triggered oversampling
oversampling_shift	oversampling shift
oversampling_ratio	oversampling ratio

Structure hal_adc_routine_rank_config_struct

Table 3-15. Structure hal_adc_routine_rank_config_struct

Member name	Function description
channel	ADC channel select
sampling_time	ADC sample time
routine_sequence	ADC routine channel sequence

Structure hal_adc_routine_config_struct

Table 3-16. Structure hal_adc_routine_config_struct

Member name	Function description
routine_sequence_conversions	routine sequence enable
routine_sequence_length	routine sequence length
routine_sequence_external_trigger_select	external trigger select for routine sequence
continuous_mode;	continuous mode
discontinuous_mode	discontinuous mode
number_of_conversions_in_discontinuous_mode	discontinuous mode length

Structure hal_adc_inserted_rank_config_struct

Table 3-17. Structure hal_adc_inserted_rank_config_struct

Member name	Function description
channel	ADC channel select
sampling_time	ADC sample time
data_offset	ADC data offset
inserted_sequence	ADC inserted channel sequence

Structure hal_adc_inserted_config_struct

Table 3-18. Structure hal_adc_inserted_config_struct

Member name	Function description
inserted_sequence_conversions	inserted sequence enable
inserted_sequence_length	ADC inserted channel sequence,0~3
inserted_sequence_external_trigger_select	ADC inserted channel sequence external trigger select
auto_convert	ADC inserted channel convert automatically
discontinuous_mode	ADC inserted channel discontinuous mode

Structure hal_adc_watchdog_config_struct

Table 3-19. Structure hal_adc_watchdog_config_struct

Member name	Function description
routine_sequence_analog_watchdog	routine sequence analog watchdog enable
inserted_sequence_analog_watchdog	inserted sequence analog watchdog enable
analog_watchdog_mode	analog watchdog mode
analog_watchdog_channel_select	analog watchdog channel select
analog_watchdog_high_threshold	ADC analog watchdog high threshold
analog_watchdog_low_threshold	ADC analog watchdog low threshold

hal_adc_struct_init

The description of hal_adc_struct_init is shown as below:

Table 3-20. Function hal_adc_struct_init

Function name	hal_adc_struct_init
Function prototype	void hal_adc_struct_init(hal_adc_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the ADC structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	

hal_struct_type	type of structure to be initialized
HAL_ADC_INIT_STRUC	ADC initialization structure
HAL_ADC_ROUTINE_CONFIG_STRUCT	ADC routine channel configuration structure
HAL_ADC_ROUTINE_RANK_CONFIG_STRUC	ADC routine rank configuration structure
HAL_ADC_INSERTED_CONFIG_STRUCT	ADC inserted channel configuration structure
HAL_ADC_INSERTED_RANK_CONFIG_STRUCT	ADC inserted rank configuration structure
HAL_ADC_IRQ_STRUC	ADC device interrupt callback function pointer structure
HAL_ADC_DMA_HANDLER_CB_STRUCT	ADC DMA callback function pointer structure
HAL_ADC_WATCHDOG_CONFIG_STRUCT	ADC watchdog configuration structure
HAL_ADC_DEV_STRUC	ADC device information structure
Input parameter{in}	
p_struct	point to ADC structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the ADC structure with the default values */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_struct_init(HAL_ADC_DEV_STRUCT, &adc_info);
```

hal_adc_deinit

The description of hal_adc_deinit is shown as below:

Table 3-21. Function hal_adc_deinit

Function name	hal_adc_deinit
Function prototype	int32_t hal_adc_deinit(hal_adc_dev_struct *adc_dev);
Function descriptions	deinitialize ADC device structure and init structure
Precondition	-

The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```
/* deinitialize ADC */

hal_adc_dev_struct adc_info;

hal_adc_deinit(&adc_info);
```

hal_adc_init

The description of hal_adc_init is shown as below:

Table 3-22. Function hal_adc_init

Function name	hal_adc_init
Function prototype	int32_t hal_adc_init(hal_adc_dev_struct *adc_dev, hal_adc_init_struct *p_init);
Function descriptions	initialize ADC
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
p_init	points to hal_adc_init_struct, struct members refer to Table 3-14. Structure hal_adc_init_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```
/* initialize ADC */

hal_adc_dev_struct adc_info;

hal_adc_init_struct adc_init_parameter;
```

```

adc_init_parameter.data_alignment = ADC_LSB_ALIGNMENT;

adc_init_parameter.resolution = ADC_RESOLUTION_12B;

adc_init_parameter.scan_mode = DISABLE;

adc_init_parameter.hardware_oversampling = DISABLE;

hal_adc_init(&adc_info,&adc_init_parameter);

```

hal_adc_calibration_start

The description of hal_adc_calibration_start is shown as below:

Table 3-23. Function hal_adc_calibration_start

Function name	hal_adc_calibration_start
Function prototype	int32_t hal_adc_calibration_start(hal_adc_dev_struct *adc_dev);
Function descriptions	ADC calibration
Precondition	called after ADC initialization but before ADC starting sampling
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```

/* ADC calibration */

hal_adc_dev_struct adc_info;

hal_adc_calibration_start(&adc_info);

```

hal_adc_routine_channel_config

The description of hal_adc_routine_channel_config is shown as below:

Table 3-24. Function hal_adc_routine_channel_config

Function name	hal_adc_routine_channel_config
Function prototype	int32_t hal_adc_routine_channel_config(hal_adc_dev_struct *adc_dev, hal_adc_routine_config_struct *p_rchannel);
Function descriptions	initialize ADC routine channel
Precondition	-
The called functions	-
Input parameter{in}	

adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
p_rchannel	points to hal_adc_routine_config_struct, struct members refer to Table 3-16. Structure hal_adc_routine_config_struct .
Output parameter{out}	
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```

/* initialize ADC routine channel */

hal_adc_dev_struct adc_info;

hal_adc_routine_config_struct adc_routine_config_parameter;

adc_routine_config_parameter.routine_sequence_conversions = ENABLE;

adc_routine_config_parameter.routine_sequence_length = 1;

adc_routine_config_parameter.routine_sequence_external_trigger_select =
ADC_EXTTRIG_ROUTINE_NONE;

adc_routine_config_parameter.continuous_mode = DISABLE;

adc_routine_config_parameter.discontinuous_mode = DISABLE;

hal_adc_routine_channel_config(&adc_info,&adc_routine_config_parameter);

```

hal_adc_routine_rank_config

The description of hal_adc_routine_rank_config is shown as below:

Table 3-25. Function hal_adc_routine_rank_config

Function name	hal_adc_routine_rank_config
Function prototype	int32_t hal_adc_routine_rank_config(hal_adc_dev_struct *adc_dev, hal_adc_routine_rank_config_struct *p_rrank);
Function descriptions	configure ADC routine rank
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
p_rrank	points to hal_adc_routine_rank_config_struct, struct members refer to Table 3-15. Structure hal_adc_routine_rank_config_struct .

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* configure ADC routine sequence: Sequence 0 */
hal_adc_dev_struct adc_info;
hal_adc_routine_rank_config_struct adc_routine_rank_config_parameter;
adc_routine_rank_config_parameter.channel = ADC_CHANNEL_0;
adc_routine_rank_config_parameter.sampling_time = ADC_SAMPLETIME_1POINT5;
adc_routine_rank_config_parameter.routine_sequence = ADC_ROUTINE_SEQUENCE_0;
hal_adc_routine_rank_config(&adc_info,&adc_routine_rank_config_parameter);
```

hal_adc_start

The description of hal_adc_start is shown as below:

Table 3-26. Function hal_adc_start

Function name	hal_adc_start
Function prototype	hal_adc_start(hal_adc_dev_struct *adc_dev);
Function descriptions	enable ADC and start the conversion of routine sequence
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```
/* enable ADC and start the conversion of routine sequence */
hal_adc_dev_struct adc_info;
hal_adc_start(&adc_info);
```

hal_adc_stop

The description of hal_adc_stop is shown as below:

Table 3-27. Function hal_adc_stop

Function name	hal_adc_stop
Function prototype	hal_adc_stop(hal_adc_dev_struct *adc_dev);
Function descriptions	stop the conversion of routine sequence and disable ADC
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```
/* stop the conversion of routine sequence and disable ADC */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_stop(&adc_info);
```

hal_adc_routine_conversion_poll

The description of hal_adc_routine_conversion_poll is shown as below:

Table 3-28. Function hal_adc_routine_conversion_poll

Function name	hal_adc_routine_conversion_poll
Function prototype	int32_t hal_adc_routine_conversion_poll(hal_adc_dev_struct *adc_dev, uint32_t timeout_ms);
Function descriptions	polling for ADC routine sequence conversion
Precondition	-
The called functions	hal_sys_basetick_count_get
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
timeout_ms	time out duration in millisecond
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_TIMEOUT
----------------	---

Example:

```
/* polling for ADC routine sequence conversion */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_routine_conversion_poll(&adc_info, 2);
```

hal_adc_routine_software_trigger_enable

The description of hal_adc_routine_software_trigger_enable is shown as below:

Table 3-29. Function hal_adc_routine_software_trigger_enable

Function name	hal_adc_routine_software_trigger_enable
Function prototype	void hal_adc_routine_software_trigger_enable(hal_adc_dev_struct *adc_dev);
Function descriptions	enable ADC routine sequence software trigger
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_routine_software_trigger_enable(&adc_info);
```

hal_adc_start_interrupt

The description of hal_adc_start_interrupt is shown as below:

Table 3-30. Function hal_adc_start_interrupt

Function name	hal_adc_start_interrupt
Function prototype	int32_t hal_adc_start_interrupt(hal_adc_dev_struct *adc_dev, hal_adc_irq_struct *p_irq);
Function descriptions	start ADC EOC interrupt
Precondition	-

The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
p_irq	points to hal_adc_irq_struct, struct members refer to Table 3-11. Structure hal_adc_irq_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```

/* start ADC EOC interrupt */

hal_adc_dev_struct adc_info;

hal_adc_irq_struct adc_irq_parameter;

void adc_irq_routine_sequence(void)
{
    /* user defined content */
}

adc_irq_parameter.adc_eoc_handle = adc_irq_routine_sequence;

hal_adc_start_interrupt (&adc_info, &adc_irq_parameter);

```

hal_adc_stop_interrupt

The description of hal_adc_stop_interrupt is shown as below:

Table 3-31. Function hal_adc_stop_interrupt

Function name	hal_adc_stop_interrupt
Function prototype	int32_t hal_adc_stop_interrupt(hal_adc_dev_struct *adc_dev);
Function descriptions	stop ADC EOC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE
----------------	---

Example:

```
/* stop ADC EOC interrupt */

hal_adc_dev_struct adc_info;

hal_adc_stop_interrupt(&adc_info);
```

hal_adc_start_dma

The description of hal_adc_start_dma is shown as below:

Table 3-32. Function hal_adc_start_dma

Function name	hal_adc_start_dma
Function prototype	int32_t hal_adc_start_dma(hal_adc_dev_struct *adc_dev, uint32_t *pdata, uint32_t length, hal_adc_dma_handle_cb_struct *dmacb);
Function descriptions	enable ADC and start the conversion of routine sequence with DMA
Precondition	-
The called functions	hal_dma_struct_init \ hal_dma_start_interrupt
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
pdata	the source memory buffer address
Input parameter{in}	
length	the number of data to be transferred from source to destination
Input parameter{in}	
dmacb	points to hal_adc_dma_handle_cb_struct, struct members refer to Table 3-12. Structure hal_adc_dma_handle_cb_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_HARDWARE

Example:

```
/* start ADC DMA transmission */

hal_adc_dev_struct adc_info;

hal_adc_dma_handle_cb_struct adc_dma_parameter;

uint16_t adc_value[8];

void adc_dma_complete(void)
```

```
{
    /* user defined content */
}

/* initialize the user callback structure */

adc_dma_parameter.full_transcom_handle = adc_dma_complete;

adc_dma_parameter.error_handle      = NULL;

hal_adc_start_dma(&adc_info, (uint32_t *)&adc_value, 8, &adc_dma_parameter);
```

hal_adc_stop_dma

The description of hal_adc_stop_dma is shown as below:

Table 3-33. Function hal_adc_stop_dma

Function name	hal_adc_stop_dma
Function prototype	int32_t hal_adc_stop_dma(hal_adc_dev_struct *adc_dev);
Function descriptions	stop the conversion of routine sequence, disable ADC DMA mode and disable ADC
Precondition	-
The called functions	hal_dma_stop
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```
/* stop ADC DMA transmission */

hal_adc_dev_struct adc_info;

hal_adc_stop_dma(&adc_info);
```

hal_adc_inserted_channel_config

The description of hal_adc_inserted_channel_config is shown as below:

Table 3-34. Function hal_adc_inserted_channel_config

Function name	hal_adc_inserted_channel_config
Function prototype	int32_t hal_adc_inserted_channel_config(hal_adc_dev_struct *adc_dev, hal_adc_inserted_config_struct *p_ichannel);

Function descriptions	initialize ADC inserted channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
p_ichannel	points to hal_adc_inserted_config_struct, struct members refer to Table 3-18. Structure hal_adc_inserted_config_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT

Example:

```
/* initialize ADC inserted channel */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_inserted_config_struct adc_inserted_config_parameter;
```

```
adc_inserted_config_parameter.inserted_sequence_conversions = ENABLE;
```

```
adc_inserted_config_parameter.inserted_sequence_length = 2;
```

```
adc_inserted_config_parameter.inserted_sequence_external_trigger_select =  
ADC_EXTTRIG_INSERTED_T0_TRGO;
```

```
adc_inserted_config_parameter.discontinuous_mode = DISABLE;
```

```
adc_inserted_config_parameter.auto_convert = DISABLE;
```

```
hal_adc_inserted_channel_config(&adc_info,&adc_inserted_config_parameter);
```

hal_adc_inserted_rank_config

The description of hal_adc_inserted_rank_config is shown as below:

Table 3-35. Function hal_adc_inserted_rank_config

Function name	hal_adc_inserted_rank_config
Function prototype	int32_t hal_adc_inserted_rank_config(hal_adc_dev_struct *adc_dev, hal_adc_inserted_rank_config_struct *p_irank);
Function descriptions	configure ADC inserted rank
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13.

	<u>Structure hal_adc_dev_struct.</u>
Input parameter{in}	
p_irank	points to hal_adc_inserted_rank_config_struct, struct members refer to <u>Table 3-17. Structure hal_adc_inserted_rank_config_struct.</u>
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT

Example:

```

/* configure ADC inserted channel: Sequence 0 */

hal_adc_dev_struct adc_info;

hal_adc_inserted_rank_config_struct adc_inserted_rank_config_parameter;

adc_inserted_rank_config_parameter.data_offset = 0;

adc_inserted_rank_config_parameter.channel = ADC_CHANNEL_0;

adc_inserted_rank_config_parameter.sampling_time = ADC_SAMPLETIME_1POINT5;

adc_inserted_rank_config_parameter.inserted_sequence =
ADC_INSERTED_SEQUENCE_0;

hal_adc_inserted_rank_config(&adc_info,&adc_inserted_rank_config_parameter);

```

hal_adc_inserted_start

The description of hal_adc_inserted_start is shown as below:

Table 3-36. Function hal_adc_inserted_start

Function name	hal_adc_inserted_start
Function prototype	int32_t hal_adc_inserted_start(hal_adc_dev_struct *adc_dev);
Function descriptions	enable ADC and start the conversion of inserted sequence
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to <u>Table 3-13. Structure hal_adc_dev_struct.</u>
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```
/* enable ADC and start the conversion of inserted sequence */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_inserted_start(&adc_info);
```

hal_adc_inserted_stop

The description of hal_adc_inserted_stop is shown as below:

Table 3-37. Function hal_adc_inserted_stop

Function name	hal_adc_inserted_stop
Function prototype	int32_t hal_adc_inserted_stop(hal_adc_dev_struct *adc_dev);
Function descriptions	stop the conversion of inserted sequence and disable ADC
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT

Example:

```
/* stop the conversion of inserted sequence and disable ADC */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_inserted_stop(&adc_info);
```

hal_adc_inserted_conversion_poll

The description of hal_adc_inserted_conversion_poll is shown as below:

Table 3-38. Function hal_adc_inserted_conversion_poll

Function name	hal_adc_inserted_conversion_poll
Function prototype	int32_t hal_adc_inserted_conversion_poll(hal_adc_dev_struct *adc_dev, uint32_t timeout_ms);
Function descriptions	polling for ADC inserted sequence conversion, this function is just for single conversion
Precondition	-
The called functions	hal_sys_basetick_count_get
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .

Input parameter{in}	
timeout_ms	time out duration in milliscond
Output parameter{out}	
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE, HAL_ERR_TIMEOUT

Example:

```
/* polling for ADC inserted sequence conversion */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_inserted_conversion_poll(&adc_info, 2);
```

hal_adc_inserted_software_trigger_enable

The description of hal_adc_inserted_software_trigger_enable is shown as below:

Table 3-39. Function hal_adc_inserted_software_trigger_enable

Function name	hal_adc_inserted_software_trigger_enable
Function prototype	void hal_adc_inserted_software_trigger_enable(hal_adc_dev_struct *adc_dev);
Function descriptions	enable ADC inserted sequence software trigger
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC inserted sequence software trigger */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_inserted_software_trigger_enable(&adc_info);
```

hal_adc_inserted_start_interrupt

The description of hal_adc_inserted_start_interrupt is shown as below:

Table 3-40. Function hal_adc_inserted_start_interrupt

Function name	hal_adc_inserted_start_interrupt
Function prototype	int32_t hal_adc_inserted_start_interrupt(hal_adc_dev_struct *adc_dev, hal_adc_irq_struct *p_irq);
Function descriptions	start ADC EOIC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
p_irq	points to hal_adc_irq_struct, struct members refer to Table 3-11. Structure hal_adc_irq_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_HARDWARE

Example:

```

/* start ADC EOIC interrupt */

hal_adc_dev_struct adc_info;

hal_adc_irq_struct adc_irq_parameter;

void adc_irq_inserted_sequence(void)
{
    /* user defined content */
}

adc_irq_parameter.adc_eoic_handle = adc_irq_inserted_sequence;

hal_adc_inserted_start_interrupt(&adc_info, &adc_irq_parameter);

```

hal_adc_inserted_stop_interrupt

The description of hal_adc_inserted_stop_interrupt is shown as below:

Table 3-41. Function hal_adc_inserted_stop_interrupt

Function name	hal_adc_inserted_stop_interrupt
Function prototype	int32_t hal_adc_inserted_stop_interrupt(hal_adc_dev_struct *adc_dev);
Function descriptions	stop ADC EOIC interrupt
Precondition	-
The called functions	-

Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT

Example:

```
/* stop ADC EOIC interrupt */
hal_adc_dev_struct adc_info;
hal_adc_inserted_stop_interrupt(&adc_info);
```

hal_adc_watchdog_config

The description of hal_adc_watchdog_config is shown as below:

Table 3-42. Function hal_adc_watchdog_config

Function name	hal_adc_watchdog_config
Function prototype	int32_t hal_adc_watchdog_config(hal_adc_dev_struct *adc_dev, hal_adc_watchdog_config_struct *p_watchdog);
Function descriptions	configure watchdog
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
p_watchdog	points to hal_adc_watchdog_config_struct, struct members refer to Table 3-19. Structure hal_adc_watchdog_config_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* configure ADC watchdog */
hal_adc_dev_struct adc_info;
hal_adc_watchdog_struct adc_watchdog_config_parameter;
adc_watchdog_config_parameter.routine_sequence_analog_watchdog = ENABLE;
```



```

adc_watchdog_config_parameter.inserted_sequence_analog_watchdog = DISABLE;

adc_watchdog_config_parameter.analog_watchdog_mode =
ADC_WATCHDOG_MODE_SINGLE_CHANNEL;

adc_watchdog_config_parameter.analog_watchdog_channel_select = ADC_CHANNEL_0;

adc_watchdog_config_parameter.analog_watchdog_high_threshold = 0xAFF;

adc_watchdog_config_parameter.analog_watchdog_low_threshold = 0x0;

hal_adc_watchdog_config(&adc_info, &adc_watchdog_config_parameter);

```

hal_adc_watchdog_interrupt_enable

The description of hal_adc_watchdog_interrupt_enable is shown as below:

Table 3-43. Function hal_adc_watchdog_interrupt_enable

Function name	hal_adc_watchdog_interrupt_enable
Function prototype	int32_t hal_adc_watchdog_interrupt_enable(hal_adc_dev_struct *adc_dev, hal_adc_irq_struct *p_irq);
Function descriptions	enable ADC watchdog interrupt
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
p_irq	points to hal_adc_irq_struct, struct members refer to Table 3-11. Structure hal_adc_irq_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/ HAL_ERR_ADDRESS

Example:

```

/* enable watchdog interrupt */

hal_adc_dev_struct adc_info;

hal_adc_irq_struct adc_irq_parameter;

void adc_irq_watchdog(void)

{

    /* user defined content */

}

```

```
adc_irq_parameter.adc_watchdog_handle = adc_irq_watchdog;

hal_adc_watchdog_interrupt_enable(&adc_info, &adc_irq_parameter);
```

hal_adc_watchdog_interrupt_disable

The description of hal_adc_watchdog_interrupt_disable is shown as below:

Table 3-44. Function hal_adc_watchdog_interrupt_disable

Function name	hal_adc_watchdog_interrupt_disable
Function prototype	int32_t hal_adc_watchdog_interrupt_disable(hal_adc_dev_struct *adc_dev);
Function descriptions	disable ADC watchdog interrupt
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/ HAL_ERR_ADDRESS

Example:

```
hal_adc_dev_struct adc_info;

hal_adc_watchdog_interrupt_disable(&adc_info, &adc_irq_parameter);
```

hal_adc_watchdog_event_poll

The description of hal_adc_watchdog_event_poll is shown as below:

Table 3-45. Function hal_adc_watchdog_event_poll

Function name	hal_adc_watchdog_event_poll
Function prototype	int32_t hal_adc_watchdog_event_poll(hal_adc_dev_struct *adc_dev, uint32_t timeout_ms);
Function descriptions	polling for ADC watchdog event conversion
Precondition	-
The called functions	hal_sys_basetick_count_get
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
timeout_ms	time out duration in millisecond
Output parameter{out}	
-	-

Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_TIMEOUT

Example:

```
/* polling for ADC inserted sequence conversion */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_watchdog_event_poll(&adc_info, 2);
```

hal_adc_irq

The description of hal_adc_irq is shown as below:

Table 3-46. Function hal_adc_irq

Function name	hal_adc_irq
Function prototype	void hal_adc_irq(hal_adc_dev_struct *adc_dev);
Function descriptions	ADC interrupt handler content function, which is merely used in ADC_CMP_IRQHandler
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the function is used in the relative interrupt routine */
```

```
hal_adc_dev_struct adc_info;
```

```
void ADC_CMP_IRQHandler (void)
```

```
{
    hal_adc_irq(&adc_info);
}
```

hal_adc_irq_handle_set

The description of hal_adc_irq_handle_set is shown as below:

Table 3-47. Function `hal_adc_irq_handle_set`

Function name	<code>hal_adc_irq_handle_set</code>
Function prototype	<code>void hal_adc_irq_handle_set(hal_adc_dev_struct *adc_dev, hal_adc_irq_struct *p_irq);</code>
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_dev</code>	points to <code>hal_adc_dev_struct</code> , struct members refer to Table 3-13. Structure <code>hal_adc_dev_struct</code> .
Input parameter{in}	
<code>p_irq</code>	point to ADC interrupt callback functions structure
<code>hal_irq_handle_cb</code>	points to <code>hal_adc_irq_struct</code> , struct members refer to Table 3-11. Structure <code>hal_adc_irq_struct</code> .
<code>NULL</code>	the structure is NULL
<code>HAL_INTERRUPT_ENABLE_ONLY</code>	The corresponding callback mechanism is out of use, while enable corresponding interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* set user-defined interrupt callback function */

hal_adc_dev_struct adc_info;

hal_adc_irq_struct adc_irq_parameter;

void adc_irq_routine_sequence (void)
{
    /* user defined content */
}

/* set the EOC handle function */

adc_irq_parameter.adc_eoc_handle = adc_irq_routine_sequence;

hal_adc_irq_handle_set(&adc_info, &adc_irq_parameter);

```

`hal_adc_irq_handle_all_reset`

The description of `hal_adc_irq_handle_all_reset` is shown as below:

Table 3-48. Function `hal_adc_irq_handle_all_reset`

Function name	<code>hal_adc_irq_handle_all_reset</code>
Function prototype	<code>void hal_adc_irq_handle_all_reset(hal_adc_dev_struct *adc_dev);</code>
Function descriptions	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_dev</code>	points to <code>hal_adc_dev_struct</code> , struct members refer to Table 3-13. Structure <code>hal_adc_dev_struct</code> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback function */
```

```
hal_adc_dev_struct adc_info;
```

```
hal_adc_irq_handle_all_reset(&adc_info);
```

`hal_adc_routine_value_get`

The description of `hal_adc_routine_value_get` is shown as below:

Table 3-49. Function `hal_adc_routine_value_get`

Function name	<code>hal_adc_routine_value_get</code>
Function prototype	<code>uint16_t hal_adc_routine_value_get(hal_adc_dev_struct *adc_dev);</code>
Function descriptions	get routine sequence conversion result
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_dev</code>	points to <code>hal_adc_dev_struct</code> , struct members refer to Table 3-13. Structure <code>hal_adc_dev_struct</code> .
Output parameter{out}	
-	-
Return value	
<code>Int16_t</code>	the routine sequence conversion result(0-0xFFFF)

Example:

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

```
hal_adc_dev_struct adc_info;
```

```
uint16_t adc_value = 0;
```

```
adc_value = hal_adc_routine_value_get (&adc_info);
```

hal_adc_inserted_value_get

The description of hal_adc_inserted_value_get is shown as below:

Table 3-50. Function hal_adc_inserted_value_get

Function name	hal_adc_inserted_value_get
Function prototype	uint16_t hal_adc_inserted_value_get(hal_adc_dev_struct *adc_dev, uint8_t inschannel_sequence);
Function descriptions	get inserted sequence conversion result
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct .
Input parameter{in}	
inschannel_sequence	ADC inserted channel sequence
<i>ADC_INSERTED_SEQUENCE_x (x=0..3)</i>	ADC inserted channel sequence
Output parameter{out}	
-	-
Return value	
uint16_t	the inserted sequence conversion result(0-0xFFFF)

Example:

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

```
hal_adc_dev_struct adc_info;
```

```
uint16_t adc_value = 0;
```

```
adc_value = hal_adc_inserted_value_get(&adc_info, ADC_INSERTED_SEQUENCE_0);
```

hal_adc_error_get

The description of hal_adc_error_get is shown as below:

Table 3-51. Function hal_adc_error_get

Function name	hal_adc_error_get
Function prototype	uint32_t hal_adc_error_get(hal_adc_dev_struct *adc_dev);
Function descriptions	get ADC error
Precondition	-
The called functions	-

Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct.
Output parameter{out}	
-	-
Return value	
uint32_t	the error state

Example:

```
/* get ADC error */

hal_adc_dev_struct adc_info;

uint32_t adc_error = 0;

adc_error = hal_adc_error_get(&adc_info);
```

hal_adc_state_get

The description of hal_adc_state_get is shown as below:

Table 3-52. Function hal_adc_state_get

Function name	hal_adc_state_get
Function prototype	uint32_t hal_adc_state_get(hal_adc_dev_struct *adc_dev);
Function descriptions	get ADC state
Precondition	-
The called functions	-
Input parameter{in}	
adc_dev	points to hal_adc_dev_struct, struct members refer to Table 3-13. Structure hal_adc_dev_struct.
Output parameter{out}	
-	-
Return value	
uint32_t	the state

Example:

```
/* get ADC state */

hal_adc_dev_struct adc_info;

uint32_t adc_state = 0;

adc_state = hal_adc_state_get (&adc_info);
```

3.3. CEC

Consumer Electronics Control (CEC) belongs to a part of HDMI (High-Definition MultimediaInterface) standard. CEC, as a kind of protocol, provides the advanced control functions of allkinds of audio-visual products in a user environment. The CEC registers are listed in chapter [3.3.1](#), the CEC firmware functions are introduced in chapter [3.3.2](#).

3.3.1. Descriptions of Peripheral registers

CEC registers are listed in the table shown as below:

Table 3-53. CEC Registers

Registers	Descriptions
CEC_CTL	Control register
CEC_CFG	Configuration register
CEC_TDATA	Transmit data register
CEC_RDATA	Receive data register
CEC_INTF	Interrupt flag register
CEC_INTEN	Interrupt enable register

3.3.2. Descriptions of Peripheral functions

CEC registers are listed in the table shown as below:

Table 3-54. CEC firmware function

Function name	Function description
hal_cec_struct_init	initialize CEC structure
hal_cec_init	initialize CEC
hal_cec_deinit	deinitialize CEC device structure and init structure
hal_cec_start	start CEC module function
hal_cec_stop	stop CEC module function
hal_cec_transmit_interrupt	transmit amounts of data by interrupt method
hal_cec_receive_interrupt	receive amounts of data by interrupt method
hal_cec_irq_handle_set	set user-defined interrupt callback function and enable CEC interrupt
hal_cec_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_cec_irq	CEC interrupt handler content function, which is merely used in cec_handler

Enum hal_cec_state_enum

Table 3-55. Enum hal_cec_state_enum

Member name	Function description
HAL_CEC_STATE_NONE	none(default value)
HAL_CEC_STATE_RESET	CEC is not initialized or disabled
HAL_CEC_STATE_READY	CEC is ready
HAL_CEC_STATE_BUSY	CEC is busy
HAL_CEC_STATE_BUSY_RX	CEC is receiveing
HAL_CEC_STATE_BUSY_TX	CEC is transmitting

Enum hal_cec_struct_type_enum

Table 3-56. Enum hal_cec_struct_type_enum

Member name	Function description
HAL_CEC_INIT_STRUCTURE	CEC initialization structure
HAL_CEC_IRQ_STRUCTURE	CEC device interrupt callback function pointer structure
HAL_CEC_DEV_STRUCTURE	CEC device information structtrue

Enum hal_cec_error_enum

Table 3-57. Enum hal_cec_error_enum

Member name	Function description
HAL_CEC_ERROR_NONE	no error
HAL_CEC_ERROR_RO	RX overrun error
HAL_CEC_ERROR_BRE	bit rising error error
HAL_CEC_ERROR_BPSE	short bit period error
HAL_CEC_ERROR_BPLE	long bit period error
HAL_CEC_ERROR_ACK	RX ACK error

_RAE	
HAL_CEC_ERROR _ARBF	arbitration lost
HAL_CEC_ERROR _TU	long bit period error
HAL_CEC_ERROR _TERR	RX ACK error
HAL_CEC_ERROR _TAERR	transmit ACK Error

Structure hal_cec_irq_struct

Table 3-58. Structure hal_cec_irq_struct

Member name	Function description
cec_tx_handle	CEC tx handler function
cec_rx_handle	CEC rx handler function

Structure hal_cec_dev_struct

Table 3-59. Structure hal_cec_dev_struct

Member name	Function description
cec_irq	CEC device interrupt callback function pointer structure
state	CEC state
error_state	CEC error state
* tx_buffer	tx buffer address
tx_count	tx buffer count
*rx_buffer	rx buffer address
rx_count	rx buffer count
*err_callback	CEC error callback fuction
*tx_callback	CEC transmit callback fuction
*rx_callback	CEC receive callback fuction
mutex	mutex locked and unlocked state
*priv	priv data

Structure hal_cec_init_struct

Table 3-60. Structure hal_cec_init_struct

Member name	Function description
signal_free_time	signal free time
reception_bit_timing _tolerance	reception bit timing tolerance
sft_start_option_bit	the SFT start option bit
listen_mode	listen mode enable bit

own_address	own address
bre_stop_receive	whether stop receive message when detected BRE
bre_generate_error	generate an error-bit when detected BRE in singlecast
blpe_generate_error	generate an error-bit when detected BPLE in singlecast
not_generate_error_broadcast	do not generate an error-bit in broadcast message

hal_cec_struct_init

The description of hal_cec_struct_init is shown as below:

Table 3-61. Function hal_cec_struct_init

Function name	hal_cec_struct_init
Function prototype	void hal_cec_struct_init(hal_cec_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the CEC structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	type of structure to be initialized
HAL_CEC_INIT_STRUCTURE	CEC initialization structure
HAL_CEC_IRQ_STRUCTURE	CEC interrupt structure
HAL_CEC_DEV_STRUCTURE	CEC device information structure
Input parameter{in}	
p_struct	point to CEC structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the CEC structure with the default values */

hal_cec_dev_struct cec_info;

hal_cec_struct_init(HAL_CEC_DEV_STRUCTURE, &cec_info);

```

hal_cec_init

The description of hal_cec_init is shown as below:

Table 3-62. Function hal_cec_init

Function name	hal_cec_init
Function prototype	int32_t hal_cec_init(hal_cec_dev_struct*cec_dev, hal_cec_init_struct*cec);
Function descriptions	initialize CEC
Precondition	-
The called functions	-
Input parameter{in}	
cec_dev	points to hal_cec_dev_struct, struct members refer to - Table 3-59. Structure hal_cec_dev_struct
Input parameter{in}	
cec	points to hal_cec_init_struct, struct members refer to - Table 3-60. Structure hal_cec_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* initialize CEC */

hal_cec_init_struct cec_init_parameter;

hal_cec_struct_init(HAL_CEC_INIT_STRUCT, &cec_init_parameter);

hal_cec_struct_init(HAL_CEC_DEV_STRUCT, &cec_info);

cec_init_parameter.signal_free_time = CEC_SFT_PROTOCOL_PERIOD;

cec_init_parameter.reception_bit_timing_tolerance = CEC_STANTARD_RTOL;

cec_init_parameter.sft_start_option_bit = CEC_SFT_START_STAOM;

cec_init_parameter.listen_mode = CEC_PARTIAL_LISTENING_MODE;

cec_init_parameter.own_address = CEC_OWN_ADDRESS2;

cec_init_parameter.bre_stop_receive = CEC_NOT_STOP_RECEPTION;

cec_init_parameter.bre_generate_error = CEC_NO_RISING_PERIOD_ERROR;

cec_init_parameter.blpe_generate_error = CEC_NO_LONG_PERIOD_ERROR;

cec_init_parameter.not_generate_error_broadcast = CEC_GEN_BROADCAST_ERROR;

hal_cec_init(&cec_info,&cec_init_parameter);

```

hal_cec_deinit

The description of hal_cec_deinit is shown as below:

Table 3-63. Function hal_cec_init

Function name	hal_cec_deinit
Function prototype	int32_t hal_cec_deinit(hal_cec_dev_struct *cec_dev);
Function descriptions	deinitialize CEC device structure and init structure
Precondition	-
The called functions	_cec_interrupt_disable
Input parameter{in}	
cec_dev	points to hal_cec_dev_struct, struct members refer to - Table 3-59. Structure hal_cec_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* deinitialize CEC */
hal_cec_dev_struct cec_info;
hal_cec_deinit(&cec_info);
```

hal_cec_start

The description of hal_cec_start is shown as below:

Table 3-64. Function hal_cec_start

Function name	hal_cec_start
Function prototype	int32_t hal_cec_start(hal_cec_dev_struct *cec_dev);
Function descriptions	start CEC module function
Precondition	-
The called functions	-
Input parameter{in}	
cec_dev	points to hal_cec_dev_struct, struct members refer to - Table 3-59. Structure hal_cec_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* start CEC */
hal_cec_dev_struct cec_info;
hal_cec_start(&cec_info);
```

hal_cec_stop

The description of hal_cec_stop is shown as below:

Table 3-65. Function hal_cec_stop

Function name	hal_cec_stop
Function prototype	int32_t hal_cec_stop(hal_cec_dev_struct *cec_dev);
Function descriptions	stop CEC module function
Precondition	-
The called functions	-
Input parameter{in}	
cec_dev	points to hal_cec_dev_struct, struct members refer to - Table 3-59. Structure hal_cec_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* stop CEC */
hal_cec_dev_struct cec_info;
hal_cec_stop(&cec_info);
```

hal_cec_transmit_interrupt

The description of hal_cec_transmit_interrupt is shown as below:

Table 3-66. Function hal_cec_transmit_interrupt

Function name	hal_cec_transmit_interrupt
Function prototype	int32_t hal_cec_transmit_interrupt(hal_cec_dev_struct *cec_dev, uint32_t tx_length, uint8_t *p_buffer, uint8_t header_addr, uint8_t destination_addr, hal_cec_user_cb p_func);
Function descriptions	transmit amounts of data by interrupt method
Precondition	-
The called functions	-
Input parameter{in}	
cec_dev	points to hal_cec_dev_struct, struct members refer to - Table 3-59. Structure hal_cec_dev_struct
Input parameter{in}	
tx_length	length of data to be sent
Input parameter{in}	
p_buffer	pointer to data buffer

Input parameter{in}	
header_addr	header address
Input parameter{in}	
destination_addr	destination address
Input parameter{in}	
p_func	CEC transmit interrupt callback function structure
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* transmit amounts of data by interrupt method */

cec_info.err_callback = HAL_CEC_ErrCallback;

uint32_t Tx_Size      = 0x0;

uint8_t Tx_Burrer[2] = {0xa5, 0x5a};

__IO uint8_t txstatus = 0U;

void HAL_CEC_TxCpltCallback(hal_cec_dev_struct *cec_dev)
{
    /* End of transmission */

    txstatus = 1;
}

hal_cec_transmit_interrupt(&cec_info, Tx_Size, Tx_Burrer, 1, 2, HAL_CEC_TxCpltCallback);

```

hal_cec_receive_interrupt

The description of hal_cec_receive_interrupt is shown as below:

Table 3-67. Function hal_cec_receive_interrupt

Function name	hal_cec_receive_interrupt
Function prototype	int32_t hal_cec_receive_interrupt(hal_cec_dev_struct *cec_dev, uint8_t *p_buffer, hal_cec_user_cb p_func);
Function descriptions	receive amounts of data by interrupt method
Precondition	-
The called functions	-
Input parameter{in}	
cec_dev	points to hal_cec_dev_struct, struct members refer to - Table 3-59. Structure hal_cec_dev_struct

Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
p_func	CEC transmit interrupt callback function structure
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* receive amounts of data by interrupt method */

cec_info.err_callback = HAL_CEC_ErrCallback;

uint8_t Rx_Burrer[CEC_MAX_BUFFER_LEN] = {0};

__IO uint8_t rxstatus = 0U;

void HAL_CEC_RxCpltCallback(hal_cec_dev_struct *cec_dev)
{
    rxstatus = 1;
}

hal_cec_receive_interrupt(&cec_info, Rx_Burrer, HAL_CEC_RxCpltCallback);

```

hal_cec_irq_handle_set

The description of hal_cec_irq_handle_set is shown as below:

Table 3-68. Function hal_cec_irq_handle_set

Function name	hal_cec_irq_handle_set
Function prototype	int32_t hal_cec_irq_handle_set(hal_cec_dev_struct *cec_dev, hal_cec_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
cec_dev	points to hal_cec_dev_struct, struct members refer to - Table 3-59. Structure hal_cec_dev_struct
Input parameter{in}	
p_irq	Structure for CEC device interrupt callback function pointer structure, the structure members can refer to Table 3-58. Structure hal_cec_irq_struct

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* set user-defined interrupt callback function */
```

```
hal_cec_dev_struct cec_info;
```

```
hal_cec_irq_struct cec_irq;
```

```
cec_irq.cec_tx_handle = cec_tx_func;
```

```
hal_cec_irq_handle_set(&cec_info,&cec_irq);
```

hal_cec_irq_handle_all_reset

The description of hal_cec_irq_handle_all_reset is shown as below:

Table 3-69. Function hal_cec_irq_handle_all_reset

Function name	hal_cec_irq_handle_all_reset
Function prototype	void hal_cec_irq_handle_all_reset(hal_cec_dev_struct *cec_dev);
Function descriptions	reset user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
cec_dev	points to hal_cec_dev_struct, struct members refer to - Table 3-59. Structure hal_cec_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* reset all user-defined interrupt callback function */
```

```
hal_cec_dev_struct cec_info;
```

```
hal_cec_irq_handle_all_reset(&cec_info);
```

hal_cec_irq

The description of hal_cec_irq is shown as below:

Table 3-70. Function hal_cec_irq

Function name	hal_cec_irq
Function prototype	void hal_cec_irq(hal_cec_dev_struct *cec_dev);
Function descriptions	CEC interrupt handler content function, which is merely used in cec_handler
Precondition	-
The called functions	-
Input parameter{in}	
cec_dev	points to hal_cec_dev_struct, struct members refer to - Table 3-59. Structure hal_cec_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* the function is used in the relative interrupt routine */
```

```
hal_cec_dev_struct cec_info;
```

```
CEC_IRQHandler (void)
```

```
{
    hal_cec_irq(&cec_info);
}
```

3.4. CMP

The general purpose comparator can work either standalone(all terminal are available on I/Os) or together with the timers. It could be used to wake up the MCU from low-power mode by an analog signal, provide a trigger source when an analog signal is in a certain condition, achieves some current control by working together with a PWM output of a timer. The CMP registers are listed in chapter [3.4.1](#), the CMP firmware functions are introduced in chapter [3.4.2](#).

3.4.1. Descriptions of Peripheral registers

CMP registers are listed in the table shown as below:

Table 3-71. CMP Registers

Registers	Descriptions
CMP_CS	Control/Status register

3.4.2. Descriptions of Peripheral functions

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

Table 3-72. CMP firmware function

Function name	Function description
hal_cmp_struct_init	comparator init struct deinitialize
hal_cmp_deinit	deinitialize comparator device structure and reset the peripheral
hal_cmp_init	initialize the comparator
hal_cmp_start	start cmp module function with event or not
hal_cmp_stop	stop cmp module function with event or not
hal_cmp_start_interrupt	start the cmp module function with interrupt
hal_cmp_stop_interrupt	stop the cmp module function with interrupt
hal_cmp_irq_handle_set	set the comparator external trigger interrupt callback function
hal_cmp_irq_handle_all_reset	reset the comparator external trigger interrupt callback function
hal_cmp_irq	cmp external trigger interrupt handle function
hal_cmp_lock	lock the comparator
hal_cmp_output_level_get	get output level of comparator
hal_cmp_state_get	get the state of comparator

Enum hal_cmp_state_enum

Table 3-73. Enum hal_cmp_state_enum

Member name	Function description
HAL_CMP_STATE_NONE	none(default value)
HAL_CMP_STATE_RESET	CMP is not initialized or disabled
HAL_CMP_STATE_BUSY	CMP is busy
HAL_CMP_STATE_TIMEOUT	CMP timeout occurs
HAL_CMP_STATE_ERROR	CMP is error
HAL_CMP_STATE_READY	CMP is ready

Enum hal_cmp_struct_type_enum

Table 3-74. Enum hal_cmp_struct_type_enum

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

HAL_CMP_INIT_STRUCTURE	CMP initialization structure
HAL_CMP_DEVICE_INFORMATION_STRUCTURE	CMP device information structure

Enum hal_cmp_noninverting_input_enum

Table 3-75. Enum hal_cmp_noninverting_input_enum

Member name	Function description
CMP0_IP_PA1	PA1 as noninverting input, only for CMP0
CMP0_IP_PA4_SWITCH_CLOSE	PA4 as noninverting input, switch between PA1 and PA4 closed, only for CMP0
CMP1_IP_PA3	PA3 as noninverting input, only for CMP1
CMP1_IP_OF_CMP0	select input of CMP0, only for CMP1(window comparator mode)

Enum hal_cmp_exti_type_enum

Table 3-76. Enum hal_cmp_exti_type_enum

Member name	Function description
CMP_EXTI_NONE	No EXTI event/interrupt enabled
CMP_EXTI_INTERRUPT_RISING	Enable EXTI interrupt for rising edge
CMP_EXTI_INTERRUPT_FALLING	Enable EXTI interrupt for falling edge
CMP_EXTI_INTERRUPT_BOTH	Enable EXTI interrupt for both edge
CMP_EXTI_EVENT_RISING	Enable EXTI event for rising edge
CMP_EXTI_EVENT_FALLING	Enable EXTI event for falling edge
CMP_EXTI_EVENT_BOTH	Enable EXTI event for both edge

Structure hal_cmp_init_struct

Table 3-77. Structure hal_cmp_init_struct

Member name	Function description
inverting_input	the comparator inverting input selection
noninverting_input	the comparator noninverting input selection
outputsel	the comparator output selection
polarity	the comparator output polarity
mode	the comparator operating mode

hysteresis	the comparator hysteresis level
exti_type	the comparator external trigger mode

Structure hal_cmp_dev_struct

Table 3-78. Structure hal_cmp_dev_struct

Member name	Function description
periph	the module instance, CMP0 or CMP1
cmp_irq	the comparator interrupt callback struct
state	the comparator state
output_level	the comparator output level

Structure hal_cmp_irq_struct

Table 3-79. Structure hal_cmp_irq_struct

Member name	Function description
output_changed_handle	the comparator output level changed interrupt callback

hal_cmp_struct_init

The description of hal_cmp_struct_init is shown as below:

Table 3-80. Function hal_cmp_struct_init

Function name	hal_cmp_struct_init
Function prototype	int32_t hal_cmp_struct_init(hal_cmp_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	comparator init struct deinitialize
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	the type of the structure
HAL_CMP_INIT_STRU CT	the comparator initialization structure
HAL_CMP_DEV_STRU CT	the comparator device structure
Input parameter{in}	
p_struct	the pointer of the structure
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```
/* CMP initializing structure initialize */
```

```
hal_cmp_init_struct cmp_init_parameter;
```

```
hal_cmp_struct_init(HAL_CMP_INIT_STRUCT, &cmp_init_parameter);
```

hal_cmp_deinit

The description of hal_cmp_deinit is shown as below:

Table 3-81. Function hal_cmp_deinit

Function name	hal_cmp_deinit
Function prototype	int32_t hal_cmp_deinit(hal_cmp_dev_struct *cmp_dev);
Function descriptions	deinitialize comparator device structure and reset the peripheral
Precondition	-
The called functions	hal_exti_internal_deinit
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* CMP device structure initialize */
```

```
hal_cmp_dev_struct cmp_dev_parameter;
```

```
hal_cmp_deinit(&cmp_dev_parameter);
```

hal_cmp_init

The description of hal_cmp_init is shown as below:

Table 3-82. Function hal_cmp_init

Function name	hal_cmp_init
Function prototype	int32_t hal_cmp_init(hal_cmp_dev_struct *cmp_dev, hal_cmp_init_struct *p_init);
Function descriptions	initialize the comparator
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Input parameter{in}	

periph	specify the module instance, CMP0 or CMP1
Input parameter{in}	
p_init	the pointer of comparator init information structure, details refer to Table 3-77. Structure hal_cmp_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* CMP initialize */

hal_cmp_dev_struct cmp_dev_parameter;

hal_cmp_init_struct cmp_init_parameter;

hal_cmp_init(&cmp_dev_parameter, CMP0, &cmp_init_parameter);
```

hal_cmp_start

The description of hal_cmp_start is shown as below:

Table 3-83. Function hal_cmp_start

Function name	hal_cmp_start
Function prototype	int32_t hal_cmp_start(hal_cmp_dev_struct *cmp_dev);
Function descriptions	start cmp module function with event or not
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start the CMP module function */

hal_cmp_dev_struct cmp_dev_parameter;

hal_cmp_start(&cmp_dev_parameter);
```

hal_cmp_stop

The description of hal_cmp_stop is shown as below:

Table 3-84. Function hal_cmp_stop

Function name	hal_cmp_stop
Function prototype	int32_t hal_cmp_stop(hal_cmp_dev_struct *cmp_dev);
Function descriptions	stop cmp module function with event or not
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* stop the CMP module function */
hal_cmp_dev_struct cmp_dev_parameter;
hal_cmp_stop(&cmp_dev_parameter);
```

hal_cmp_start_interrupt

The description of hal_cmp_start_interrupt is shown as below:

Table 3-85. Function hal_cmp_start_interrupt

Function name	hal_cmp_start_interrupt
Function prototype	int32_t hal_cmp_start_interrupt(hal_cmp_dev_struct *cmp_dev, hal_cmp_irq_struct *p_irq)
Function descriptions	start the comparator with interrupt
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Input parameter{in}	
p_irq	the comparator interrupt callback struct pointer, details refer to Table 3-79. Structure hal_cmp_irq_struct
Output parameter{out}	
-	-

Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start the CMP module function in interrupt mode */

hal_cmp_dev_struct cmp_dev_parameter;

hal_cmp_irq_struct cmp_irq;

void xxx_function(void)
{
    /* user defined content */
}

cmp_irq.output_changed_handle = xxx_function;

hal_cmp_start_interrupt(&cmp_dev_parameter, &cmp_irq);
```

hal_cmp_stop_interrupt

The description of hal_cmp_stop_interrupt is shown as below:

Table 3-86. Function hal_cmp_stop_interrupt

Function name	hal_cmp_stop_interrupt
Function prototype	int32_t hal_cmp_stop_interrupt(hal_cmp_dev_struct *cmp_dev);
Function descriptions	stop the comparator with interrupt
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* stop the CMP module function and interrupt */

hal_cmp_dev_struct cmp_dev_parameter;

hal_cmp_stop_interrupt(&cmp_dev_parameter);
```

hal_cmp_irq_handle_set

The description of hal_cmp_irq_handle_set is shown as below:

Table 3-87. Function hal_cmp_irq_handle_set

Function name	hal_cmp_irq_handle_set
Function prototype	void hal_cmp_irq_handle_set(hal_cmp_dev_struct *cmp_dev, hal_cmp_irq_handle_cb irq_handler);
Function descriptions	set the comparator external trigger interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Input parameter{in}	
p_irq	the comparator interrupt callback struct pointer, details refer to Table 3-79. Structure hal_cmp_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* set the comparator external trigger interrupt callback function */

hal_cmp_dev_struct cmp_dev_parameter;

hal_cmp_irq_struct cmp_irq;

void xxx_function(void)
{
    /* user defined content */
}

cmp_irq.output_changed_handle = xxx_function;

hal_cmp_irq_handle_set(&cmp_dev_parameter, &cmp_irq);

```

hal_cmp_irq_handle_all_reset

The description of hal_cmp_irq_handle_all_reset is shown as below:

Table 3-88. Function hal_cmp_irq_handle_all_reset

Function name	hal_cmp_irq_handle_all_reset
Function prototype	void hal_cmp_irq_handle_all_reset(hal_cmp_dev_struct *cmp_dev);

Function descriptions	reset the comparator external trigger interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the comparator external trigger interrupt callback function */
hal_cmp_dev_struct cmp_dev_parameter;
hal_cmp_irq_handle_all_reset(&cmp_dev_parameter);
```

hal_cmp_irq

The description of hal_cmp_irq is shown as below:

Table 3-89. Function hal_cmp_irq

Function name	hal_cmp_irq
Function prototype	void hal_cmp_irq(hal_cmp_dev_struct *cmp_dev);
Function descriptions	cmp external trigger interrupt handler content function
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* cmp external trigger interrupt handler content function */
hal_cmp_dev_struct cmp_dev_parameter;
hal_cmp_irq(&cmp_dev_parameter);
```

hal_cmp_lock

The description of hal_cmp_lock is shown as below:

Table 3-90. Function hal_cmp_lock

Function name	hal_cmp_lock
Function prototype	int32_t hal_cmp_lock(hal_cmp_dev_struct *cmp_dev);
Function descriptions	lock the comparator
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_ALREADY_DONE

Example:

```
/* lock the comparator */
hal_cmp_dev_struct cmp_dev_parameter;
hal_cmp_lock(&cmp_dev_parameter);
```

hal_cmp_output_level_get

The description of hal_cmp_output_level_get is shown as below:

Table 3-91. Function hal_cmp_output_level_get

Function name	hal_cmp_output_level_get
Function prototype	uint32_t hal_cmp_output_level_get(hal_cmp_dev_struct *cmp_dev);
Function descriptions	get output level of comparator
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Output parameter{out}	
-	-
Return value	
uint32_t	Output level, 0 or 1

Example:

```
/* get output level of comparator */
```

```

hal_cmp_dev_struct cmp_dev_parameter;

uint32_t output_level;

output_level = hal_cmp_output_level_get (&cmp_dev_parameter);

```

hal_cmp_state_get

The description of hal_cmp_state_get is shown as below:

Table 3-92. Function hal_cmp_state_get

Function name	hal_cmp_state_get
Function prototype	hal_cmp_state_enum hal_cmp_state_get(hal_cmp_dev_struct *cmp_dev);
Function descriptions	get the state of comparator
Precondition	-
The called functions	-
Input parameter{in}	
cmp_dev	the pointer of comparator device information structure, details refer to Table 3-78. Structure hal_cmp_dev_struct
Output parameter{out}	
-	-
Return value	
hal_cmp_state_enum	CMP device state

Example:

```

/* get output level of comparator */

hal_cmp_dev_struct cmp_dev_parameter;

hal_cmp_state_enum cmp_state;

cmp_state = hal_cmp_state_get(&cmp_dev_parameter);

```

3.5. 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 [3.5.1](#), the CRC firmware functions are introduced in chapter [3.5.2](#).

3.5.1. Descriptions of Peripheral registers

CRC registers are listed in the table shown as below:

Table 3-93. CRC Registers

Registers	Descriptions
CRC_DATA	CRC data register

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

3.5.2. Descriptions of Peripheral functions

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

Table 3-94. CRC firmware function

Function name	Function description
hal_crc_struct_init	initialize the CRC structure with the default values
hal_crc_init	initialize CRC
hal_crc_deinit	deinitialize CRC calculation unit
hal_crc_single_data_calculate	CRC calculate single data
hal_crc_block_data_calculate	CRC calculate a data array

Enum hal_crc_state_enum

Table 3-95. Enum hal_crc_state_enum

Member name	Function description
HAL_CRC_STATE_NONE	NONE(default value)
HAL_CRC_STATE_RESET	RESET
HAL_CRC_STATE_BUSY	BUSY
HAL_CRC_STATE_TIMEOUT	TIMEOUT
HAL_CRC_STATE_ERROR	ERROR
HAL_CRC_STATE_READY	READY

Enum hal_crc_struct_type_enum

Table 3-96. Enum hal_crc_struct_type_enum

Member name	Function description
HAL_CRC_INIT_STRUCT	CRC initialization structure
HAL_CRC_DEV_STRUCT	CRC device information structure

Structure hal_crc_dev_struct

Table 3-97. Structure hal_crc_dev_struct

Member name	Function description
state	CRC state
mutex	mutex locked and unlocked state
*priv	priv data

Structure hal_crc_init_struct

Table 3-98. Structure hal_crc_init_struct

Member name	Function description
polynomial_value	polynomial value
init_data_value	init data value
input_data_reverse_mode	input data reverse mode
output_data_reverse	output data reverse
polynomial_size	polynomial size

hal_crc_struct_init

The description of hal_crc_struct_init is shown as below:

Table 3-99. Function hal_crc_struct_init

Function name	hal_crc_struct_init
Function prototype	int32_t hal_crc_struct_init(hal_crc_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the CRC structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	type of structure to be initialized
HAL_CRC_INIT_STRUCTURE	CRC initialization structure
HAL_CRC_DEV_STRUCTURE	CRC device information structrue
Input parameter{in}	
p_struct	point to the structure to be deinitialized
Input parameter{in}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* initialize the CRC structure with the default values */
hal_crc_dev_struct crc_info;
hal_crc_struct_init(HAL_CRC_DEV_STRUCTURE, &crc_info);
```

hal_crc_init

The description of hal_crc_init is shown as below:

Table 3-100. Function hal_crc_init

Function name	hal_crc_init
Function prototype	int32_t hal_crc_init(hal_crc_dev_struct *crc_dev, hal_crc_init_struct *p_crc_init);
Function descriptions	Initialize the CRC
Precondition	-
The called functions	-
Input parameter{in}	
crc_dev	CRC device information structue, the structure members can refer to Table 3-97. Structure hal_crc_dev_struct .
Input parameter{in}	
p_crc_init	points to hal_crc_init_struct, the struct members refer to Table 3-98. Structure hal_crc_init_struct .
Onput parameter{in}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* initialize CRC */
hal_crc_init_struct crc_init_parameter;
hal_crc_dev_struct crc_info;
hal_crc_struct_init(HAL_CRC_INIT_STRUCT, &crc_init_parameter);
hal_crc_struct_init(HAL_CRC_DEV_STRUCT, &crc_info);
crc_init_parameter.polynomial_value = 0x04C11DB7;
crc_init_parameter.init_data_value = 0xFFFFFFFF;
crc_init_parameter.input_data_reverse_mode = CRC_INPUT_REVERSE_NONE;
crc_init_parameter.output_data_reverse = CRC_OUTPUT_REVERSE_DISABLE;
crc_init_parameter.polynomial_size = CRC_POLYNOMIAL_SIZE_32BIT;
hal_crc_init(&crc_info,&crc_init_parameter);

```

hal_crc_deinit

The description of hal_crc_deinit is shown as below:

Table 3-101. Function hal_crc_deinit

Function name	hal_crc_deinit
Function prototype	int32_t hal_crc_deinit(hal_crc_dev_struct *crc_dev);
Function descriptions	deinitialize CRC
Precondition	-
The called functions	
Input parameter{in}	
crc_dev	CRC device information structue, the struct members refer to Table 3-97.

	<u>Structure <i>hal_crc_dev_struct</i></u>
Onput parameter{in}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* deinitialize CRC */
```

```
hal_crc_dev_struct crc_info;
```

```
hal_crc_struct_init(HAL_CRC_DEV_STRUCT, &crc_info);
```

```
hal_crc_deinit(&crc_info);
```

hal_crc_single_data_calculate

The description of hal_crc_single_data_calculate is shown as below:

Table 3-102. Function hal_crc_single_data_calculate

Function name	hal_crc_single_data_calculate
Function prototype	uint32_t hal_crc_single_data_calculate(hal_crc_dev_struct *crc_dev, uint32_t sdata, uint8_t data_format);
Function descriptions	CRC calculate single data
Precondition	-
The called functions	
Input parameter{in}	
crc_dev	CRC device information structrue, the struct members refer to <u>Table 3-97.</u> <u>Structure <i>hal_crc_dev_struct</i></u>
Input parameter{in}	
sdata	specify input data
Input parameter{in}	
data_format	input data format
INPUT_FORMAT_WORD	input data in word format
INPUT_FORMAT_HALFWORD	input data in half-word format
INPUT_FORMAT_BYTE	input data in byte format
Output parameter{in}	
-	-
Return value	
int32_t	CRC calculate value

Example:

```

/* CRC calculate a 32-bit data */
uint32_t valcrc = 0;
uint32_t val_32 = 0xabcd1234;
hal_crc_dev_struct crc_info;
hal_crc_struct_init(HAL_CRC_DEV_STRUCT, &crc_info);
valcrc = hal_crc_single_data_calculate(&crc_info, val_32, INPUT_FORMAT_WORD);

```

hal_crc_block_data_calculate

The description of hal_crc_block_data_calculate is shown as below:

Table 3-103. Function hal_crc_block_data_calculate

Function name	hal_crc_block_data_calculate
Function prototype	uint32_t hal_crc_block_data_calculate(hal_crc_dev_struct *crc_dev, void *array, uint32_t size, uint8_t data_format)
Function descriptions	CRC calculate a data array
Precondition	-
The called functions	
Input parameter{in}	
crc_dev	CRC device information structtrue, the struct members refer to Table 3-97. Structure hal_crc_dev_struct .
Input parameter{in}	
array	pointer to the input data array
Input parameter{in}	
size	size of the array
Input parameter{in}	
data_format	input data format
INPUT_FORMAT_WORD	input data in word format
INPUT_FORMAT_HALFWORD	input data in half-word format
INPUT_FORMAT_BYTE	input data in byte format
Output parameter{in}	
-	-
Return value	
int32_t	CRC calculate value

Example:

```

/*calculate the 32-bit CRC value of a 32-bit data array */
uint32_t valcrc = 0;
uint32_t val_32[4] = {0xabcd1234, 0x3456cdef, 0x789a1524, 0x12346792};
hal_crc_dev_struct crc_info;

```

```

hal_crc_struct_init(HAL_CRC_DEV_STRUCT, &crc_info);
valcrc              =      hal_crc_block_data_calculate(&crc_info, val_32,      val_size,
INPUT_FORMAT_WORD);

```

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

3.6.1. Descriptions of Peripheral registers

CTC registers are listed in the table shown as below:

Table 3-104. 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

3.6.2. Descriptions of Peripheral functions

CTC registers are listed in the table shown as below:

Table 3-105. CTC firmware function

Function name	Function description
hal_ctc_deinit	reset the CTC peripheral
hal_ctc_init	initialize CTC init structure
hal_ctc_irc48m_trim_value_config	configure the IRC48M trim value
hal_ctc_struct_init	initialize the CTC device structure with the default values
hal_ctc_start	start CTC peripheral
hal_ctc_stop	stop CTC peripheral
hal_ctc_irq	CTC interrupt handler content function, which is merely used in ctc_handler
hal_ctc_irq_handle_set	set user-defined interrupt callback function
hal_ctc_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_ctc_start_interrupt	start CTC with interrupt
hal_ctc_stop_interrupt	stop CTC with interrupt

Enum hal_ctc_struct_type_enum

Table 3-106. Enum hal_ctc_struct_type_enum

Member name	Function description
HAL_CTC_INIT_STRUCT	CTC initialization structure
HAL_CTC_IRQ_STRUCT	CTC initialization IRQ structure
HAL_CTC_DEV_STRUCT	CTC initialization device structure

Enum hal_ctc_error_enum

Table 3-107. Enum hal_ctc_error_enum

Member name	Function description
HAL_CTC_ERROR_NONE	no error
HAL_CTC_ERROR_SYSTEM	CTC internal error: if problem of clocking, enable/disable, wrong state
HAL_CTC_ERROR_CONFIG	configuration error occurred

Enum hal_ctc_state_enum

Table 3-108. Enum hal_ctc_state_enum

Member name	Function description
HAL_CTC_STATE_NONE	NONE
HAL_CTC_STATE_RESET	RESET
HAL_CTC_STATE_BUSY	BUSY
HAL_CTC_STATE_TIMEOUT	TIMEOUT
HAL_CTC_STATE_ERROR	ERROR
HAL_CTC_STATE_READY	READY

Structure hal_ctc_irq_struct

Table 3-109. Structure hal_ctc_irq_struct

Member name	Function description
ctc_ckok_handle	CTC clock trim OK handler function
ctc_ckwarn_handle	CTC clock trim warning handler function
ctc_err_handle	CTC error interrupt handler function
ctc_eref_handle	CTC expect reference handler function

Structure hal_ctc_init_struct

Table 3-110. Structure hal_ctc_init_struct

Member name	Function description
cal_style	calculation mode
source	reference source

polarity	reference signal source polarity
prescaler	reference signal source prescaler
limit_value	clock trim base limit value
reload_value	CTC counter reload value
frequency	reference signal source frequency

Structure hal_ctc_dev_struct

Table 3-111. Structure hal_ctc_dev_struct

Member name	Function description
ctc_irq	CTC device interrupt callback function pointer structure
error_state	CTC error state
state	CTC state
mutex	mutex locked and unlocked state
*priv	priv data

hal_ctc_deinit

The description of hal_ctc_deinit is shown as below:

Table 3-112. Function hal_ctc_deinit

Function name	hal_ctc_deinit
Function prototype	int32_t hal_ctc_deinit(hal_ctc_dev_struct *ctc_dev)
Function descriptions	reset CTC peripheral
Precondition	-
The called functions	hal_rcu_periph_reset_enable / hal_rcu_periph_reset_disable
Input parameter{in}	
ctc_dev	point to hal_ctc_dev_struct, the structure members can refer to Table 3-111. Structure hal_ctc_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* reset CTC peripheral */
hal_ctc_dev_struct ctc_info;
hal_ctc_deinit(&ctc_info);
```

hal_ctc_init

The description of hal_ctc_init is shown as below:

Table 3-113. Function `hal_ctc_init`

Function name	<code>hal_ctc_init</code>
Function prototype	<code>int32_t hal_ctc_init(hal_ctc_dev_struct *ctc_dev, hal_ctc_init_struct *ctc_init)</code>
Function descriptions	initialize CTC init structure
Precondition	-
The called functions	-
Input parameter{in}	
<code>ctc_dev</code>	point to <code>hal_ctc_dev_struct</code> , the structure members can refer to Table 3-111. Structure <code>hal_ctc_dev_struct</code>
Output parameter{out}	
<code>ctc_init</code>	point to <code>hal_ctc_init_struct</code> , the structure members can refer to Table 3-110. Structure <code>hal_ctc_init_struct</code>
Return value	
<code>int32_t</code>	<code>HAL_ERR_ADDRESS</code> , <code>HAL_ERR_NONE</code>

Example:

```

/* initialize CTC init structure */

hal_ctc_dev_struct ctc_info;

hal_ctc_init_struct ctc_init_parameter;

hal_ctc_struct_init(HAL_CTC_INIT_STRUCT, &ctc_init_parameter);

ctc_init_parameter.source = CTC_REFSOURCE_LXTAL;

ctc_init_parameter.cal_style = CTC_VALUE_AUTO_CAL;

ctc_init_parameter.polarity = CTC_REFSOURCE_POLARITY_FALLING;

ctc_init_parameter.prescaler = CTC_REFSOURCE_PSC_OFF;

hal_ctc_init(&ctc_info,&ctc_init_parameter);

```

`hal_ctc_irc48m_trim_value_config`

The description of `hal_ctc_irc48m_trim_value_config` is shown as below:

Table 3-114. Function `hal_ctc_irc48m_trim_value_config`

Function name	<code>hal_ctc_irc48m_trim_value_config</code>
Function prototype	<code>void hal_ctc_irc48m_trim_value_config(hal_ctc_dev_struct *ctc_dev, uint8_t trim_value)</code>
Function descriptions	configure the IRC48M trim value
Precondition	-
The called functions	-
Input parameter{in}	

ctc_dev	point to hal_ctc_dev_struct, the structure members can refer to Table 3-111. Structure hal_ctc_dev_struct
trim_value	6-bit IRC48M trim value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the IRC48M trim value */
hal_ctc_dev_struct ctc_info;
hal_ctc_irc48m_trim_value_config (&ctc_info, 0x30);
```

hal_ctc_struct_init

The description of hal_ctc_struct_init is shown as below:

Table 3-115. Function hal_ctc_struct_init

Function name	hal_ctc_struct_init
Function prototype	void hal_ctc_struct_init(hal_ctc_struct_type_enum hal_struct_type, void *p_struct)
Function descriptions	initialize the CTC device structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	point to hal_ctc_struct_type_enum, the enum members can refer to Table 3-106. Enum hal_ctc_struct_type_enum
Output parameter{out}	
p_struct	point to hal_ctc_dev_struct, the structure members can refer to Table 3-111. Structure hal_ctc_dev_struct
Return value	
-	-

Example:

```
/* initialize the CTC device structure with the default values */
hal_ctc_dev_struct ctc_info;
hal_ctc_struct_init(HAL_CTC_DEV_STRUCT, &ctc_info);
```

hal_ctc_start

The description of hal_ctc_start is shown as below:

Table 3-116. Function `hal_ctc_start`

Function name	<code>hal_ctc_start</code>
Function prototype	<code>int32_t hal_ctc_start(hal_ctc_dev_struct *ctc_dev)</code>
Function descriptions	start CTC module function
Precondition	-
The called functions	-
Input parameter{in}	
<code>ctc_dev</code>	point to <code>hal_ctc_dev_struct</code> , the structure members can refer to Table 3-111. Structure <code>hal_ctc_dev_struct</code>
Output parameter{out}	
-	-
Return value	
<code>int32_t</code>	<code>HAL_ERR_ADDRESS</code> , <code>HAL_ERR_NONE</code>

Example:

```
/* start CTC module function */
```

```
hal_ctc_dev_struct ctc_info;
```

```
hal_ctc_start(&ctc_info);
```

`hal_ctc_stop`

The description of `hal_ctc_stop` is shown as below:

Table 3-117. Function `hal_ctc_stop`

Function name	<code>hal_ctc_stop</code>
Function prototype	<code>int32_t hal_ctc_stop(hal_ctc_dev_struct *ctc_dev)</code>
Function descriptions	stop CTC module function
Precondition	-
The called functions	-
Input parameter{in}	
<code>ctc_dev</code>	point to <code>hal_ctc_dev_struct</code> , the structure members can refer to Table 3-111. Structure <code>hal_ctc_dev_struct</code>
Output parameter{out}	
-	-
Return value	
<code>int32_t</code>	<code>HAL_ERR_ADDRESS</code> , <code>HAL_ERR_NONE</code>

Example:

```
/* stop CTC module function */
```

```
hal_ctc_dev_struct ctc_info;
```

```
hal_ctc_stop(&ctc_info);
```


hal_ctc_irq

The description of hal_ctc_irq is shown as below:

Table 3-118. Function hal_ctc_irq

Function name	hal_ctc_irq
Function prototype	void hal_ctc_irq(hal_ctc_dev_struct *ctc_dev)
Function descriptions	CTC interrupt handler content function, which is merely used in ctc_handler
Precondition	-
The called functions	-
Input parameter{in}	
ctc_dev	point to hal_ctc_dev_struct, the structure members can refer to Table 3-111. Structure hal_ctc_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* CTC interrupt handler content function */
```

```
hal_ctc_dev_struct ctc_info;
```

```
hal_ctc_irq(&ctc_info);
```

hal_ctc_irq_handle_set

The description of hal_ctc_irq_handle_set is shown as below:

Table 3-119. Function hal_ctc_irq_handle_set

Function name	hal_ctc_irq_handle_set
Function prototype	void hal_ctc_irq_handle_set(hal_ctc_dev_struct *ctc_dev, hal_ctc_irq_struct *p_irq)
Function descriptions	set user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
ctc_dev	point to hal_ctc_dev_struct, the structure members can refer to Table 3-111. Structure hal_ctc_dev_struct
p_irq	Structure for PMU device interrupt callback function pointer structure, the structure members can refer to Table 3-109. Structure hal_ctc_irq_struct
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* set user-defined interrupt callback function */
```

```
hal_ctc_dev_struct ctc_info;
```

```
hal_ctc_irq_struct ctc_irq;
```

```
void ctc_irq_test(void *p);
```

```
ctc_irq.ctc_ckok_handle = ctc_irq_test;
```

```
hal_ctc_irq_handle_set(&ctc_info, &ctc_irq);
```

hal_ctc_irq_handle_all_reset

The description of hal_ctc_irq_handle_all_reset is shown as below:

Table 3-120. Function hal_ctc_irq_handle_all_reset

Function name	hal_ctc_irq_handle_all_reset
Function prototype	void hal_ctc_irq_handle_all_reset(hal_ctc_dev_struct *ctc_dev)
Function descriptions	reset all user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
ctc_dev	point to hal_ctc_dev_struct, the structure members can refer to Table 3-111. Structure hal_ctc_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset user-defined interrupt callback function */
```

```
hal_ctc_dev_struct ctc_info;
```

```
hal_ctc_irq_handle_reset(&ctc_info);
```

hal_ctc_start_interrupt

The description of hal_ctc_start_interrupt is shown as below:

Table 3-121. Function hal_ctc_start_interrupt

Function name	hal_ctc_start_interrupt
---------------	-------------------------

Function prototype	int32_t hal_ctc_start_interrupt(hal_ctc_dev_struct *ctc_dev, hal_ctc_irq_struct *p_irq)
Function descriptions	start CTC with interrupt
Precondition	-
The called functions	-
Input parameter{in}	
ctc_dev	point to hal_ctc_dev_struct, the structure members can refer to Table 3-111. Structure hal_ctc_dev_struct
p_irq	the callback handler of CTC interrupt
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* start CTC with interrupt */
hal_ctc_dev_struct ctc_info;
hal_ctc_irq_struct ctc_irq;
void ctc_irq_test(void *p);
ctc_irq.ctc_ckok_handle = ctc_irq_test;
hal_ctc_start_interrupt(&ctc_info, &ctc_irq);

```

hal_ctc_stop_interrupt

The description of hal_ctc_stop_interrupt is shown as below:

Table 3-122. Function hal_ctc_stop_interrupt

Function name	hal_ctc_stop_interrupt
Function prototype	int32_t hal_ctc_stop_interrupt(hal_ctc_dev_struct *ctc_dev)
Function descriptions	stop CTC with interrupt
Precondition	-
The called functions	-
Input parameter{in}	
ctc_dev	point to hal_ctc_dev_struct, the structure members can refer to Table 3-111. Structure hal_ctc_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* stop CTC with interrupt */
```

```
hal_ctc_dev_struct ctc_info;
```

```
hal_ctc_stop_interrupt(&ctc_info);
```

3.7. 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 [3.7.1](#), the DAC firmware functions are introduced in chapter [3.7.2](#). GD32F330 and GD32F310 do not have DAC peripheral.

3.7.1. Descriptions of Peripheral registers

DAC registers are listed in the table shown as below:

Table 3-123. DAC Registers

Registers	Descriptions
DAC_CTL	DAC control register
DAC_SWT	DAC software trigger register
DAC_R12DH	DAC 12-bit right-aligned data holding register
DAC_L12DH	DAC 12-bit left-aligned data holding register
DAC_R8DH	DAC 8-bit right-aligned data holding register
DAC_DO	DAC data output register
DAC_STAT	DAC status register

3.7.2. Descriptions of Peripheral functions

DAC registers are listed in the table shown as below:

Table 3-124. DAC firmware function

Function name	Function description
hal_dac_init	initialize DAC
hal_dac_deinit	deinitialize DAC device structure and init structure
hal_dac_struct_init	initialize DAC structure
hal_dac_start	start DAC module function
hal_dac_stop	stop DAC module function
hal_dac_start_interrupt	start DAC under error interrupt
hal_dac_stop_interrupt	stop DAC under error interrupt
hal_dac_start_dma	start DAC with DMA
hal_dac_stop_dma	stop DAC with DMA

Function name	Function description
hal_dac_irq	DAC interrupt handler content function, which is merely used in dac_handler
hal_dac_irq_handle_set	set user-defined interrupt callback function and enable dac interrupt
hal_dac_irq_handle_all_reset	reset all user-defined interrupt callback function

Enum hal_dac_state_enum

Table 3-125. Enum hal_dac_state_enum

Member name	Function description
DAC_STATE_NONE	none(default value)
DAC_STATE_RESET	DAC is not initialized or disabled
DAC_STATE_BUSY	DAC is busy
DAC_STATE_TIMEOUT	DAC timeout occurs
DAC_STATE_ERROR	DAC is error
DAC_STATE_READY	DAC is ready

Enum hal_dac_struct_type_enum

Table 3-126. Enum hal_dac_struct_type_enum

Member name	Function description
HAL_DAC_INIT_STRUCTURE	DAC initialization structure
HAL_DAC_IRQ_STRUCTURE	DAC device interrupt callback function pointer structure
HAL_DAC_DMA_HANDLER_CALLBACK_STRUCTURE	DAC DMA callback function pointer structure
HAL_DAC_DEVICE_STRUCTURE	DAC device information structure

Enum hal_dac_error_enum

Table 3-127. Enum hal_dac_error_enum

Member name	Function description
DAC_ERROR_NONE	no error

DAC_ERROR_SYS TEM	DAC internal error: if problem of clocking, enable/disable, wrong state
DAC_ERROR_DMA _UNDERFLOW	DAC DMA underflow error
DAC_ERROR_CON FIG	configuration error occurs

Structure hal_dac_irq_struct

Table 3-128. Structure hal_dac_irq_struct

Member name	Function description
dac_underflow_han dle	DAC underflow handler function

Structure hal_dac_dma_handle_cb_struct

Table 3-129. Structure hal_dac_dma_handle_cb_struct

Member name	Function description
full_transcom_han dle	DAC DMA transfer complete interrupt handler function
half_transcom_han dle	DAC DMA transfer complete interrupt handler function
error_handle	DAC DMA underflow error handler function

Structure hal_dac_dev_struct

Table 3-130. Structure hal_dac_dev_struct

Member name	Function description
dac_irq	DAC device interrupt callback function pointer structure
*p_dma_dac	DMA device information structtrue
dac_dma	DMA callback function pointer structure
error_state	DAC error state
state	DAC state
mutex	mutex locked and unlocked state
*priv	priv data

Structure hal_dac_init_struct

Table 3-131. Structure hal_dac_init_struct

Member name	Function description
output_waveform_s election	output waveform selection
noise_wave_bit_wid	noise wave bit width

th	
output_buffer_enable	output buffer enable
trigger_enable	trigger enable
trigger_selection	trigger selection
aligned_mode	aligned mode
output_value	output value

hal_dac_init

The description of hal_dac_init is shown as below:

Table 3-132. Function hal_dac_init

Function name	hal_dac_init
Function prototype	int32_t hal_dac_init(hal_dac_dev_struct *dac_dev, hal_dac_init_struct *dac);
Function descriptions	initialize DAC
Precondition	-
The called functions	-
Input parameter{in}	
dac_dev	points to hal_dac_dev_struct, struct members refer to - Table 3-130. Structure hal_dac_dev_struct
Input parameter{in}	
dac	points to hal_dac_init_struct, struct members refer to - Table 3-131. Structure hal_dac_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* initialize DAC */

hal_dac_init_struct dac_init_parameter;

hal_dac_dev_struct dac_info;

dac_init_parameter.output_waveform_selection = DAC_WAVE_DISABLE;

dac_init_parameter.trigger_enable = ENABLE;

dac_init_parameter.trigger_selection = DAC_TRIGGER_EXTI_9;

dac_init_parameter.aligned_mode = DAC_R12_ALIGNED_MODE;

dac_init_parameter.output_buffer_enable = ENABLE;

```

```
dac_init_parameter.output_value = 0;
```

```
hal_dac_init(&dac_info,&dac_init_parameter);
```

hal_dac_deinit

The description of hal_dac_deinit is shown as below:

Table 3-133. Function hal_dac_deinit

Function name	hal_dac_deinit
Function prototype	int32_t hal_dac_deinit(hal_dac_dev_struct *dac_dev);
Function descriptions	deinitialize DAC device structure and init structure
Precondition	-
The called functions	hal_rcu_periph_reset_enable / hal_rcu_periph_reset_disable
Input parameter{in}	
dac_dev	points to hal_dac_dev_struct, struct members refer to Table 3-130. Structure hal_dac_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

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

```
hal_dac_dev_struct dac_info;
```

```
hal_dac_deinit(&dac_info);
```

hal_dac_struct_init

The description of hal_dac_struct_init is shown as below:

Table 3-134. Function hal_dac_struct_init

Function name	hal_dac_struct_init
Function prototype	void hal_dac_struct_init(hal_dac_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the DAC structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	type of structure to be initialized
HAL_DAC_INIT_STRU CT	DAC initialization structure
HAL_DAC_IRQ_STRU	DAC interrupt structure

CT	
HAL_DAC_DMA_HANDLER_CB_STRUCT	DAC DMA callback structure
HAL_DAC_DEV_STRUCT	DAC device structure
Input parameter{in}	
p_struct	point to DAC structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the DAC structure with the default values */
hal_dac_dev_struct dac_info;
hal_dac_struct_init(HAL_DAC_DEV_STRUCT, &dac_info);
```

hal_dac_start

The description of hal_dac_start is shown as below:

Table 3-135. Function hal_dac_start

Function name	hal_dac_start
Function prototype	int32_t hal_dac_start(hal_dac_dev_struct *dac_dev);
Function descriptions	start DAC module function
Precondition	-
The called functions	-
Input parameter{in}	
dac_dev	points to hal_dac_dev_struct, struct members refer to - Table 3-130. Structure hal_dac_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* start DAC */
hal_dac_dev_struct dac_info;
hal_dac_start(&dac_info);
```

hal_dac_stop

The description of hal_dac_stop is shown as below:

Table 3-136. Function hal_dac_stop

Function name	hal_dac_stop
Function prototype	int32_t hal_dac_stop(hal_dac_dev_struct *dac_dev);
Function descriptions	stop DAC module function
Precondition	-
The called functions	-
Input parameter{in}	
dac_dev	points to hal_dac_dev_struct, struct members refer to - Table 3-130. Structure hal_dac_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* stop DAC */
hal_dac_dev_struct dac_info;
hal_dac_stop(&dac_info);
```

hal_dac_start_interrupt

The description of hal_dac_start_interrupt is shown as below:

Table 3-137. Function hal_dac_start_interrupt

Function name	hal_dac_start_interrupt
Function prototype	int32_t hal_dac_start_interrupt(hal_dac_dev_struct *dac_dev, hal_dac_irq_struct *p_irq);
Function descriptions	start DAC under error interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dac_dev	points to hal_dac_dev_struct, struct members refer to - Table 3-130. Structure hal_dac_dev_struct
Input parameter{in}	
p_irq	Structure for ADC device interrupt callback function pointer structure, the structure members can refer to Table 3-128. Structure hal_dac_irq_struct
Output parameter{out}	

-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* start DAC under error interrupt */

uint8_t convertarr[10] = {0x00, 0x33, 0x66, 0x99, 0xCC, 0xFF, 0xCC, 0x99, 0x66, 0x33};

hal_dac_init_struct dac_init_parameter;

hal_dac_dev_struct dac_info;

hal_dac_irq_struct dac_irq;

hal_dac_start_dma(&dac_info, (uint32_t*)convertarr, 10, DAC_R8_ALIGNED_MODE, &dac_dma_handle_cb);

dac_irq.dac_underflow_handle = dac_underflow_func;

hal_dac_irq_handle_set(&dac_info, &dac_irq);

hal_dac_start_interrupt(&dac_info, &dac_irq);
```

hal_dac_stop_interrupt

The description of hal_dac_stop_interrupt is shown as below:

Table 3-138. Function hal_dac_stop_interrupt

Function name	hal_dac_stop_interrupt
Function prototype	int32_t hal_dac_stop_interrupt(hal_dac_dev_struct *dac_dev);
Function descriptions	stop DAC under error interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dac_dev	points to hal_dac_dev_struct, struct members refer to - Table 3-130. Structure hal_dac_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* stop DAC under error interrupt */

uint8_t convertarr[10] = {0x00, 0x33, 0x66, 0x99, 0xCC, 0xFF, 0xCC, 0x99, 0x66, 0x33};

hal_dac_init_struct dac_init_parameter;
```

```
hal_dac_dev_struct dac_info;
```

```
hal_dac_start_dma(&dac_info,(uint32_t*)convertarr,10,DAC_R8_ALIGNED_MODE,&dac_d  
ma_handle_cb);
```

```
dac_irq.dac_underflow_handle = dac_underflow_func;
```

```
hal_dac_irq_handle_set(&dac_info,&dac_irq);
```

```
hal_dac_stop_interrupt(&dac_info);
```

hal_dac_start_dma

The description of hal_dac_start_dma is shown as below:

Table 3-139. Function hal_dac_start_dma

Function name	hal_dac_start_dma
Function prototype	int32_t hal_dac_start_dma(hal_dac_dev_struct *dac_dev, uint32_t* pdata, uint32_t length, uint32_t aligned_mode, hal_dac_dma_handle_cb_struct *dmacb);
Function descriptions	start DAC with DMA
Precondition	-
The called functions	-
Input parameter{in}	
dac_dev	points to hal_dac_dev_struct, struct members refer to - Table 3-130. Structure hal_dac_dev_struct
Input parameter{in}	
p_data	the source memory buffer address
Input parameter{in}	
length	the number of data to be transferred from source to destination
Input parameter{in}	
aligned_mode	aligned mode to be choose
DAC_R12_ALIGNED_MODE	12-bit right-aligned
DAC_L12_ALIGNED_MODE	12-bit left-aligned
DAC_R8_ALIGNED_MODE	8-bit right-aligned
Input parameter{in}	
dmacb	points to hal_dac_dma_handle_cb_struct, struct members refer to - Table 3-129. Structure hal_dac_dma_handle_cb_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* start DAC with DMA */

uint8_t convertarr[10] = {0x00, 0x33, 0x66, 0x99, 0xCC, 0xFF, 0xCC, 0x99, 0x66, 0x33};

hal_dac_init_struct dac_init_parameter;

hal_dac_dev_struct dac_info;

hal_dac_start_dma(&dac_info, (uint32_t*)convertarr, 10, DAC_R8_ALIGNED_MODE, &dac_d
ma_handle_cb);

dac_irq.dac_underflow_handle = dac_underflow_func;

hal_dac_start_dma(&dac_info, (uint32_t*)convertarr, 10, DAC_R8_ALIGNED_MODE, &dac_d
ma_handle_cb);

hal_dac_start(&dac_info);
```

hal_dac_stop_dma

The description of hal_dac_stop_dma is shown as below:

Table 3-140. Function hal_dac_stop_dma

Function name	hal_dac_stop_dma
Function prototype	int32_t hal_dac_stop_dma(hal_dac_dev_struct *dac_dev);
Function descriptions	stop DAC with DMA
Precondition	-
The called functions	-
Input parameter{in}	
dac_dev	points to hal_dac_dev_struct, struct members refer to - Table 3-130. Structure hal_dac_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* stop DAC with DMA */

hal_dac_dev_struct dac_info;

hal_dac_stop_dma(&dac_info);
```

hal_dac_irq

The description of hal_dac_irq is shown as below:

Table 3-141. Function `hal_dac_irq`

Function name	<code>hal_dac_irq</code>
Function prototype	<code>int32_t hal_dac_irq(hal_dac_dev_struct *dac_dev);</code>
Function descriptions	DAC interrupt handler content function, which is merely used in <code>dac_handler</code>
Precondition	-
The called functions	-
Input parameter{in}	
<code>dac_dev</code>	points to <code>hal_dac_dev_struct</code> , struct members refer to - Table 3-130. Structure <code>hal_dac_dev_struct</code>
Output parameter{out}	
-	-
Return value	
<code>int32_t</code>	<code>HAL_ERR_ADDRESS</code> , <code>HAL_ERR_NONE</code>

Example:

```
/* the function is used in the relative interrupt routine */
```

```
hal_dac_dev_struct dac_info;
```

```
TIMER5_DAC_IRQHandler(void)
```

```
{
    hal_dac_irq(&dac_info);
}
```

`hal_dac_irq_handle_set`

The description of `hal_dac_irq_handle_set` is shown as below:

Table 3-142. Function `hal_dac_irq_handle_set`

Function name	<code>hal_dac_irq_handle_set</code>
Function prototype	<code>int32_t hal_dac_irq_handle_set(hal_dac_dev_struct *dac_dev, hal_dac_irq_struct *p_irq);</code>
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
<code>dac_dev</code>	points to <code>hal_dac_dev_struct</code> , struct members refer to - Table 3-130. Structure <code>hal_dac_dev_struct</code>
Input parameter{in}	
<code>p_irq</code>	Structure for ADC device interrupt callback function pointer structure, the structure members can refer to Table 3-128. Structure

	<u>hal_dac_irq_struct</u>
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* set user-defined interrupt callback function */

hal_dac_dev_struct dac_info;

hal_dac_irq_struct dac_irq;

dac_irq.dac_underflow_handle = dac_underflow_func;

hal_dac_irq_handle_set(&dac_info,&dac_irq);
```

hal_dac_irq_handle_all_reset

The description of hal_dac_irq_handle_all_reset is shown as below:

Table 3-143. Function hal_dac_irq_handle_all_reset

Function name	hal_dac_irq_handle_all_reset
Function prototype	int32_t hal_dac_irq_handle_all_reset(hal_dac_dev_struct *dac_dev);
Function descriptions	reset user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
dac_dev	points to hal_dac_dev_struct, struct members refer to - <u>Table 3-130. Structure hal_dac_dev_struct</u>
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* reset all user-defined interrupt callback function */

hal_dac_dev_struct dac_info;

hal_dac_irq_handle_all_reset(&dac_info);
```

3.8. SYS

SYS contains DBG and Basetick modules, which provides the configuration function of the system Debug mode and timer source, and it also provides a timer with an accuracy of 1ms. Section [3.8.1](#) describes the register list of DBG and SysTick, and Section [3.8.2](#) describes the SYS library functions.

3.8.1. Descriptions of Peripheral registers

SYS DBG & SysTick registers are listed in the table shown as below:

Table 3-144. SYS DBG & SysTick Registers

Registers	Descriptions
SYS_DBG_ID	DBG ID code register
SYS_DBG_CTL0	DBG control register 0
SYS_DBG_CTL1	DBG control register 1
SYS_SYSTICK_CTRL	SysTick Control and Status Register
SYS_SYSTICK_LOAD	SysTick Reload Value Register
SYS_SYSTICK_VAL	SysTick Current Value Register
SYS_SYSTICK_CALIB	SysTick Calibration Register

3.8.2. Descriptions of Peripheral functions

SYS firmware function are listed in the table shown as below:

Table 3-145. SYS firmware function

Function name	Function description
hal_sys_deinit	deinitialize SYS(DBG&Basetick)
hal_sys_debug_init	Initialize DBG device
hal_sys_timesource_init	Initialize timebase source
hal_sys_basetick_count_get	get the basetick count
hal_sys_basetick_timeout_check	check whether the delay is finished
hal_sys_basetick_delay_ms	set the basetick delay
hal_sys_basetick_suspend	suspend the basetick timer
hal_sys_basetick_resume	resume the basetick timer
hal_sys_basetick_irq	basetick interrupt handler
hal_sys_basetick_irq_handle_set	set user-defined interrupt callback function
hal_sys_basetick_irq_handle_all_reset	reset all user-defined interrupt callback function

Enum hal_sys_debug_cfg_enum

Table 3-146. Enum hal_sys_debug_cfg_enum

Member name	Function description
SYS_DEBUG_DISABLE	Disable Debug
SYS_DEBUG_SERIAL_WIRE	Serial wire Debug

Enum hal_sys_timebase_source_enum

Table 3-147. Enum hal_sys_timebase_source_enum

Member name	Function description
SYS_TIMEBASE_SOURCE_SYSTICK	Select SysTick as timebase source
SYS_TIMEBASE_SOURCE_TIMER0	Select TIMER0 as timebase source
SYS_TIMEBASE_SOURCE_TIMER1	Select TIMER1 as timebase source
SYS_TIMEBASE_SOURCE_TIMER2	Select TIMER2 as timebase source
SYS_TIMEBASE_SOURCE_TIMER5	Select TIMER5 as timebase source
SYS_TIMEBASE_SOURCE_TIMER13	Select TIMER13 as timebase source
SYS_TIMEBASE_SOURCE_TIMER14	Select TIMER14 as timebase source
SYS_TIMEBASE_SOURCE_TIMER15	Select TIMER15 as timebase source
SYS_TIMEBASE_SOURCE_TIMER16	Select TIMER16 as timebase source

hal_sys_deinit

The description of hal_sys_deinit is shown as below:

Table 3-148. Function hal_sys_deinit

Function name	hal_sys_deinit
Function prototype	int32_t hal_sys_deinit(void);
Function descriptions	deinitialize SYS(DBG&Basetick)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```
/* Deinitialize SYS(DBG & Basetick) */
```

```
hal_sys_deinit();
```

hal_sys_init

The description of hal_sys_init is shown as below:

Table 3-149. Function hal_sys_init

Function name	hal_sys_debug_init
Function prototype	int32_t hal_sys_debug_init(hal_sys_debug_cfg_enum debug_cfg);
Function descriptions	Initialize DBG device
Precondition	-
The called functions	-
Input parameter{in}	
debug_cfg	DBG configuration information refer to Table 3-146. Enum hal_sys_debug_cfg_enum
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```
/* initialize DBG */
```

```
hal_sys_debug_init (SYS_DEBUG_SERIAL_WIRE);
```

hal_sys_timesource_init

The description of hal_sys_timesource_init is shown as below:

Table 3-150. Function hal_sys_timesource_init

Function name	hal_sys_timesource_init
Function prototype	int32_t hal_sys_timesource_init(hal_sys_timebase_source_enum timebase_source);
Function descriptions	Initialize timebase source
Precondition	-
The called functions	-
Input parameter{in}	
timebase_source	System timerbase configuration information refer to Table 3-147. Enum hal_sys_timebase_source_enum
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```
/* SYS timebase initialize */
```

```
hal_sys_timesource_init (SYS_TIMEBASE_SOURCE_SYSTICK);
```

hal_sys_basetick_count_get

The description of hal_sys_basetick_count_get is shown as below:

Table 3-151. Function hal_sys_basetick_count_get

Function name	hal_sys_basetick_count_get
Function prototype	uint32_t hal_sys_basetick_count_get(void);
Function descriptions	get the basetick count
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
int32_t	32-bits current basetick counter in ms

Example:

```
/* get the basetick count function */
uint32 basetick_cout = 0x00000000u;
basetick_cout = hal_sys_basetick_count_get ();
```

hal_sys_basetick_timeout_check

The description of hal_sys_basetick_timeout_check is shown as below:

Table 3-152. Function hal_sys_basetick_timeout_check

Function name	hal_sys_basetick_timeout_check
Function prototype	FlagStatus hal_sys_basetick_timeout_check(uint32_t time_start, uint32_t delay);
Function descriptions	check whether the delay is finished
Precondition	-
The called functions	-
Input parameter{in}	
time_start	the start time of the system Delay (accuracy:ms).
Input parameter{in}	
delay	system delay (accuracy:ms)
Output parameter{out}	
-	-
Return value	
FlagStatus	SET, RESET

Example:

```
uint32_t count = 0;

/* Get current system tick counter*/

count = hal_sys_basetick_count_get();

/* check whether the delay is finished */

while(RESET == hal_sys_basetick_timeout_check(count, 500));

{

    /* delay not finished */

}

/* delay finished */
```

hal_sys_basetick_delay_ms

The description of hal_sys_basetick_delay_ms is shown as below:

Table 3-153. Function hal_sys_basetick_delay_ms

Function name	hal_sys_basetick_delay_ms
Function prototype	void hal_sys_basetick_delay_ms(uint32_t time_ms);
Function descriptions	set the basetick delay
Precondition	-
The called functions	-
Input parameter{in}	
time_ms	Timebase delay set(accuray:ms)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* Set 500ms system time delay */

hal_sys_basetick_delay_ms(500);
```

hal_sys_basetick_suspend

The description of hal_sys_basetick_suspend is shown as below:

Table 3-154. Function hal_sys_basetick_suspend

Function name	hal_sys_basetick_suspend
Function prototype	void hal_sys_basetick_suspend(void);

Function descriptions	suspend the basetick timer
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* suspend system tick */
hal_sys_basetick_suspend();
```

hal_sys_basetick_resume

The description of hal_sys_basetick_resume is shown as below:

Table 3-155. Function hal_sys_basetick_resume

Function name	hal_sys_basetick_resume
Function prototype	void hal_sys_basetick_resume(void);
Function descriptions	resume the basetick timer
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

```
/* resume system tick */
hal_sys_basetick_resume ();
```

hal_sys_basetick_irq

The description of hal_sys_basetick_irq is shown as below:

Table 3-156. Function hal_sys_basetick_irq

Function name	hal_sys_basetick_irq
Function prototype	void hal_sys_basetick_irq(void);
Function descriptions	basetick interrupt handler
Precondition	-

The called functions	SysTick_Handler()/ TIMERx_IRQHandler()
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* the function is used in the relative interrupt routine */

SysTick_Handler (void)

```
{
    hal_sys_basetick_irq ();
}
```

hal_sys_basetick_irq_handle_set

The description of hal_sys_basetick_irq_handle_set is shown as below:

Table 3-157. Function hal_sys_basetick_irq_handle_set

Function name	hal_sys_basetick_irq_handle_set
Function prototype	void hal_sys_basetick_irq_handle_set(hal_sys_basetick_irq_handle_cb irq_handler);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
irq_handler	Pointer of user defined interrupt callback function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set user-defined interrupt callback function */

hal_sys_basetick_irq_handle_cb sys_basetick_user_irq(void)

```
{
}
```

```
hal_sys_basetick_irq_handle_set (sys_basetick_user_irq);
```

hal_sys_basetick_irq_handle_all_reset

The description of hal_sys_basetick_irq_handle_all_reset is shown as below:

Table 3-158. Function hal_sys_basetick_irq_handle_all_reset

Function name	hal_sys_basetick_irq_handle_all_reset
Function prototype	void hal_sys_basetick_irq_handle_all_reset(void);
Function descriptions	reset user-defined interrupt callback function to NULL
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback function */
```

```
hal_sys_basetick_irq_handle_all_reset()
```

3.9. DMA

The direct memory access (DMA) controller provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. The DMA registers are listed in chapter [3.9.1](#) the DMA firmware functions are introduced in chapter [3.9.2](#).

3.9.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

Table 3-159. DMA Registers

Registers	Descriptions
DMA_INTF	DMA interrupt flag register
DMA_INTC	DMA interrupt flag clear register
DMA_CHxCTL (x=0..6)	DMA channel x control register
DMA_CHxCNT (x=0..6)	DMA channel x counter register

Registers	Descriptions
DMA_CHxPADDR (x=0..6)	DMA channel x peripheral base address register
DMA_CHxMADDR (x=0..6)	DMA channel x memory base address register

3.9.2. Descriptions of Peripheral functions

DMA registers are listed in the table shown as below:

Table 3-160. DMA firmware function

Function name	Function description
hal_dma_init	initialize DMA channel
hal_dma_struct_init	initialize the DMA structure with the default values
hal_dma_deinit	initialize DAC structure
hal_dma_start	start DMA transferring
hal_dma_stop	stop DMA transferring
hal_dma_irq	DMA interrupt handler content function
hal_dma_irq_handle_set	set user-defined interrupt callback function
hal_dma_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_dma_start_poll	start transferring amounts of data, poll transmit process and completed status
hal_dma_start_interrupt	start transferring amounts of data by interrupt method
hal_dma_stop_interrupt	stop transferring amounts of data by interrupt method
hal_dma_channel_remap_enable	reset all user-defined interrupt callback function
hal_dma_channel_remap_disable	disable the DMA channels remapping

Enum dma_channel_enum

Table 3-161. dma_channel_enum

Member name	Function description
DMA_CH0	DMA Channel0
DMA_CH1	DMA Channel1
DMA_CH2	DMA Channel2
DMA_CH3	DMA Channel3
DMA_CH4	DMA Channel4
DMA_CH5	DMA Channel5
DMA_CH6	DMA Channel6

Enum hal_dma_struct_type_enum

Table 3-162. hal_dma_struct_type_enum

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

HAL_DMA_INIT_STRUCTURE	DMA initialize structure
HAL_DMA_DEVICE_INFORMATION_STRUCTURE	DMA device information structure
HAL_DMA_INTERRUPT_CALLBACK_FUNCTION_POINTER_STRUCTURE	DMA device interrupt callback function pointer structure

Enum hal_dma_transfer_state_enum

Table 3-163. hal_dma_transfer_state_enum

Member name	Function description
HAL_DMA_TRANSFER_STATE_HALF	half transfer
HAL_DMA_TRANSFER_STATE_FULL	full transfer

Enum hal_dma_error_enum

Table 3-164. hal_dma_error_enum

Member name	Function description
HAL_DMA_ERROR_NONE	no error
HAL_DMA_ERROR_TRANSFER	DMA transfer error
HAL_DMA_ERROR_NOTTRANSFER	DMA not in transfer error
HAL_DMA_ERROR_TIMEOUT	DMA transfer timeout error
HAL_DMA_ERROR_UNSupport	DMA transfer mode configure error

Enum hal_dma_state_enum

Table 3-165. hal_dma_state_enum

Member name	Function description
HAL_DMA_STATE_NONE	None (default value)
HAL_DMA_STATE_RESET	reset
HAL_DMA_STATE_BUSY	busy
HAL_DMA_STATE_TIMEOUT	timeout

Member name	Function description
HAL_DMA_STATE_ERROR	error
HAL_DMA_STATE_READY	ready

Structure hal_dma_irq_struct

Table 3-166. Structure hal_dma_irq_struct

Member name	Function description
error_handle	channel error interrupt handler pointer
half_finish_handle	channel half transfer finish interrupt handler pointer
full_finish_handle	channel full transfer finish interrupt handler pointer

Structure hal_dma_dev_struct

Table 3-167. Structure hal_dma_dev_struct

Member name	Function description
channel	DMA channel
dma_irq	DMA device interrupt callback function pointer structure
transfer_state	DMA device transfer state
error_state	DMA device error state
state	DMA device state
*p_periph	user private periph
mutex	DMA device mutex

Structure hal_dma_init_struct

Table 3-168. Structure hal_dma_init_struct

Member name	Function description
direction	transfer direction
priority	channel priority
mode	transfer mode
periph_inc	peripheral increasing mode
memory_inc	memory increasing mode
periph_width	transfer data size of peripheral
memory_width	transfer data size of memory

hal_dma_init

The description of hal_dma_init is shown as below:

Table 3-169. Function hal_dma_init

Function name	hal_dma_init
---------------	--------------

Function prototype	int32_t hal_dma_init(hal_dma_dev_struct *dma_dev, dma_channel_enum channelx, hal_dma_init_struct *dma);
Function descriptions	initialize DMA channelx (x=0..6)
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Input parameter{in}	
channelx	DMA channel
DMA_CHx (x=0..6)	Select DMA channel
Input parameter{in}	
dma	points to hal_dac_init_struct, struct members refer to - Table 3-168. Structure hal_dma_init_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE, HAL_ERR_VAL

Example:

```

/* initialize DMA channel0 */
hal_dma_init_struct dma_init_parameter;
hal_dma_dev_struct dma_info;
dma_init_parameter.direction = DMA_DIR_MEMORY_TO_PERIPH;
dma_init_parameter.periph_inc = DISABLE;
dma_init_parameter.memory_inc = ENABLE;
dma_init_parameter.periph_width = DMA_PERIPH_SIZE_8BITS;
dma_init_parameter.memory_width = DMA_MEMORY_SIZE_8BITS;
dma_init_parameter.mode = DMA_NORMAL_MODE;
dma_init_parameter.priority = DMA_PRIORITY_LEVEL_LOW;
hal_dma_init(&dma_info, DMA_CH1, &dma_init_parameter);

```

hal_dma_struct_init

The description of hal_dma_struct_init is shown as below:

Table 3-170. Function hal_dma_struct_init

Function name	hal_dma_struct_init
Function prototype	int32_t hal_dma_struct_init(hal_dma_struct_type_enum struct_type, void *p_struct);

Function descriptions	initialize the DMA structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	type of structure to be initialized
HAL_DMA_INIT_STR UCT	DMA initialize structure
HAL_DMA_DEV_STR UCT	DMA device information structure
HAL_DMA_IRQ_STRU CT	DMA device interrupt callback function pointer structure
Input parameter{in}	
p_struct	point to DMA structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE, HAL_ERR_VAL

Example:

```
/* initialize the DMA structure with the default values */
hal_dma_dev_struct dma_info;
hal_dma_struct_init(HAL_DMA_DEV_STRUCT, &dma_info);
```

hal_dma_deinit

The description of hal_dma_deinit is shown as below:

Table 3-171. Function hal_dma_deinit

Function name	hal_dma_deinit
Function prototype	int32_t hal_dma_deinit(hal_dma_dev_struct *dma_dev);
Function descriptions	deinitialize DMA device structure and init structure
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* deinitialize DMA device structure and init structure */
hal_dma_dev_struct dma_info;
hal_dma_deinit (&dma_info);
```

hal_dma_start

The description of hal_dma_start is shown as below:

Table 3-172. Function hal_dma_start

Function name	hal_dma_start
Function prototype	int32_t hal_dma_start(hal_dma_dev_struct *dma_dev, uint32_t src_addr, \ uint32_t dst_addr, uint16_t length);
Function descriptions	start DMA transferring
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Input parameter{in}	
src_addr	the source memory buffer address
Input parameter{in}	
dst_addr	the destination memory buffer address
Input parameter{in}	
length	the number of data to be transferred from source to destination
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_BUSY, HAL_ERR_NONE

Example:

```
/* start DMA transfer */
#define TRANSFER_NUM    1024
int src_array[TRANSFER_NUM];
int dst_array[TRANSFER_NUM];
hal_dma_dev_struct dma_info;
hal_dma_start(&dma_info, src_array, dst_array, TRANSFER_NUM);
```

hal_dma_stop

The description of hal_dma_stop is shown as below:

Table 3-173. Function hal_dma_stop

Function name	hal_dma_stop
Function prototype	int32_t hal_dma_stop(hal_dma_dev_struct *dma_dev);
Function descriptions	stop DMA transferring
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE, HAL_ERR_ALREADY_DONE

Example:

```
/* stop DMA transfer */
hal_dma_dev_struct dma_info;
hal_dma_stop(&dma_info);
```

hal_dma_irq

The description of hal_dma_irq is shown as below:

Table 3-174. Function hal_dma_irq

Function name	hal_dma_irq
Function prototype	int32_t hal_dma_irq(hal_dma_dev_struct *dma_dev);
Function descriptions	DMA interrupt handler content function, which is merely used in dma_handler
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* the function is used in the relative interrupt routine */
hal_dma_dev_struct dma_info;
```

```
void DMA_Channel1_2_IRQHandler(void)
{
    hal_dma_irq(&dma_info);
}
```

hal_dma_irq_handle_set

The description of hal_dma_irq_handle_set is shown as below:

Table 3-175. Function hal_dma_irq_handle_set

Function name	hal_dma_irq_handle_set
Function prototype	int32_t hal_dma_irq_handle_set(hal_dma_dev_struct *dma_dev, hal_dma_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Input parameter{in}	
p_irq	DMA device interrupt callback function pointer structure, the structure members can refer to - Table 3-166. Structure hal_dma_irq_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* set user-defined interrupt callback function */
hal_dma_dev_struct dma_info;
hal_dma_irq_struct dma_irq;
dma_irq.full_finish_handle = dma_full_finish_func;
hal_dma_irq_handle_set(&dma_info,&dma_irq);
```

hal_dma_irq_handle_all_reset

The description of hal_dma_irq_handle_all_reset is shown as below:

Table 3-176. Function hal_dma_irq_handle_all_reset

Function name	hal_dma_irq_handle_all_reset
Function prototype	int32_t hal_dma_irq_handle_all_reset(hal_dma_dev_struct *dma_dev);
Function	reset all user-defined interrupt callback function

descriptions	
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* reset all user-defined interrupt callback function */
hal_dma_dev_struct dma_info;
hal_dma_irq_handle_all_reset(&dma_info);
```

hal_dma_start_poll

The description of hal_dma_start_poll is shown as below:

Table 3-177. Function hal_dma_start_poll

Function name	hal_dma_start_poll
Function prototype	int32_t hal_dma_start_poll(hal_dma_dev_struct *dma_dev, hal_dma_transfer_state_enum transfer_state, uint32_t timeout_ms);
Function descriptions	start transferring amounts of data, poll transmit process and completed status
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Input parameter{in}	
transfer_state	DMA transfer state
HAL_DMA_TARNSFER_HALF	half transfer
HAL_DMA_TARNSFER_FULL	full transfer
Input parameter{in}	
timeout_ms	time out duration
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_ADDRESS, HAL_ERR_HARDWARE, HAL_ERR_NONE, HAL_ERR_NO_SUPPORT, HAL_ERR_ALREADY_DONE, HAL_ERR_TIMEOUT
----------------	--

Example:

```
/* start amounts of data, poll transmit process and completed status*/
#define TIMEOUT    10
hal_dma_dev_struct dma_info;
hal_dma_start_poll(&dma_info, HAL_DMA_TARNSEFER_FULL, TIMEOUT);
```

hal_dma_start_interrupt

The description of hal_dma_start_interrupt is shown as below:

Table 3-178. Function hal_dma_start_interrupt

Function name	hal_dma_start_interrupt
Function prototype	int32_t hal_dma_start_interrupt(hal_dma_dev_struct *dma_dev, uint32_t src_addr, uint32_t dst_addr, uint16_t length, hal_dma_irq_struct *p_irq)
Function descriptions	start transferring amounts of data by interrupt method
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Input parameter{in}	
src_addr	the source memory buffer address
Input parameter{in}	
dst_addr	the destination memory buffer address
Input parameter{in}	
length	he number of data to be transferred from source to destination
Input parameter{in}	
p_irq	DMA device interrupt callback function pointer structure, the structure members can refer to - Table 3-166. Structure hal_dma_irq_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* start amounts of data by interrupt method */
uint8_t src_arr[10] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A};
```

```
uint8_t dst_arr[10];
hal_dma_init_struct dma_init_parameter;
hal_dma_dev_struct dma_info;
hal_dma_irq_struct dma_irq;
dma_irq.full_finish_handle = dma_full_finish_func;
dma_init_parameter.direction = DMA_DIR_MEMORY_TO_MEMORY;
dma_init_parameter.periph_inc = DISABLE;
dma_init_parameter.memory_inc = ENABLE;
dma_init_parameter.periph_width = DMA_PERIPH_SIZE_8BITS;
dma_init_parameter.memory_width = DMA_MEMORY_SIZE_8BITS;
dma_init_parameter.mode = DMA_NORMAL_MODE;
dma_init_parameter.priority = DMA_PRIORITY_LEVEL_LOW;
hal_dma_init(&dma_info, DMA_CH1, &dma_init_parameter);
hal_dma_irq_handle_set(&dma_info, &dma_irq);
hal_dma_start_interrupt(&dma_info, src_arr, dst_arr, 10, &dma_irq);
```

hal_dma_stop_interrupt

The description of hal_dma_stop_interrupt is shown as below:

Table 3-179. Function hal_dma_stop_interrupt

Function name	hal_dma_stop_interrupt
Function prototype	int32_t hal_dma_stop_interrupt(hal_dma_dev_struct *dma_dev);
Function descriptions	stop transferring amounts of data by interrupt method
Precondition	-
The called functions	-
Input parameter{in}	
dma_dev	points to hal_dma_dev_struct, struct members refer to - Table 3-167. Structure hal_dma_dev_struct .
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE, HAL_ERR_ALREADY_DONE

Example:

```
/* stop amounts of data by interrupt method */
hal_dma_stop_interrupt (&dma_info);
```

hal_dma_channel_remap_enable

The description of hal_dma_channel_remap_enable is shown as below:

Table 3-180. Function hal_dma_channel_remap_enable

Function name	hal_dma_channel_remap_enable
Function prototype	int32_t hal_dma_channel_remap_enable(uint32_t dma_remap);
Function descriptions	enable the DMA channels remapping
Precondition	-
The called functions	-
Input parameter{in}	
dma_remap	specify the DMA channels to remap
DMA_REMAP_TIMER16_CH0_UP	remap TIMER16 channel0 and UP DMA requests to channel1 (default channel0)
DMA_REMAP_TIMER15_CH0_UP	remap TIMER15 channel2 and UP DMA requests to channel3 (default channel2)
DMA_REMAP_USART0_RX	remap USART0 Rx DMA request to channel4 (default channel2)
DMA_REMAP_USART0_TX	remap USART0 Tx DMA request to channel3 (default channel1)
DMA_REMAP_ADC	remap ADC DMA requests from channel0 to channel1
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE, HAL_ERR_ALREADY_DONE

Example:

```
/* enable the DMA channels remapping */
hal_dma_channel_remap_enable (DMA_REMAP_USART0_TX);
```

hal_dma_channel_remap_disable

The description of hal_dma_channel_remap_disable is shown as below:

Table 3-181. Function hal_dma_channel_remap_disable

Function name	hal_dma_channel_remap_disable
Function prototype	int32_t hal_dma_channel_remap_disable(uint32_t dma_remap);
Function descriptions	disable the DMA channels remapping
Precondition	-
The called functions	-
Input parameter{in}	
dma_remap	specify the DMA channels to disable remapping
DMA_REMAP_TIMER16_CH0_UP	remap TIMER16 channel0 and UP DMA requests to channel1 (default channel0)
DMA_REMAP_TIMER15_CH0_UP	remap TIMER15 channel2 and UP DMA requests to channel3 (default channel2)

15_CH0_UP	channel2)
DMA_REMAP_USART0_RX	remap USART0 Rx DMA request to channel4 (default channel2)
DMA_REMAP_USART0_TX	remap USART0 Tx DMA request to channel3 (default channel1)
DMA_REMAP_ADC	remap ADC DMA requests from channel0 to channel1
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE, HAL_ERR_ALREADY_DONE

Example:

```
/* disable the DMA channels remapping */
hal_dma_channel_remap_disable (DMA_REMAP_USART0_TX);
```

3.10. EXTI

EXTI is the interrupt/event controller in the MCU. It contains up to 19 independent edge detectors and generates interrupt requests or events to the processor. The EXTI registers are listed in chapter [3.10.1](#), the EXTI firmware functions are introduced in chapter [3.10.2](#).

3.10.1. Descriptions of Peripheral registers

EXTI registers are listed in the table shown as below:

Table 3-182 EXTI registers

Registers	Descriptions
EXTI_INTEN	Interrupt enable register
EXTI_EVEN	Event enable register
EXTI_RTEN	Rising edge trigger enable register
EXTI_FTEN	Falling edge trigger enable register
EXTI_SWIEV	Software interrupt event register
EXTI_PD	Pending register

3.10.2. Descriptions of Peripheral functions

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

Table 3-183. EXTI firmware function

Function name	Function description
hal_exti_gpio_deinit	deinitialize the EXTI gpio
hal_exti_internal_deinit	deinitialize the EXTI internal line
hal_exti_gpio_init	initialize the configuration of EXTI gpio
hal_exti_internal_init	initialize the configuration of EXTI internal
hal_exti_gpio_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_exti_gpio_irq_handle_all_reset	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_exti_gpio_irq	EXTI GPIO interrupt handle function

Enum hal_exti_type_enum

Table

3-184. hal_exti_type_enum

enum name	enum description
EXTI_EVENT_TRIG_RISING	event rising edge trigger
EXTI_EVENT_TRIG_FALLING	event falling edge trigger
EXTI_EVENT_TRIG_BOTH	event rising edge and falling edge trigger
EXTI_INTERRUPT_TRIG_RISING	interrupt rising edge trigger
EXTI_INTERRUPT_TRIG_FALLING	interrupt falling edge trigger
EXTI_INTERRUPT_TRIG_BOTH	interrupt rising edge and falling edge trigger

Enum hal_exti_line_enum

Table 3-185. hal_exti_line_enum

enum name	enum description
EXTI_PIN_0	EXTI line 0
EXTI_PIN_1	EXTI line 1
EXTI_PIN_2	EXTI line 2
EXTI_PIN_3	EXTI line 3
EXTI_PIN_4	EXTI line 4
EXTI_PIN_5	EXTI line 5
EXTI_PIN_6	EXTI line 6
EXTI_PIN_7	EXTI line 7
EXTI_PIN_8	EXTI line 8
EXTI_PIN_9	EXTI line 9
EXTI_PIN_10	EXTI line 10
EXTI_PIN_11	EXTI line 11
EXTI_PIN_12	EXTI line 12
EXTI_PIN_13	EXTI line 13
EXTI_PIN_14	EXTI line 14
EXTI_PIN_15	EXTI line 15
EXTI_LVD_16	EXTI line 16
EXTI_RTC_ALARM_17	EXTI line 17
EXTI_USBFS_WAKEUP_18	EXTI line 18
EXTI_RTC_TAMPER_TIMESTAMP_19	EXTI line 19
EXTI_COMP_OUTPUT_T_21	EXTI line 20
EXTI_COMP_OUTPUT	EXTI line 21

enum name	enum description
_22	
EXTI_USART0_WAKEUP_25	EXTI line 22
EXTI_CEC_WAKEUP_27	EXTI line 23

Enum hal_exti_internal_line_enum

Table 3-186. hal_exti_internal_line_enum

enum name	enum description
EXTI_LINE_16_LVD	Low Voltage Detection EXTI line
EXTI_LINE_17_RTC_ALARM	RTC alarm EXTI line
EXTI_LINE_18_USBF_WAKEUP	USB WAKEUP EXTI line
EXTI_LINE_19_RTC_TAMPER_TIMESTAMP	RTC tamper timestamp EXTI line
EXTI_LINE_21_CMP0_OUTPUT	CMP0 output EXTI line
EXTI_LINE_22_CMP1_OUTPUT	CMP1 output EXTI line
EXTI_LINE_25_USART0_WAKEUP	USART0 WAKEUP EXTI line
EXTI_LINE_27_CEC_WAKEUP	CEC WAKEUP EXTI line

Enum hal_exti_irq_index

Table 3-187. hal_exti_irq_index

enum name	enum description
EXTI_0_1_IRQHandler_USED	EXTI line 0 and 1 interrupts
EXTI_2_3_IRQHandler	EXTI line 2 and 3 interrupts

enum name	enum description
ler_USED	
EXTI_4_15_IRQHandler_USED	EXTI line 4 to 15 interrupts

hal_exti_gpio_deinit

The description of hal_exti_gpio_deinit is shown as below:

Table 3-188. Function hal_exti_gpio_deinit

Function name	hal_exti_gpio_deinit
Function prototype	int32_t hal_exti_gpio_deinit(uint32_t gpio_periph, uint32_t pin)
Function descriptions	deinitialize the EXTI gpio
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO pin
<i>GPIOx</i>	x = A,B,C, D,F
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	x=0..15
<i>GPIO_PIN_ALL</i>	all pins
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_VAL / HAL_ERR_NONE

Example:

```
/* deinitialize the EXTI GPIO */
```

```
hal_exti_gpio_deinit(GPIOA, GPIO_PIN_9);
```


hal_exti_internal_deinit

The description of hal_exti_internal_deinit is shown as below:

Table 3-189. Function hal_exti_internal_deinit

Function name	hal_exti_internal_deinit
Function prototype	void hal_exti_internal_deinit(hal_exti_internal_line_enum line)
Function descriptions	deinitialize the EXTI internal line
Precondition	-
The called functions	-
Input parameter{in}	
line	EXTI internal line, the enum members can refer to Table 3-186. hal_exti_internal_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the RTC_ALARM line */
hal_exti_internal_deinit(EXTI_LINE_17_RTC_ALARM);
```

hal_exti_gpio_init

The description of hal_exti_gpio_init is shown as below:

Table 3-190. Function hal_exti_gpio_init

Function name	hal_exti_gpio_init
Function prototype	int32_t hal_exti_gpio_init(uint32_t gpio_periph, uint32_t pin, uint32_t pull, hal_exti_type_enum exti_type)
Function descriptions	initialize the configuration of EXTI gpio
Precondition	-

The called functions	-
Input parameter{in}	
gpio_periph	gpio periph
GPIOx	x = A,B,C, D,F
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	x=0..15
GPIO_PIN_ALL	all pins
Input parameter{in}	
pull	gpio pin with pull-up or pull-down resistor
HAL_GPIO_PULL_NONE	floating mode, no pull-up and pull-down resistors
HAL_GPIO_PULL_UP	with pull-up resistor
HAL_GPIO_PULL_DOWN	with pull-down resistor
Input parameter{in}	
exti_type	EXTI type, the enum members can refer to Table 3-184. hal_exti_type_enum
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_VAL / HAL_ERR_ALREADY_DONE / HAL_ERR_NONE

Example:

```
/* initialize the configuration of EXTI gpio */
```

```
hal_exti_gpio_init(GPIOA, GPIO_PIN_9, HAL_GPIO_PULL_NONE, EXTI_EVENT_TRIG_RISING);
```

hal_exti_internal_init

The description of hal_exti_internal_init is shown as below:

Table 3-191. Function `hal_exti_internal_init`

Function name	<code>hal_exti_internal_init</code>
Function prototype	<code>void hal_exti_internal_init(hal_exti_internal_line_enum line, hal_exti_type_enum exti_type)</code>
Function descriptions	initialize the configuration of EXTI internal
Precondition	-
The called functions	-
Input parameter{in}	
line	EXTI internal line, the enum members can refer to Table 3-186. <code>hal_exti_internal_line_enum</code>
Input parameter{in}	
exti_type	EXTI type, the enum members can refer to Table 3-184. <code>hal_exti_type_enum</code>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the configuration of EXTI internal */
```

```
hal_exti_internal_init (EXTI_LINE_17_RTC_ALARM, EXTI_EVENT_TRIG_RISING);
```

`hal_exti_gpio_irq_handle_set`

The description of `hal_exti_gpio_irq_handle_set` is shown as below:

Table 3-192. Function `hal_exti_gpio_irq_handle_set`

Function name	<code>hal_exti_gpio_irq_handle_set</code>
Function prototype	<code>int32_t hal_exti_gpio_irq_handle_set(hal_gpio_irq_handle_cb irq_handler)</code>
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered

Precondition	-
The called functions	-
Input parameter{in}	
irq_handler	configuration the EXTI callback function, implemented by the user himself
输出参数{out}	
-	-
返回值	
int32_t	HAL_ERR_ADDRESS / HAL_ERR_NONE

Example:

```

/* set EXTI line 13 interrupt callback function */
void s_gpio_irq_handle(uint32_t id, uint32_t event)
{
    if(EXTI_PIN_13 == id){
        gd_eval_led_toggle(LED2);
    }
}

hal_exti_gpio_irq_handle_set(exti_irq_handler);

```

hal_exti_gpio_irq_handle_all_reset

The description of hal_exti_gpio_irq_handle_all_reset is shown as below:

Table 3-193. Function hal_exti_gpio_irq_handle_all_reset

Function name	hal_exti_gpio_irq_handle_all_reset
Function prototype	void hal_exti_gpio_irq_handle_all_reset(void)
Function descriptions	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-

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

Example:

```
/* reset all user-defined interrupt callback function */
hal_exti_gpio_irq_handle_all_reset();
```

hal_exti_gpio_irq

The description of hal_exti_gpio_irq is shown as below:

Table 3-194. Function hal_exti_gpio_irq

Function name	hal_exti_gpio_irq
Function prototype	void hal_exti_gpio_irq(hal_exti_irq_index index)
Function descriptions	EXTI_GPIO interrupt handler content function, which is merely used in EXTI_GPIO_handler
Precondition	-
The called functions	-
Input parameter{in}	
index	indicate this function will be called by which interrupt handler entry, the enum members can refer to Table 3-187. hal_exti_irq_index
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* EXTI_GPIO interrupt handler content function */
```

```
hal_exti_gpio_irq(EXTI_0_1_IRQHandler_USED);
```

3.11. FMC

There is flash controller and option byte for GD32F3x0 series. The FMC registers are listed in chapter [3.11.1](#) the FMC firmware functions are introduced in chapter [3.11.2](#).

3.11.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

Table 3-1. FMC registers

Registers	Descriptions
FMC_WS	FMC wait state register
FMC_KEY	FMC unlock key register
FMC_OBKEY	FMC option bytes unlock key register
FMC_STAT	FMC status register
FMC_CTL	FMC control register
FMC_ADDR	FMC address register
FMC_OBSTAT	FMC option bytes status register
FMC_WP	FMC write protection register
FMC_WSEN	FMC wait state enable register
FMC_PID	FMC product ID register

3.11.2. Descriptions of Peripheral functions

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

Table 3-2. FMC firmware function

Function name	Function description
hal_fmc_unlock	unlock the main FMC operation
hal_fmc_lock	lock the main FMC operation
hal_fmc_wait_state_enable	fmc wait state enable
hal_fmc_wait_state_disable	fmc wait state disable
hal_fmc_ready_wait	check whether FMC is ready or not
hal_fmc_wscnt_set	set the wait state counter value
hal_fmc_region_read	read the target region
hal_fmc_word_program	write flash target address in word
hal_fmc_halfword_program	write flash target address in halfword
hal_fmc_region_write	write flash target region with amounts of data
hal_fmc_page_erase	erase the page which start address locating in
hal_fmc_mass_erase	erase the whole chip

Function name	Function description
hal_fmc_region_erase	erase flash target region
hal_fmc_irq	fmc interrupt handler content function, which is merely used in fmc_handler
hal_fmc_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_fmc_irq_handle_all_reset	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_ob_unlock	unlock option byte
hal_ob_lock	lock option byte
hal_ob_reset	reset option byte
hal_ob_erase	erase option byte
hal_ob_security_protection_config	configure security protection
hal_ob_user_write	write option byte user
hal_ob_data_program	program option byte data
hal_ob_wp_enable	enable the targeted address region written protection
hal_ob_wp_disable	disable the targeted address region written protection
hal_ob_parm_get	get currently efficient option byte
hal_ob_write_protection_enable	enable option byte write protection(OB_WP) depending on current option byte
hal_ob_parm_config	configure option byte parameters thoroughly

Enum fmc_state_enum

Table 3-3. fmc_state_enum

enum name	enum description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_PGERR	program error
FMC_PGAERR	program alignment error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error
FMC_OB_HSPC	option byte security protection high level

Structure ob_parm_struct

Table 3-4. Structure ob_parm_struct

Member name	Function description
spc	option byte parameter spc
user	option byte parameter user
data0	option byte parameter data0
data1	option byte parameter data1

Member name	Function description
wp0	option byte parameter wp0
wp1	option byte parameter wp1

Structure hal_sector_addr_range_struct

Table 3-5. Structure hal_sector_addr_range_struct

Member name	Function description
sector_start_addr	sector actual start address
sector_end_addr	sector actual end address

Structure hal_fmc_irq_struct

Table 3-6. Structure hal_fmc_irq_struct

Member name	Function description
error_handle	flash error interrupt
finish_handle	end of operation interrupt

Structure hal_ob_parm_config_struct

Table 3-7. Structure hal_ob_parm_config_struct

Member name	Function description
ob_type	ob_type: OB_TYPE_WRP,OB_TYPE_SPC,OB_TYPE_USER and OB_TYPE_DATA. one or more parameters can be selected which are shown as above
wrp_state	wrp_state: OB_WRP_DISABLE or OB_WRP_ENABLE
wrp_addr	wrp_addr: specifies the target start address
wrp_size	wrp_size: specifies the data size to be write protected
spc_level	spc_level: OB_SPC_0, OB_SPC_1 or OB_SPC_2. only one parameter can be selected which is shown as above
user	user: the value is used to configure OB_USER
data_value	data_value: the low byte of data_value is used to configure OB_DATA0 , the high byte of data_value is used to configure OB_DATA1

hal_fmc_unlock

The description of hal_fmc_unlock is shown as below:

Table 3-8. Function hal_fmc_unlock

Function name	hal_fmc_unlock
Function prototype	void hal_fmc_unlock(void);
Function descriptions	unlock the main FMC operation
Precondition	-
Input parameter{in}	

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

Example:

```
/* unlock the main FMC operation */
```

```
hal_fmc_unlock();
```

hal_fmc_lock

The description of hal_fmc_lock is shown as below:

Table 3-9. Function hal_fmc_lock

Function name	hal_fmc_lock
Function prototype	void hal_fmc_lock(void);
Function descriptions	lock the main FMC operation
Precondition	hal_fmc_unlock
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the main FMC operation */
```

```
hal_fmc_lock();
```

hal_fmc_wait_state_enable

The description of hal_fmc_wait_state_enable is shown as below:

Table 3-10. Function hal_fmc_wait_state_enable

Function name	hal_fmc_wait_state_enable
Function prototype	void hal_fmc_wait_state_enable(void);
Function descriptions	enable fmc wait state
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* enable fmc wait state */
```

```
hal_fmc_wait_state_enable();
```

hal_fmc_wait_state_disable

The description of hal_fmc_wait_state_disable is shown as below:

Table 3-11. Function hal_fmc_wait_state_disable

Function name	hal_fmc_wait_state_disable
Function prototype	void hal_fmc_wait_state_disable(void);
Function descriptions	disable fmc wait state
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable fmc wait state */
```

```
hal_fmc_wait_state_disable();
```

hal_fmc_ready_wait

The description of hal_fmc_ready_wait is shown as below:

Table 3-12. Function hal_fmc_ready_wait

Function name	hal_fmc_ready_wait
Function prototype	fmc_state_enum hal_fmc_ready_wait(uint32_t timeout);
Function descriptions	check whether FMC is ready or not
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* check whether FMC is ready or not */
```

```
hal_fmc_ready_wait(0xFF);
```

hal_fmc_wscnt_set

The description of hal_fmc_wscnt_set is shown as below:

Table 3-13. Function hal_fmc_wscnt_set

Function name	hal_fmc_wscnt_set
Function prototype	void hal_fmc_wscnt_set(uint8_t wscnt);
Function descriptions	set the wait state counter value
Precondition	-
Input parameter{in}	
wscnt	wait state counter value
WS_WSCNT_0	FMC 0 wait state added
WS_WSCNT_1	FMC 1 wait state added
WS_WSCNT_2	FMC 2 wait state added
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
hal_fmc_wscnt_set (WS_WSCNT_1);
```

hal_fmc_region_read

The description of hal_fmc_region_read is shown as below:

Table 3-14. Function hal_fmc_region_read

Function name	hal_fmc_region_read
Function prototype	void hal_fmc_region_read(uint32_t start_addr, uint8_t *data, uint32_t size);
Function descriptions	read flash target region
Precondition	-
Input parameter{in}	
start_addr	target region start address

Input parameter{in}	
data	pointer to read result
Input parameter{in}	
size	target region size
Output parameter{out}	
data	pointer to read result
Return value	
-	-

Example:

```
/* read flash target region */
```

```
uint8_t *data;
```

```
hal_fmc_region_read(0x08004000, data, 0x400);
```

hal_fmc_word_program

The description of hal_fmc_word_program is shown as below:

Table 3-15. Function hal_fmc_word_program

Function name	hal_fmc_word_program
Function prototype	fmc_state_enum hal_fmc_word_program(uint32_t addr, uint32_t data);
Function descriptions	write flash target address in word
Precondition	hal_fmc_unlock
Input parameter{in}	
addr	target address
Input parameter{in}	
data	target data
Output parameter{out}	
-	-
Return value	
fmc_state_enum	refer to Enum fmc state enum

Example:

```
/* write flash target address in word */
```

```
hal_fmc_word_program(0x8004000, 0x12345678);
```

hal_fmc_halfword_program

The description of hal_fmc_halfword_program is shown as below:

Table 3-16. Function `hal_fmc_halfword_program`

Function name	hal_fmc_halfword_program
Function prototype	fmc_state_enum hal_fmc_halfword_program(uint32_t addr, uint16_t data);
Function descriptions	write flash target address in halfwords words
Precondition	hal_fmc_unlock
Input parameter{in}	
addr	target address
Input parameter{in}	
data	target data
Output parameter{out}	
-	-
Return value	
fmc_state_enum	refer to Enum fmc state enum

Example:

```
/* write flash target address in halfword */
```

```
fmc_state_enum state = hal_fmc_halfword_program (0x08004000, 0xaabb);
```

hal_fmc_region_write

The description of `hal_fmc_region_write` is shown as below:

Table 3-17. Function `hal_fmc_region_write`

Function name	hal_fmc_region_write
Function prototype	int32_t hal_fmc_region_write(uint32_t start_addr, uint8_t *data, uint32_t size);
Function descriptions	write flash target region with amounts of data
Precondition	hal_fmc_unlock
Input parameter{in}	
start_addr	target region start address
Input parameter{in}	
data	pointer to read result
Input parameter{in}	
size	target region size
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
/* write flash target region with amounts of data */
```

```
uint8_t data[0x400] = { 0 };
```

```
fmc_state_enum state = hal_fmc_region_write (0x08004000, data, 0x400);
```

hal_fmc_page_erase

The description of hal_fmc_page_erase is shown as below:

Table 3-18. Function hal_fmc_page_erase

Function name	hal_fmc_page_erase
Function prototype	fmc_state_enum hal_fmc_page_erase(uint32_t start_addr);
Function descriptions	erase the page which start address locating in
Precondition	hal_fmc_unlock
Input parameter{in}	
start_addr	target region start address
Output parameter{out}	
-	-
Return value	
fmc_state_enum	refer to Enum fmc state enum

Example:

```
/* erase the page which start address locating in */
```

```
fmc_state_enum state = hal_fmc_page_erase(0x8004000);
```

hal_fmc_mass_erase

The description of hal_fmc_mass_erase is shown as below:

Table 3-19. Function hal_fmc_mass_erase

Function name	hal_fmc_mass_erase
Function prototype	fmc_state_enum hal_fmc_mass_erase(void);
Function descriptions	erase the whole flash
Precondition	hal_fmc_unlock
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	refer to Enum fmc state enum

Example:

```
/* erase the whole flash */
```

```
fmc_state_enum state = hal_fmc_mass_erase();
```

hal_fmc_region_erase

The description of hal_fmc_region_erase is shown as below:

Table 3-20. Function hal_fmc_region_erase

Function name	hal_fmc_region_erase
Function prototype	int32_t hal_fmc_region_erase(uint32_t start_addr, uint32_t size);
Function descriptions	erase flash target region
Precondition	hal_fmc_unlock
Input parameter{in}	
start_addr	target region start address
Input parameter{in}	
size	target region size
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
/* erase flash target region */
hal_fmc_region_erase(0x8004000, 0x400);
```

hal_fmc_irq

The description of hal_fmc_irq is shown as below:

Table 3-21. Function hal_fmc_irq

Function name	hal_fmc_irq
Function prototype	void hal_fmc_irq();
Function descriptions	fmc interrupt handler content function, which is merely used in fmc_handler
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* process the FMC interrupt */
```

```
hal_uart_irq();
```

hal_fmc_irq_handle_set

The description of hal_fmc_irq_handle_set is shown as below:

Table 3-22. Function hal_fmc_irq_handle_set

Function name	hal_fmc_irq_handle_set
Function prototype	void hal_fmc_irq_handle_set(hal_fmc_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
Input parameter{in}	
p_irq	pointer to fmc interrupt handler callback function Structure hal_fmc_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* erase the FMC option byte */
```

```
hal_fmc_irq_struct p_irq;
```

```
hal_fmc_irq_handle_set(&p_irq);
```

hal_fmc_irq_handle_all_reset

The description of hal_fmc_irq_handle_all_reset is shown as below:

Table 3-23. Function hal_fmc_irq_handle_all_reset

Function name	hal_fmc_irq_handle_all_reset
Function prototype	void hal_fmc_irq_handle_all_reset(hal_fmc_irq_struct *p_irq);
Function descriptions	reset user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
Input parameter{in}	
p_irq	pointer to fmc interrupt handler callback function Structure hal_fmc_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_fmc_irq_struct p_irq;

hal_fmc_irq_handle_reset(&p_irq);
```

hal_ob_unlock

The description of hal_ob_unlock is shown as below:

Table 3-24. Function hal_ob_unlock

Function name	hal_ob_unlock
Function prototype	void hal_ob_unlock(void);
Function descriptions	unlock option byte
Precondition	hal_fmc_unlock
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock option byte */

void hal_ob_unlock(void);
```

hal_ob_lock

The description of hal_ob_lock is shown as below:

Table 3-25. Function hal_ob_lock

Function name	hal_ob_lock
Function prototype	void hal_ob_lock(void);
Function descriptions	lock option byte
Precondition	hal_ob_unlock
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock option byte */
```

```
hal_ob_lock();
```

hal_ob_reset

The description of hal_ob_reset is shown as below:

Table 3-26. Function hal_ob_reset

Function name	hal_ob_reset
Function prototype	void hal_ob_reset(void);
Function descriptions	reset option byte
Precondition	hal_ob_unlock
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset option byte */
```

```
hal_ob_reset();
```

hal_ob_erase

The description of hal_ob_erase is shown as below:

Table 3-27. Function hal_ob_erase

Function name	hal_ob_erase
Function prototype	fmc_state_enum hal_ob_erase(void);
Function descriptions	erase option byte
Precondition	hal_ob_unlock
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	refer to Enum fmc state enum

Example:

```
/* erase option byte */
```

```
fmc_state_enum state = hal_ob_erase();
```

hal_ob_security_protection_config

The description of hal_ob_security_protection_config is shown as below:

Table 3-28. Function hal_ob_security_protection_config

Function name	hal_ob_security_protection_config
Function prototype	fmc_state_enum hal_ob_security_protection_config(uint8_t ob_spc);
Function descriptions	configure option byte security protection
Precondition	hal_ob_unlock
Input parameter{in}	
ob_spc	specify security protection code
<i>FMC_NSPC</i>	no security protection
<i>FMC_LSPC</i>	low security protection
<i>FMC_HSPC</i>	high security protection
Output parameter{out}	
-	-
Return value	
fmc_state_enum	refer to Enum fmc_state_enum

Example:

```
/* get the FMC data option byte */
```

```
fmc_state_enum state = hal_ob_security_protection_config(FMC_LSPC);
```

hal_ob_user_write

The description of hal_ob_user_write is shown as below:

Table 3-29. Function hal_ob_user_write

Function name	hal_ob_user_write
Function prototype	fmc_state_enum hal_ob_user_write(uint8_t ob_user);
Function descriptions	write option byte user
Precondition	hal_ob_unlock
Input parameter{in}	
ob_user	user option byte
<i>OB_FWDGT_HW</i>	hardware free watchdog timer
<i>OB_DEEPSLEEP_RST</i>	no reset when entering deepsleep mode
<i>OB_STDBY_RST</i>	no reset when entering standby mode
<i>OB_BOOT1_SET_1</i>	Set BOOT1 bit is 1

<i>OB_VDDA_DISABLE</i>	disable VDDA monitor
<i>OB_SRAM_PARITY_ENABLE</i>	enable SRAM parity check
Output parameter{out}	
-	-
Return value	
fmc_state_enum	refer to Enum fmc_state_enum

Example:

```
/* write option byte user */
```

```
fmc_state_enum state = hal_ob_user_write (OB_FWDGT_HW);
```

hal_ob_data_program

The description of hal_ob_data_program is shown as below:

Table 3-30. Function hal_ob_data_program

Function name	hal_ob_data_program
Function prototype	fmc_state_enum hal_ob_data_program(uint16_t ob_data);
Function descriptions	program the FMC data option byte
Precondition	hal_ob_unlock
Input parameter{in}	
ob_data	option byte data value
Output parameter{out}	
-	-
Return value	
fmc_state_enum	refer to Enum fmc_state_enum

Example:

```
/* program the FMC data option byte */
```

```
fmc_state_enum state = hal_ob_data_program (0x1357) ;
```

hal_ob_wp_enable

The description of hal_ob_wp_enable is shown as below:

Table 3-31. Function hal_ob_wp_enable

Function name	hal_ob_wp_enable
Function prototype	hal_sector_addr_range_struct hal_ob_wp_enable(uint32_t start_addr, uint32_t data_size);
Function	enable the targeted address region written protection

descriptions	
Precondition	hal_ob_unlock
Input parameter{in}	
start_addr	target region start address
Input parameter{in}	
data_size	target region size
Output parameter{out}	
-	-
Return value	
hal_sector_addr_range	refer to Structure hal_sector_addr_range_struct

Example:

```
/* enable the targeted address region written protection */
```

```
hal_sector_addr_range_struct sector = hal_ob_wp_enable(0x8004000, 0x1000);
```

hal_ob_wp_disable

The description of hal_ob_wp_disable is shown as below:

Table 3-32. Function hal_ob_wp_disable

Function name	hal_ob_wp_disable
Function prototype	hal_sector_addr_range_struct hal_ob_wp_disable(uint32_t start_addr, uint32_t data_size);
Function descriptions	disable the targeted address region written protection
Precondition	hal_ob_unlock
Input parameter{in}	
start_addr	target region start address
Input parameter{in}	
data_size	target region size
Output parameter{out}	
-	-
Return value	
hal_sector_addr_range	refer to Structure hal_sector_addr_range_struct

Example:

```
/* disable the targeted address region written protection */
```

```
hal_sector_addr_range_struct sector = hal_ob_wp_disable(0x8004000, 0x1000);
```

hal_ob_parm_get

The description of hal_ob_parm_get is shown as below:

Table 3-33. Function hal_ob_parm_get

Function name	hal_ob_parm_get
Function prototype	int32_t hal_ob_parm_get(ob_parm_struct *p_parm)
Function descriptions	get currently efficient option byte
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
p_parm	refer to Structure ob_parm_struct
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
/* get option byte parameters, stored in register FMC_OBSTAT and FMC_WP */
ob_parm_struct ob_parm_get;
hal_ob_parm_get(&ob_parm_get);
```

hal_ob_write_protection_enable

The description of hal_ob_write_protection_enable is shown as below:

Table 3-34. Function hal_ob_write_protection_enable

Function name	hal_ob_write_protection_enable
Function prototype	fmc_state_enum hal_ob_write_protection_enable(uint16_t ob_wp)
Function descriptions	enable option byte write protection(OB_WP) depending on current option byte
Precondition	-
Input parameter{in}	
ob_wp	0-0xFFFF
Output parameter{out}	
-	-
Return value	
fmc_state_enum	refer to Enum fmc_state_enum

Example:

```
/* enable option byte write protection(OB_WP) depending on current option byte */
hal_ob_write_protection_enable(0x01);
```

hal_ob_parm_config

The description of hal_ob_parm_config is shown as below:

Table 3-35. Function hal_ob_parm_config

Function name	hal_ob_parm_config
Function prototype	int32_t hal_ob_parm_config(hal_ob_parm_config_struct *ob_parm);
Function descriptions	configure option byte parameters thoroughly
Precondition	hal_ob_unlock
Input parameter{in}	
ob_parm	refer to Structure hal_ob_parm_config_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```

/* configure option byte parameters thoroughly */

hal_ob_parm_config_struct ob_parm_config;

ob_parm_config.ob_type    =    OB_TYPE_WRP|OB_TYPE_SPC|OB_TYPE_USER    |
OB_TYPE_DATA;

ob_parm_config.spc_level = OB_SPC_1;

ob_parm_config.user = 0x43;

ob_parm_config.data_value = 0x1234;

ob_parm_config.wrp_addr = 0x8006000;

ob_parm_config.wrp_size = 0x5000;

ob_parm_config.wrp_state = OB_WRP_ENABLE;

hal_ob_parm_config(&ob_parm_config);

```

3.12. FWDGT

The free watchdog timer (FWDGT) is a hardware timing circuitry that can be used to detect system failures due to software malfunctions. It's suitable for the situation that requires an independent environment and lower timing accuracy. The FWDGT registers are listed in chapter [3.12.1](#) the FWDGT firmware functions are introduced in chapter [3.12.2](#)

3.12.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

Table 3-195. FWDGT Registers

Registers	Descriptions
FWDGT_CTL	Control register
FWDGT_PSC	Prescaler register
FWDGT_RLD	Reload register
FWDGT_STAT	Status register
FWDGT_WND	window register

3.12.2. Descriptions of Peripheral functions

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

Table 3-196. FWDGT firmware function

Function name	Function description
hal_fwdgt_struct_init	initialize the specified structure
hal_fwdgt_init	initialize FWDGT
hal_fwdgt_deinit	deinitialize FWDGT

Enum hal_fwdgt_state_enum

Table

3-197. hal_fwdgt_state_enum

enum name	enum description
HAL_FWDGT_STATE_NONE	none
HAL_FWDGT_STATE_RESET	FWDGT status reset
HAL_FWDGT_STATE_READY	FWDGT ready
HAL_FWDGT_STATE_BUSY	FWDGT busy

Enum `hal_fwdgt_struct_type_enum`

Table 3-198. `hal_fwdgt_struct_type_enum`

enum name	enum description
HAL_FWDGT_INIT_STRUCT	FWDGT initialize structure
HAL_FWDGT_DEV_STRUCT	FWDGT device information structure

Structure `hal_fwdgt_dev_struct`

Table 3-199. `hal_fwdgt_dev_struct`

Member name	Function description
state	FWDGT status
mutex	mutex locked and unlocked state

Structure `hal_fwdgt_init_struct`

Table 3-200. `hal_fwdgt_init_struct`

Member name	Function description
fwdgt_pre_select	FWDGT prescaler value
wdgt_cnt_value	FWDGT counter window value
fwdgt_cnt_reload_value	FWDGT counter reload value

`hal_fwdgt_struct_init`

The description of `hal_fwdgt_struct_init` is shown as below:

Table 3-201. Function `hal_fwdgt_struct_init`

Function name	<code>hal_fwdgt_struct_init</code>
Function prototype	<code>void hal_fwdgt_struct_init(hal_fwdgt_struct_type_enum hal_struct_type, void *p_struct);</code>
Function descriptions	initialize the specified structure
Precondition	-

The called functions	-
Input parameter{in}	
hal_struct_type	pointer to hal_fwdgt_struct_type_enum, refer to Table 3-198. hal_fwdgt_struct_type_enum
Input parameter{in}	
*p_struct	pointer to the FWDGT structure that contains configuration information
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the FWDGT STRUCT */
```

```
hal_fwdgt_struct_init(HAL_FWDGT_INIT_STRUCT, &fwdgt_init_parameter);
```

hal_fwdgt_init

The description of hal_fwdgt_init is shown as below:

Table 3-202. Function hal_fwdgt_init

Function name	hal_fwdgt_init
Function prototype	int32_t hal_fwdgt_init(hal_fwdgt_dev_struct *fwdgt_dev, hal_fwdgt_init_struct *p_fwdgt_init);
Function descriptions	initialize FWDGT
Precondition	-
The called functions	-
Input parameter{in}	
fwdgt_dev	pointer to FWDGT device information structure, structure member refer to Table 3-199. hal_fwdgt_dev_struct
Input parameter{in}	

p_fwdgt_init	pointer to FWDGT initialize structure, structure member refer to Table 3-200. <u>hal_fwdgt_init_struct</u>
Output parameter{out}	
-	-
Return value	
error code	HAL_ERR_VAL / HAL_ERR_NONE

例如:

```
/* initialize the FWDGT */
```

```
hal_fwdgt_init(&fwdgt_info,&fwdgt_init_parameter);
```

hal_fwdgt_deinit

The description of hal_fwdgt_deinit is shown as below:

Table 3-203. Function hal_fwdgt_deinit

Function name	hal_fwdgt_deinit
Function prototype	void hal_fwdgt_deinit(hal_fwdgt_dev_struct *fwdgt_dev);
Function descriptions	
Precondition	-
The called functions	-
Input parameter{in}	
fwdgt_dev	pointer to FWDGT device information structure, structure member refer to Table 3-199. <u>hal_fwdgt_dev_struct</u>
Output parameter{out}	
-	-
Return value	
-	-

例如:

```
/* deinitialize the configuration of FWDGT */
```

```
hal_fwdgt_deinit(&fwdgt_info);
```

3.13. GPIO

GPIO is used to implement logic input/output functions for the devices. The GPIO registers are listed in chapter [3.13.1](#), the GPIO firmware functions are introduced in chapter [3.13.2](#).

3.13.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

Table 3-204. GPIO Registers

Registers	Descriptions
GPIOx_CTL	GPIO port control register
GPIOx_OMODE	GPIO port output mode register
GPIOx_OSPD0	GPIO port output speed register 0
GPIOx_PUD	GPIO port pull-up/pull-down register
GPIOx_ISTAT	GPIO port input status register
GPIOx_OCTL	GPIO port output control register
GPIOx_BOP	GPIO port bit operation register
GPIOx_LOCK	GPIO port configuration lock register
GPIOx_AFSEL0	GPIO alternate function selected register 0
GPIOx_AFSEL1	GPIO alternate function selected register 1
GPIOx_BC	GPIO bit clear register
GPIOx_TG	GPIO port bit toggle register
GPIOx_OSPD1	GPIO port output speed register 1

3.13.2. Descriptions of Peripheral functions

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

Table 3-205. GPIO firmware function

Function name	Function description
hal_gpio_init	initialize GPIO
hal_gpio_bit_set	set GPIO pin bit
hal_gpio_bit_reset	reset GPIO pin bit
hal_gpio_struct_init	initialize the GPIO initialization structure with the default

Function name	Function description
	values
hal_gpio_deinit	deinitialize GPIO

Enum hal_gpio_mode_enum

Table 3-206. Enum hal_gpio_mode_enum

Member name	Function description
GPIO_MODE_ANALOG	analog mode
GPIO_MODE_INPUT	input floating mode
GPIO_MODE_OUTPUT_PP	output push pull mode
GPIO_MODE_OUTPUT_OD	output open drain mode
GPIO_MODE_AF_PP	alternate function push pull mode
GPIO_MODE_AF_OD	alternate function open drain mode

Enum hal_gpio_pull_enum

Table 3-207. Enum hal_gpio_pull_enum

Member name	Function description
GPIO_PULL_NONE	floating mode, no pull-up and pull-down resistors
GPIO_PULL_UP	with pull-up resistor
GPIO_PULL_DOWN	with pull-down resistor

Enum hal_gpio_ospeed_enum

Table 3-208. Enum hal_gpio_ospeed_enum

Member name	Function description
GPIO_OSPEED_2MHZ	output max speed 2MHz
GPIO_OSPEED_10MHZ	output max speed 10MHz
GPIO_OSPEED_50MHZ	output max speed 50MHz
GPIO_OSPEED_MAX	GPIO very high output speed, max speed more than 50MHz

Enum hal_gpio_af_enum

Table 3-209. Enum hal_gpio_af_enum

Member name	Function description
GPIO_AF_0	alternate function 0 selected
GPIO_AF_1	alternate function 1 selected
GPIO_AF_2	alternate function 2 selected
GPIO_AF_3	alternate function 3 selected
GPIO_AF_4	alternate function 4 selected (port A,B only)
GPIO_AF_5	alternate function 5 selected (port A,B only)
GPIO_AF_6	alternate function 6 selected (port A,B only)

GPIO_AF_7	alternate function 7 selected (port A,B only)
-----------	---

Structure hal_gpio_init_struct

Table 3-210. Structure hal_gpio_init_struct

Member name	Function description
mode	GPIOx output\input mode, refer to Table 3-206. Enum hal_gpio_mode_enum
pull	GPIOx pull-up\down mode, refer to Table 3-207. Enum hal_gpio_pull_enum .
ospeed	output max speed, refer to Table 3-208. Enum hal_gpio_ospeed_enum
af	GPIOx alternate function select, refer to Table 3-209. Enum hal_gpio_af_enum .

hal_gpio_init

The description of hal_gpio_init is shown as below:

Table 3-211. Function hal_gpio_init

Function name	hal_gpio_init
Function prototype	int32_t hal_gpio_init(uint32_t gpio_periph, uint32_t pin, hal_gpio_init_struct *p_init);
Function descriptions	initialize the GPIO
Precondition	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
pin	gpio pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	All pins
Input parameter{in}	
p_init	Structure for initialization, the structure members can refer to Table 3-210. Structure hal_gpio_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

```
/* configure GPIO pin : PB6 */
```

```
hal_gpio_init_struct gpio_init_parameter;
```

```
gpio_init_parameter.mode = GPIO_MODE_OUTPUT_PP;
```

```
gpio_init_parameter.pull = GPIO_PULL_NONE;
```

```

gpio_init_parameter.ospeed = GPIO_OSPEED_50MHZ;
gpio_init_parameter.af = GPIO_AF_0;
hal_gpio_init(GPIOB, GPIO_PIN_6, &gpio_init_parameter);

```

hal_gpio_bit_set

The description of hal_gpio_bit_set is shown as below:

Table 3-212. Function hal_gpio_bit_set

Function name	hal_gpio_bit_set
Function prototype	void hal_gpio_bit_set(uint32_t gpio_periph, uint32_t pin);
Function descriptions	set GPIO pin bit
Precondition	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* set PA0*/
hal_gpio_bit_set (GPIOA, GPIO_PIN_0);

```

hal_gpio_bit_reset

The description of hal_gpio_bit_reset is shown as below:

Table 3-213. Function hal_gpio_bit_reset

Function name	hal_gpio_bit_reset
Function prototype	void hal_gpio_bit_reset(uint32_t gpio_periph, uint32_t pin);
Function descriptions	reset GPIO pin bit
Precondition	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
pin	GPIO pin

<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset PA0*/
hal_gpio_bit_reset (GPIOA, GPIO_PIN_0);
```

hal_gpio_struct_init

The description of hal_gpio_struct_init is shown as below:

Table 3-214. Function hal_gpio_struct_init

Function name	hal_gpio_struct_init
Function prototype	int32_t hal_gpio_struct_init(hal_gpio_init_struct *p_init);
Function descriptions	initialize the GPIO initialization structure with the default values
Precondition	-
The called functions	
Input parameter{in}	
p_init	GPIO initialization structure
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* initialize the GPIO initialization structure */
hal_gpio_init_struct gpio_init_parameter;
hal_gpio_struct_init(&gpio_init_parameter);
```

hal_gpio_deinit

The description of hal_gpio_deinit is shown as below:

Table 3-215. Function hal_gpio_deinit

Function name	hal_gpio_deinit
Function prototype	int32_t hal_gpio_deinit(uint32_t gpio_periph, uint32_t pin);
Function descriptions	deinitialize GPIO
Precondition	-
Input parameter{in}	
gpio_periph	GPIO port

<i>GPIOx</i>	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
pin	gpio pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```
* deinitialize GPIO pin : PB6 */
hal_gpio_deinit(GPIOB, GPIO_PIN_6);
```

3.14. I2C

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. The I2C registers are listed in chapter [3.14.1](#), the I2C firmware functions are introduced in chapter [3.14.2](#).

3.14.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

Table 3-216. I2C Registers

Registers	Descriptions
I2C_CTL0	Control register 0
I2C_CTL1	Control register 1
I2C_SADDR0	Slave address register 0
I2C_SADDR1	Slave address register 1
I2C_DATA	Transfer buffer register
I2C_STAT0	Transfer status register 0
I2C_STAT1	Transfer status register 1
I2C_CKCFG	Clock configure register
I2C_RT	Rise time register
I2C_FMPCFG	Fast mode plus configure register

3.14.2. Descriptions of Peripheral functions

I2C firmware functions are listed in the table shown as below:

Table 3-217. I2C firmware function

Function name	Function description
hal_i2c_struct_init	initialize the I2C structure with the default values
hal_i2c_deinit	deinitialize I2C module
hal_i2c_init	initialize I2C module
hal_i2c_error_irq	I2C Error Interrupt handler
hal_i2c_event_irq	I2C Event Interrupt handler
hal_i2c_start	start I2C module function
hal_i2c_stop	stop I2C module function
hal_i2c_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_i2c_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_i2c_master_transmit_poll	transmit amounts of data in master mode, poll transmit process and completed status
hal_i2c_master_receive_poll	receive amounts of data in master mode, poll receive process and completed status
hal_i2c_slave_transmit_poll	transmit amounts of data in slave mode, poll transmit process and completed status
hal_i2c_slave_receive_poll	receive amounts of data in slave mode, poll receive process and completed status
hal_i2c_memory_write_poll	write amounts of data to memory, poll transmit process and completed status
hal_i2c_memory_read_poll	read amounts of data from memory, poll transmit process and completed status
hal_i2c_master_transmit_interrupt	transmit amounts of data in master mode by interrupt method
hal_i2c_master_receive_interrupt	receive amounts of data in master mode by interrupt method
hal_i2c_slave_transmit_interrupt	transmit amounts of data in slave mode by interrupt method
hal_i2c_slave_receive_interrupt	receive amounts of data in slave mode by interrupt method
hal_i2c_memory_write_interrupt	write amounts of data to memory by interrupt method
hal_i2c_memory_read_interrupt	read amounts of data from memory by interrupt method
hal_i2c_master_transmit_dma	transmit amounts of data in master mode by dma method
hal_i2c_master_receive_dma	receive amounts of data in master mode by dma method
hal_i2c_slave_transmit_dma	transmit amounts of data in slave mode by dma method
hal_i2c_slave_receive_dma	receive amounts of data in slave mode by dma method
hal_i2c_memory_write_dma	write amounts of data to memory by dma method
hal_i2c_memory_read_dma	read amounts of data from memory by dma method

Function name	Function description
hal_i2c_device_ready_check	check whether the device is ready for access
hal_i2c_master_serial_transmit_interrupt	serial transmit amounts of data in master mode by interrupt method
hal_i2c_master_serial_receive_interrupt	serial receive amounts of data in master mode by interrupt method
hal_i2c_slave_serial_transmit_interrupt	serial transmit amounts of data in slave mode by interrupt method
hal_i2c_slave_serial_receive_interrupt	serial receive amounts of data in slave mode by interrupt method
hal_i2c_address_listen_interrupt_enable	enable address listen in slave mode by interrupt method
hal_i2c_address_listen_interrupt_disable	disable address listen in slave mode by interrupt method

Enum i2c_flag_enum

Table 3-218. Enum i2c_flag_enum

Member name	Function description
I2C_FLAG_SBSEND	start condition sent out in master mode
I2C_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode
I2C_FLAG_BTC	byte transmission finishes
I2C_FLAG_ADD10SEND	header of 10-bit address is sent in master mode
I2C_FLAG_STPDET	stop condition detected in slave mode
I2C_FLAG_RBNE	I2C_DATA is not empty during receiving
I2C_FLAG_TBE	I2C_DATA is empty during transmitting
I2C_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus
I2C_FLAG_LOSTARB	arbitration lost in master mode
I2C_FLAG_AERR	acknowledge error
I2C_FLAG_OUERR	over-run or under-run situation occurs in slave mode
I2C_FLAG_PECERR	PEC error when receiving data
I2C_FLAG_SMBTO	timeout signal in SMBus mode
I2C_FLAG_SMBALT	SMBus alert status
I2C_FLAG_MASTER	a flag indicating whether I2C block is in master or slave mode
I2C_FLAG_I2CBSY	busy flag
I2C_FLAG_TR	whether the I2C is a transmitter or a receiver
I2C_FLAG_RXGC	general call address (00h) received
I2C_FLAG_DEFSMB	default address of SMBus device
I2C_FLAG_HSTSMB	SMBus host header detected in slave mode
I2C_FLAG_DUMOD	dual flag in slave mode indicating which address is matched in dual-address

	mode
--	------

Enum i2c_interrupt_flag_enum

Table 3-219. Enum i2c_interrupt_flag_enum

Member name	Function description
I2C_INT_FLAG_SB SEND	start condition sent out in master mode interrupt flag
I2C_INT_FLAG_AD DSEND	address is sent in master mode or received and matches in slave mode interrupt flag
I2C_INT_FLAG_BT C	byte transmission finishes interrupt flag
I2C_INT_FLAG_AD D10SEND	header of 10-bit address is sent in master mode interrupt flag
I2C_INT_FLAG_ST PDET	stop condition detected in slave mode interrupt flag
I2C_INT_FLAG_RB NE	I2C_DATA is not empty during receiving interrupt flag
I2C_INT_FLAG_TB E	I2C_DATA is empty during transmitting interrupt flag
I2C_INT_FLAG_BE RR	a bus error occurs indication a unexpected start or stop condition on I2C bus interrupt flag
I2C_INT_FLAG_LO STARB	arbitration lost in master mode interrupt flag
I2C_INT_FLAG_AE RR	acknowledge error interrupt flag
I2C_INT_FLAG_OU ERR	over-run or under-run situation occurs in slave mode interrupt flag
I2C_INT_FLAG_PE CERR	PEC error when receiving data interrupt flag
I2C_INT_FLAG_SM BTO	timeout signal in SMBus mode interrupt flag
I2C_INT_FLAG_SM BALT	SMBus alert status interrupt flag

Enum i2c_interrupt_enum

Table 3-220. Enum i2c_interrupt_enum

Member name	Function description
I2C_INT_ERR	error interrupt enable
I2C_INT_EV	event interrupt enable
I2C_INT_BUF	buffer interrupt enable

Enum hal_i2c_struct_type_enum

Table 3-221. Enum hal_i2c_struct_type_enum

Member name	Function description
HAL_I2C_INIT_STRU CT	I2C initialize structure
HAL_I2C_DEV_STRU CT	I2C device information structure
HAL_I2C_IRQ_STRU CT	I2C device interrupt callback function pointer structure

Enum hal_i2c_run_state_enum

Table 3-222. Enum hal_i2c_run_state_enum

Member name	Function description
HAL_I2C_STATE_R EADY	I2C is ready for use
HAL_I2C_STATE_B USY	I2C transfer process is ongoing
HAL_I2C_STATE_ MEMORY_BUSY_T X	I2C write memory process is ongoing
HAL_I2C_STATE_ MEMORY_BUSY_R X	I2C read memory process is ongoing
HAL_I2C_STATE_L ISTEN	I2C addressing listen is ongoing in slave mode
HAL_I2C_STATE_B USY_LISTEN	I2C addressing listen and data transfer is ongoing

Structure i2c_buffer_struct

Table 3-223. Structure i2c_buffer_struct

Member name	Function description
buffer	pointer to transfer buffer
length	transfer length
pos	transfer position

Structure hal_i2c_irq_struct

Table 3-224. Structure hal_i2c_irq_struct

Member name	Function description
event_handle	event callback function

Member name	Function description
error_handle	error callback function

Structure hal_i2c_slave_address_struct

Table 3-225. Structure hal_i2c_slave_address_struct

Member name	Function description
device_address	device address
memory_address	memory address
address_size	memory address size
address_complete	addressing complete flag, initialize to RESET
address_count	address count, initialize to zero
second_addressing	Second Address, initialize to RESET

Structure hal_i2c_dev_struct

Table 3-226. Structure hal_i2c_dev_struct

Member name	Function description
periph	I2C port
i2c_irq	device interrupt callback function pointer
p_dma_rx	DMA receive pointer
p_dma_tx	DMA transmit pointer
txbuffer	transmit buffer
rxbuffer	receive buffer
slave_address	slave address
transfer_option	transfer option
last_error	the last error code
error_state	error state
tx_state	transmit state
rx_state	receive state
previous_state	previous state
rx_callback	receive callback function pointer
tx_callback	transmit callback function pointer
mutex	mutex locked and unlocked state

Structure hal_i2c_init_struct

Table 3-227. Structure hal_i2c_init_struct

Member name	Function description
duty_cycle	duty cycle in fast mode or fast mode plus
clock_speed	I2C clock speed
address_format	I2C addformat, 7bits or 10bits
own_address1	I2C own address

Member name	Function description
dual_address	dual-address mode switch
own_address2	I2C own address2 in dual-address mode
general_call	whether or not to response to a general call
no_stretch	whether to stretch SCL low when data is not ready in slave mode

hal_i2c_struct_init

The description of hal_i2c_struct_init is shown as below:

Table 3-228. Function hal_i2c_struct_init

Function name	hal_i2c_struct_init
Function prototype	void hal_i2c_struct_init(hal_i2c_struct_type_enum struct_type, void *p_struct);
Function descriptions	initialize the I2C structure with the default values
Precondition	-
Input parameter{in}	
struct_type	refer to hal_i2c_struct_type_enum
HAL_I2C_INIT_STRUC T	I2C initialize structure
HAL_I2C_DEV_STRUC T	I2C device information structure
HAL_I2C_IRQ_STRUC T	I2C device interrupt callback function pointer structure
Input parameter{in}	
p_struct	pointer to I2C structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the I2C structure with the default values */
hal_i2c_init_struct i2c0_init_parameter;
hal_i2c_struct_init(HAL_I2C_INIT_STRUCTURE, &i2c0_init_parameter);
```

hal_i2c_deinit

The description of hal_i2c_deinit is shown as below:

Table 3-229. Function hal_i2c_deinit

Function name	hal_i2c_deinit
Function prototype	void hal_i2c_deinit(hal_i2c_dev_struct *i2c_dev);

Function descriptions	deinitialize I2C
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, Table 3-226. Structure hal_i2c_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize I2C */
hal_i2c_dev_struct i2c0_info;
hal_i2c_deinit(&i2c0_info);
```

hal_i2c_init

The description of hal_i2c_init is shown as below:

Table 3-230. Function hal_i2c_init

Function name	hal_i2c_init
Function prototype	int32_t hal_i2c_init(hal_i2c_dev_struct *i2c_dev, uint32_t periph, hal_i2c_init_struct *i2c_init)
Function descriptions	initialize I2C
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
periph	specify which I2C is initialized
I2Cx	(x=0,1)
Input parameter{in}	
i2c_init	the initialization data needed to initialize I2C, structural member reference Table 3-227. Structure hal_i2c_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_VAL/HAL_ERR_ADDRESS

Example:

```
/* initialize I2C */
```



```

hal_i2c_dev_struct i2c0_info;

hal_i2c_init_struct i2c0_init_parameter;

i2c0_init_parameter.duty_cycle = I2C_DTCY_2;

i2c0_init_parameter.clock_speed = 400000;

i2c0_init_parameter.address_format = I2C_ADDFORMAT_7BITS;

i2c0_init_parameter.own_address1 = 0x82;

i2c0_init_parameter.dual_address = I2C_DUADEN_DISABLE;

i2c0_init_parameter.own_address2 = 0x00;

i2c0_init_parameter.general_call = I2C_GCEN_DISABLE;

i2c0_init_parameter.no_stretch = I2C_SCLSTRETCH_DISABLE;

hal_i2c_init(&i2c0_info, I2C0, &i2c0_init_parameter);

```

hal_i2c_error_irq

The description of hal_i2c_error_irq is shown as below:

Table 3-231. Function hal_i2c_error_irq

Function name	hal_i2c_error_irq
Function prototype	void hal_i2c_error_irq(hal_i2c_dev_struct *i2c_dev);
Function descriptions	I2C error interrupt handler
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* I2C error interrupt handler*/

hal_i2c_dev_struct i2c0_info;

hal_i2c_error_irq(&i2c0_info);

```

hal_i2c_event_irq

The description of hal_i2c_event_irq is shown as below:

Table 3-232. Function hal_i2c_event_irq

Function name	hal_i2c_event_irq
Function prototype	void hal_i2c_event_irq(hal_i2c_dev_struct *i2c_dev);
Function descriptions	I2C event interrupt handler
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C event interrupt handler */
hal_i2c_dev_struct i2c0_info;
hal_i2c_event_irq(&i2c0_info);
```

hal_i2c_start

The description of hal_i2c_start is shown as below:

Table 3-233. Function hal_i2c_start

Function name	hal_i2c_start
Function prototype	void hal_i2c_start(hal_i2c_dev_struct * i2c_dev);
Function descriptions	start I2C module function
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start I2C module function */
hal_i2c_dev_struct i2c0_info;
hal_i2c_start(&i2c0_info);
```

hal_i2c_stop

The description of hal_i2c_stop is shown as below:

Table 3-234. Function hal_i2c_stop

Function name	hal_i2c_stop
Function prototype	void hal_i2c_stop(hal_i2c_dev_struct *i2c_dev);
Function descriptions	stop I2C module function
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* stop I2C module function */
hal_i2c_dev_struct i2c0_info;
hal_i2c_stop(&i2c0_info);
```

hal_i2c_irq_handle_set

The description of hal_i2c_irq_handle_set is shown as below:

Table 3-235. Function hal_i2c_irq_handle_set

Function name	hal_i2c_irq_handle_set
Function prototype	void hal_i2c_irq_handle_set(hal_i2c_dev_struct *i2c_dev, hal_i2c_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_irq	the structure that contains callback handlers of I2C interrupt, structural member reference Table 3-224. Structure hal_i2c_irq_struct
Output parameter{out}	
-	-
Return value	

Example:

```
/* set user-defined interrupt callback function */

hal_i2c_dev_struct i2c0_info;

hal_i2c_irq_struct i2c0_irq_parameter;

hal_i2c_irq_handle_set(&i2c0_info, &i2c0_irq_parameter);
```

hal_i2c_irq_handle_all_reset

The description of hal_i2c_irq_handle_all_reset is shown as below:

Table 3-236. Function hal_i2c_irq_handle_all_reset

Function name	hal_i2c_irq_handle_all_reset
Function prototype	void hal_i2c_irq_handle_all_reset(hal_i2c_dev_struct *i2c_dev);
Function descriptions	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback function */

hal_i2c_dev_struct i2c0_info;

hal_i2c_irq_handle_all_reset(&i2c0_info);
```

hal_i2c_master_transmit_poll

The description of hal_i2c_master_transmit_poll is shown as below:

Table 3-237. Function hal_i2c_master_transmit_poll

Function name	hal_i2c_master_transmit_poll
Function prototype	int32_t hal_i2c_master_transmit_poll(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	transmit amounts of data in master mode, poll transmit process and

	completed status
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226 . Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* transmit amounts of data in master mode, poll transmit process and completed status */
hal_i2c_dev_struct i2c0_info;
uint8_t tx_buffer[] = "I2C communication based on polling";
#define TXBUFFERSIZE (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))
hal_i2c_master_transmit_poll(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE, 10000);
```

hal_i2c_master_receive_poll

The description of hal_i2c_master_receive_poll is shown as below:

Table 3-238. Function hal_i2c_master_receive_poll

Function name	hal_i2c_master_receive_poll
Function prototype	int32_t hal_i2c_master_receive_poll(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	receive amounts of data in master mode, poll receive process and completed status
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226 . Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer

Input parameter{in}	
length	length of data to be read
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* receive amounts of data in master mode, poll receive process and completed status */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
#define RXBUFFERSIZE 10
```

```
uint8_t rx_buffer[RXBUFFERSIZE];
```

```
hal_i2c_master_receive_poll(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE, 10000);
```

hal_i2c_slave_transmit_poll

The description of hal_i2c_slave_transmit_poll is shown as below:

Table 3-239. Function hal_i2c_slave_transmit_poll

Function name	hal_i2c_slave_transmit_poll
Function prototype	int32_t hal_i2c_slave_transmit_poll(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	transmit amounts of data in slave mode, poll transmit process and completed status
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY
----------------	---

Example:

```
/* transmit amounts of data in slave mode, poll transmit process and completed status */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
uint8_t tx_buffer[] = "I2C communication based on polling";
```

```
#define TXBUFFERSIZE (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))
```

```
hal_i2c_slave_transmit_poll(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE, 10000);
```

hal_i2c_slave_receive_poll

The description of hal_i2c_slave_receive_poll is shown as below:

Table 3-240. Function hal_i2c_slave_receive_poll

Function name	hal_i2c_slave_receive_poll
Function prototype	int32_t hal_i2c_slave_receive_poll(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	receive amounts of data in slave mode, poll receive process and completed status
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be read
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* receive amounts of data in slave mode, poll receive process and completed status */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
#define RXBUFFERSIZE 10
```

```
uint8_t rx_buffer[RXBUFFERSIZE];
```

```
hal_i2c_slave_receive_poll(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE, 10000);
```

hal_i2c_memory_write_poll

The description of hal_i2c_memory_write_poll is shown as below:

Table 3-241. Function hal_i2c_memory_write_poll

Function name	hal_i2c_memory_write_poll
Function prototype	int32_t hal_i2c_memory_write_poll(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	write amounts of data to memory, poll transmit process and completed status
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_VAL

Example:

```
/* write amounts of data to memory, poll transmit process and completed status */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
uint8_t tx_buffer[] = "I2C write memory test";
```

```
#define TXBUFFERSIZE (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))
```

```
hal_i2c_master_transmit_poll(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE, 10000);
```

Note: Users need to write according to the characteristics of memory. Such as AT24C02 is byte writing or page writing (8 bytes) only. If more than 8 bytes are to be written, paging is required first and then call hal_i2c_master_transmit_poll function to write.

hal_i2c_memory_read_poll

The description of hal_i2c_memory_read_poll is shown as below:

Table 3-242. Function hal_i2c_memory_read_poll

Function name	hal_i2c_memory_read_poll
Function prototype	int32_t hal_i2c_memory_read_poll(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	read amounts of data from memory, poll receive process and completed status
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be read
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_VAL

Example:

```
/* read amounts of data from memory, poll receive process and completed status */
hal_i2c_dev_struct i2c0_info;

#define RXBUFFERSIZE 10

uint8_t rx_buffer[RXBUFFERSIZE];

hal_i2c_memory_read_poll(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE, 10000);
```

hal_i2c_master_transmit_interrupt

The description of hal_i2c_master_transmit_interrupt is shown as below:

Table 3-243. Function hal_i2c_master_transmit_interrupt

Function name	hal_i2c_master_transmit_interrupt
Function prototype	int32_t hal_i2c_master_transmit_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);

Function descriptions	transmit amounts of data in master mode by interrupt method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```

/* transmit amounts of data in master mode by interrupt method */

hal_i2c_dev_struct i2c0_info;

void fun_user(hal_i2c_dev_struct *i2c_dev);

uint8_t tx_buffer[] = "I2C communication based on interrupt";

#define TXBUFFERSIZE (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))

hal_i2c_master_transmit_interrupt(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE,
&fun_user);

```

hal_i2c_master_receive_interrupt

The description of hal_i2c_master_receive_interrupt is shown as below:

Table 3-244. Function hal_i2c_master_receive_interrupt

Function name	hal_i2c_master_receive_interrupt
Function prototype	int32_t hal_i2c_master_receive_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	receive amounts of data in master mode by interrupt method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	

p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be read
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```

/* receive amounts of data in master mode by interrupt method */

hal_i2c_dev_struct i2c0_info;

void fun_user(hal_i2c_dev_struct *i2c_dev);

#define RXBUFFERSIZE 10

uint8_t rx_buffer[RXBUFFERSIZE];

hal_i2c_master_receive_interrupt(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE,
&fun_user);

```

hal_i2c_slave_transmit_interrupt

The description of hal_i2c_slave_transmit_interrupt is shown as below:

Table 3-245. Function hal_i2c_slave_transmit_interrupt

Function name	hal_i2c_slave_transmit_interrupt
Function prototype	int32_t hal_i2c_slave_transmit_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	transmit amounts of data in slave mode by interrupt method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	

-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* transmit amounts of data in slave mode by interrupt method */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
void fun_user(hal_i2c_dev_struct *i2c_dev);
```

```
uint8_t tx_buffer[] = "I2C communication based on interrupt";
```

```
#define TXBUFFERSIZE (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))
```

```
hal_i2c_slave_transmit_interrupt(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE, &fun_user);
```

hal_i2c_slave_receive_interrupt

The description of hal_i2c_slave_receive_interrupt is shown as below:

Table 3-246. Function hal_i2c_slave_receive_interrupt

Function name	hal_i2c_slave_receive_interrupt
Function prototype	int32_t hal_i2c_slave_receive_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	receive amounts of data in slave mode by interrupt method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be receive
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* receive amounts of data in slave mode by interrupt method */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
void fun_user(hal_i2c_dev_struct *i2c_dev);
```

```
#define RXBUFFERSIZE 10
```

```
uint8_t rx_buffer[RXBUFFERSIZE];
```

```
hal_i2c_slave_receive_interrupt(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE,  
&fun_user);
```

hal_i2c_memory_write_interrupt

The description of hal_i2c_memory_write_interrupt is shown as below:

Table 3-247. Function hal_i2c_memory_write_interrupt

Function name	hal_i2c_memory_write_interrupt
Function prototype	int32_t hal_i2c_memory_write_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	write amounts of data to memory by interrupt method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be write
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT, HAL_ERR_VAL

Example:

```
/* write amounts of data to memory by interrupt method */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
void fun_user(hal_i2c_dev_struct *i2c_dev);
```

```
uint8_t tx_buffer[] = "I2C write memory test";
```

```
#define TXBUFFERSIZE (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))
```

```
hal_i2c_memory_write_interrupt(&i2c0_info,      (uint8_t*)tx_buffer,      TXBUFFERSIZE,
&fun_user);
```

Note: Users need to write according to the characteristics of memory. Such as AT24C02 is byte writing or page writing (8 bytes) only. If more than 8 bytes are to be written, paging is required first and then call hal_i2c_memory_write_interrupt function to write.

hal_i2c_memory_read_interrupt

The description of hal_i2c_memory_read_interrupt is shown as below:

Table 3-248. Function hal_i2c_memory_read_interrupt

Function name	hal_i2c_memory_read_interrupt
Function prototype	int32_t hal_i2c_memory_read_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	read amounts of data from memory by interrupt method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be read
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT, HAL_ERR_VAL

Example:

```
/* read amounts of data from memory by interrupt method */

hal_i2c_dev_struct i2c0_info;

void fun_user(hal_i2c_dev_struct *i2c_dev);

#define RXBUFFERSIZE  10

uint8_t rx_buffer[RXBUFFERSIZE];

hal_i2c_memory_read_interrupt(&i2c0_info,      (uint8_t*)rx_buffer,      RXBUFFERSIZE,
&fun_user);
```

hal_i2c_master_transmit_dma

The description of hal_i2c_master_transmit_dma is shown as below:

Table 3-249. Function hal_i2c_master_transmit_dma

Function name	hal_i2c_master_transmit_dma
Function prototype	int32_t hal_i2c_master_transmit_dma(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	transmit amounts of data in master mode by dma method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```

/* transmit amounts of data in master mode by dma method */
hal_i2c_dev_struct i2c0_info;

void fun_user(hal_i2c_dev_struct *i2c_dev);

uint8_t tx_buffer[] = "I2C communication based on interrupt";

#define TXBUFFERSIZE  (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))

hal_i2c_master_transmit_dma(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE, &fun_user);

```

hal_i2c_master_receive_dma

The description of hal_i2c_master_receive_dma is shown as below:

Table 3-250. Function hal_i2c_master_receive_dma

Function name	hal_i2c_master_receive_dma
Function prototype	int32_t hal_i2c_master_receive_dma(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);

Function descriptions	receive amounts of data in master mode by dma method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be receive
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* receive amounts of data in master mode by dma method */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
void fun_user(hal_i2c_dev_struct *i2c_dev);
```

```
#define RXBUFFERSIZE 10
```

```
uint8_t rx_buffer[RXBUFFERSIZE];
```

```
hal_i2c_master_receive_dma(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE, &fun_user);
```

hal_i2c_slave_transmit_dma

The description of hal_i2c_slave_transmit_dma is shown as below:

Table 3-251. Function hal_i2c_slave_transmit_dma

Function name	hal_i2c_slave_transmit_dma
Function prototype	int32_t hal_i2c_slave_transmit_dma(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	transmit amounts of data in slave mode by dma method
Precondition	
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer

Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* transmit amounts of data in slave mode by dma method */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
void fun_user(hal_i2c_dev_struct *i2c_dev);
```

```
uint8_t tx_buffer[] = "I2C communication based on interrupt";
```

```
#define TXBUFFERSIZE (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))
```

```
hal_i2c_slave_transmit_dma(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE, &fun_user);
```

hal_i2c_slave_receive_dma

The description of hal_i2c_slave_receive_dma is shown as below:

Table 3-252. Function hal_i2c_slave_receive_dma

Function name	hal_i2c_slave_receive_dma
Function prototype	int32_t hal_i2c_slave_receive_dma(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	receive amounts of data in slave mode by dma method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be receive
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY
----------------	---

Example:

```
/* receive amounts of data in slave mode by dma method */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
void fun_user(hal_i2c_dev_struct *i2c_dev);
```

```
#define RXBUFFERSIZE 10
```

```
uint8_t rx_buffer[RXBUFFERSIZE];
```

```
hal_i2c_slave_receive_dma(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE, &fun_user);
```

hal_i2c_memory_write_dma

The description of hal_i2c_memory_write_dma is shown as below:

Table 3-253. Function hal_i2c_memory_write_dma

Function name	hal_i2c_memory_write_dma
Function prototype	int32_t hal_i2c_memory_write_dma(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	write amounts of data to memory by dma method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be write
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT, HAL_ERR_VAL

Example:

```
/* write amounts of data to memory by dma method */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
void fun_user(hal_i2c_dev_struct *i2c_dev);

uint8_t tx_buffer[] = "I2C write memory test";

#define TXBUFFERSIZE  (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))

hal_i2c_memory_write_dma(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE, &fun_user);
```

Note: Users need to write according to the characteristics of memory. Such as AT24C02 is byte writing or page writing (8 bytes) only. If more than 8 bytes are to be written, paging is required first and then call hal_i2c_memory_write_dma function to write.

hal_i2c_memory_read_dma

The description of hal_i2c_memory_read_dma is shown as below:

Table 3-254. Function hal_i2c_memory_read_dma

Function name	hal_i2c_memory_read_dma
Function prototype	int32_t hal_i2c_memory_read_dma(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	read amounts of data from memory by dma method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be read
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT, HAL_ERR_VAL

Example:

```
/* read amounts of data from memory by dma method */

hal_i2c_dev_struct i2c0_info;

void fun_user(hal_i2c_dev_struct *i2c_dev);

#define RXBUFFERSIZE  10

uint8_t rx_buffer[RXBUFFERSIZE];
```

```
hal_i2c_memory_read_dma(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE, &fun_user);
```

hal_i2c_device_ready_check

The description of hal_i2c_device_ready_check is shown as below:

Table 3-255. Function hal_i2c_device_ready_check

Function name	hal_i2c_device_ready_check
Function prototype	int32_t hal_i2c_device_ready_check(hal_i2c_dev_struct *i2c_dev, uint32_t timeout_ms);
Function descriptions	check whether the device is ready for access
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_NONE

Example:

```
/* check whether the device is ready for access */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
hal_i2c_device_ready_check(&i2c0_info, 1000);
```

hal_i2c_master_serial_transmit_interrupt

The description of hal_i2c_master_serial_transmit_interrupt is shown as below:

Table 3-256. Function hal_i2c_master_serial_transmit_interrupt

Function name	hal_i2c_master_serial_transmit_interrupt
Function prototype	int32_t hal_i2c_master_serial_transmit_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	serial transmit amounts of data in master mode by interrupt method
Precondition	
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	

length	length of data to be sent
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* serial transmit amounts of data in master mode by interrupt method */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
void fun_user(hal_i2c_dev_struct *i2c_dev);
```

```
uint8_t tx_buffer[] = "I2C serial communication";
```

```
#define TXBUFFERSIZE (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))
```

```
hal_i2c_master_serial_transmit_interrupt(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE,
&fun_user);
```

hal_i2c_master_serial_receive_interrupt

The description of hal_i2c_master_serial_receive_interrupt is shown as below:

Table 3-257. Function hal_i2c_master_serial_receive_interrupt

Function name	hal_i2c_master_serial_receive_interrupt
Function prototype	int32_t hal_i2c_master_serial_receive_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	serial receive amounts of data in master mode by interrupt method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be receive
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY
----------------	---

Example:

```
/* serial receive amounts of data in master mode by interrupt method */

hal_i2c_dev_struct i2c0_info;

void fun_user(hal_i2c_dev_struct *i2c_dev);

#define RXBUFFERSIZE 10

uint8_t rx_buffer[RXBUFFERSIZE];

hal_i2c_master_serial_receive_interrupt(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE,
&fun_user);
```

hal_i2c_slave_serial_transmit_interrupt

The description of hal_i2c_slave_serial_transmit_interrupt is shown as below:

Table 3-258. Function hal_i2c_slave_serial_transmit_interrupt

Function name	hal_i2c_slave_serial_transmit_interrupt
Function prototype	int32_t hal_i2c_slave_serial_transmit_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	serial transmit amounts of data in slave mode by interrupt method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* serial transmit amounts of data in slave mode by interrupt method */

hal_i2c_dev_struct i2c0_info;
```

```
void fun_user(hal_i2c_dev_struct *i2c_dev);

uint8_t tx_buffer[] = " I2C serial communication";

#define TXBUFFERSIZE  (uint32_t)(sizeof(tx_buffer) / sizeof(*(tx_buffer)))

hal_i2c_slave_serial_transmit_interrupt(&i2c0_info, (uint8_t*)tx_buffer, TXBUFFERSIZE,
&fun_user);
```

hal_i2c_slave_serial_receive_interrupt

The description of hal_i2c_slave_serial_receive_interrupt is shown as below:

Table 3-259. Function hal_i2c_slave_serial_receive_interrupt

Function name	hal_i2c_slave_serial_receive_interrupt
Function prototype	int32_t hal_i2c_slave_serial_receive_interrupt(hal_i2c_dev_struct *i2c_dev, uint8_t *p_buffer, uint32_t length, hal_i2c_user_cb p_user_func);
Function descriptions	serial receive amounts of data in slave mode by interrupt method
Precondition	-
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be receive
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_ADDRESS/HAL_ERR_TIMEOUT/HAL_ERR_BUSY

Example:

```
/* serial receive amounts of data in slave mode by interrupt method */

hal_i2c_dev_struct i2c0_info;

void fun_user(hal_i2c_dev_struct *i2c_dev);

#define RXBUFFERSIZE  10

uint8_t rx_buffer[RXBUFFERSIZE];

hal_i2c_slave_serial_receive_interrupt(&i2c0_info, (uint8_t*)rx_buffer, RXBUFFERSIZE,
&fun_user);
```

hal_i2c_address_listen_interrupt_enable

The description of hal_i2c_address_listen_interrupt_enable is shown as below:

Table 3-260. Function hal_i2c_address_listen_interrupt_enable

Function name	hal_i2c_address_listen_interrupt_enable
Function prototype	int32_t hal_i2c_address_listen_interrupt_enable(hal_i2c_dev_struct *i2c_dev, hal_i2c_user_cb p_user_func);
Function descriptions	enable address listen in slave mode by interrupt method
Precondition	
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE/HAL_ERR_BUSY/HAL_ERR_ADDRESS

Example:

```
/* enable address listen in slave mode by interrupt method */
hal_i2c_dev_struct i2c0_info;
void fun_user(hal_i2c_dev_struct *i2c_dev);
hal_i2c_address_listen_interrupt_enable(&i2c0_info, &fun_user);
```

hal_i2c_address_listen_interrupt_disable

The description of hal_i2c_address_listen_interrupt_disable is shown as below:

Table 3-261. Function hal_i2c_address_listen_interrupt_disable

Function name	hal_i2c_address_listen_interrupt_disable
Function prototype	int32_t hal_i2c_address_listen_interrupt_disable(hal_i2c_dev_struct *i2c_dev);
Function descriptions	disable address listen in slave mode by interrupt method
Precondition	
Input parameter{in}	
i2c_dev	I2C device information structure, structural member reference Table 3-226. Structure hal_i2c_dev_struct
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_NONE/HAL_ERR_BUSY/HAL_ERR_ADDRESS
---------	---

Example:

```
/* disable address listen in slave mode by interrupt method */
```

```
hal_i2c_dev_struct i2c0_info;
```

```
hal_i2c_address_listen_interrupt_disable(&i2c0_info);
```

3.15. SMBUS

SMBUS (System Management Bus) is a simple single-ended dual-wire bus, which belongs to a derivative bus form of I2C. It can realize lightweight communication requirements. Generally speaking, SMBUS is most commonly used in computer motherboard, mainly for power transmission ON/OFF instruction communication. The I2C registers are listed in chapter [3.15.1](#), the SMBUS firmware functions are introduced in chapter [3.15.2](#).

3.15.1. Descriptions of Peripheral registers

I2C-SMBUS registers are listed in the table shown as below:

Table 3-262. I2C Registers

Registers	Descriptions
I2C_CTL0	Control register 0
I2C_CTL1	Control register 1
I2C_SADDR0	Slave address register 0
I2C_SADDR1	Slave address register 1
I2C_DATA	Transfer buffer register
I2C_STAT0	Transfer status register 0
I2C_STAT1	Transfer status register 1
I2C_CKCFG	Clock configure register
I2C_RT	Rise time register
I2C_FMPCFG	Fast mode plus configure register

3.15.2. Descriptions of Peripheral functions

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

Table 3-263. SMBUS firmware function

Function name	Function description
hal_smbus_struct_init	initialize the SMBUS structure with the default values
hal_smbus_deinit	deinitialize SMBUS interface
hal_smbus_init	initialize SMBUS registers

Function name	Function description
hal_smbus_enable_alert_interrupt	Enable the SMBUS alert mode with Interrupt
hal_smbus_event_irq	SMBUS event interrupt handler
hal_smbus_error_irq	SMBUS error interrupt handler
hal_smbus_start	start SMBUS transfer
hal_smbus_stop	stop SMBUS transfer
hal_smbus_irq_handle_set	set user-defined interrupt callback function
hal_smbus_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_smbus_master_transmit_interrupt	SMBUS master transmit in interrupt mode
hal_smbus_master_receive_interrupt	SMBUS master receive with interrupt mode
hal_smbus_slave_transmit_interrupt	SMBUS slave transmit in interrupt mode
hal_smbus_slave_receive_interrupt	SMBUS slave receive with interrupt mode
hal_smbus_disable_alert_interrupt	Disable the SMBUS alert mode with Interrupt

Enum hal_smbus_run_state_enum

Table 3-264. Enum hal_smbus_run_state_enum

Member name	Function description
HAL_SMBUS_STATE_READY	SMBUS ready state
HAL_SMBUS_STATE_FREE	SMBUS free state
HAL_SMBUS_STATE_BUSY	SMBUS busy state

Enum hal_smbus_struct_type_enum

Table 3-265. Enum hal_smbus_struct_type_enum

Member name	Function description
HAL_SMBUS_INIT_S_STRUCT	SMBUS initialize structure
HAL_SMBUS_DEV_S_STRUCT	SMBUS device information structure
HAL_SMBUS_IRQ_S_STRUCT	SMBUS device interrupt callback function pointer structure

Structure hal_smbus_irq_struct

Table 3-266. Structure hal_smbus_irq_struct

Member name	Function description
event_handle	event handle
error_handle	error handle

Structure `smbus_buffer_struct`

Table 3-267. `smbus_buffer_struct`

Member name	Function description
<code>buffer</code>	pointer to transfer buffer
<code>length</code>	transfer length
<code>pos</code>	transfer position

Structure `hal_smbus_init_struct`

Table 3-268. Structure `hal_smbus_init_struct`

Member name	Function description
<code>clock_speed</code>	SMBUS clock speed
<code>address_format</code>	SMBUS addformat
<code>own_address1</code>	SMBUS own address
<code>dual_address</code>	dual-address mode switch
<code>own_address2</code>	SMBUS own address2 in dual-address mode
<code>smbus_type</code>	SMBUS type (SMBUS_DEVICE, SMBUS_HOST)
<code>smbus_pec</code>	SMBUS PEC calculation enable
<code>smbus_arp</code>	SMBUS ARP protocol enable
<code>general_call</code>	whether or not to response to a general call
<code>no_stretch</code>	whether to stretch SCL low when data is not ready in slave mode

Structure `hal_smbus_dev_struct`

Table 3-269. Structure `hal_smbus_dev_struct`

Member name	Function description
<code>periph</code>	SMBUS port
<code>slave_address</code>	slave address
<code>smbus_irq</code>	SMBUS interrupt request structure
<code>txbuffer</code>	SMBUS transmit buffer structure
<code>rxbuffer</code>	SMBUS receive buffer structure
<code>smbus_pec_transfer</code>	SMBus PEC transfer
<code>last_error</code>	SMBUS last error
<code>error_state</code>	SMBUS error state
<code>tx_state</code>	SMBUS transmit state enum
<code>rx_state</code>	SMBUS receive state enum
<code>*rx_callback</code>	SMBUS receive callback function
<code>*tx_callback</code>	SMBUS transmit callback function
<code>*priv</code>	private data
<code>mutex</code>	mutex locked and unlocked state

hal_smbus_struct_init

The description of hal_smbus_struct_init is shown as below:

Table 3-270. Function hal_smbus_struct_init

Function name	hal_smbus_struct_init
Function prototype	void hal_smbus_struct_init(hal_smbus_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the SMBUS structure with the default values
Precondition	-
Input parameter{in}	
hal_struct_type	Structural type
HAL_SMBUS_INIT_STRUCT	SMBUS initialize structure
HAL_SMBUS_DEV_STRUCT	SMBUS device information structure
HAL_SMBUS_IRQ_STRUCT	SMBUS device interrupt callback function pointer structure
Input parameter{in}	
p_struct	point to SMBUS structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* init HAL_SMBUS_INIT_STRUCT */
hal_smbus_init_struct i2c0_smbus_init_parameter;
hal_smbus_struct_init(HAL_SMBUS_INIT_STRUCT, &i2c0_smbus_init_parameter);
```

hal_smbus_deinit

The description of hal_smbus_deinit is shown as below:

Table 3-271. Function hal_smbus_deinit

Function name	hal_smbus_deinit
Function prototype	void hal_smbus_deinit(hal_smbus_dev_struct *smbus_dev);
Function descriptions	deinitialize SMBUS
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize SMBUS */

hal_smbus_dev_struct i2c0_smbus_info;

hal_smbus_deinit(&i2c0_smbus_info);
```

hal_smbus_init

The description of hal_smbus_init is shown as below:

Table 3-272. Function hal_smbus_init

Function name	hal_smbus_init
Function prototype	int32_t hal_smbus_init(hal_smbus_dev_struct *smbus_dev, uint32_t periph, hal_smbus_init_struct *p_init);
Function descriptions	initialize SMBUS
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Input parameter{in}	
periph	specify which SMBUS is initialized
Input parameter{in}	
p_init	the initialization data needed to initialize SMBUS, structural member reference Table 3-268. Structure hal_smbus_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```
/* initialize SMBUS registers */

hal_smbus_dev_struct i2c0_smbus_info;

hal_smbus_init_struct i2c0_smbus_init_parameter;

hal_smbus_init(&i2c0_smbus_info, I2C0, &i2c0_smbus_init_parameter);
```

hal_smbus_enable_alert_interrupt

The description of hal_smbus_enable_alert_interrupt is shown as below:

Table 3-273. Function hal_smbus_slave_receive_interrupt

Function name	hal_smbus_enable_alert_interrupt
Function prototype	int32_t hal_smbus_enable_alert_interrupt(hal_smbus_dev_struct *smbus_dev);
Function descriptions	Enable the SMBUS alert mode with Interrupt.
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Output parameter{out}	
-	-
Return value	
error code	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* enable the SMBUS alert mode with Interrupt */

hal_smbus_dev_struct i2c0_smbus_info;

hal_smbus_enable_alert_interrupt(&i2c0_smbus_info);
```

hal_smbus_event_irq

The description of hal_smbus_event_irq is shown as below:

Table 3-274. Function hal_smbus_event_irq

Function name	hal_smbus_event_irq
Function prototype	void hal_smbus_event_irq(hal_smbus_dev_struct *smbus_dev);
Function descriptions	SMBUS event interrupt handler
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 event interrupt */
```

```

hal_smbus_dev_struct i2c0_smbus_info;

void I2C0_EV_IRQHandler(void)
{
    hal_smbus_event_irq(&i2c0_smbus_info);
}

```

hal_smbus_error_irq

The description of hal_smbus_error_irq is shown as below:

Table 3-275. Function hal_smbus_error_irq

Function name	hal_smbus_error_irq
Function prototype	void hal_smbus_error_irq(hal_smbus_dev_struct *smbus_dev);
Function descriptions	SMBUS error interrupt handler
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* I2C0 error interrupt */

hal_smbus_dev_struct i2c0_smbus_info;

void I2C0_ER_IRQHandler(void)
{
    hal_smbus_error_irq(&i2c0_smbus_info);
}

```

hal_smbus_start

The description of hal_smbus_start is shown as below:

Table 3-276. Function hal_smbus_start

Function name	hal_smbus_start
Function prototype	void hal_smbus_start(hal_smbus_dev_struct *smbus_dev);
Function descriptions	start SMBUS module function

Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start SMBUS module function */
hal_smbus_dev_struct i2c0_smbus_info;
hal_smbus_start(&i2c0_smbus_info);
```

hal_smbus_stop

The description of hal_smbus_stop is shown as below:

Table 3-277. Function hal_smbus_stop

Function name	hal_smbus_stop
Function prototype	void hal_smbus_stop(hal_smbus_dev_struct *smbus_dev);
Function descriptions	stop SMBUS module function
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* stop SMBUS module function */
hal_smbus_dev_struct i2c0_smbus_info;
hal_smbus_stop(&i2c0_smbus_info);
```

hal_smbus_irq_handle_set

The description of hal_smbus_irq_handle_set is shown as below:

Table 3-278. Function hal_smbus_irq_handle_set

Function name	hal_smbus_irq_handle_set
---------------	--------------------------

Function prototype	void hal_smbus_irq_handle_set(hal_smbus_dev_struct *smbus_dev, hal_smbus_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Input parameter{in}	
p_irq	the structure that contains callback handlers of SMBUS interrupt Table 3-266. Structure hal_smbus_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable I2C0 dual-address */

void _smbus_alert_event(void *smbus_dev)
{
    LED_ON;
}

i2c0_smbus_info.smbus_irq.error_handle = _smbus_alert_event;

hal_smbus_irq_handle_set(&i2c0_smbus_info, &i2c0_smbus_info.smbus_irq);

```

hal_smbus_irq_handle_all_reset

The description of hal_smbus_irq_handle_all_reset is shown as below:

Table 3-279. Function hal_smbus_irq_handle_all_reset

Function name	hal_smbus_irq_handle_all_reset
Function prototype	void hal_smbus_irq_handle_all_reset(hal_smbus_dev_struct *smbus_dev);
Function descriptions	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback */
```

```
hal_smbus_dev_struct i2c0_smbus_info;
```

```
hal_smbus_irq_handle_all_reset(&i2c0_smbus_info);
```

hal_smbus_master_transmit_interrupt

The description of hal_smbus_master_transmit_interrupt is shown as below:

Table 3-280. Function hal_smbus_master_transmit_interrupt

Function name	hal_smbus_master_transmit_interrupt
Function prototype	int32_t hal_smbus_master_transmit_interrupt(hal_smbus_dev_struct *smbus_dev, uint8_t *p_buffer, uint32_t length, hal_smbus_user_cb p_user_func)
Function descriptions	SMBUS master transmit in interrupt mode
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
error code	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_BUSY

Example:

```
/* SMBUS master transmit in interrupt mode */
```

```
hal_smbus_dev_struct i2c0_smbus_info;
```

```
buffertx[2] = {0, 0};
```

```
hal_smbus_master_transmit_interrupt(&i2c0_smbus_info, &buffertx, 2, NULL);
```

hal_smbus_master_receive_interrupt

The description of hal_smbus_master_receive_interrupt is shown as below:

Table 3-281. Function hal_smbus_master_receive_interrupt

Function name	hal_smbus_master_receive_interrupt
Function prototype	int32_t hal_smbus_master_receive_interrupt(hal_smbus_dev_struct *smbus_dev, uint8_t *p_buffer, uint32_t length, hal_smbus_user_cb p_user_func)
Function descriptions	SMBUS master receive with interrupt
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be read
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
error code	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
/* SMBUS master receive in interrupt mode */
hal_smbus_dev_struct i2c0_smbus_info;
uint8_t bufferrx[2];
hal_smbus_master_receive_interrupt(&i2c0_smbus_info, &bufferrx, 2, NULL);
```

hal_smbus_slave_transmit_interrupt

The description of hal_smbus_slave_transmit_interrupt is shown as below:

Table 3-282. Function hal_smbus_slave_transmit_interrupt

Function name	hal_smbus_slave_transmit_interrupt
Function prototype	int32_t hal_smbus_slave_transmit_interrupt(hal_smbus_dev_struct *smbus_dev, uint8_t *p_buffer, uint32_t length, hal_smbus_user_cb p_user_func)
Function descriptions	SMBUS slave transmit in interrupt mode

Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
error code	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
/* SMBUS slave transmit in interrupt mode */
hal_smbus_dev_struct i2c0_smbus_info;
buffertx[2] = {0, 0};
hal_smbus_slave_transmit_interrupt (&i2c0_smbus_info, &buffertx, 2, NULL);
```

hal_smbus_slave_receive_interrupt

The description of hal_smbus_slave_receive_interrupt is shown as below:

Table 3-283. Function hal_smbus_slave_receive_interrupt

Function name	hal_smbus_slave_receive_interrupt
Function prototype	int32_t hal_smbus_slave_receive_interrupt(hal_smbus_dev_struct *smbus_dev, uint8_t *p_buffer, uint32_t length, hal_smbus_user_cb p_user_func)
Function descriptions	SMBUS slave receive with interrupt
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be read
Input parameter{in}	

p_user_func	call back function for user
Output parameter{out}	
-	-
Return value	
error code	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
/* SMBUS slave receive in interrupt mode */
hal_smbus_dev_struct i2c0_smbus_info;
uint8_t bufferrx[2];
hal_smbus_slave_receive_interrupt(&i2c0_smbus_info, &buffertx, 2, NULL);
```

hal_smbus_disable_alert_interrupt

The description of hal_smbus_disable_alert_interrupt is shown as below:

Table 3-284. Function hal_smbus_disable_alert_interrupt

Function name	hal_smbus_disable_alert_interrupt
Function prototype	int32_t hal_smbus_disable_alert_interrupt(hal_smbus_dev_struct *smbus_dev);
Function descriptions	Disable the SMBUS alert mode with Interrupt.
Precondition	-
Input parameter{in}	
smbus_dev	SMBUS device information structure, structural member reference Table 3-269. Structure hal_smbus_dev_struct
Output parameter{out}	
-	-
Return value	
error code	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* disable the SMBUS alert mode with Interrupt */
hal_smbus_dev_struct i2c0_smbus_info;
hal_smbus_disable_alert_interrupt(&i2c0_smbus_info);
```

3.16. NVIC

Nested Vectored Interrupt Controller (NVIC). The NVIC registers are listed in chapter [3.16.1](#), the NVIC firmware functions are introduced in chapter [3.16.2](#)

3.16.1. Descriptions of Peripheral registers

Table 3-285. NVIC Registers

Registers	Descriptions
ISER ⁽¹⁾	Interrupt Set Enable Register
ICER ⁽¹⁾	Interrupt Clear Enable Register
ISPR ⁽¹⁾	Interrupt Set Pending Register
ICPR ⁽¹⁾	Interrupt Clear Pending Register
IABR ⁽¹⁾	Interrupt Active bit Register
ITNS ⁽¹⁾	Interrupt Non-Secure State Register
IPR ⁽¹⁾	Interrupt Priority Register
CPUID ⁽²⁾	CPUID Base Register
ICSR ⁽²⁾	Interrupt Control and State Register
VTOR ⁽²⁾	Vector Table Offset Register
AIRCR ⁽²⁾	Application Interrupt and Reset Control Register
SCR ⁽²⁾	System Control Register
CCR ⁽²⁾	Configuration Control Register
SHPR ⁽²⁾	System Handlers Priority Registers
SHCSR ⁽²⁾	System Handler Control and State Register

1. refer to the structure NVIC_Type, is defined in the core_cm23.h file

2. refer to the structure SCB_Type, is defined in the core_cm23.h file

3.16.2. Descriptions of Peripheral functions

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

Table 3-286. NVIC firmware function

Function name	Function description
hal_nvic_irq_priority_group_set	Set NVIC request priority
hal_nvic_irq_enable	Enable NVIC peripheral request
hal_nvic_set_priority	Set priority of system NVIC requests

Enum IRQn_Type

Table 3-287. IRQn_Type

Member name	Function description
WWDGT_IRQn	window watchDog timer interrupt
LVD_IRQn	LVD through EXTI line detect interrupt
RTC_IRQn	RTC through EXTI line interrupt
FMC_IRQn	FMC interrupt
RCU_IRQn	RCU interrupt
EXTI0_1_IRQn	EXTI line 0 and 1 interrupts
EXTI2_3_IRQn	EXTI line 2 and 3 interrupts
EXTI4_15_IRQn	EXTI line 4 to 15 interrupts
DMA_Channel0_IRQn	DMA channel 0 interrupt
DMA_Channel1_2_IRQn	DMA channel 1 and channel 2 interrupts
DMA_Channel3_4_IRQn	DMA channel 3 and channel 4 interrupts
ADC_CMP_IRQn	ADC, CMP interrupts
TIMER0_BRK_UP_TRG_COM_IRQn	TIMER0 break, update, trigger and commutation interrupts
TIMER0_Channel_I_IRQn	TIMER0 channel capture compare interrupts
TIMER2_IRQn	TIMER2 interrupt
TIMER5_IRQn	TIMER5 interrupt
TIMER13_IRQn	TIMER13 interrupt
TIMER14_IRQn	TIMER14 interrupt
TIMER15_IRQn	TIMER15 interrupt
TIMER16_IRQn	TIMER16 interrupt
I2C0_EV_IRQn	I2C0 event interrupt
I2C1_EV_IRQn	I2C1 event interrupt
SPI0_IRQn	SPI0 interrupt
SPI1_IRQn	SPI1 interrupt
USART0_IRQn	USART0 interrupt
USART1_IRQn	USART1 interrupt
I2C0_ER_IRQn	I2C0 error interrupt
I2C1_ER_IRQn	I2C1 error interrupt

hal_nvic_irq_priority_group_set

The description of hal_nvic_irq_priority_group_set is shown as below:

Table 3-288. Function hal_nvic_irq_priority_group_set

Function name	hal_nvic_irq_priority_group_set
Function prototype	void hal_nvic_irq_priority_group_set(uint32_t nvic_prigroup);
Function descriptions	set NVIC request priority
Precondition	-
The called functions	-
Input parameter{in}	
nvic_prigroup	NVIC interrupt request, the enum members can refer to Table 3-287. IRQn_Type
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set priority group NVIC_PRIGROUP_PRE0_SUB4 */
hal_nvic_irq_priority_group_set (NVIC_PRIGROUP_PRE0_SUB4);
```

hal_nvic_irq_enable

The description of hal_nvic_irq_enable is shown as below:

Table 3-289. Function hal_nvic_irq_enable

Function name	hal_nvic_irq_enable
Function prototype	void hal_nvic_irq_enable(IRQn_Type nvic_irq, uint8_t nvic_irq_pre_priority, uint8_t nvic_irq_sub_priority);
Function descriptions	enable peripherals NVIC request
Precondition	-
The called functions	-
Input parameter{in}	
nvic_irq	NVIC interrupt request, the enum members can refer to Table 3-287. IRQn_Type
Input parameter{in}	
nvic_irq_pre_priority	Set pre priority
Input parameter{in}	
nvic_irq_sub_priority	Set sub priority

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 NVIC request */
hal_nvic_irq_enable((USART0, 0, 0);
```

3.17. PMU

The Power management unit (PMU), provides three types of power saving modes, including Sleep, Deep-sleep and Standby mode. The PMU registers are listed in chapter [3.17.1](#), the PMU firmware functions are introduced in chapter [3.17.2](#).

3.17.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

Table 3-290. PMU Registers

Registers	Descriptions
PMU_CTL	Control register
PMU_CS	Control and status register

3.17.2. Descriptions of Peripheral functions

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

Table 3-291. PMU firmware function

Function name	Function description
hal_pmu_deinit	reset PMU peripheral
hal_pmu_struct_init	initialize the PMU structure with the default values
hal_pmu_lvd_init	configure EXTI_16 and then configure low voltage detector threshold
hal_pmu_wakeup_pin_enable	enable PMU wakeup pin
hal_pmu_lvd_start	start LVD detector
hal_pmu_lvd_stop	stop LVD detector
hal_pmu_wakeup_pin_disable	disable PMU wakeup pin
hal_pmu_lvd_irq	PMU interrupt handler content function
hal_pmu_lvd_irq_handle_set	set user-defined interrupt callback function
hal_pmu_lvd_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_pmu_lvd_start_interrupt	start PMU lvd with interrupt mode

Function name	Function description
hal_pmu_lvd_stop_interrupt	stop PMU lvd with interrupt mode

Enum hal_pmu_lvd_voltage_enum

Table 3-292. Enum hal_pmu_lvd_voltage_enum

Member name	Function description
PMU_LVDT_0	voltage threshold is 2.1V
PMU_LVDT_1	voltage threshold is 2.3V
PMU_LVDT_2	voltage threshold is 2.4V
PMU_LVDT_3	voltage threshold is 2.6V
PMU_LVDT_4	voltage threshold is 2.7V
PMU_LVDT_5	voltage threshold is 2.9V
PMU_LVDT_6	voltage threshold is 3.0V
PMU_LVDT_7	voltage threshold is 3.1V

Enum hal_pmu_struct_type_enum

Table 3-293. Enum hal_pmu_struct_type_enum

Member name	Function description
HAL_PMU_INIT_STRUCT	PMU initialization structure
HAL_PMU_IRQ_STRUCT	PMU initialization IRQ structure
HAL_PMU_DEV_STRUCT	PMU initialization device structure

Enum hal_pmu_error_enum

Table 3-294. Enum hal_pmu_error_enum

Member name	Function description
HAL_PMU_ERROR_NONE	no error
HAL_PMU_ERROR_SYSTEM	PMU internal error: if problem of clocking, enable/disable, wrong state
HAL_PMU_ERROR_CONFIG	configuration error occurred

Enum hal_pmu_state_enum

Table 3-295. Enum hal_pmu_state_enum

Member name	Function description
HAL_PMU_STATE_NONE	NONE
HAL_PMU_STATE_RESET	RESET
HAL_PMU_STATE_BUSY	BUSY
HAL_PMU_STATE_TIMEOUT	TIMEOUT
HAL_PMU_STATE_ERROR	ERROR
HAL_PMU_STATE_READY	READY

Structure hal_pmu_init_struct

Table 3-296. Structure hal_pmu_init_struct

Member name	Function description
int_event_mode	interrupt or event mode to monitor the LVD
trig_type	the trigger edge of the LVD event
lvd_threshold	select low voltage detector threshold

Structure hal_pmu_lvd_irq_struct

Table 3-297. Structure hal_pmu_lvd_irq_struct

Member name	Function description
pmu_lvd_handle	LVD handler function

Structure hal_pmu_dev_struct

Table 3-298. Structure hal_pmu_dev_struct

Member name	Function description
pmu_lvd_irq	PMU device interrupt callback function pointer structure
error_state	PMU error state
state	PMU state
mutex	mutex locked and unlocked state
priv	priv data

hal_pmu_deinit

The description of hal_pmu_deinit is shown as below:

Table 3-299. Function hal_pmu_deinit

Function name	hal_pmu_deinit
Function prototype	int32_t hal_pmu_deinit(hal_pmu_dev_struct *pmu_dev)
Function descriptions	reset PMU peripheral
Precondition	-
The called functions	hal_rcu_periph_reset_enable / hal_rcu_periph_reset_disable
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* reset PMU peripheral */
```

```
hal_pmu_dev_struct pmu_info;
```

```
hal_pmu_deinit(&pmu_info);
```

hal_pmu_struct_init

The description of hal_pmu_struct_init is shown as below:

Table 3-300. Function hal_pmu_struct_init

Function name	hal_pmu_struct_init
Function prototype	void hal_pmu_struct_init(hal_pmu_struct_type_enum hal_struct_type, void *p_struct)
Function descriptions	initialize the PMU structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	point to hal_pmu_struct_type_enum, the enum members can refer to Table 3-293. Enum hal_pmu_struct_type_enum
Output parameter{out}	
p_struct	point to PMU structure that contains the configuration information
Return value	
-	-

Example:

```
/* initialize the PMU structure */
```

```
hal_pmu_dev_struct pmu_info;
```

```
hal_pmu_struct_init(HAL_PMU_DEV_STRUCT, &pmu_info);
```

hal_pmu_lvd_init

The description of hal_pmu_lvd_init is shown as below:

Table 3-301. Function hal_pmu_lvd_init

Function name	hal_pmu_lvd_init
Function prototype	int32_t hal_pmu_lvd_init(hal_pmu_dev_struct *pmu_dev, hal_pmu_init_struct *pmu_lvd_init)
Function descriptions	configure EXTI_16 and then configure low voltage detector threshold
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
-	-

Output parameter{out}	
pmu_lvd_init	point to hal_pmu_init_struct, the structure members can refer to Table 3-296. Structure hal_pmu_init_struct
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* configure EXTI_16 and then configure low voltage detector threshold */

hal_pmu_dev_struct pmu_info;

hal_pmu_lvd_init_struct pmu_init_parameter;

pmu_init_parameter.lvd_threshold = PMU_LVDT_6;

pmu_init_parameter.int_event_mode = PMU_LVD_EVENT_MODE;

pmu_init_parameter.trig_type = PMU_LVD_TRIG_RISING;

hal_pmu_lvd_init(&pmu_info,&pmu_init_parameter);

```

hal_pmu_wakeup_pin_enable

The description of hal_pmu_wakeup_pin_enable is shown as below:

Table 3-302. Function hal_pmu_wakeup_pin_enable

Function name	hal_pmu_wakeup_pin_enable
Function prototype	int32_t hal_pmu_wakeup_pin_enable(hal_pmu_dev_struct *pmu_dev, uint32_t wakeup_pin)
Function descriptions	enable PMU wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
wakeup_pin	wakeup_pin
<i>PMU_WAKEUP_PIN0</i>	WKUP Pin 0 (PA0)
<i>PMU_WAKEUP_PIN1</i>	WKUP Pin 1 (PC13)
<i>PMU_WAKEUP_PIN4</i>	WKUP Pin 4 (PC5)
<i>PMU_WAKEUP_PIN5</i>	WKUP Pin 5 (PB5)
<i>PMU_WAKEUP_PIN6</i>	WKUP Pin 6 (PB15)
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* enable PMU wakeup pin PA0 */

hal_pmu_dev_struct pmu_info;

hal_pmu_wakeup_pin_enable(&pmu_info, PMU_WAKEUP_PIN0);
```

hal_pmu_lvd_start

The description of hal_pmu_lvd_start is shown as below:

Table 3-303. Function hal_pmu_lvd_start

Function name	hal_pmu_lvd_start
Function prototype	int32_t hal_pmu_lvd_start(hal_pmu_dev_struct *pmu_dev)
Function descriptions	start LVD detector
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* start LVD detector */

hal_pmu_dev_struct pmu_info;

hal_pmu_lvd_start(&pmu_info);
```

hal_pmu_lvd_stop

The description of hal_pmu_lvd_stop is shown as below:

Table 3-304. Function hal_pmu_lvd_stop

Function name	hal_pmu_lvd_stop
Function prototype	int32_t hal_pmu_lvd_stop(hal_pmu_dev_struct *pmu_dev)
Function descriptions	stop LVD detector
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* stop LVD detector */
```

```
hal_pmu_dev_struct pmu_info;
```

```
hal_pmu_lvd_stop(&pmu_info);
```

hal_pmu_wakeup_pin_disable

The description of hal_pmu_wakeup_pin_disable is shown as below:

Table 3-305. Function hal_pmu_wakeup_pin_enable

Function name	hal_pmu_wakeup_pin_disable
Function prototype	int32_t hal_pmu_wakeup_pin_disable(hal_pmu_dev_struct *pmu_dev, uint32_t wakeup_pin)
Function descriptions	disable PMU wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
wakeup_pin	wakeup_pin
PMU_WAKEUP_PIN0	WKUP Pin 0 (PA0)
PMU_WAKEUP_PIN1	WKUP Pin 1 (PC13)
PMU_WAKEUP_PIN4	WKUP Pin 4 (PC5)
PMU_WAKEUP_PIN5	WKUP Pin 5 (PB5)
PMU_WAKEUP_PIN6	WKUP Pin 6 (PB15)
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* disable PMU wakeup pin PA0 */
```

```
hal_pmu_dev_struct pmu_info;
```

```
hal_pmu_wakeup_pin_disable(&pmu_info, PMU_WAKEUP_PIN0);
```

hal_pmu_lvd_irq

The description of hal_pmu_lvd_irq is shown as below:

Table 3-306. Function hal_pmu_lvd_irq

Function name	hal_pmu_lvd_irq
Function prototype	void hal_pmu_lvd_irq(hal_pmu_dev_struct *pmu_dev)
Function descriptions	PMU interrupt handler content function
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PMU interrupt handler content function,which is merely used in pmu_lvd_handler */
hal_pmu_dev_struct pmu_info;
hal_pmu_lvd_irq(&pmu_info);
```

hal_pmu_lvd_irq_handle_set

The description of hal_pmu_lvd_irq_handle_set is shown as below:

Table 3-307. Function hal_pmu_lvd_irq_handle_set

Function name	hal_pmu_lvd_irq_handle_set
Function prototype	void hal_pmu_lvd_irq_handle_set(hal_pmu_dev_struct *pmu_dev, hal_pmu_lvd_irq_struct *irq_handle)
Function descriptions	set user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
irq_handle	Structure for PMU device interrupt callback function pointer structure, the structure members can refer to Table 3-297. Structure hal_pmu_lvd_irq_struct
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* set user-defined interrupt callback function */

hal_pmu_dev_struct pmu_info;

hal_pmu_lvd_irq_struct lvd_irq;

void lvd_irq_test(void *p);

lvd_irq.pmu_lvd_handle = lvd_irq_test;

hal_pmu_lvd_irq_handle_set(&pmu_info, &lvd_irq);
```

hal_pmu_lvd_irq_handle_reset

The description of hal_pmu_lvd_irq_handle_reset is shown as below:

Table 3-308. Function hal_pmu_lvd_irq_handle_all_reset

Function name	hal_pmu_lvd_irq_handle_reset
Function prototype	void hal_pmu_lvd_irq_handle_all_reset(hal_pmu_dev_struct *pmu_dev)
Function descriptions	reset all user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback function */

hal_pmu_dev_struct pmu_info;

hal_pmu_lvd_irq_handle_reset(&pmu_info);
```

hal_pmu_lvd_start_interrupt

The description of hal_pmu_lvd_start_interrupt is shown as below:

Table 3-309. Function hal_pmu_lvd_start_interrupt

Function name	hal_pmu_lvd_start_interrupt
---------------	-----------------------------

Function prototype	int32_t hal_pmu_lvd_start_interrupt(hal_pmu_dev_struct *pmu_dev, hal_pmu_lvd_irq_struct *irq_handle)
Function descriptions	start PMU lvd with interrupt mode
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
irq_handle	Structure for PMU device interrupt callback function pointer structure, the structure members can refer to Table 3-297. Structure hal_pmu_lvd_irq_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* start PMU lvd by interrupt method */

hal_pmu_dev_struct pmu_info;

hal_pmu_lvd_irq_struct lvd_irq;

void lvd_irq_test(void *p);

lvd_irq.pmu_lvd_handle = lvd_irq_test;

hal_pmu_lvd_start_interrupt(&pmu_info, &lvd_irq);

```

hal_pmu_lvd_stop_interrupt

The description of hal_pmu_lvd_stop_interrupt is shown as below:

Table 3-310. Function hal_pmu_lvd_stop_interrupt

Function name	hal_pmu_lvd_stop_interrupt
Function prototype	int32_t hal_pmu_lvd_stop_interrupt(hal_pmu_dev_struct *pmu_dev)
Function descriptions	stop PMU lvd with interrupt mode
Precondition	-
The called functions	-
Input parameter{in}	
pmu_dev	point to hal_pmu_dev_struct, the structure members can refer to Table 3-298. Structure hal_pmu_dev_struct
Output parameter{out}	
-	-
Return value	

<code>int32_t</code>	<code>HAL_ERR_ADDRESS, HAL_ERR_NONE</code>
----------------------	--

Example:

```
/* stop PMU lvd by interrupt method */
```

```
hal_pmu_dev_struct pmu_info;
```

```
hal_pmu_lvd_stop (&pmu_info);
```

3.18. RCU

RCU is the reset and clock unit. Reset Control includes the control of three kinds of reset: power reset, system reset and backup domain reset. The Clock Control unit provides a range of frequencies and clock functions. The RCU registers are listed in chapter [3.18.1](#), the RCU HAL firmware functions are introduced in chapter [3.18.2](#).

3.18.1. Descriptions of Peripheral registers

RCU registers are listed in the table shown as below:

Table 3-311. RCU Registers

Registers	Descriptions
RCU_CTL0	Control register 0
RCU_CFG0	Clock configuration register 0
RCU_INT	Clock interrupt register
RCU_APB2RST	APB2 reset register
RCU_APB1RST	APB1 reset register
RCU_AHBEN	AHB enable register
RCU_APB2EN	APB2 enable register
RCU_APB1EN	APB1 enable register
RCU_BDCTL	Backup domain control register
RCU_RSTSCK	Reset source/clock register
RCU_AHBRST	AHB reset register
RCU_CFG1	Clock configuration register 1
RCU_CFG2	Clock configuration register 2
RCU_CTL1	Control register 1
RCU_ADDCTL	Additional clock control register
RCU_ADDINT	Additional clock interrupt register
RCU_ADDAPB1EN	APB1 additional enable register
RCU_ADDAPB1RST	APB1 additional reset register
RCU_VKEY	Unlock voltage register
RCU_DSV	Deep-sleep mode voltage register

3.18.2. Descriptions of Peripheral functions

Table 3-312. RCU firmware function

Function name	Function description
hal_rcu_osci_config	configure the RCU oscillators
hal_rcu_clock_out_config	configure the clock out to output on CKOUT pin
hal_rcu_deinit	deinitialize the RCU
hal_rcu_struct_init	initialize the RCU structure with the default values
hal_rcu_periph_clk_enable	enable the peripherals clock
hal_rcu_periph_clk_disable	disable the peripherals clock
hal_rcu_hxtal_clock_monitor_enable	enable the HXTAL clock monitor
hal_rcu_hxtal_clock_monitor_disable	disable the HXTAL clock monitor
hal_rcu_periph_reset_enable	reset the peripherals
hal_rcu_periph_reset_disable	disable reset the peripheral
hal_rcu_periph_clock_config	initialize the RCU extended peripherals(RTC, Usart0, ADC, USBFS, CEC) clocks
hal_rcu_periph_clkfreq_get	get the peripherals clock frequency
hal_rcu_osci_config_get	get the RCU oscillators configuration
hal_rcu_clock_config	configure the RCU clock
hal_rcu_clock_config_get	get the RCU clock configuration
hal_SystemCoreClockUpdate	update the SystemCoreClock with current core clock retrieved from cpu registers
hal_rcu_irq	RCU interrupt handler content function, which is merely used in RCU_handler
hal_rcu_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_rcu_irq_handle_all_reset	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered

Enum hal_rcu_periph_enum

Table 3-313. Enum hal_rcu_periph_enum

Member name	Function description
RCU_DMA	DMA clock
RCU_CRC	CRC clock
RCU_GPIOA	GPIOA clock
RCU_GPIOB	GPIOB clock
RCU_GPIOC	GPIOC clock

Member name	Function description
RCU_GPIOD	GPIOD clock
RCU_GPIOF	GPIOF clock
RCU_TSI	TSI clock
RCU_CFGCMP	CFGCMP clock
RCU_ADC	ADC clock
RCU_TIMER0	TIMER0 clock
RCU_SPI0	SPI0 clock
RCU_USART0	USART0 clock
RCU_TIMER14	TIMER14 clock
RCU_TIMER15	TIMER15 clock
RCU_TIMER16	TIMER16 clock
RCU_TIMER1	TIMER1 clock
RCU_TIMER2	TIMER2 clock
RCU_TIMER13	TIMER13 clock
RCU_WWDGT	WWDGT clock
RCU_SPI1	SPI1 clock
RCU_USART1	USART1 clock
RCU_I2C0	I2C0 clock
RCU_I2C1	I2C1 clock
RCU_PMU	PMU clock
RCU_DAC	DAC clock (only for GD32F350)
RCU_CEC	CEC clock (only for GD32F350)
RCU_TIMER5	TIMER5 clock (only for GD32F350)
RCU_USBFS	USBFS clock (only for GD32F350)
RCU_RTC	RTC clock
RCU_CTC	CTC clock

Enum hal_rcu_periph_sleep_enum

Table 3-314. Enum hal_rcu_periph_sleep_enum

Member name	Function description
RCU_SRAM_SLP	SRAM clock
RCU_FMC_SLP	FMC clock

Enum hal_rcu_periph_reset_enum

Table 3-315. Enum hal_rcu_periph_reset_enum

Member name	Function description
RCU_GPIOARST	GPIOA reset
RCU_GPIOBRST	GPIOB reset

Member name	Function description
RCU_GPIOCRST	GPIOC reset
RCU_GPIODRST	GPIOD reset
RCU_GPIOFRST	GPIOF reset
RCU_TSIRST	TSI reset
RCU_CFGCMRST	CFGCMP reset
RCU_ADCRST	ADC reset
RCU_TIMER0RST	TIMER0 reset
RCU_SPI0RST	SPI0 reset
RCU_USART0RST	USART0 reset
RCU_TIMER14RST	TIMER14 reset
RCU_TIMER15RST	TIMER15 reset
RCU_TIMER16RST	TIMER16 reset
RCU_TIMER1RST	TIMER1 reset
RCU_TIMER2RST	TIMER2 reset
RCU_TIMER13RST	TIMER13 reset
RCU_WWDGTRST	WWDGT reset
RCU_SPI1RST	SPI1 reset
RCU_USART1RST	USART1 reset
RCU_I2C0RST	I2C0 reset
RCU_I2C1RST	I2C1 reset
RCU_PMURST	PMU reset
RCU_DACRST	DAC reset (only for GD32F350)
RCU_CECRST	CEC reset (only for GD32F350)
RCU_TIMER5RST	TIMER5 reset (only for GD32F350)
RCU_USBFSRST	USBFS reset (only for GD32F350)
RCU_CTCRST	CTC reset

Enum hal_rcu_flag_enum

Table 3-316. Enum hal_rcu_flag_enum

Member name	Function description
RCU_FLAG_IRC40KSTB	IRC40K stabilization flags
RCU_FLAG_LXTALSTB	LXTAL stabilization flags
RCU_FLAG_IRC8MSTB	IRC8M stabilization flags
RCU_FLAG_HXTALSTB	HXTAL stabilization flags
RCU_FLAG_PLLSTB	PLL stabilization flags

Member name	Function description
RCU_FLAG_IRC28 MSTB	IRC28M stabilization flags
RCU_FLAG_IRC48 MSTB	IRC48M stabilization flags
RCU_FLAG_V12RS T	V12 domain Power reset flags
RCU_FLAG_OBLR ST	Option byte loader reset flags
RCU_FLAG_EPRS T	External PIN reset flags
RCU_FLAG_PORR ST	Power reset flags
RCU_FLAG_SWRS T	Software reset flags
RCU_FLAG_FWDG TRST	Free Watchdog timer reset flags
RCU_FLAG_WWD GTRST	Window watchdog timer reset flags
RCU_FLAG_LPRST	Low-power reset flags

Enum hal_rcu_int_flag_enum

Table 3-317. Enum hal_rcu_int_flag_enum

Member name	Function description
RCU_INT_FLAG_IR C40KSTB	IRC40K stabilization interrupt flag
RCU_INT_FLAG_L XTALSTB	LXTAL stabilization interrupt flag
RCU_INT_FLAG_IR C8MSTB	IRC8M stabilization interrupt flag
RCU_INT_FLAG_H XTALSTB	HXTAL stabilization interrupt flag
RCU_INT_FLAG_P LLSTB	PLL stabilization interrupt flag
RCU_INT_FLAG_IR C28MSTB	IRC28M stabilization interrupt flag
RCU_INT_FLAG_C KM	CKM interrupt flag
RCU_INT_FLAG_IR C48MSTB	IRC48M stabilization interrupt flag

Enum hal_rcu_int_flag_clear_enum

Table 3-318. Enum hal_rcu_int_flag_clear_enum

Member name	Function description
RCU_INT_FLAG_IR C40KSTB_CLR	IRC40K stabilization interrupt flags clear
RCU_INT_FLAG_L XTALSTB_CLR	LXTAL stabilization interrupt flags clear
RCU_INT_FLAG_IR C8MSTB_CLR	IRC8M stabilization interrupt flags clear
RCU_INT_FLAG_H XTALSTB_CLR	HXTAL stabilization interrupt flags clear
RCU_INT_FLAG_P LLSTB_CLR	PLL stabilization interrupt flags clear
RCU_INT_FLAG_IR C28MSTB_CLR	IRC28M stabilization interrupt flags clear
RCU_INT_FLAG_C KM_CLR	CKM interrupt flags clear
RCU_INT_FLAG_IR C48MSTB_CLR	IRC48M stabilization interrupt flag clear

Enum hal_rcu_int_enum

Table 3-319. Enum hal_rcu_int_enum

Member name	Function description
RCU_INT_IRC40KS TB	IRC40K stabilization interrupt
RCU_INT_LXTALS TB	LXTAL stabilization interrupt
RCU_INT_IRC8MS TB	IRC8M stabilization interrupt
RCU_INT_HXTALS TB	HXTAL stabilization interrupt
RCU_INT_PLLSTB	PLL stabilization interrupt
RCU_INT_IRC28M STB	IRC28M stabilization interrupt
RCU_INT_IRC48M STB	IRC48M stabilization interrupt

Enum hal_rcu_adc_clksrc_enum

Table 3-320. Enum hal_rcu_adc_clksrc_enum

Member name	Function description
RCU_ADCCCK_IRC2_8M_DIV2	ADC clock source select IRC28M/2
RCU_ADCCCK_IRC2_8M	ADC clock source select IRC28M
RCU_ADCCCK_APB2_DIV2	ADC clock source select APB2/2
RCU_ADCCCK_AHB_DIV3	ADC clock source select AHB/3
RCU_ADCCCK_APB2_DIV4	ADC clock source select APB2/4
RCU_ADCCCK_AHB_DIV5	ADC clock source select AHB/5
RCU_ADCCCK_APB2_DIV6	ADC clock source select APB2/6
RCU_ADCCCK_AHB_DIV7	ADC clock source select AHB/7
RCU_ADCCCK_APB2_DIV8	ADC clock source select APB2/8
RCU_ADCCCK_AHB_DIV9	ADC clock source select AHB/9

Enum hal_rcu_clock_freq_enum

Table 3-321. Enum hal_rcu_clock_freq_enum

Member name	Function description
CK_SYS	system clock
CK_AHB	AHB clock
CK_APB1	APB1 clock
CK_APB2	APB2 clock
CK_ADC	ADC clock
CK_CEC	CEC clock
CK_USART	USART clock

Enum hal_rcu_osci_type_enum

Table 3-322. Enum hal_rcu_osci_type_enum

Member name	Function description
RCU_HXTAL	HXTAL
RCU_LXTAL	LXTAL

Member name	Function description
RCU_IRC8M	IRC8M
RCU_IRC28M	IRC28M
RCU_IRC48M	IRC48M
RCU_IRC40K	IRC40K
RCU_PLL_CK	PLL

Enum hal_rcu_struct_type_enum

Table 3-323. Enum hal_rcu_struct_type_enum

Member name	Function description
HAL_RCU_CLK_STRUCTURE	RCU clock structure type
HAL_RCU_OSCILLATOR_STRUCTURE	RCU oscillator structure type
HAL_RCU_PERIPHERAL_CLOCK_STRUCTURE	RCU peripheral clock structure type

Enum hal_rcu_rtc_clksrc_enum

Table 3-324. Enum hal_rcu_rtc_clksrc_enum

Member name	Function description
RCU_RTC_CLKSRC_NONE	no clock selected
RCU_RTC_CLKSRC_LXTAL	LXTAL selected as RTC source clock
RCU_RTC_CLKSRC_IRC40K	IRC40K selected as RTC source clock
RCU_RTC_CLKSRC_HXTAL_DIV32	HXTAL/32 selected as RTC source clock

Enum hal_rcu_usart_clksrc_enum

Table 3-325. Enum hal_rcu_usart_clksrc_enum

Member name	Function description
RCU_USART0_CLKSRC_APB2	CK_USART0 select CK_APB2
RCU_USART0_CLKSRC_SYS	CK_USART0 select CK_SYS
RCU_USART0_CLKSRC_LXTAL	CK_USART0 select LXTAL
RCU_USART0_CLKSRC_IRC8M	CK_USART0 select IRC8M

Enum hal_rcu_usbfs_clksrc_enum

Table 3-326. Enum hal_rcu_usbfs_clksrc_enum

Member name	Function description
RCU_PLLCLK_USB FS_DIV1_5	USBFS clock prescaler select CK_PLL/1.5
RCU_PLLCLK_USB FS_DIV1	USBFS clock prescaler select CK_PLL
RCU_PLLCLK_USB FS_DIV2_5	USBFS clock prescaler select CK_PLL/2.5
RCU_PLLCLK_USB FS_DIV2	USBFS clock prescaler select CK_PLL/2
RCU_PLLCLK_USB FS_DIV3_5	USBFS clock prescaler select CK_PLL/3.5
RCU_PLLCLK_USB FS_DIV3	USBFS clock prescaler select CK_PLL/3

Enum hal_rcu_cec_clksrc_enum

Table 3-327. Enum hal_rcu_cec_clksrc_enum

Member name	Function description
RCU_CEC_CLKSR C_DIV244	CK_CEC clock source select IRC8M/244
RCU_CEC_CLKSR C_LXTAL	CK_CEC clock source select LXTAL

Enum hal_rcu_sysclk_src_enum

Table 3-328. Enum hal_rcu_sysclk_src_enum

Member name	Function description
RCU_SYSCLK_SR C_IRC8M	system clock source select IRC8M
RCU_SYSCLK_SR C_HXTAL	system clock source select HXTAL
RCU_SYSCLK_SR C_PLL	system clock source select PLL

Enum hal_rcu_ck48mclk_src_enum

Table 3-329. Enum hal_rcu_ck48mclk_src_enum

Member name	Function description
RCU_USB_CK48M SRC_PLL48M	CK48M clock source select PLL48M

Member name	Function description
RCU_USB_CK48M SRC_IRC48M	CK48M clock source select IRC48M

Enum hal_rcu_sysclk_ahbdiv_enum

Table 3-330. Enum hal_rcu_sysclk_ahbdiv_enum

Member name	Function description
RCU_SYSCLK_AH BDIV1	AHB prescaler select CK_SYS
RCU_SYSCLK_AH BDIV2	AHB prescaler select CK_SYS/2
RCU_SYSCLK_AH BDIV4	AHB prescaler select CK_SYS/4
RCU_SYSCLK_AH BDIV8	AHB prescaler select CK_SYS/8
RCU_SYSCLK_AH BDIV16	AHB prescaler select CK_SYS/16
RCU_SYSCLK_AH BDIV32	AHB prescaler select CK_SYS/32
RCU_SYSCLK_AH BDIV64	AHB prescaler select CK_SYS/64
RCU_SYSCLK_AH BDIV128	AHB prescaler select CK_SYS/128
RCU_SYSCLK_AH BDIV256	AHB prescaler select CK_SYS/256
RCU_SYSCLK_AH BDIV512	AHB prescaler select CK_SYS/512

Enum hal_rcu_ahbclk_apb1div_enum

Table 3-331. Enum hal_rcu_ahbclk_apb1div_enum

Member name	Function description
RCU_AHBCLK_AP B1DIV1	RCU_APB1_CKAHB_DIV1
RCU_AHBCLK_AP B1DIV2	RCU_APB1_CKAHB_DIV2
RCU_AHBCLK_AP B1DIV4	RCU_APB1_CKAHB_DIV4
RCU_AHBCLK_AP B1DIV8	RCU_APB1_CKAHB_DIV8
RCU_AHBCLK_AP B1DIV16	RCU_APB1_CKAHB_DIV16

Enum hal_rcu_ahbclk_apb2div_enum

Table 3-332. Enum hal_rcu_ahbclk_apb2div_enum

Member name	Function description
RCU_AHBCLK_APB2DIV1	RCU_APB2_CK_AHB_DIV1
RCU_AHBCLK_APB2DIV2	RCU_APB2_CK_AHB_DIV2
RCU_AHBCLK_APB2DIV4	RCU_APB2_CK_AHB_DIV4
RCU_AHBCLK_APB2DIV8	RCU_APB2_CK_AHB_DIV8
RCU_AHBCLK_APB2DIV16	RCU_APB2_CK_AHB_DIV16

Enum hal_rcu_osc_state_enum

Table 3-333. Enum hal_rcu_osc_state_enum

Member name	Function description
RCU_OSC_NONE	the oscillator is not configured
RCU_OSC_OFF	the oscillator deactivation
RCU_OSC_ON	the oscillator activation
RCU_OSC_BYPASS	external clock source for the oscillator
RCU_OSC_ADCCTL	IRC28M may be clock source of ADC

Enum hal_rcu_pll_src_enum

Table 3-334. Enum hal_rcu_pll_src_enum

Member name	Function description
RCU_PLL_SRC_HXTAL_IRC48M	PLL clock source select HXTAL or IRC48M
RCU_PLL_SRC_IRC8M_DIV2	PLL clock source select IRC8M/2

Enum hal_rcu_pll_prediv_enum

Table 3-335. Enum hal_rcu_pll_prediv_enum

Member name	Function description
RCU_PLL_PREDIV1	PLL not divided
RCU_PLL_PREDIV	PLL divided by 2

Member name	Function description
2	
RCU_PLL_PREDIV 3	PLL divided by 3
RCU_PLL_PREDIV 4	PLL divided by 4
RCU_PLL_PREDIV 5	PLL divided by 5
RCU_PLL_PREDIV 6	PLL divided by 6
RCU_PLL_PREDIV 7	PLL divided by 7
RCU_PLL_PREDIV 8	PLL divided by 8
RCU_PLL_PREDIV 9	PLL divided by 9
RCU_PLL_PREDIV 10	PLL divided by 10
RCU_PLL_PREDIV 11	PLL divided by 11
RCU_PLL_PREDIV 12	PLL divided by 12
RCU_PLL_PREDIV 13	PLL divided by 13
RCU_PLL_PREDIV 14	PLL divided by 14
RCU_PLL_PREDIV 15	PLL divided by 15
RCU_PLL_PREDIV 16	PLL divided by 16

Enum hal_rcu_pll_mul_enum

Table 3-336. Enum hal_rcu_pll_mul_enum

Member name	Function description
RCU_PLL_MULT2	PLL source clock multiply by 2
RCU_PLL_MULT3	PLL source clock multiply by 3
RCU_PLL_MULT4	PLL source clock multiply by 4
RCU_PLL_MULT5	PLL source clock multiply by 5
RCU_PLL_MULT6	PLL source clock multiply by 6
RCU_PLL_MULT7	PLL source clock multiply by 7
RCU_PLL_MULT8	PLL source clock multiply by 8

Member name	Function description
RCU_PLL_MULT9	PLL source clock multiply by 9
RCU_PLL_MULT10	PLL source clock multiply by 10
RCU_PLL_MULT11	PLL source clock multiply by 11
RCU_PLL_MULT12	PLL source clock multiply by 12
RCU_PLL_MULT13	PLL source clock multiply by 13
RCU_PLL_MULT14	PLL source clock multiply by 14
RCU_PLL_MULT15	PLL source clock multiply by 15
RCU_PLL_MULT16	PLL source clock multiply by 16
RCU_PLL_MULT17	PLL source clock multiply by 17
RCU_PLL_MULT18	PLL source clock multiply by 18
RCU_PLL_MULT19	PLL source clock multiply by 19
RCU_PLL_MULT20	PLL source clock multiply by 20
RCU_PLL_MULT21	PLL source clock multiply by 21
RCU_PLL_MULT22	PLL source clock multiply by 22
RCU_PLL_MULT23	PLL source clock multiply by 23
RCU_PLL_MULT24	PLL source clock multiply by 24
RCU_PLL_MULT25	PLL source clock multiply by 25
RCU_PLL_MULT26	PLL source clock multiply by 26
RCU_PLL_MULT27	PLL source clock multiply by 27
RCU_PLL_MULT28	PLL source clock multiply by 28
RCU_PLL_MULT29	PLL source clock multiply by 29
RCU_PLL_MULT30	PLL source clock multiply by 30
RCU_PLL_MULT31	PLL source clock multiply by 31
RCU_PLL_MULT32	PLL source clock multiply by 32
RCU_PLL_MULT33	PLL source clock multiply by 33
RCU_PLL_MULT34	PLL source clock multiply by 34
RCU_PLL_MULT35	PLL source clock multiply by 35
RCU_PLL_MULT36	PLL source clock multiply by 36
RCU_PLL_MULT37	PLL source clock multiply by 37
RCU_PLL_MULT38	PLL source clock multiply by 38
RCU_PLL_MULT39	PLL source clock multiply by 39
RCU_PLL_MULT40	PLL source clock multiply by 40
RCU_PLL_MULT41	PLL source clock multiply by 41
RCU_PLL_MULT42	PLL source clock multiply by 42
RCU_PLL_MULT43	PLL source clock multiply by 43
RCU_PLL_MULT44	PLL source clock multiply by 44
RCU_PLL_MULT45	PLL source clock multiply by 45
RCU_PLL_MULT46	PLL source clock multiply by 46
RCU_PLL_MULT47	PLL source clock multiply by 47
RCU_PLL_MULT48	PLL source clock multiply by 48

Member name	Function description
RCU_PLL_MULT49	PLL source clock multiply by 49
RCU_PLL_MULT50	PLL source clock multiply by 50
RCU_PLL_MULT51	PLL source clock multiply by 51
RCU_PLL_MULT52	PLL source clock multiply by 52
RCU_PLL_MULT53	PLL source clock multiply by 53
RCU_PLL_MULT54	PLL source clock multiply by 54
RCU_PLL_MULT55	PLL source clock multiply by 55
RCU_PLL_MULT56	PLL source clock multiply by 56
RCU_PLL_MULT57	PLL source clock multiply by 57
RCU_PLL_MULT58	PLL source clock multiply by 58
RCU_PLL_MULT59	PLL source clock multiply by 59
RCU_PLL_MULT60	PLL source clock multiply by 60
RCU_PLL_MULT61	PLL source clock multiply by 61
RCU_PLL_MULT62	PLL source clock multiply by 62
RCU_PLL_MULT63	PLL source clock multiply by 63
RCU_PLL_MULT64	PLL source clock multiply by 64

Enum hal_rcu_pll_presel_enum

Table 3-337. Enum hal_rcu_pll_presel_enum

Member name	Function description
RCU_PLL_PRESEL_HXTAL	PLL clock source preselection HXTAL
RCU_PLL_PRESEL_IRC48M	PLL clock source preselection IRC48M

Enum hal_rcu_ckout_src_enum

Table 3-338. Enum hal_rcu_ckout_src_enum

Member name	Function description
RCU_CKOUT_SRC_NONE	no clock selected
RCU_CKOUT_SRC_IRC28M	CK_OUT clock source select IRC28M
RCU_CKOUT_SRC_IRC40K	CK_OUT clock source select IRC40K
RCU_CKOUT_SRC_LXTAL	CK_OUT clock source select LXTAL
RCU_CKOUT_SRC_CKSYS	CK_OUT clock source select CKSYS
RCU_CKOUT_SRC	CK_OUT clock source select IRC8M

Member name	Function description
_IRC8M	
RCU_CLKOUT_SRC_HXTAL	CK_OUT clock source select HXTAL
RCU_CLKOUT_SRC_CKPLL_DIV2	CK_OUT clock source select PLL/2
RCU_CLKOUT_SRC_CKPLL_DIV1	CK_OUT clock source select PLL

Enum hal_rcu_ckout_div_enum

Table 3-339. Enum hal_rcu_ckout_div_enum

Member name	Function description
RCU_CLKOUT_DIV_1	CK_OUT is divided by 1
RCU_CLKOUT_DIV_2	CK_OUT is divided by 2
RCU_CLKOUT_DIV_4	CK_OUT is divided by 4
RCU_CLKOUT_DIV_8	CK_OUT is divided by 8
RCU_CLKOUT_DIV_16	CK_OUT is divided by 16
RCU_CLKOUT_DIV_32	CK_OUT is divided by 32
RCU_CLKOUT_DIV_64	CK_OUT is divided by 64
RCU_CLKOUT_DIV_128	CK_OUT is divided by 128

Structure hal_rcu_periphclk_struct

Table 3-340. Structure hal_rcu_periphclk_struct

Member name	Function description
uint32_t periph_clock_type	peripherals clock type selection
hal_rcu_rtc_clksrc_enum	RTC clock source selection
hal_rcu_usart_clksrc_enum	usart0 clock source selection
hal_rcu_adc_clksrc_enum	ADC clock source selection
hal_rcu_usbfs_clksr	USBFS clock source selection

c_enum	
hal_rcu_cec_clksrc_enum	CEC clock source selection

Structure hal_rcu_clk_struct

Table 3-341. Structure hal_rcu_clk_struct

Member name	Function description
uint32_t clock_type	rcu clock type to be configured
hal_rcu_sysclk_src_enum	the system clock source
hal_rcu_ck48mclk_src_enum	the ck48m clock source
hal_rcu_sysclk_ahb_div_enum	the AHB clock divider
hal_rcu_ahbclk_apb1_div_enum	the APB1 clock divider
hal_rcu_ahbclk_apb2div_enum	the APB2 clock divider

Structure hal_rcu_irq_struct

Table 3-342. Structure hal_rcu_irq_struct

Member name	Function description
pll_stable_handle	PLL clock stable interrupt
irc40k_stable_handle	IRC40K clock stable interrupt
irc8m_stable_handle	IRC8M clock stable interrupt
irc28m_stable_handle	IRC28M clock stable interrupt
irc48m_stable_handle	IRC28M clock stable interrupt
lxtal_stable_handle	LXTAL clock stable interrupt
hxtal_stable_handle	HXTAL clock stable interrupt
hxtal_stuck_handle	HXTAL clock stuck interrupt

Structure hal_rcu_hxtal_struct

Table 3-343. Structure hal_rcu_hxtal_struct

Member name	Function description
ControlStatus	the oscillators configure flag
hal_rcu_osc_state_	the oscillators state

enum	
------	--

Structure hal_rcu_lxtal_struct

Table 3-344. Structure hal_rcu_lxtal_struct

Member name	Function description
ControlStatus	the oscillators configure flag
hal_rcu_osc_state_ enum	the oscillators state

Structure hal_rcu_irc8m_struct

Table 3-345. Structure hal_rcu_irc8m_struct

Member name	Function description
ControlStatus	the oscillators configure flag
uint8_t adjust_value	the oscillators adjust value
hal_rcu_osc_state_ enum	the oscillators state

Structure hal_rcu_irc28m_struct

Table 3-346. Structure hal_rcu_irc28m_struct

Member name	Function description
ControlStatus	the oscillators configure flag
uint8_t adjust_value	the oscillators adjust value
hal_rcu_osc_state_ enum	the oscillators state

Structure hal_rcu_irc48m_struct

Table 3-347. Structure hal_rcu_irc48m_struct

Member name	Function description
ControlStatus	the oscillators configure flag
hal_rcu_osc_state_ enum	the oscillators state

Structure hal_rcu_irc40k_struct

Table 3-348. Structure hal_rcu_irc40k_struct

Member name	Function description
ControlStatus	the oscillators configure flag
hal_rcu_osc_state_ enum	the oscillators state

Structure hal_rcu_pll_struct

Table 3-349. Structure hal_rcu_pll_struct

Member name	Function description
ControlStatus	the oscillators configure flag
hal_rcu_osc_state_enum	the oscillators state
hal_rcu_pll_src_enum	PLL entry clock source
hal_rcu_pll_prediv_enum	predivision factor of PLL input clock
hal_rcu_pll_mul_enum	multiplication factor of PLL input clock
hal_rcu_pll_presel_enum	source preselection of PLL input clock

Structure hal_rcu_osci_struct

Table 3-350. Structure hal_rcu_osci_struct

Member name	Function description
hal_rcu_hxtal_struct	HXTAL status structure
hal_rcu_lxtal_struct	LXTAL status structure
hal_rcu_irc8m_struct	IRC8M status structure
hal_rcu_irc28m_struct	IRC28M status structure
hal_rcu_irc48m_struct	IRC48M status structure
hal_rcu_irc40k_struct	IRC40K status structure
hal_rcu_pll_struct	PLL status structure

hal_rcu_osci_config

The description of hal_rcu_osci_config is shown as below:

Table 3-351. Function hal_rcu_osci_config

Function name	hal_rcu_osci_config
Function prototype	int32_t hal_rcu_osci_config(hal_rcu_osci_struct *rcu_osci)
Function descriptions	configure the RCU oscillators
Precondition	-
The called functions	hal_rcu_deinit hals_rcu_system_clock_source_get

	<div> <div>hals_rcu_osci_off</div> <div>hals_rcu_osci_bypass_mode_disable</div> <div>hals_rcu_osci_on</div> <div>hals_rcu_osci_stab_wait</div> <div>hal_sys_basetick_count_get</div> <div>hal_sys_basetick_timeout_check</div> <div>hal_rcu_periph_clk_enable</div> <div>hals_pmu_backup_write_enable</div> <div>hals_rcu_irc8m_adjust_value_set</div> <div>hals_rcu_irc28m_adjust_value_set</div> <div>hals_rcu_pll_preselection_config</div> <div>hals_rcu_hxtal_prediv_config</div> <div>hals_rcu_pll_config</div> </div>
Input parameter{in}	
rcu_osci	points to hal_rcu_osci_struct, struct members refer to - Table 3-350. Structure hal_rcu_osci_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_TIMEOUT, HAL_ERR_VAL, HAL_ERR_NONE

Example:

```

/* configure the RCU oscillators */

hal_rcu_osci_struct rcu_osci_parameter;

rcu_osci_parameter.irc8m.need_configure = ENABLE;

rcu_osci_parameter.irc8m.state = RCU_OSC_ON;

rcu_osci_parameter.irc8m.adjust_value = 0;

rcu_osci_parameter.pll.need_configure = ENABLE;

rcu_osci_parameter.pll.state = RCU_OSC_ON;

rcu_osci_parameter.pll.pll_source = RCU_PLL_SRC_IRC8M_DIV2;

rcu_osci_parameter.pll.pll_mul = RCU_PLL_MULT27;

rcu_osci_parameter.pll.pll_presel = RCU_PLL_PRESEL_HXTAL;

if(HAL_ERR_NONE != hal_rcu_osci_config(&rcu_osci_parameter)){

    while(1);

}

```

hal_rcu_clock_out_config

The description of hal_rcu_clock_out_config is shown as below:

Table 3-352. Function hal_rcu_clock_out_config

Function name	hal_rcu_clock_out_config
Function prototype	void hal_rcu_clock_out_config(hal_rcu_ckout_src_enum ckout_src, hal_rcu_ckout_div_enum ckout_div)
Function descriptions	configure the clock out to output on CKOUT pin
Precondition	-
The called functions	hal_gpio_init hals_rcu_ckout_config
Input parameter{in}	
ckout_src	points to hal_rcu_ckout_src_enum, enum members refer to - Table 3-338. Enum hal_rcu_ckout_src_enum
Input parameter{in}	
ckout_div	points to hal_rcu_ckout_div_enum, enum members refer to - Table 3-339. Enum hal_rcu_ckout_div_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the clock out to output on CKOUT pin */
hal_rcu_clock_out_config(RCU_CKOUT_SRC_NONE, RCU_CLKOUT_DIV1);
```

hal_rcu_deinit

The description of hal_rcu_deinit is shown as below:

Table 3-353. Function hal_rcu_deinit

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

Example:

```
/* deinitialize the RCU */
```

```
hal_rcu_deinit();
```

hal_rcu_struct_init

The description of hal_rcu_struct_init is shown as below:

Table 3-354. Function hal_rcu_struct_init

Function name	hal_rcu_struct_init
Function prototype	void hal_rcu_struct_init(hal_rcu_struct_type_enum rcu_struct_type, void *p_struct)
Function descriptions	initialize the RCU structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
rcu_struct_type	points to hal_rcu_struct_type_enum, enum members refer to - Table 3-323. Enum hal_rcu_struct_type_enum
Input parameter{in}	
p_struct	point to RCU structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the RCU structure with the default values */
```

```
hal_rcu_clk_struct rcu_clk_parameter;
```

```
hal_rcu_struct_init(HAL_RCU_CLK_STRUCT, &rcu_clk_parameter);
```

hal_rcu_periph_clk_enable

The description of hal_rcu_periph_clk_enable is shown as below:

Table 3-355. Function hal_rcu_periph_clk_enable

Function name	hal_rcu_periph_clk_enable
Function prototype	void hal_rcu_periph_clk_enable(hal_rcu_periph_enum periph)
Function descriptions	enable the peripherals clock
Precondition	-
The called functions	-
Input parameter{in}	
periph	points to hal_rcu_periph_enum, enum members refer to - Table 3-313.

	<u>Enum hal_rcu_periph_enum</u>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the peripherals clock */
```

```
hal_rcu_periph_clk_enable(RCU_DMA);
```

hal_rcu_periph_clk_disable

The description of hal_rcu_periph_clk_disable is shown as below:

Table 3-356. Function hal_rcu_periph_clk_disable

Function name	hal_rcu_periph_clk_disable
Function prototype	void hal_rcu_periph_clk_disable(hal_rcu_periph_enum periph)
Function descriptions	disable the peripherals clock
Precondition	-
The called functions	-
Input parameter{in}	
periph	points to hal_rcu_periph_enum, enum members refer to - <u>Table 3-313.</u> <u>Enum hal_rcu_periph_enum</u>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the peripherals clock */
```

```
hal_rcu_periph_clk_disable(RCU_DMA);
```

hal_rcu_hxtal_clock_monitor_enable

The description of hal_rcu_hxtal_clock_monitor_enable is shown as below:

Table 3-357. Function hal_rcu_hxtal_clock_monitor_enable

Function name	hal_rcu_hxtal_clock_monitor_enable
Function prototype	void hal_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 */
```

```
hal_rcu_hxtal_clock_monitor_enable();
```

hal_rcu_hxtal_clock_monitor_disable

The description of hal_rcu_hxtal_clock_monitor_disable is shown as below:

Table 3-358. Function hal_rcu_hxtal_clock_monitor_disable

Function name	hal_rcu_hxtal_clock_monitor_disable
Function prototype	void hal_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 */
```

```
hal_rcu_hxtal_clock_monitor_disable();
```

hal_rcu_periph_reset_enable

The description of hal_rcu_periph_reset_enable is shown as below:

Table 3-359. Function hal_rcu_periph_reset_enable

Function name	hal_rcu_periph_reset_enable
Function prototype	void hal_rcu_periph_reset_enable(hal_rcu_periph_reset_enum periph_reset)
Function descriptions	reset the peripherals
Precondition	-
The called functions	-
Input parameter{in}	

periph_reset	points to hal_rcu_periph_reset_enum, enum members refer to - Table 3-315. Enum hal_rcu_periph_reset_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the peripherals */
```

```
hal_rcu_periph_reset_enable(RCU_GPIOARST);
```

hal_rcu_periph_reset_disable

The description of hal_rcu_periph_reset_disable is shown as below:

Table 3-360. Function hal_rcu_periph_reset_disable

Function name	hal_rcu_periph_reset_disable
Function prototype	void hal_rcu_periph_reset_disable(hal_rcu_periph_reset_enum periph_reset)
Function descriptions	disable reset the peripherals
Precondition	-
The called functions	-
Input parameter{in}	
periph_reset	points to hal_rcu_periph_reset_enum, enum members refer to - Table 3-315. Enum hal_rcu_periph_reset_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable reset the peripherals */
```

```
hal_rcu_periph_reset_disable(RCU_GPIOARST);
```

hal_rcu_periph_clock_config

The description of hal_rcu_periph_clock_config is shown as below:

Table 3-361. Function hal_rcu_periph_clock_config

Function name	hal_rcu_periph_clock_config
Function prototype	int32_t hal_rcu_periph_clock_config(hal_rcu_periphclk_struct *periph_clk)
Function descriptions	initialize the RCU extended peripherals(RTC, Usart0, ADC, USBFS, CEC) clocks

Precondition	-
The called functions	hal_rcu_periph_clk_enable hals_pmu_backup_write_enable hal_sys_basetick_count_get hal_sys_basetick_timeout_check hals_rcu_bkp_reset_enable hals_rcu_bkp_reset_disable hals_rcu_osc_stab_wait hals_rcu_rtc_clock_config hal_rcu_periph_clk_disable hals_rcu_usart_clock_config hals_rcu_adc_clock_config hals_rcu_cec_clock_config hals_rcu_usbfs_clock_config
Input parameter{in}	
periph_clk	points to hal_rcu_periphclk_struct, struct members refer to - Table 3-340. Structure hal_rcu_periphclk_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_TIMEOUT, HAL_ERR_VAL, HAL_ERR_NONE

Example:

```

/* initialize the RCU extended peripherals(RTC, Usart0, ADC, USBFS, CEC) clocks */
hal_rcu_periphclk_struct rcu_periphclk_parameter;

rcu_periphclk_parameter.periph_clock_type = RCU_PERIPH_CLKTYPE_ADC;
rcu_periphclk_parameter.adc_clock_source = RCU_ADCK_APB2_DIV2;
if(HAL_ERR_NONE != hal_rcu_periph_clock_config(&rcu_periphclk_parameter)){
    while(1);
}

```

hal_rcu_periph_clkfreq_get

The description of hal_rcu_periph_clkfreq_get is shown as below:

Table 3-362. Function hal_rcu_periph_clkfreq_get

Function name	hal_rcu_periph_clkfreq_get
Function prototype	uint32_t hal_rcu_periph_clkfreq_get(uint32_t periph_clk)
Function descriptions	get the peripherals clock frequency

Precondition	-
The called functions	hals_rcu_clock_freq_get
Input parameter{in}	
periph_clk	peripherals clock type
RCU_PERIPH_CLKTY PE_NONE	no clock type
RCU_PERIPH_CLKTY PE_RTC	RTC clock type
RCU_PERIPH_CLKTY PE_USART0	usart0 clock type
RCU_PERIPH_CLKTY PE_ADC	ADC clock type
RCU_PERIPH_CLKTY PE_APB1TIMER	APB1 timer clock type
RCU_PERIPH_CLKTY PE_APB2TIMER	APB2 timer clock type
RCU_PERIPH_CLKTY PE_CEC	CEC clock type
Output parameter{out}	
-	-
Return value	
int32_t	periph_freq

Example:

```
/* get the peripherals clock frequency */
```

```
hal_rcu_periph_clkfreq_get(RCU_PERIPH_CLKTYPE_USART0);
```

hal_rcu_osci_config_get

The description of hal_rcu_osci_config_get is shown as below:

Table 3-363. Function hal_rcu_osci_config_get

Function name	hal_rcu_osci_config_get
Function prototype	void hal_rcu_osci_config_get(hal_rcu_osci_struct *rcu_osci)
Function descriptions	get the RCU oscillators configuration
Precondition	-
The called functions	-
Input parameter{in}	
rcu_osci	points to hal_rcu_osci_struct, struct members refer to - Table 3-350. Structure hal_rcu_osci_struct
Output parameter{out}	
-	-

Return value	
-	-

Example:

```

/* get the RCU oscillators configuration */
hal_rcu_osci_struct rcu_osci_parameter;

rcu_osci_parameter.hxtal.need_configure = ENABLE;

rcu_osci_parameter.hxtal.state = RCU_OSC_BYPASS;

rcu_osci_parameter.pll.need_configure = ENABLE;

rcu_osci_parameter.pll.state = RCU_OSC_ON;

rcu_osci_parameter.pll.pll_source = RCU_PLL_SRC_HXTAL_IRC48M;

rcu_osci_parameter.pll.pll_mul = RCU_PLL_MULT2;

rcu_osci_parameter.pll.pll_presele = RCU_PLL_PRESEL_HXTAL;

rcu_osci_parameter.pll.pre_div = RCU_PLL_PREDIV16;

hal_rcu_osci_config_get(&rcu_osci_parameter);

```

hal_rcu_clock_config

The description of hal_rcu_clock_config is shown as below:

Table 3-364. Function hal_rcu_clock_config

Function name	hal_rcu_clock_config
Function prototype	int32_t hal_rcu_clock_config(hal_rcu_clk_struct *rcu_clk)
Function descriptions	configure the RCU clock
Precondition	-
The called functions	hals_rcu_flag_get hals_rcu_system_clock_source_config hal_sys_basetick_count_get hal_sys_basetick_timeout_check hals_rcu_ck48m_clock_config hals_rcu_ahb_clock_config hals_rcu_apb1_clock_config hals_rcu_apb2_clock_config hals_rcu_clock_freq_get hal_sys_timesource_init
Input parameter{in}	
rcu_clk	points to hal_rcu_clk_struct, struct members refer to - Table 3-341. Structure hal_rcu_clk_struct

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_TIMEOUT, HAL_ERR_NONE

Example:

```

/* configure RCU clock */

hal_rcu_clk_struct rcu_clk_parameter;

rcu_clk_parameter.clock_type = RCU_CLKTYPE_SYSCCLK | RCU_CLKTYPE_AHBCLK |
RCU_CLKTYPE_APB1CLK | RCU_CLKTYPE_APB2CLK;

rcu_clk_parameter.sysclk_source = RCU_SYSCCLK_SRC_IRC8M;

rcu_clk_parameter.ahbclk_divider = RCU_SYSCCLK_AHBDIV1;

rcu_clk_parameter.apb1clk_divider = RCU_AHBCLK_APB1DIV1;

rcu_clk_parameter.apb2clk_divider = RCU_AHBCLK_APB2DIV1;

if(HAL_ERR_NONE != hal_rcu_clock_config(&rcu_clk_parameter)){
    while(1);
}

```

hal_rcu_clock_config_get

The description of hal_rcu_clock_config_get is shown as below:

Table 3-365. Function hal_rcu_clock_config_get

Function name	hal_rcu_clock_config_get
Function prototype	void hal_rcu_clock_config_get(hal_rcu_clk_struct *rcu_clk)
Function descriptions	get the RCU clock configuration
Precondition	-
The called functions	-
Input parameter{in}	
rcu_clk	points to hal_rcu_clk_struct, struct members refer to - Table 3-341. Structure hal_rcu_clk_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure RCU clock */

```

```

hal_rcu_clk_struct rcu_clk_parameter;

rcu_clk_parameter.clock_type = RCU_CLKTYPE_SYSCCLK | RCU_CLKTYPE_AHBCLK |
RCU_CLKTYPE_APB1CLK | RCU_CLKTYPE_APB2CLK;

rcu_clk_parameter.sysclk_source = RCU_SYSCCLK_SRC_IRC8M;

rcu_clk_parameter.ahbclk_divider = RCU_SYSCCLK_AHBDIV1;

rcu_clk_parameter.apb1clk_divider = RCU_AHBCLK_APB1DIV1;

rcu_clk_parameter.apb2clk_divider = RCU_AHBCLK_APB2DIV1;

hal_rcu_clock_config_get(&rcu_clk_parameter);

```

hal_SystemCoreClockUpdate

The description of hal_SystemCoreClockUpdate is shown as below:

Table 3-366. Function hal_SystemCoreClockUpdate

Function name	hal_SystemCoreClockUpdate
Function prototype	int32_t hal_SystemCoreClockUpdate(void)
Function descriptions	update the SystemCoreClock with current core clock retrieved from cpu registers
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
int32_t	g_systemcoreclock

Example:

```

/* update the SystemCoreClock with current core clock retrieved from cpu registers */

Int32_t sys_clk = hal_SystemCoreClockUpdate();

```

hal_rcu_irq

The description of hal_rcu_irq is shown as below:

Table 3-367. Function hal_rcu_irq

Function name	hal_rcu_irq
Function prototype	void hal_rcu_irq(void)
Function descriptions	RCU interrupt handler content function, which is merely used in RCU_handler

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

Example:

```
/* RCU interrupt handler content function,which is merely used in RCU_handler */
hal_rcu_irq();
```

hal_rcu_irq_handle_set

The description of hal_rcu_irq_handle_set is shown as below:

Table 3-368. Function hal_rcu_irq_handle_set

Function name	hal_rcu_irq_handle_set
Function prototype	void hal_rcu_irq_handle_set(hal_rcu_irq_struct *prcu_irq)
Function descriptions	set user-defined interrupt callback function,which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	hals_rcu_interrupt_enable hals_rcu_interrupt_disable
Input parameter{in}	
prcu_irq	points to hal_rcu_irq_struct, struct members refer to - Table 3-342. Structure hal_rcu_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_irq_handle_cb hxtal_cb;
hal_rcu_irq_struct hal_rcu_irq;
hal_rcu_irq.hxtal_stable_handle = hxtal_cb;
/* set HXTAL stable interrupt handler */
hal_rcu_irq_handle_set(&hal_rcu_irq);
```


hal_rcu_irq_handle_all_reset

The description of hal_rcu_irq_handle_all_reset is shown as below:

Table 3-369. Function hal_rcu_irq_handle_all_reset

Function name	hal_rcu_irq_handle_all_reset
Function prototype	void hal_rcu_irq_handle_all_reset(void);
Function descriptions	reset all user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined callback functions */
```

```
hal_rcu_irq_handle_all_reset();
```

3.19. RTC

The Real-time Clock (RTC) is usually used as a clock-calendar. The ones in the Backup Domain consist of a 32-bit up-counter, an alarm, a prescaler, a divider and the RTC clock configuration register. The RTC registers are listed in chapter [3.19.1](#), the RTC firmware functions are introduced in chapter [3.19.2](#).

3.19.1. Descriptions of Peripheral registers

RTC registers are listed in the table shown as below:

Table 3-370. RTC Registers

Registers	Descriptions
RTC_TIME	RTC time of day register
RTC_DATE	RTC date register
RTC_CTL	RTC control register

Registers	Descriptions
RTC_STAT	RTC status register
RTC_PSC	RTC time prescaler register
RTC_ALRM0TD	RTC alarm 0 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_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

3.19.2. Descriptions of Peripheral functions

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

Table 3-371. RTC firmware function

Function name	Function description
hal_rtc_init	init RTC
hal_rtc_alarm_output_config	configure RTC alarm output
hal_rtc_alarm_config	configure RTC alarm
hal_rtc_tamp_config	configure RTC tamper
hal_rtc_timestamp_config	configure RTC time stamp
hal_rtc_calib_config	configure RTC calibration
hal_rtc_refclock_detection_config	configure RTC reference clock detection function
hal_rtc_struct_init	initialize the RTC structure with the default values
hal_rtc_deinit	deinitialize RTC device structure and init structure
hal_rtc_interrupt_enable	enable RTC interrupt
hal_rtc_interrupt_disable	disable RTC interrupt
hal_rtc_irq	RTC interrupt handler content function, which is merely used in rtc_handler
hal_rtc_irq_handle_set	set user-defined interrupt callback function
hal_rtc_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_rtc_current_time_get	get current time and date
hal_rtc_alarm_get	get RTC alarm time

Function name	Function description
hal_rtc_alarm_event_poll	poll for RTC alarm event
hal_rtc_timestamp_get	get RTC timestamp time and date
hal_rtc_timestamp_event_poll	poll for RTC timestamp event
hal_rtc_tamper0_event_poll	poll for RTC tamper0 event
hal_rtc_tamper1_event_poll	poll for RTC tamper1 event
hal_rtc_daylight_saving_time_adjust	adjust the daylight saving time by adding or subtracting one hour from the current time

Enum hal_rtc_state_enum

Table 3-372. Enum hal_rtc_state_enum

Member name	Function description
HAL_RTC_STATE_NONE	NONE(default value)
HAL_RTC_STATE_RESET	RESET
HAL_RTC_STATE_BUSY	BUSY
HAL_RTC_STATE_TIMEOUT	TIMEOUT
HAL_RTC_STATE_ERROR	ERROR
HAL_RTC_STATE_READY	READY

Enum hal_rtc_error_enum

Table 3-373. Enum hal_rtc_error_enum

Member name	Function description
HAL_RTC_ERROR_NONE	no error
HAL_RTC_ERROR_SYSTEM	RTC internal error
HAL_RTC_ERROR_CONFIG	configuration error occurs

Enum hal_rtc_struct_type_enum

Table 3-374. Enum hal_rtc_struct_type_enum

Member name	Function description
HAL_RTC_INIT_STRUCTURE	RTC init structure

HAL_RTC_ALARM_OUTPUT_CONFIG_STRUCT	RTC alarm output config structure
HAL_RTC_ALARM_CONFIG_STRUCT	RTC alarm config structure
HAL_RTC_TAMPER_CONFIG_STRUCT	RTC tamper config structure
HAL_RTC_TIMESTAMP_STRUCT	RTC timestamp config structure
HAL_RTC_IRQ_STRUCT	RTC irq structure
HAL_RTC_DEV_STRUCT	RTC device structure

Structure hal_rtc_irq_struct

Table 3-375. Structure hal_rtc_irq_struct

Member name	Function description
rtc_timestamp_handler	RTC timestamp handler function
rtc_alarm_handler	RTC alarm handler function
rtc_tamper0_handler	RTC tamper0 handler function
rtc_tamper1_handler	RTC tamper1 handler function

Structure hal_rtc_dev_struct

Table 3-376. Structure hal_rtc_dev_struct

Member name	Function description
rtc_irq	RTC device interrupt callback function pointer structure
error_state	RTC error state
state	RTC state
mutex	mutex
priv	priv data

Structure hal_rtc_timestamp_struct

Table 3-377. hal_rtc_timestamp_struct

Member name	Function description
rtc_timestamp_month	RTC time-stamp month value
rtc_timestamp_date	RTC time-stamp date value
rtc_timestamp_weekday	RTC time-stamp weekday value

rtc_timestamp_hour	RTC time-stamp hour value
rtc_timestamp_minute	RTC time-stamp minute value
rtc_timestamp_second	RTC time-stamp second value
rtc_timestamp_subsecond	RTC time-stamp subsecond value
rtc_am_pm	RTC time-stamp AM/PM value

Structure hal_rtc_init_struct

Table 3-378. hal_rtc_init_struct

Member name	Function description
rtc_year	RTC year value
rtc_month	RTC month value
rtc_date	RTC date value
rtc_day_of_week	RTC weekday value
rtc_hour	RTC hour value
rtc_minute	RTC minute value
rtc_second	RTC second value
rtc_subsecond	RTC subsecond value
rtc_am_pm	RTC AM/PM value
rtc_daylight_saving	RTC daylight saving
rtc_clock_format	RTC display format of clock system
rtc_psc_factor_s	RTC synchronous prescaler value
rtc_psc_factor_a	RTC asynchronous prescaler value

Structure hal_rtc_alarm_output_config_struct

Table 3-379. hal_rtc_alarm_output_config_struct

Member name	Function description
rtc_alarm_output_polarity	alarm output polarity
rtc_alarm_output_type	alarm output type control/PC13 output value

Structure hal_rtc_alarm_config_struct

Table 3-380. hal_rtc_alarm_config_struct

Member name	Function description
rtc_alarm_am_pm	RTC alarm AM/PM value
rtc_alarm_weekday_mask	RTC alarm weekday mask

rtc_alarm_hour_mask	RTC alarm hour mask
rtc_alarm_minute_mask	RTC alarm minute mask
rtc_alarm_second_mask	RTC alarm second mask
rtc_alarm_subsecond_mask	RTC alarm subsecond mask
rtc_weekday_or_date	specify RTC alarm is on date or weekday
rtc_alarm_day	RTC alarm date or weekday value
rtc_alarm_hour	RTC alarm hour value
rtc_alarm_minute	RTC alarm minute value
rtc_alarm_second	RTC alarm second value
rtc_alarm_subsecond	RTC alarm subsecond value

Structure hal_rtc_tamper_config_struct

Table 3-381. hal_rtc_tamper_config_struct

Member name	Function description
rtc_tamper_filter	RTC tamper consecutive samples needed during a voltage level detection
rtc_tamper_sample_frequency	RTC tamper sampling frequency during a voltage level detection
rtc_tamper_precharge_time	RTC tamper precharge duration if precharge feature is enabled
rtc_tamper_precharge_enable	RTC tamper precharge feature during a voltage level detection
rtc_tamper_with_time_stamp	RTC tamper time-stamp feature
rtc_tamper0_trigger	RTC tamper0 trigger
rtc_tamper1_trigger	RTC tamper1 trigger
rtc_tamper0_source	RTC tamper0 source
rtc_tamper1_source	RTC tamper1 source

hal_rtc_init

The description of hal_rtc_init is shown as below:

Table 3-382. Function hal_rtc_init

Function name	hal_rtc_init
Function prototype	int32_t hal_rtc_init(hal_rtc_dev_struct *rtc_dev, hal_rtc_init_struct *rtc_init);
Function descriptions	initialize RTC

Precondition	-
The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal_rtc_dev_struct
Input parameter{in}	
rtc_init	points to hal_rtc_init_struct, struct members refer to Table 3-378. hal_rtc_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* initialize RTC */

hal_rtc_dev_struct rtc_info;
hal_rtc_init_struct rtc_init_parameter;
rtc_init_parameter.rtc_hour = 16;
rtc_init_parameter.rtc_minute = 30;
rtc_init_parameter.rtc_second = 0;
rtc_init_parameter.rtc_daylight_saving = RTC_DAYLIGHT_SAVING_NONE;
rtc_init_parameter.rtc_year = 23;
rtc_init_parameter.rtc_month = RTC_JUL;
rtc_init_parameter.rtc_date = 5;
rtc_init_parameter.rtc_day_of_week = RTC_WEDSDAY;
rtc_init_parameter.rtc_clock_format = RTC_24H_FORMAT;
rtc_init_parameter.rtc_psc_factor_a = 127;
rtc_init_parameter.rtc_psc_factor_s = 255;
hal_rtc_init(&rtc_info,&rtc_init_parameter);

```

hal_rtc_alarm_output_config

The description of hal_rtc_alarm_output_config is shown as below:

Table 3-383. Function hal_rtc_alarm_output_config

Function name	hal_rtc_alarm_output_config
Function prototype	int32_t hal_rtc_alarm_output_config(hal_rtc_dev_struct *rtc_dev, hal_rtc_alarm_output_config_struct *rtc_alarm_output_config);
Function descriptions	configure RTC alarm output
Precondition	-

The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal_rtc_dev_struct
Input parameter{in}	
rtc_alarm_output_config	points to hal_rtc_alarm_output_config_struct, struct members refer to Table 3-379. hal_rtc_alarm_output_config_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* configure RTC alarm output */

hal_rtc_dev_struct rtc_info;

hal_rtc_alarm_output_config_struct rtc_alarm_output_config;

rtc_alarm_output_config.rtc_alarm_output_polarity = RTC_ALARM_HIGH;

rtc_alarm_output_config.rtc_alarm_output_type = RTC_ALARM_OUTPUT_OD;

hal_rtc_alarm_output_config (&rtc_info,& rtc_alarm_output_config);
```

hal_rtc_alarm_config

The description of hal_rtc_alarm_config is shown as below:

Table 3-384. Function hal_rtc_alarm_config

Function name	hal_rtc_alarm_config
Function prototype	int32_t hal_rtc_alarm_config(hal_rtc_dev_struct *rtc_dev, hal_rtc_alarm_config_struct *rtc_alarm_config);
Function descriptions	configure RTC alarm
Precondition	-
The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal_rtc_dev_struct
Input parameter{in}	
rtc_alarm_config	points to hal_rtc_alarm_config_struct, struct members refer to Table 3-380. hal_rtc_alarm_config_struct
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE, HAL_ERR_VAL, HAL_ERR_TIMEOUT
----------------	--

Example:

```

/* configure RTC alarm */

hal_rtc_dev_struct rtc_info;

rtc_alarm_config rtc_alarm_config_parameter;

rtc_alarm_config_parameter.rtc_alarm_hour = 16;

rtc_alarm_config_parameter.rtc_alarm_minute = 30;

rtc_alarm_config_parameter.rtc_alarm_second = 15;

rtc_alarm_config_parameter.rtc_alarm_subsecond = 0;

rtc_alarm_config_parameter.rtc_alarm_hour_mask                =
RTC_ALARM_HOUR_MASK_ENABLE;

rtc_alarm_config_parameter.rtc_alarm_minute_mask             =
RTC_ALARM_MINUTE_MASK_ENABLE;

rtc_alarm_config_parameter.rtc_alarm_second_mask             =
RTC_ALARM_SECOND_MASK_DISABLE;

rtc_alarm_config_parameter.rtc_alarm_subsecond_mask = RTC_MASKSSC_0_14;

rtc_alarm_config_parameter.rtc_alarm_weekday_mask            =
RTC_ALARM_WEEKDAY_MASK_ENABLE;

rtc_alarm_config_parameter.rtc_weekday_or_date = RTC_ALARM_SELECT_DATE;

rtc_alarm_config_parameter.rtc_alarm_day = 1;

hal_rtc_alarm_config(&rtc_info,&rtc_alarm_config_parameter);

```

hal_rtc_tamp_config

The description of hal_rtc_tamp_config is shown as below:

Table 3-385. Function hal_rtc_tamp_config

Function name	hal_rtc_tamp_config
Function prototype	int32_t hal_rtc_tamp_config(hal_rtc_dev_struct *rtc_dev, hal_rtc_tamper_config_struct *rtc_tamper_config);
Function descriptions	configure RTC tamper
Precondition	-
The called functions	-
Input parameter{in}	

rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal rtc dev struct
Input parameter{in}	
rtc_tamper_config	points to hal_rtc_tamper_config_struct, struct members refer to Table 3-381. hal rtc tamper config struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* configure RTC tamper */

hal_rtc_dev_struct rtc_info;

hal_rtc_tamper_config_struct rtc_tamper_config_parameter;

rtc_tamper_config_parameter.rtc_tamper_filter = RTC_FLT_EDGE;

rtc_tamper_config_parameter.rtc_tamper_sample_frequency = RTC_FREQ_DIV32768;

rtc_tamper_config_parameter.rtc_tamper_precharge_time = RTC_PRCH_1C;

rtc_tamper_config_parameter.rtc_tamper_precharge_enable = RTC_PRCH_ENALE;

rtc_tamper_config_parameter.rtc_tamper_with_timestamp =
RTC_TAMPER_TIMESTAMP_DISALE;

rtc_tamper_config_parameter.rtc_tamper0_trigger =
RTC_TAMPER_TRIGGER_EDGE_FALLING;

rtc_tamper_config_parameter.rtc_tamper0_source = ENABLE;

rtc_tamper_config_parameter.rtc_tamper1_source = DISABLE;

hal_rtc_tamp_config(&rtc_info,&rtc_tamper_config_parameter);
```

hal_rtc_timestamp_config

The description of hal_rtc_timestamp_config is shown as below:

Table 3-386. Function hal_rtc_timestamp_config

Function name	hal_rtc_timestamp_config
Function prototype	int32_t hal_rtc_timestamp_config(hal_rtc_dev_struct *rtc_dev, uint32_t rtc_timestamp_config);
Function descriptions	configure RTC time stamp
Precondition	-
The called functions	-

Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal_rtc_dev_struct
Input parameter{in}	
rtc_timestamp_config	RTC timestamp configuration
RTC_TIMESTAMP_RISING_EDGE	rising edge is valid event edge for time-stamp event
RTC_TIMESTAMP_FALLING_EDGE	falling edge is valid event edge for time-stamp event
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* configure RTC time stamp */
```

```
hal_rtc_dev_struct rtc_info;
```

```
hal_rtc_timestamp_config(&rtc_info, RTC_TIMESTAMP_RISING_EDGE);
```

hal_rtc_calib_config

The description of hal_rtc_calib_config is shown as below:

Table 3-387. Function hal_rtc_calib_config

Function name	hal_rtc_calib_config
Function prototype	int32_t hal_rtc_calib_config(hal_rtc_dev_struct *rtc_dev, uint8_t rtc_calib_config);
Function descriptions	configure RTC calibration
Precondition	-
The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal_rtc_dev_struct
Input parameter{in}	
rtc_calib_config	RTC calibration configuration
RTC_CALIBRATION_NONE	calibration output none
RTC_CALIBRATION_OUTPUT_512HZ	calibration output of 512Hz is enable
RTC_CALIBRATION_OUTPUT_1HZ	calibration output of 1Hz is enable
Output parameter{out}	

-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* configure RTC calibration */
```

```
hal_rtc_dev_struct rtc_info;
```

```
hal_rtc_calib_config(&rtc_info, RTC_CALIBRATION_OUTPUT_512HZ);
```

hal_rtc_refclock_detection_config

The description of hal_rtc_refclock_detection_config is shown as below:

Table 3-388. Function hal_rtc_refclock_detection_config

Function name	hal_rtc_refclock_detection_config
Function prototype	int32_t hal_rtc_refclock_detection_config(hal_rtc_dev_struct *rtc_dev, ControlStatus refclock_detection);
Function descriptions	configure RTC reference clock detection function
Precondition	-
The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal_rtc_dev_struct
Input parameter{in}	
refclock_detection	RTC reference clock detection
<i>DISABLE</i>	disable RTC reference clock detection function
<i>ENABLE</i>	enable RTC reference clock detection function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* configure RTC reference clock detection function */
```

```
hal_rtc_dev_struct rtc_info;
```

```
hal_rtc_refclock_detection_config(&rtc_info, ENABLE);
```

hal_rtc_struct_init

The description of hal_rtc_struct_init is shown as below:

Table 3-389. Function hal_rtc_struct_init

Function name	hal_rtc_struct_init
Function prototype	void hal_rtc_struct_init(hal_rtc_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the RTC structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	points to hal_rtc_struct_type_enum, struct members refer to Table 3-374. Enum hal rtc struct type enum
Input parameter{in}	
p_struct	points to RTC structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
void	-

Example:

```
/* initialize RTC init structure */
hal_rtc_dev_struct rtc_info;
hal_rtc_struct_init(HAL_RTC_DEV_STRUCT, &rtc_info);
```

hal_rtc_deinit

The description of hal_rtc_deinit is shown as below:

Table 3-390. Function hal_rtc_deinit

Function name	hal_rtc_deinit
Function prototype	int32_t hal_rtc_deinit(hal_rtc_dev_struct *rtc_dev);
Function descriptions	deinitialize RTC device structure and init structure
Precondition	-
The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal rtc dev struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* deinitialize RTC */
```

```
hal_rtc_dev_struct rtc_info;
```

```
hal_rtc_deinit(&rtc_info);
```

hal_rtc_interrupt_enable

The description of hal_rtc_interrupt_enable is shown as below:

Table 3-391. Function hal_rtc_interrupt_enable

Function name	hal_rtc_interrupt_enable
Function prototype	int32_t hal_rtc_interrupt_enable(hal_rtc_dev_struct *rtc_dev, hal_rtc_irq_struct *p_irq);
Function descriptions	enable RTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal_rtc_dev_struct
Input parameter{in}	
p_irq	points to hal_rtc_irq_struct, struct members refer to Table 3-375. Structure hal_rtc_irq_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* enable RTC interrupt */
hal_rtc_dev_struct rtc_info;
hal_rtc_irq_struct rtc_irq;
hal_rtc_interrupt_enable(&rtc_info, &rtc_irq);
```

hal_rtc_interrupt_disable

The description of hal_rtc_interrupt_disable is shown as below:

Table 3-392. Function hal_rtc_interrupt_disable

Function name	hal_rtc_interrupt_disable
Function prototype	int32_t hal_rtc_interrupt_disable(hal_rtc_dev_struct *rtc_dev);
Function descriptions	disable RTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	

rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal_rtc_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* disable RTC interrupt */
hal_rtc_dev_struct rtc_info;
hal_rtc_interrupt_disable(&rtc_info);
```

hal_rtc_irq

The description of hal_rtc_irq is shown as below:

Table 3-393. Function hal_rtc_irq

Function name	hal_rtc_irq
Function prototype	void hal_rtc_irq(hal_rtc_dev_struct *rtc_dev);
Function descriptions	RTC interrupt handler content function, which is merely used in rtc_handler
Precondition	-
The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal_rtc_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the function is used in the relative interrupt routine */
hal_rtc_dev_struct rtc_info;
void RTC_IRQHandler(void)
{
    hal_rtc_irq(&rtc_info);
}
```

hal_rtc_irq_handle_set

The description of hal_rtc_irq_handle_set is shown as below:

Table 3-394. Function hal_rtc_irq_handle_set

Function name	hal_rtc_irq_handle_set
Function prototype	void hal_rtc_irq_handle_set(hal_rtc_dev_struct *rtc_dev, hal_rtc_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal rtc dev struct
Input parameter{in}	
p_irq	points to hal_rtc_irq_struct, struct members refer to Table 3-375. Structure hal rtc irq struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set RTC interrupt callback function */
hal_rtc_dev_struct rtc_info;
hal_rtc_irq_struct rtc_irq;
hal_rtc_irq_handle_set(&rtc_info, &rtc_irq);
```

hal_rtc_irq_handle_all_reset

The description of hal_rtc_irq_handle_all_reset is shown as below:

Table 3-395. Function hal_rtc_irq_handle_all_reset

Function name	hal_rtc_irq_handle_all_reset
Function prototype	void hal_rtc_irq_handle_all_reset(hal_rtc_dev_struct *rtc_dev);
Function descriptions	reset all user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
rtc_dev	points to hal_rtc_dev_struct, struct members refer to Table 3-376. Structure hal rtc dev struct
Output parameter{out}	

-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* configure RTC reference clock detection function */
```

```
hal_rtc_dev_struct rtc_info;
```

```
hal_rtc_irq_handle_all_reset(&rtc_info);
```

hal_rtc_current_time_get

The description of hal_rtc_current_time_get is shown as below:

Table 3-396. Function hal_rtc_current_time_get

Function name	hal_rtc_current_time_get
Function prototype	void hal_rtc_current_time_get(hal_rtc_init_struct *rtc_calendar);
Function descriptions	get current time and date
Precondition	-
The called functions	-
Input parameter{in}	
rtc_calendar	points to hal_rtc_init_struct, struct members refer to Table 3-378. hal_rtc_init_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* get rtc current time */
```

```
hal_rtc_init_struct rtc_init;
```

```
hal_rtc_current_time_get(&rtc_init);
```

hal_rtc_alarm_get

The description of hal_rtc_alarm_get is shown as below:

Table 3-397. Function hal_rtc_alarm_get

Function name	hal_rtc_alarm_get
Function prototype	void hal_rtc_alarm_get(hal_rtc_alarm_config_struct *rtc_alarm_time);
Function descriptions	get RTC alarm time
Precondition	-
The called functions	-

Input parameter{in}	
rtc_alarm_time	points to hal_rtc_alarm_config_struct, struct members refer to Table 3-380. hal_rtc_alarm_config_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* get rtc alarm time */
hal_rtc_alarm_config_struct rtc_alarm_time;
hal_rtc_alarm_get(&rtc_alarm_time);
```

hal_rtc_alarm_event_poll

The description of hal_rtc_alarm_event_poll is shown as below:

Table 3-398. Function hal_rtc_alarm_event_poll

Function name	hal_rtc_alarm_event_poll
Function prototype	int32_t hal_rtc_alarm_event_poll(uint32_t timeout_ms);
Function descriptions	poll for RTC alarm event
Precondition	-
The called functions	-
Input parameter{in}	
timeout_ms	the time cost for event polling 0x0 – 0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
Int32_t	HAL_ERR_TIMEOUT, HAL_ERR_HARDWARE, HAL_ERR_NONE

Example:

```
/* poll for RTC alarm event with 100 ms timeout */
hal_rtc_alarm_event_poll (100);
```

hal_rtc_timestamp_get

The description of hal_rtc_timestamp_get is shown as below:

Table 3-399. Function hal_rtc_timestamp_get

Function name	hal_rtc_timestamp_get
Function prototype	void hal_rtc_timestamp_get(hal_rtc_timestamp_struct *rtc_timestamp);
Function descriptions	get RTC timestamp time and date

Precondition	-
The called functions	-
Input parameter{in}	
rtc_timestamp	points to hal_rtc_timestamp_struct, struct members refer to Table 3-377. hal_rtc_timestamp_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* get rtc timestamp time */
hal_rtc_timestamp_struct rtc_timestamp;
hal_rtc_timestamp_get(&rtc_alarm_time);
```

hal_rtc_timestamp_event_poll

The description of hal_rtc_timestamp_event_poll is shown as below:

Table 3-400. Function hal_rtc_timestamp_event_poll

Function name	hal_rtc_timestamp_event_poll
Function prototype	int32_t hal_rtc_timestamp_event_poll(uint32_t timeout_ms);
Function descriptions	poll for RTC timestamp event
Precondition	-
The called functions	-
Input parameter{in}	
timeout_ms	the time cost for event polling, 0x0 – 0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
Int32_t	HAL_ERR_TIMEOUT, HAL_ERR_HARDWARE, HAL_ERR_NONE

Example:

```
/* poll for RTC timestamp event with 100 ms timeout */
hal_rtc_timestamp_event_poll(100);
```

hal_rtc_tamper0_event_poll

The description of hal_rtc_tamper0_event_poll is shown as below:

Table 3-401. Function hal_rtc_tamper0_event_poll

Function name	hal_rtc_tamper0_event_poll
---------------	----------------------------

Function prototype	int32_t hal_rtc_tamper0_event_poll(uint32_t timeout_ms);
Function descriptions	poll for RTC tamper0 event
Precondition	-
The called functions	-
Input parameter{in}	
timeout_ms	the time cost for event polling, 0x0 – 0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
Int32_t	HAL_ERR_TIMEOUT, HAL_ERR_HARDWARE, HAL_ERR_NONE

Example:

```
/* poll for RTC tamper0 event with 100 ms timeout */
```

```
hal_rtc_tamper0_event_poll(100);
```

hal_rtc_tamper1_event_poll

The description of hal_rtc_tamper1_event_poll is shown as below:

Table 3-402. Function hal_rtc_tamper1_event_poll

Function name	hal_rtc_tamper1_event_poll
Function prototype	int32_t hal_rtc_tamper1_event_poll(uint32_t timeout_ms);
Function descriptions	poll for RTC tamper1 event
Precondition	-
The called functions	-
Input parameter{in}	
timeout_ms	the time cost for event polling, 0x0 – 0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
Int32_t	HAL_ERR_TIMEOUT, HAL_ERR_HARDWARE, HAL_ERR_NONE

Example:

```
/* poll for RTC tamper1 event with 100 ms timeout */
```

```
hal_rtc_tamper1_event_poll(100);
```

hal_rtc_daylight_saving_time_adjust

The description of hal_rtc_daylight_saving_time_adjust is shown as below:

Table 3-403. Function hal_rtc_daylight_saving_time_adjust

Function name	hal_rtc_daylight_saving_time_adjust
Function prototype	void hal_rtc_daylight_saving_time_adjust(uint32_t operation, uint32_t

	record);
Function descriptions	ajust the daylight saving time by adding or subtracting one hour from the current time
Precondition	-
The called functions	-
Input parameter{in}	
operation	hour ajustment operation
<i>RTC_DAYLIGHT_SAVI NG_ADD_1H</i>	add one hour
<i>RTC_DAYLIGHT_SAVI NG_SUB_1H</i>	substract one hour
<i>RTC_DAYLIGHT_SAVI NG_NONE</i>	no add or subtract one hour
Input parameter{in}	
record	daylight saving mark operation
<i>RTC_RECORD_DAYLI GHTSAVING_SET</i>	set daylight saving mark
<i>RTC_RECORD_DAYLI GHTSAVING_RESET</i>	reset daylight saving mark
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* ajust the daylight saving time by adding one hour from the current time */
hal_rtc_daylight_saving_time_adjust(RTC_DAYLIGHT_SAVING_ADD_1H,
RTC_RECORD_DAYLIGHTSAVING_SET);
```

3.20. SPI

The SPI module can communicate with external devices using the SPI protocol The SPI/I2S registers are listed in chapter [3.20.1](#), the SPI firmware functions are introduced in chapter [3.20.2](#).

3.20.1. Descriptions of Peripheral registers

SPI/I2S registers are listed in the table shown as below:

Table 3-404. SPI Registers

Registers	Descriptions
SPI_CTL0	SPI control register 0
SPI_CTL1	SPI control register 1
SPI_STAT	SPI status register
SPI_DATA	SPI data register
SPI_CRCPOLY	SPI CRC polynomial register
SPI_RCRC	SPI receive CRC register
SPI_TCRC	SPI transmit CRC register
SPI_I2SCTL	SPI/I2S control register
SPI_I2SPSC	SPI/I2S clock prescaler register
SPI_QCTL	Quad-SPI mode control register of SPI1

3.20.2. Descriptions of Peripheral functions

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

Table 3-405. DAC firmware function

Function name	Function description
hal_spi_deinit	deinitialize SPI
hal_spi_struct_init	initialize SPI structure
hal_spi_init	initialize SPI
hal_spi_transmit_poll	transmit amounts of data, poll transmit process and completed status, the function is blocking
hal_spi_receive_poll	receive amounts of data, poll receive process and completed status, the function is blocking
hal_spi_transmit_receive_poll	transmit and receive amounts of data, poll receive process and completed status, the function is blocking
hal_spi_transmit_interrupt	transmit amounts of data by interrupt method, the function is non-blocking
hal_spi_receive_interrupt	receive amounts of data by interrupt method, the function is non-blocking
hal_spi_transmit_receive_interrupt	transmit and receive amounts of data by interrupt method , the function is non-blocking
hal_spi_transmit_dma	transmit amounts of data by DMA method, the function is non-blocking
hal_spi_receive_dma	receive amounts of data by DMA method, the function is non-blocking
hal_spi_transmit_receive_dma	transmit and receive amounts of data by DMA method , the function is non-blocking
hal_spi_irq	SPI interrupt handler content function, which is merely used in spi_handler

Function name	Function description
hal_spi_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_spi_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_spi_start	start SPI module function
hal_spi_stop	stop SPI module function
hal_spi_abort	SPI abort function
hal_spi_dma_pause	SPI DMA pause function
hal_spi_dma_resume	SPI DMA resume function
hal_spi_dma_stop	SPI DMA stop function

Enum hal_spi_struct_type_enum

Table 3-406. Enum hal_spi_struct_type_enum

Member name	Function description
HAL_SPI_INIT_STRUCT	SPI initialization structure
HAL_SPI_DEV_STRUCT	SPI device information structure

Enum hal_spi_run_state_enum

Table 3-407. Enum hal_spi_run_state_enum

Member name	Function description
HAL_SPI_STATE_READY	peripheral Initialized and ready for use
HAL_SPI_STATE_BUSY	an internal process is ongoing
HAL_SPI_STATE_BUSY_TX	data transmission process is ongoing
HAL_SPI_STATE_BUSY_RX	data reception process is ongoing
HAL_SPI_STATE_BUSY_TX_RX	data Transmission and Reception process is ongoing
HAL_SPI_STATE_ERROR	SPI error state
HAL_SPI_STATE_ABORT	SPI abort is ongoing

Enum hal_spi_trans_mode_enum

Table 3-408. Enum hal_spi_trans_mode_enum

Member name	Function description
SPI_TRANSMODE_FULLDUPLEX	SPI receive and send data at full duplex communication
SPI_TRANSMODE_RECEIVEONLY	SPI only receive data
SPI_TRANSMODE_BDRECEIVE	bidirectional receive data
SPI_TRANSMODE_BDTRANSMIT	bidirectional transmit data
SPI_TRANSMODE_QUADRECEIVE	quad-wire receive data

Member name	Function description
VE	
SPI_TRANSMODE_QUADTRANS MIT	quad-wire transmit data

Enum hal_spi_prescaler_enum

Table 3-409. Enum hal_spi_prescaler_enum

Member name	Function description
SPI_PSC_2	SPI clock prescaler factor is 2
SPI_PSC_4	SPI clock prescaler factor is 4
SPI_PSC_8	SPI clock prescaler factor is 8
SPI_PSC_16	SPI clock prescaler factor is 16
SPI_PSC_32	SPI clock prescaler factor is 32
SPI_PSC_64	SPI clock prescaler factor is 64
SPI_PSC_128	SPI clock prescaler factor is 128
SPI_PSC_256	SPI clock prescaler factor is 256

Structure hal_spi_irq_struct

Table 3-410. hal_spi_irq_struct

Member name	Function description
receive_handler	SPI receive complete callback function
transmit_handler	SPI transmit complete callback function
transmit_receive_handler	SPI transmit and receive complete callback function
error_handle	SPI error complete callback function

Struct hal_spi_buffer_struct

Table 3-411. hal_spi_buffer_struct

Member name	Function description
buffer	pointer to SPI transfer buffer
length	SPI transfer length
pos	SPI transfer position

Structure hal_spi_dev_struct

Table 3-412. hal_spi_dev_struct

Member name	Function description
periph	SPI peripheral
spi_irq	SPI device interrupt callback function structure
p_dma_rx	DMA receive pointer

p_dma_tx	DMA transmit pointer
txbuffer	transmit buffer
rxbuffer	receive buffer
rx_callback	receive callback function pointer
tx_callback	transmit callback function pointer
tx_rx_callback	transmit and receive callback function pointer
state	SPI communication state
error_code	SPI error code
mutex	hal_mutex_enum

Structure hal_spi_user_callback_struct

Table 3-413. hal_spi_user_callback_struct

Member name	Function description
complete_func	SPI user complete callback function
error_func	SPI user error callback function

Structure hal_spi_init_struct

Table 3-414. hal_spi_init_struct

Member name	Function description
device_mode	SPI master or slave
trans_mode	SPI transtype
frame_size	SPI frame size
nss	SPI NSS control by hardware or software
endian	SPI big endian or little endian
clock_polarity_phase	SPI clock phase and polarity
prescaler	SPI prescaler value
crc_calculation	SPI CRC function selection
crc_poly	SPI CRC polynomial value
nssp_mode	SPI NSSP mode selection
ti_mode	SPI TI mode selection

hal_spi_deinit

The description of hal_spi_deinit is shown as below:

Table 3-415. Function hal_spi_deinit

Function name	hal_spi_deinit
Function prototype	void hal_spi_deinit (hal_spi_dev_struct *spi);
Function descriptions	deinitialize SPI

Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_spi_dev_struct spi0_info;

/* deinitialize the device information structure */

hal_spi_deinit(&spi0_info);
```

hal_spi_struct_init

The description of hal_spi_struct_init is shown as below:

Table 3-416. Function hal_spi_struct_init

Function name	hal_spi_struct_init
Function prototype	void hal_spi_struct_init(hal_spi_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the SPI structure
Precondition	-
Input parameter{in}	
hal_struct_type	refer to Enum hal_spi_struct_type_enum
Input parameter{in}	
p_struct	point to SPI structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_spi_dev_struct spi0_info;

/* initialize a spi device information structure */

hal_spi_struct_init(HAL_SPI_DEV_STRUCT, &spi0_info);
```

hal_spi_init

The description of hal_spi_init is shown as below:

Table 3-417. Function hal_spi_init

Function name	hal_spi_init
Function prototype	int32_t hal_spi_init(hal_spi_dev_struct *spi, uint32_t periph, hal_spi_init_struct *p_init);
Function descriptions	initialize SPI
Precondition	-
Input parameter{in}	
spi	SPI device information struct, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
periph	SPI peripheral
<i>SPIx</i>	x=0,1
Input parameter{in}	
p_init	pointer to SPI initial structure, the structure members can refer to Structure hal_spi_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code
HAL_ERR_NONE	No error, everything OK
HAL_ERR_ADDRESSES	Out of address range error
HAL_ERR_VAL	Illegal value

Example:

```

hal_spi_dev_struct spi0_info;
hal_spi_init_struct spi0_init_parameter;

/* initialize the structures */
hal_spi_struct_init(HAL_SPI_DEV_STRUCT, &spi0_info);
hal_spi_struct_init(HAL_SPI_INIT_STRUCT, &spi0_init_parameter);

/* initialize SPI0 */
spi0_init_parameter.trans_mode      = SPI_TRANSMODE_FULLDUPLEX;
spi0_init_parameter.device_mode    = SPI_MASTER;
spi0_init_parameter.frame_size     = SPI_FRAME_SIZE_8BIT;

```

```

spi0_init_parameter.clock_polarity_phase = SPI_CK_PL_HIGH_PH_2EDGE;

spi0_init_parameter.nss = SPI_NSS_SOFT;

spi0_init_parameter.prescale = SPI_PSC_8;

spi0_init_parameter.endian = SPI_ENDIAN_MSB;

spi0_init_parameter.crc_calculation = SPI_CRC_DISABLE;

spi0_init_parameter.ti_mode = SPI_TIMODE_DISABLE;

spi0_init_parameter.nssp_mode = SPI_NSSP_DISABLE;

hal_spi_init(&spi0_info, SPI0, &spi0_init_parameter);

```

hal_spi_transmit_poll

The description of hal_spi_transmit_poll is shown as below:

Table 3-418. Function hal_spi_transmit_poll

Function name	hal_spi_transmit_poll
Function prototype	int32_t hal_spi_transmit_poll(hal_spi_dev_struct *spi, uint8_t *p_txbuffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	transmit amounts of data, poll transmit process and completed status, the function is blocking
Precondition	-
Input parameter{in}	
spi	SPI device information struct, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
p_txbuffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	error code
HAL_ERR_NONE	No error, everything OK
HAL_ERR_VAL	Illegal value
HAL_ERR_BUSY	busy
HAL_ERR_TIMEOUT	Timeout

Example:

```
#define ARRAYNUM(arr_naname)      (uint32_t)(sizeof(arr_naname) / sizeof(*(arr_naname)))

#define TRANSMIT_SIZE              (ARRAYNUM(txbuffer))

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F};

/* transmit using polling mode */

hal_spi_transmit_poll(&spi0_info, txbuffer, TRANSMIT_SIZE, 0x1FFFF);
```

hal_spi_receive_poll

The description of hal_spi_receive_poll is shown as below:

Table 3-419. Function hal_spi_receive_poll

Function name	hal_spi_receive_poll
Function prototype	int32_t hal_spi_receive_poll(hal_spi_dev_struct *spi, uint8_t *p_rxbuffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	receive amounts of data, poll receive process and completed status, the function is blocking
Precondition	-
Input parameter{in}	
spi	SPI device information structue, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
p_rxbuffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	error code
HAL_ERR_NONE	No error, everything OK
HAL_ERR_HARDWARE	peripheral error flag set
HAL_ERR_VAL	Illegal value
HAL_ERR_BUSY	busy
HAL_ERR_TIMEOUT	timeout

Example:

```
#define TRANSFER_SIZE
```

16

```
uint8_t rxbuffer[TRANSFER_SIZE];

/* receive using polling mode */

hal_spi_receive_poll(&spi0_info, rxbuffer, TRANSFER_SIZE, 0x7FFFFFFF);
```

hal_spi_transmit_receive_poll

The description of hal_spi_transmit_receive_poll is shown as below:

Table 3-420. Function hal_spi_transmit_receive_poll

Function name	hal_spi_transmit_receive_poll
Function prototype	int32_t hal_spi_transmit_receive_poll(hal_spi_dev_struct *spi, uint8_t *p_txbuffer, uint8_t *p_rxbuffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	transmit and receive amounts of data, poll receive process and completed status, the function is blocking
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
p_txbuffer	pointer to txbuffer
Input parameter{in}	
p_rxbuffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	error code
HAL_ERR_NONE	No error, everything OK
HAL_ERR_VAL	Illegal value
HAL_ERR_BUSY	busy
HAL_ERR_TIMEOUT	timeout
HAL_ERR_HARDWARE	peripheral error flag set

Example:

```
#define ARRAYNUM(arr_name) ((uint32_t)(sizeof(arr_name) / sizeof(*(arr_name))))

#define TRANSMIT_SIZE (ARRAYNUM(txbuffer))
```

```
uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F};
```

```
uint8_t rxbuffer[TRANSFER_SIZE];
```

```
/* transmit and receive using polling mode */
```

```
hal_spi_transmit_receive_poll(&spi0_info, txbuffer, rxbuffer, TRANSFER_SIZE, 6000);
```

hal_spi_transmit_interrupt

The description of hal_spi_transmit_interrupt is shown as below:

Table 3-421. Function hal_spi_transmit_interrupt

Function name	hal_spi_transmit_interrupt
Function prototype	int32_t hal_spi_transmit_interrupt(hal_spi_dev_struct *spi, uint8_t *p_txbuffer, uint32_t length, hal_spi_user_callback_struct *p_user_func);
Function descriptions	transmit amounts of data by interrupt method, the function is non-blocking
Precondition	-
Input parameter{in}	
spi	SPI device information struct, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
p_txbuffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	error code
HAL_ERR_NONE	No error, everything OK
HAL_ERR_VAL	Illegal value
HAL_ERR_BUSY	busy

Example:

```
hal_spi_user_callback_struct user_call;
```

```
#define ARRAYNUM(arr_name) ((uint32_t)(sizeof(arr_name) / sizeof(*(arr_name))))
```

```
#define TRANSMIT_SIZE (ARRAYNUM(txbuffer))
```

```
uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F};
```

```
void spi_callback(hal_spi_dev_struct *spi)
{
    printf("\n\r SPI interrupt transmit success ");
}

/* transmit using interrupt mode */

user_call.complete_func = spi_callback;

hal_spi_transmit_interrupt(&spi0_info, txbuffer, TRANSMIT_SIZE, &user_call);
```

hal_spi_receive_interrupt

The description of hal_spi_receive_interrupt is shown as below:

Table 3-422. Function hal_spi_receive_interrupt

Function name	hal_spi_receive_interrupt
Function prototype	int32_t hal_spi_receive_interrupt(hal_spi_dev_struct *spi, uint8_t *p_rxbuffer, uint32_t length, hal_spi_user_callback_struct *p_user_func);
Function descriptions	receive amounts of data by interrupt method, the function is non-blocking
Precondition	-
Input parameter{in}	
spi	SPI device information struct, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
p_rxbuffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	error code
HAL_ERR_NONE	No error, everything OK
HAL_ERR_VAL	Illegal value
HAL_ERR_BUSY	busy

Example:

```
hal_spi_user_callback_struct user_call;
```

```
#define TRANSFER_SIZE
```

16


```
uint8_t rxbuffer[TRANSFER_SIZE];

void spi_callback(hal_spi_dev_struct *spi)
{
    printf("\n\r SPI interrupt receive success ");
}

/* receive using interrupt mode */

user_call.complete_func = spi_callback;

hal_spi_receive_interrupt(&spi0_info, rxbuffer, TRANSMIT_SIZE, &user_call);
```

hal_spi_transmit_receive_interrupt

The description of hal_spi_transmit_receive_interrupt is shown as below:

Table 3-423. Function hal_spi_transmit_receive_interrupt

Function name	hal_spi_transmit_receive_interrupt
Function prototype	int32_t hal_spi_transmit_receive_interrupt(hal_spi_dev_struct *spi, uint8_t *p_txbuffer, uint8_t *p_rxbuffer, uint32_t length, hal_spi_user_callback_struct *p_user_func);
Function descriptions	transmit and receive amounts of data by interrupt method, the function is non-blocking
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
p_txbuffer	pointer to txbuffer
Input parameter{in}	
p_rxbuffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to callback function for user
Output parameter{out}	
-	-
Return value	
int32_t	error code
HAL_ERR_NONE	No error, everything OK
HAL_ERR_VAL	Illegal value
HAL_ERR_BUSY	busy

Example:

```
hal_spi_user_callback_struct user_call;

#define ARRAYNUM(arr_name)      (uint32_t)(sizeof(arr_name) / sizeof(*(arr_name)))

#define TRANSMIT_SIZE          (ARRAYNUM(txbuffer))

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F};

uint8_t rxbuffer[TRANSFER_SIZE];

void spi_callback(hal_spi_dev_struct *spi)
{
    printf("\n\r  SPI interrupt transmit and receive success ");
}

/* transmit and receive using interrupt mode */

user_call.complete_func = spi_callback;

hal_spi_transmit_receive_interrupt(&spi0_info,  txbuffer,  rxbuffer  ,TRANSMIT_SIZE,
&user_call);
```

hal_spi_transmit_dma

The description of hal_spi_transmit_dma is shown as below:

Table 3-424. Function hal_spi_transmit_dma

Function name	hal_spi_transmit_dma
Function prototype	int32_t hal_spi_transmit_dma(hal_spi_dev_struct *spi, uint8_t *p_txbuffer, \ uint32_t length, hal_spi_user_callback_struct *p_user_func);
Function descriptions	transmit amounts of data by DMA method, the function is non-blocking
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
p_txbuffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to call back function for user
Output parameter{out}	

-	-
Return value	
int32_t	error code
<i>HAL_ERR_NONE</i>	No error, everything OK
<i>HAL_ERR_VAL</i>	Illegal value
<i>HAL_ERR_BUSY</i>	busy

Example:

```

hal_spi_user_callback_struct user_call;

#define ARRAYNUM(arr_name)      (uint32_t)(sizeof(arr_name) / sizeof(*(arr_name)))

#define TRANSMIT_SIZE          (ARRAYNUM(txbuffer))

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F};

void spi_callback(hal_spi_dev_struct *spi)
{
    printf("\n\r  SPI DMA transmit success ");
}

/* transmit using DMA mode */

user_call.complete_func = spi_callback;

hal_spi_transmit_dma(&spi0_info, txbuffer, TRANSMIT_SIZE, &user_call);

```

hal_spi_receive_dma

The description of hal_spi_receive_dma is shown as below:

Table 3-425. Function hal_spi_receive_dma

Function name	hal_spi_receive_dma
Function prototype	int32_t hal_spi_receive_dma(hal_spi_dev_struct *spi, uint8_t *p_rxbuffer, uint32_t length, hal_spi_user_callback_struct *p_user_func);
Function descriptions	receive amounts of data by DMA method, the function is non-blocking
Precondition	-
Input parameter{in}	
spi	SPI device information struct, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
p_rxbuffer	pointer to rxbuffer
Input parameter{in}	

length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	error code
<i>HAL_ERR_NONE</i>	No error, everything OK
<i>HAL_ERR_VAL</i>	Illegal value
<i>HAL_ERR_BUSY</i>	busy

Example:

```
hal_spi_user_callback_struct user_call;

#define TRANSFER_SIZE                16

uint8_t rxbuffer[TRANSFER_SIZE];

void spi_callback(hal_spi_dev_struct *spi)
{
    printf("\n\r  SPI DMA receive success ");
}

/* receive using DMA mode */

user_call.complete_func = spi_callback;

hal_spi_receive_dma(&spi0_info, rxbuffer, TRANSFER_SIZE, &user_call);
```

hal_spi_transmit_receive_dma

The description of hal_spi_transmit_receive_dma is shown as below:

Table 3-426. Function hal_spi_transmit_receive_dma

Function name	hal_spi_transmit_receive_dma
Function prototype	int32_t hal_spi_transmit_receive_dma(hal_spi_dev_struct *spi, uint8_t *p_txbuffer, uint8_t *p_rxbuffer, uint32_t length, hal_spi_user_callback_struct *p_user_func);
Function descriptions	transmit and receive amounts of data by DMA method , the function is non-blocking
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct

Input parameter{in}	
p_txbuffer	pointer to txbuffer
Input parameter{in}	
p_rxbuffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	error code
<i>HAL_ERR_NONE</i>	No error, everything OK
<i>HAL_ERR_VAL</i>	Illegal value
<i>HAL_ERR_BUSY</i>	busy

Example:

```

hal_spi_user_callback_struct user_call;

#define ARRAYNUM(arr_name)      (uint32_t)(sizeof(arr_name) / sizeof(*(arr_name)))

#define TRANSMIT_SIZE          (ARRAYNUM(txbuffer))

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F};

uint8_t rxbuffer[TRANSFER_SIZE];

void spi_callback(hal_spi_dev_struct *spi)
{
    printf("\n\r  SPI DMA transmit and receive success ");
}

/* transmit and receive using DMA mode */

user_call.complete_func = spi_callback;

hal_spi_transmit_receive_dma(&spi0_info, txbuffer, rxbuffer, TRANSMIT_SIZE, &user_call);

```

hal_spi_irq

The description of hal_spi_irq is shown as below:

Table 3-427. Function hal_spi_irq

Function name	hal_spi_irq
----------------------	-------------

Function prototype	void hal_spi_irq(hal_spi_dev_struct * spi);
Function descriptions	SPI interrupt handler content function, which is merely used in spi_handler
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* process the SPI interrupt */
```

```
hal_spi_irq(&spi0_info);
```

hal_spi_irq_handle_set

The description of hal_spi_irq_handle_set is shown as below:

Table 3-428. Function hal_spi_irq_handle_set

Function name	hal_spi_irq_handle_set
Function prototype	void hal_spi_irq_handle_set(hal_spi_dev_struct *spi, hal_spi_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Input parameter{in}	
p_irq	point to the structure that contains callback handlers of SPI interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_spi_irq_struct spi_irq;
```

```
hal_spi_struct_init(HAL_SPI_IRQ_INIT_STRUCT, &spi_irq);
```

```
/* set the spi transmit interrupt handle function */
```

```
spi_irq.transmit_handler = _spi_transmit_interrupt;
```

```
hal_spi_irq_handle_set(&spi0_info, &spi_irq);
```

hal_spi_irq_handle_all_reset

The description of hal_spi_irq_handle_all_reset is shown as below:

Table 3-429. Function hal_spi_irq_handle_all_reset

Function name	hal_spi_irq_handle_all_reset
Function prototype	void hal_spi_irq_handle_all_reset(hal_spi_dev_struct *spi);
Function descriptions	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback */
```

```
hal_spi_irq_handle_all_reset(&spi0_info);
```

hal_spi_start

The description of hal_spi_start is shown as below:

Table 3-430. Function hal_spi_start

Function name	hal_spi_start
Function prototype	void hal_spi_start(hal_spi_dev_struct *spi);
Function descriptions	start SPI module function
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start SPI module */
```

```
hal_spi_start(&spi0_info);
```

hal_spi_stop

The description of hal_spi_stop is shown as below:

Table 3-431. Function hal_spi_stop

Function name	hal_spi_stop
Function prototype	void hal_spi_stop(hal_spi_dev_struct *spi);
Function descriptions	stop SPI module function
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* stop SPI module */
```

```
hal_spi_stop(&spi0_info);
```

hal_spi_abort

The description of hal_spi_abort is shown as below:

Table 3-432. Function hal_spi_abort

Function name	hal_spi_abort
Function prototype	int32_t hal_spi_abort(hal_spi_dev_struct *spi);
Function descriptions	SPI abort function
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code
HAL_ERR_NONE	No error, everything OK

<i>HAL_ERR_ADDRESSES</i>	Out of address range error
--------------------------	----------------------------

Example:

```
/* abort SPI module */
hal_spi_abort(&spi0_info);
```

hal_spi_dma_pause

The description of hal_spi_dma_pause is shown as below:

Table 3-433. Function hal_spi_dma_pause

Function name	hal_spi_dma_pause
Function prototype	void hal_spi_dma_pause(hal_spi_dev_struct *spi);
Function descriptions	SPI DMA pause function
Precondition	-
Input parameter{in}	
spi	SPI device information struct, the structure members can refer to Structure hal_spi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* pause SPI DMA transfer */
hal_spi_dma_pause(&spi0_info);
```

hal_spi_dma_resume

The description of hal_spi_dma_resume is shown as below:

Table 3-434. Function hal_spi_dma_resume

Function name	hal_spi_dma_resume
Function prototype	void hal_spi_dma_resume(hal_spi_dev_struct *spi);
Function descriptions	SPI DMA resume function
Precondition	-
Input parameter{in}	
spi	SPI device information struct, the structure members can refer to Structure hal_spi_dev_struct
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* resume SPI DMA transfer */
hal_spi_dma_resume(&spi0_info);
```

hal_spi_dma_stop

The description of hal_spi_dma_stop is shown as below:

Table 3-435. Function hal_spi_dma_stop

Function name	hal_spi_dma_stop
Function prototype	void hal_spi_dma_stop(hal_spi_dev_struct *spi);
Function descriptions	SPI DMA stop function
Precondition	-
Input parameter{in}	
spi	SPI device information structtrue, the structure members can refer to Structure hal_spi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* stop SPI DMA transfer */
hal_spi_dma_stop(&spi0_info);
```

3.21. I2S

The I2S module can communicate with external devices using the I2S protocol The SPI/I2S registers are listed in chapter [3.21.1](#), the I2S firmware functions are introduced in chapter [3.21.2](#).

3.21.1. Descriptions of Peripheral registers

SPI/I2S registers are listed in the table shown as below:

Table 3-436. SPI/I2S Registers

Registers	Descriptions
SPI_CTL0	SPI control register 0
SPI_CTL1	SPI control register 1
SPI_STAT	SPI status register
SPI_DATA	SPI data register
SPI_CRCPOLY	SPI CRC polynomial register
SPI_RCRC	SPI receive CRC register
SPI_TCRC	SPI transmit CRC register
SPI_I2SCTL	SPI/I2S control register
SPI_I2SPSC	SPI/I2S clock prescaler register
SPI_QCTL	Quad-SPI mode control register of SPI1

3.21.2. Descriptions of Peripheral functions

I2S HAL firmware functions are listed in the table shown as below:

Table 3-437. I2S firmware function

Function name	Function description
hal_i2s_deinit	deinitialize I2S
hal_i2s_struct_init	initialize I2S structure
hal_i2s_init	initialize I2S
hal_i2s_transmit_poll	transmit amounts of data, poll transmit process, the function is blocking
hal_i2s_receive_poll	receive amounts of data, poll receive process, the function is blocking
hal_i2s_transmit_interrupt	transmit amounts of data by interrupt method ,the function is non-blocking
hal_i2s_receive_interrupt	receive amounts of data by interrupt method ,the function is non-blocking
hal_i2s_transmit_dma	transmit amounts of data by DMA method ,the function is non-blocking
hal_i2s_receive_dma	receive amounts of data by DMA method ,the function is non-blocking
hal_i2s_irq	I2S interrupt handler content function
hal_i2s_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_i2s_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_i2s_start	start I2S module function
hal_i2s_stop	stop I2S module function
hal_i2s_dma_pause	I2S DMA pause function

Function name	Function description
hal_i2s_dma_resume	I2S DMA resume function
hal_i2s_dma_stop	I2S DMA stop function

Enum hal_i2s_struct_type_enum

Table 3-438. Enum hal_i2s_struct_type_enum

Member name	Function description
HAL_I2S_INIT_STRUCT	I2S initialization structure
HAL_I2S_DEV_STRUCT	I2S device information structure

Enum hal_i2s_run_state_enum

Table 3-439. Enum hal_i2s_run_state_enum

Member name	Function description
HAL_I2S_STATE_RESET	I2S not yet initialized or disabled
HAL_I2S_STATE_READY	I2S initialized and ready for use
HAL_I2S_STATE_BUSY	I2S internal process is ongoing
HAL_I2S_STATE_BUSY_TX	data transmission process is ongoing
HAL_I2S_STATE_BUSY_RX	data reception process is ongoing
HAL_I2S_STATE_TIMEOUT	I2S timeout state
HAL_I2S_STATE_ERROR	I2S error state

Enum hal_i2s_standard_enum

Table 3-440. Enum hal_i2s_standard_enum

Member name	Function description
I2S_STD_PHILLIPS	I2S phillips standard
I2S_STD_MSB	I2S MSB standard
I2S_STD_LSB	I2S LSB standard
I2S_STD_PCMSHORT	I2S PCM short standard
I2S_STD_PCMLONG	I2S PCM long standard

Enum hal_i2s_audiosample_enum

Table 3-441. Enum hal_i2s_audiosample_enum

Member name	Function description
I2S_AUDIOSAMPLE_8K	I2S audio sample rate is 8KHz
I2S_AUDIOSAMPLE_11K	I2S audio sample rate is 11KHz
I2S_AUDIOSAMPLE_16K	I2S audio sample rate is 16KHz
I2S_AUDIOSAMPLE_22K	I2S audio sample rate is 22KHz
I2S_AUDIOSAMPLE_32K	I2S audio sample rate is 32KHz
I2S_AUDIOSAMPLE_44K	I2S audio sample rate is 44KHz

Member name	Function description
I2S_AUDIOSAMPLE_48K	I2S audio sample rate is 48KHz
I2S_AUDIOSAMPLE_96K	I2S audio sample rate is 96KHz
I2S_AUDIOSAMPLE_192K	I2S audio sample rate is 192KHz

Structure hal_i2s_buffer_struct

Table 3-442. hal_i2s_buffer_struct

Member name	Function description
buffer	pointer to I2S transfer buffer
length	I2S transfer length
pos	I2S transfer position

Structure hal_i2s_irq_struct

Table 3-443. hal_i2s_irq_struct

Member name	Function description
receive_handler	I2S receive complete callback function
transmit_handler	I2S transmit complete callback function
error_handle	I2S error complete callback function

Structure hal_i2s_dev_struct

Table 3-444. hal_i2s_dev_struct

Member name	Function description
periph	I2S peripheral
i2s_irq	I2S device interrupt callback function pointer
p_dma_rx	DMA receive pointer
p_dma_tx	DMA transmit pointer
txbuffer	transmit buffer
rxbuffer	receive buffer
rx_callback	receive callback function pointer
tx_callback	transmit callback function pointer
error_callback	error callback function pointer
state	I2S communication state
error_code	I2S error code
mutex	hal_mutex_enum

Structure hal_i2s_user_callback_struct

Table 3-445. hal_i2s_user_callback_struct

Member name	Function description
complete_func	I2S user complete callback function

error_func	I2S user error callback function
------------	----------------------------------

Structure hal_i2s_init_struct

Table 3-446. hal_i2s_init_struct

Member name	Function description
mode	I2S operation mode
standard	I2S standard
frameformat	I2S frame format
mckout	I2S master clock output
audiosample	I2S audio sample rate
ckpl	I2S idle state clock polarity

hal_i2s_deinit

The description of hal_i2s_deinit is shown as below:

Table 3-447. Function hal_i2s_deinit

Function name	hal_i2s_deinit
Function prototype	void hal_i2s_deinit (hal_i2s_dev_struct *i2s);
Function descriptions	deinitialize I2S
Precondition	-
Input parameter{in}	
I2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_i2s_dev_struct i2s0_info;

/* deinitialize the device information structure */

hal_i2s_deinit(&i2s0_info);
```

hal_i2s_struct_init

The description of hal_i2s_struct_init is shown as below:

Table 3-448. Function hal_i2s_struct_init

Function name	hal_i2s_struct_init
Function prototype	void hal_i2s_struct_init(hal_i2s_struct_type_enum hal_struct_type, void

	*p_struct);
Function descriptions	initialize the I2S structure
Precondition	-
Input parameter{in}	
hal_struct_type	结构体类型
<i>HAL_I2S_INIT_STR</i> <i>UCT</i>	初始化结构体
<i>HAL_I2S_DEV_STR</i> <i>UCT</i>	设备信息结构体
Input parameter{in}	
p_struct	指向结构体的空指针
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_i2s_dev_struct i2s0_info;
```

```
/* initialize a i2s device information structure */
```

```
hal_i2s_struct_init(HAL_I2S_DEV_STRUCT, &i2s0_info);
```

hal_i2s_init

The description of hal_i2s_init is shown as below:

Table 3-449. Function hal_i2s_init

Function name	hal_i2s_init
Function prototype	int32_t hal_i2s_init(hal_i2s_dev_struct *i2s, uint32_t periph, hal_i2s_init_struct *p_init);
Function descriptions	initialize I2S
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Input parameter{in}	
periph	I2S peripheral
<i>SPIx</i>	x=0
Input parameter{in}	
p_init	pointer to I2S initial structure,the structure members can refer to Structure hal_i2s_init_struct

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```

hal_i2s_dev_struct i2s0_info;

hal_i2s_init_struct i2s0_init_parameter;

/* initialize the structures */

hal_i2s_struct_init(HAL_I2S_INIT_STRUCT, &i2s0_init_parameter);

hal_i2s_struct_init(HAL_I2S_DEV_STRUCT, &i2s0_info);

/* initialize SPI0 */

i2s0_init_parameter.mode = I2S_MODE_MASTERTX;

i2s0_init_parameter.standard = I2S_STD_PHILLIPS;

i2s0_init_parameter.ckpl = I2S_CKPL_LOW;

i2s0_init_parameter.frameformat = I2S_FRAMEFORMAT_DT16B_CH16B;

i2s0_init_parameter.mckout = I2S_MCKOUT_DISABLE;

i2s0_init_parameter.audiosample = I2S_AUDIOSAMPLE_8K;

hal_i2s_init(&i2s0_info, SPI0, &i2s0_init_parameter);

```

hal_i2s_transmit_poll

The description of hal_i2s_transmit_poll is shown as below:

Table 3-450. Function hal_i2s_transmit_poll

Function name	hal_i2s_transmit_poll
Function prototype	int32_t hal_i2s_transmit_poll(hal_i2s_dev_struct *i2s, uint16_t *p_buffer, uint16_t length, uint32_t timeout_ms);
Function descriptions	transmit amounts of data, poll transmit process and completed status, the function is blocking
Precondition	-
Input parameter{in}	
I2s	pointer to I2S device information structure, the structure members can refer to Structure hal_i2s_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	

length	length of data to be sent
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```
#define ARRAYNUM(arr_name)      (uint32_t)(sizeof(arr_name) / sizeof(*(arr_name)))

#define TRANSMIT_SIZE          (ARRAYNUM(txbuffer))

uint16_t txbuffer[] = {0x0001, 0x0203, 0x0405, 0x0607, 0x0809, 0x0A0B, 0x0C0D, 0x0E0F};

/* transmit using polling mode */

hal_i2s_transmit_poll(&i2s0_info, txbuffer, TRANSMIT_SIZE, 0x1FFFF);
```

hal_i2s_receive_poll

The description of hal_i2s_receive_poll is shown as below:

Table 3-451. Function hal_i2s_receive_poll

Function name	hal_i2s_receive_poll
Function prototype	int32_t hal_i2s_receive_poll(hal_i2s_dev_struct *i2s, uint16_t *p_buffer, uint16_t length, uint32_t timeout_ms);
Function descriptions	receive amounts of data, poll receive process and completed status, the function is blocking
Precondition	-
Input parameter{in}	
I2s	pointer to I2S device information structure, the structure members can refer to Structure hal_i2s_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```
#define TRANSFER_SIZE                16

uint16_t rxbuffer[TRANSFER_SIZE];

/* receive using polling mode */

hal_i2s_receive_poll(&i2s0_info, rxbuffer, TRANSFER_SIZE, 0x7FFFFFFF);
```

hal_i2s_transmit_interrupt

The description of hal_i2s_transmit_interrupt is shown as below:

Table 3-452. Function hal_i2s_transmit_interrupt

Function name	hal_i2s_transmit_interrupt
Function prototype	int32_t hal_i2s_transmit_interrupt(hal_i2s_dev_struct *i2s, uint16_t *p_buffer, uint16_t length, hal_i2s_user_callback_struct *p_user_func);
Function descriptions	transmit amounts of data by interrupt method, the function is non-blocking
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```
hal_i2s_user_callback_struct user_call;

#define ARRAYNUM(arr_name)      (uint32_t)(sizeof(arr_name) / sizeof(*(arr_name)))

#define TRANSMIT_SIZE          (ARRAYNUM(txbuffer))

uint16_t txbuffer[] = {0x0001, 0x0203, 0x0405, 0x0607, 0x0809, 0x0A0B, 0x0C0D, 0x0E0F};

void i2s_callback(hal_i2s_dev_struct *i2s)
{
    printf("\n\r I2S interrupt transmit success ");
```

```

}

/* transmit using interrupt mode */

user_call.complete_func = i2s_callback;

hal_i2s_transmit_interrupt(&i2s0_info, txbuffer, TRANSMIT_SIZE,&user_call);

```

hal_i2s_receive_interrupt

The description of hal_i2s_receive_interrupt is shown as below:

Table 3-453. Function hal_i2s_receive_interrupt

Function name	hal_i2s_receive_interrupt
Function prototype	int32_t hal_i2s_receive_interrupt(hal_i2s_dev_struct *i2s, uint16_t *p_buffer, uint16_t length, hal_i2s_user_callback_struct *p_user_func);
Function descriptions	receive amounts of data by interrupt method ,the function is non-blocking
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```

hal_i2s_user_callback_struct user_call;

#define TRANSFER_SIZE          16

void i2s_callback(hal_i2s_dev_struct *i2s)
{
    printf("\n\r  I2S interrupt receive success ");
}

```

```
uint16_t rxbuffer[TRANSFER_SIZE];

/* receive using interrupt mode */

user_call.complete_func = i2s_callback;

hal_i2s_receive_interrupt(&i2s0_info, rxbuffer, TRANSMIT_SIZE,&user_call);
```

hal_i2s_transmit_dma

The description of hal_i2s_transmit_dma is shown as below:

Table 3-454. Function hal_i2s_transmit_dma

Function name	hal_i2s_transmit_dma
Function prototype	int32_t hal_i2s_transmit_dma(hal_i2s_dev_struct *i2s, uint16_t *p_buffer, uint16_t length, hal_i2s_user_callback_struct *p_user_func);
Function descriptions	transmit amounts of data by DMA method ,the function is non-blocking
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Input parameter{in}	
p_buffer	pointer to txbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```
hal_i2s_user_callback_struct user_call;

#define ARRAYNUM(arr_name)      (uint32_t)(sizeof(arr_name) / sizeof(*(arr_name)))

#define TRANSMIT_SIZE          (ARRAYNUM(txbuffer))

uint16_t txbuffer[] = {0x0001, 0x0203, 0x0405, 0x0607, 0x0809, 0x0A0B, 0x0C0D, 0x0E0F};

void i2s_callback(hal_i2s_dev_struct *i2s)
{
    printf("\n\r I2S interrupt DMA transmit success ");
```

```

}

/* transmit using DMA mode */

user_call.complete_func = i2s_callback;

hal_i2s_transmit_dma(&i2s0_info, txbuffer, TRANSMIT_SIZE, &user_call);

```

hal_i2s_receive_dma

The description of hal_i2s_receive_dma is shown as below:

Table 3-455. Function hal_i2s_receive_dma

Function name	hal_i2s_receive_dma
Function prototype	int32_t hal_i2s_receive_dma(hal_i2s_dev_struct *i2s, uint16_t *p_buffer, uint16_t length, hal_i2s_user_callback_struct *p_user_func);
Function descriptions	receive amounts of data by DMA method, the function is non-blocking
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure, the structure members can refer to Structure hal_i2s_dev_struct
Input parameter{in}	
p_buffer	pointer to rxbuffer
Input parameter{in}	
length	length of data to be sent
Input parameter{in}	
p_user_func	pointer to call back function for user
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_VAL

Example:

```

hal_i2s_user_callback_struct user_call;

#define TRANSFER_SIZE          16

void i2s_callback(hal_i2s_dev_struct *i2s)
{
    printf("\n\r  I2S DMA receive success ");
}

uint16_t rxbuffer[TRANSFER_SIZE];

```

```
/* receive using DMA mode */
```

```
user_call.complete_func = i2s_callback;
```

```
hal_i2s_receive_dma(&i2s0_info, rxbuffer, TRANSMIT_SIZE,&user_call);
```

hal_i2s_irq

The description of hal_i2s_irq is shown as below:

Table 3-456. Function hal_i2s_irq

Function name	hal_i2s_irq
Function prototype	void hal_i2s_irq(hal_i2s_dev_struct *i2s);
Function descriptions	handler I2S interrupt request
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* process the I2S interrupt */
```

```
hal_i2s_irq(&i2s0_info);
```

hal_i2s_irq_handle_set

The description of hal_i2s_irq_handle_set is shown as below:

Table 3-457. Function hal_i2s_irq_handle_set

Function name	hal_i2s_irq_handle_set
Function prototype	void hal_i2s_irq_handle_set(hal_i2s_dev_struct *i2s, hal_i2s_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Input parameter{in}	
p_irq	point to the structure that contains callback handlers of I2S interrupt
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
hal_i2s_irq_struct i2s_irq;

hal_i2s_struct_init(HAL_I2S_IRQ_INIT_STRUCT, &i2s_irq);

/* set the i2s transmit interrupt handle function */

i2s_irq.transmit_handler = _i2s_transmit_interrupt;

hal_i2s_irq_handle_set(&i2s0_info, &i2s_irq);
```

hal_i2s_irq_handle_all_reset

The description of hal_i2s_irq_handle_all_reset is shown as below:

Table 3-458. Function hal_i2s_irq_handle_all_reset

Function name	hal_i2s_irq_handle_all_reset
Function prototype	void hal_i2s_irq_handle_all_reset(hal_i2s_dev_struct *i2s);
Function descriptions	reset all user-defined interrupt callback function
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback */

hal_i2s_irq_handle_all_reset(&i2s0_info);
```

hal_i2s_start

The description of hal_i2s_start is shown as below:

Table 3-459. Function hal_i2s_start

Function name	hal_i2s_start
Function prototype	void hal_i2s_start(hal_i2s_dev_struct *i2s);
Function	start I2S module

descriptions	
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start I2S module */
hal_i2s_start(&i2s0_info);
```

hal_i2s_stop

The description of hal_i2s_stop is shown as below:

Table 3-460. Function hal_i2s_stop

Function name	hal_i2s_stop
Function prototype	void hal_i2s_stop(hal_i2s_dev_struct *i2s);
Function descriptions	stop I2S module
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* stop I2S module */
hal_i2s_stop(&i2s0_info);
```

hal_i2s_dma_pause

The description of hal_i2s_dma_pause is shown as below:

Table 3-461. Function hal_i2s_dma_pause

Function name	hal_i2s_dma_pause
Function prototype	void hal_i2s_dma_pause(hal_i2s_dev_struct *i2s);

Function descriptions	pause I2S DMA
Precondition	-
Input parameter{in}	
I2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* pause I2S DMA transfer */
hal_i2s_dma_pause(&i2s0_info);
```

hal_i2s_dma_resume

The description of hal_i2s_dma_resume is shown as below:

Table 3-462. Function hal_i2s_dma_resume

Function name	hal_i2s_dma_resume
Function prototype	void hal_i2s_dma_resume(hal_i2s_dev_struct *i2s);
Function descriptions	resume I2S DMA
Precondition	-
Input parameter{in}	
I2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* resume I2S DMA transfer */
hal_i2s_dma_resume(&i2s0_info);
```

hal_i2s_dma_stop

The description of hal_i2s_dma_stop is shown as below:

Table 3-463. Function hal_i2s_dma_stop

Function name	hal_i2s_dma_stop
----------------------	------------------

Function prototype	void hal_i2s_dma_stop(hal_i2s_dev_struct *i2s);
Function descriptions	stop I2S DMA and disable DMA channel
Precondition	-
Input parameter{in}	
i2s	pointer to I2S device information structure,the structure members can refer to Structure hal_i2s_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* stop I2S DMA transfer */
hal_i2s_dma_stop(&i2s0_info);
```

3.22. TIMER

The timers have a 16-bit counter that can be used as an unsigned counter and supports both input capture and output compare. Timers (TIMERx) are divided into five sorts: advanced timer (TIMER0), general level0 timer (TIMERx, x=1, 2), general level2 timer (TIMER13), general level2 timer (TIMERx, x=15, 16), basic timer (TIMER5). The specific functions of different types of timer are different. The TIMER registers are listed in chapter [3.22.1](#), the TIMER firmware functions are introduced in chapter [3.22.2](#).

3.22.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

Table 3-464. TIMERx Registers

Registers	Descriptions
TIMER_CTL0(TIMERx, x=0, 1, 2, 5, 13, 14, 15, 16)	Control register 0
TIMERx_CTL1(TIMERx, x=0, 1, 2, 5, 14, 15, 16)	Control register 1
TIMERx_SMCFG(TIMERx, x=0, 1, 2, 14)	Slave mode configuration register
TIMERx_DMAINTEN(TIMERx, x=0, 1, 2, 5, 13, 14, 15, 16)	DMA and interrupt enable register
TIMERx_INTF(TIMERx, x=0, 1, 2, 5, 13, 14, 15, 16)	Interrupt flag register
TIMERx_SWEVG(TIMERx, x=0, 1, 2, 5, 13, 14, 15, 16)	Software event generation register
TIMERx_CHCTL0(TIMERx, x=0, 1, 2, 13, 14, 15, 16)	Channel control register 0
TIMERx_CHCTL1(TIMERx, x=0, 1, 2)	Channel control register 1
TIMERx_CHCTL2(TIMERx, x=0, 1, 2, 13, 14, 15, 16)	Channel control register 2

Registers	Descriptions
TIMERx_CNT(TIMERx, x=0, 1, 2, 5, 13, 14, 15, 16)	Counter register
TIMERx_PSC(TIMERx, x=0, 1, 2, 5, 13, 14, 15, 16)	Prescaler register
TIMERx_CAR(TIMERx, x=0, 1, 2, 5, 13, 14, 15, 16)	Counter auto reload register
TIMERx_CREP(TIMERx, x=0, 14, 15, 16)	Counter repetition register
TIMERx_CH0CV(TIMERx, x=0, 1, 2, 13, 14, 15, 16)	Channel 0 capture/compare value register
TIMERx_CH1CV(TIMERx, x=0, 1, 2, 14)	Channel 1 capture/compare value register
TIMERx_CH2CV(TIMERx, x=0, 1, 2)	Channel 2 capture/compare value register
TIMERx_CH3CV(TIMERx, x=0, 1, 2)	Channel 3 capture/compare value register
TIMERx_CCHP(TIMERx, x=0, 14, 15, 16)	Complementary channel protection register
TIMERx_DMCFG(TIMERx, x=0, 1, 2, 14, 15, 16)	DMA configuration register
TIMERx_DMATB(TIMERx, x=0, 1, 2, 14, 15, 16)	DMA transfer buffer register
TIMERx_IRMP(TIMERx, x=13)	Channel input remap register
TIMERx_CFG(TIMERx, x=0, 1, 2, 13, 14, 15, 16)	Configuration register

3.22.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

Table 3-465. TIMERx firmware function

Function name	Function description
hal_timer_init	initialize TIMER timebase
hal_timer_input_capture_config	configure TIMER input capture mode
hal_timer_output_compare_config	configure TIMER output compare mode
hal_timer_break_config	configure TIMER break function
hal_timer_ocpre_clear_source_config	configure TIMER OCPRE clear source
hal_timer_ci0_input_select	configure TIMER ci0 trigger input
hal_timer_single_pulse_mode_config	configure TIMER single pulse mode
hal_timer_clock_source_config	configure TIMER clock source
hal_timer_slave_mode_config	configure TIMER slave mode
hal_timer_decoder_config	configure TIMER decoder mode
hal_timer_hall_sensor_config	configure TIMER hall sensor mode
hal_timer_struct_init	initialize TIMER structure with the default values
hal_timer_deinit	deinitialize TIMER and device structure
hal_timer_counter_start	start TIMER counter
hal_timer_counter_stop	stop TIMER counter
hal_timer_counter_start_interrupt	start TIMER counter and update interrupt

Function name	Function description
hal_timer_counter_stop_interrupt	stop TIMER counter and update interrupt
hal_timer_counter_start_dma	start TIMER counter and update DMA request
hal_timer_counter_stop_dma	stop TIMER counter and update DMA request
hal_timer_input_capture_start	start TIMER channel input capture mode
hal_timer_input_capture_stop	stop TIMER channel input capture mode
hal_timer_input_capture_start_interrupt	start TIMER channel input capture mode and channel interrupt
hal_timer_input_capture_stop_interrupt	stop TIMER channel input capture mode and channel interrupt
hal_timer_input_capture_start_dma	start TIMER channel input capture mode and channel DMA request
hal_timer_input_capture_stop_dma	stop TIMER channel input capture mode and channel DMA request
hal_timer_output_compare_start	start TIMER channel output compare mode
hal_timer_output_compare_stop	stop TIMER channel output compare mode
hal_timer_output_compare_start_interrupt	start TIMER channel output compare mode and channel interrupt
hal_timer_output_compare_stop_interrupt	stop TIMER channel output compare mode and channel interrupt
hal_timer_output_compare_start_dma	start TIMER channel output compare mode and channel DMA request
hal_timer_output_compare_stop_dma	stop TIMER channel output compare mode and channel DMA request
hal_timer_output_compare_complementary_channel_start	start TIMER complementary channel output compare mode
hal_timer_output_compare_complementary_channel_stop	stop TIMER complementary channel output compare mode
hal_timer_output_compare_complementary_channel_start_interrupt	start TIMER complementary channel output compare mode and channel interrupt
hal_timer_output_compare_complementary_channel_stop_interrupt	stop TIMER complementary channel output compare mode and channel interrupt
hal_timer_output_compare_complementary_channel_start_dma	start TIMER complementary channel output compare mode and channel DMA request
hal_timer_output_compare_complementary_channel_stop_dma	stop TIMER complementary channel output compare mode and channel DMA request
hal_timer_single_pulse_mode_channel_config	TIMER single pulse mode configure
hal_timer_single_pulse_start	start TIMER single pulse mode
hal_timer_single_pulse_stop	stop TIMER single pulse mode
hal_timer_single_pulse_start_interrupt	start TIMER single pulse mode and channel interrupt
hal_timer_single_pulse_stop_interrupt	stop TIMER single pulse mode and channel interrupt
hal_timer_single_pulse_complementary_channel_start	start TIMER complementary channel single pulse mode

Function name	Function description
el_start	mode
hal_timer_single_pulse_complementary_chann el_stop	stop TIMER complementary channel single pulse mode
hal_timer_single_pulse_complementary_chann el_start_interrupt	start TIMER complementary channel single pulse mode and channel interrupt
hal_timer_single_pulse_complementary_chann el_stop_interrupt	stop TIMER complementary channel single pulse mode and channel interrupt
hal_timer_slave_mode_interrupt_config	configure TIMER slave mode and interrupt
hal_timer_decoder_start	start TIMER decoder mode
hal_timer_decoder_stop	stop TIMER decoder mode
hal_timer_decoder_start_interrupt	start TIMER decoder mode and channel interrupt
hal_timer_decoder_stop_interrupt	stop TIMER decoder mode and channel interrupt
hal_timer_decoder_start_dma	start TIMER decoder mode and channel DMA request
hal_timer_decoder_stop_dma	stop TIMER decoder mode and channel DMA request
hal_timer_hall_sensor_start	start TIMER hall sensor mode
hal_timer_hall_sensor_stop	stop TIMER hall sensor mode
hal_timer_hall_sensor_start_interrupt	start TIMER hall sensor mode and channel interrupt
hal_timer_hall_sensor_stop_interrupt	stop TIMER hall sensor mode and channel interrupt
hal_timer_hall_sensor_start_dma	start TIMER hall sensor mode and channel DMA request
hal_timer_hall_sensor_stop_dma	stop TIMER hall sensor mode and channel DMA request
hal_timer_dma_transfer_write_start	start TIMER DMA transfer mode for writing data to TIMER
hal_timer_dma_transfer_write_stop	stop TIMER DMA transfer mode for writing data to TIMER
hal_timer_dma_transfer_read_start	start TIMER DMA transfer mode for read data from TIMER
hal_timer_dma_transfer_read_stop	stop TIMER DMA transfer mode for read data from TIMER
hal_timer_commutation_event_config	configure TIMER commutation event
hal_timer_commutation_event_interrupt_config	configure TIMER commutation event and enable CMT interrupt
hal_timer_commutation_event_dma_config	configure TIMER commutation event and enable CMT DMA request
hal_timer_irq_handle_set	set user-defined interrupt callback function
hal_timer_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_timer_irq	TIMER interrupt handler content function, which is merely used in timer_handler

Enum hal_timer_state_enum

Table 3-466. Enum hal_timer_state_enum

Member name	Function description
HAL_TIMER_STATE_NONE	NONE (default value)
HAL_TIMER_STATE_RESET	RESET
HAL_TIMER_STATE_BUSY	BUSY
HAL_TIMER_STATE_TIMEOUT	TIMEOUT
HAL_TIMER_STATE_ERROR	ERROR
HAL_TIMER_STATE_READY	READY

Enum hal_timer_error_enum

Table 3-467. Enum hal_timer_error_enum

Member name	Function description
HAL_TIMER_ERROR_NONE	No error
HAL_TIMER_ERROR_SYSTEM	TIMER internal error: if problem of clocking, enable/disable, wrong state
HAL_TIMER_ERROR_DMA	DMA transfer error
HAL_TIMER_ERROR_CONFIG	Configuration error occurs

Enum hal_timer_service_channel_enum

Table 3-468. Enum hal_timer_service_channel_enum

Member name	Function description
HAL_TIMER_SERVICE_CHANNEL_NONE	None channel service
HAL_TIMER_SERVICE_CHANNEL_0	Service channel 0
HAL_TIMER_SERVICE_CHANNEL_1	Service channel 1
HAL_TIMER_SERVICE_CHANNEL_2	Service channel 2
HAL_TIMER_SERVICE_CHANNEL_3	Service channel 3

Enum hal_timer_struct_type_enum

Table 3-469. Enum hal_timer_struct_type_enum

Member name	Function description
HAL_TIMER_INIT_STRUCT	TIMER basic initialize configuration structure
HAL_TIMER_INPUT_CAPTURE_STRUCT	TIMER input capture configuration structure
HAL_TIMER_OUTPUT_COMPARE_STRUCT	TIMER output compare configuration structure
HAL_TIMER_BREAK_STRUCT	TIMER break and complementary channel protection configuration structure
HAL_TIMER_CLEAR_SOURCE_STRUCT	TIMER OCPRE clear source configuration structure
HAL_TIMER_CLOCK_SOURCE_STRUCT	TIMER clock source configuration structure
HAL_TIMER_SLAVE_MODE_STRUCT	TIMER slave mode configuration structure

HAL_TIMER_DECODER_STRUCT	TIMER decoder mode configuration structure
HAL_TIMER_DECODER_DMA_CONFIG_STRUCT	TIMER decoder mode DMA transfer configuration structure
HAL_TIMER_HALL_SENSOR_STRUCT	TIMER hall sensor mode configuration structure
HAL_TIMER_SINGLE_PULSE_STRUCT	TIMER single pulse mode configuration structure
HAL_TIMER_DMA_TRANSFER_CONFIG_STRUCT	TIMER DMA transfer configuration structure
HAL_TIMER_IRQ_STRUCT	TIMER interrupt user callback function pointer structure
HAL_TIMER_DMA_HANDLE_CB_STRUCT	TIMER DMA interrupt user callback function pointer structure
HAL_TIMER_DEV_STRUCT	TIMER device information structure

Enum hal_timer_dma_transfer_start_address_enum

Table 3-470. Enum hal_timer_dma_transfer_start_address_enum

Member name	Function description
TIMER_DMA_START_ADDRESS_CTL0	DMA transfer address is TIMER_CTL0
TIMER_DMA_START_ADDRESS_CTL1	DMA transfer address is TIMER_CTL1
TIMER_DMA_START_ADDRESS_SMCFG	DMA transfer address is TIMER_SMCFG
TIMER_DMA_START_ADDRESS_DMAINTEN	DMA transfer address is TIMER_DMAINTEN
TIMER_DMA_START_ADDRESS_INTF	DMA transfer address is TIMER_INTF
TIMER_DMA_START_ADDRESS_SWEVG	DMA transfer address is TIMER_SWEVG
TIMER_DMA_START_ADDRESS_CHCTL0	DMA transfer address is TIMER_CHCTL0
TIMER_DMA_START_ADDRESS_CHCTL1	DMA transfer address is TIMER_CHCTL1
TIMER_DMA_START_ADDRESS_CHCTL2	DMA transfer address is TIMER_CHCTL2
TIMER_DMA_START_ADDRESS_CNT	DMA transfer address is TIMER_CNT
TIMER_DMA_START_ADDRESS_PSC	DMA transfer address is TIMER_PSC
TIMER_DMA_START_ADDRESS_CAR	DMA transfer address is TIMER_CAR
TIMER_DMA_START_ADDRESS_CREP	DMA transfer address is TIMER_CREP
TIMER_DMA_START_ADDRESS_CH0CV	DMA transfer address is TIMER_CH0CV
TIMER_DMA_START_ADDRESS_CH1CV	DMA transfer address is TIMER_CH1CV
TIMER_DMA_START_ADDRESS_CH2CV	DMA transfer address is TIMER_CH2CV
TIMER_DMA_START_ADDRESS_CH3CV	DMA transfer address is TIMER_CH3CV
TIMER_DMA_START_ADDRESS_CCHP	DMA transfer address is TIMER_CCHP
TIMER_DMA_START_ADDRESS_DMACFG	DMA transfer address is TIMER_DMACFG

Enum hal_timer_dma_transfer_length_enum

Table 3-471. Enum hal_timer_dma_transfer_length_enum

Member name	Function description
TIMER_DMACFG_DMATC_1TRANSFER	DMA transfer 1 time

TIMER_DMACFG_DMATC_2TRANSFER	DMA transfer 2 time
TIMER_DMACFG_DMATC_3TRANSFER	DMA transfer 3 time
TIMER_DMACFG_DMATC_4TRANSFER	DMA transfer 4 time
TIMER_DMACFG_DMATC_5TRANSFER	DMA transfer 5 time
TIMER_DMACFG_DMATC_6TRANSFER	DMA transfer 6 time
TIMER_DMACFG_DMATC_7TRANSFER	DMA transfer 7 time
TIMER_DMACFG_DMATC_8TRANSFER	DMA transfer 8 time
TIMER_DMACFG_DMATC_9TRANSFER	DMA transfer 9 time
TIMER_DMACFG_DMATC_10TRANSFER	DMA transfer 10 time
TIMER_DMACFG_DMATC_11TRANSFER	DMA transfer 11 time
TIMER_DMACFG_DMATC_12TRANSFER	DMA transfer 12 time
TIMER_DMACFG_DMATC_13TRANSFER	DMA transfer 13 time
TIMER_DMACFG_DMATC_14TRANSFER	DMA transfer 14 time
TIMER_DMACFG_DMATC_15TRANSFER	DMA transfer 15 time
TIMER_DMACFG_DMATC_16TRANSFER	DMA transfer 16 time
TIMER_DMACFG_DMATC_17TRANSFER	DMA transfer 17 time
TIMER_DMACFG_DMATC_18TRANSFER	DMA transfer 18 time

Enum hal_timer_trgo_selection_enum

Table 3-472. Enum hal_timer_trgo_selection_enum

Member name	Function description
TIMER_TRI_OUT_SRC_RESET	The UPG bit as trigger output
TIMER_TRI_OUT_SRC_ENABL	The counter enable signal TIMER_CTL0_CEN as trigger output
TIMER_TRI_OUT_SRC_UPDATE	Update event as trigger output
TIMER_TRI_OUT_SRC_CH0	A capture or a compare match occurred in channel0 as trigger output TRGO
TIMER_TRI_OUT_SRC_O0CPRE	O0CPRE as trigger output
TIMER_TRI_OUT_SRC_O1CPRE	O1CPRE as trigger output
TIMER_TRI_OUT_SRC_O2CPRE	O2CPRE as trigger output
TIMER_TRI_OUT_SRC_O3CPRE	O3CPRE as trigger output

Enum hal_timer_output_compare_enum

Table 3-473. Enum hal_timer_output_compare_enum

Member name	Function description
TIMER_OC_MODE_TIMING	Timing mode
TIMER_OC_MODE_ACTIVE	Active mode, set on match
TIMER_OC_MODE_INACTIVE	Inactive mode, clear on match
TIMER_OC_MODE_TOGGLE	Toggle mode
TIMER_OC_MODE_LOW	Force low mode

TIMER_OC_MODE_HIGH	Force high mode
TIMER_OC_MODE_PWM0	PWM0 mode
TIMER_OC_MODE_PWM1	PWM1 mode

Enum hal_timer_clock_source_enum

Table 3-474. Enum hal_timer_clock_source_enum

Member name	Function description
TIMER_CLOCK_SOURCE_CK_TIMER	Clock input source selection:CK_TIMER, internal clock
TIMER_CLOCK_SOURCE_ITI0	Clock input source selection:ITI0,external clock mode 0
TIMER_CLOCK_SOURCE_ITI1	Clock input source selection:ITI1, external clock mode 0
TIMER_CLOCK_SOURCE_ITI2	Clock input source selection:ITI2, external clock mode 0
TIMER_CLOCK_SOURCE_ITI3	Clock input source selection:ITI3, external clock mode 0
TIMER_CLOCK_SOURCE_CIOFE0	Clock input source selection:CIOFE0, external clock mode 0
TIMER_CLOCK_SOURCE_CI1FE1	Clock input source selection:CI1FE1, external clock mode 0
TIMER_CLOCK_SOURCE_CIOFED	Clock input source selection:CIOFED, external clock mode 0
TIMER_CLOCK_SOURCE_ETIMODE0	Clock input source selection:ETI in external clock mode0
TIMER_CLOCK_SOURCE_ETIMODE1	Clock input source selection:ETI in external clock mode1

Enum hal_timer_slave_mode_enum

Table 3-475. Enum hal_timer_slave_mode_enum

Member name	Function description
TIMER_SLAVE_DISABLE_MODE	Slave mode disable
TIMER_SLAVE_MODE_RESTART_MODE	Restart mode
TIMER_SLAVE_MODE_PAUSE_MODE	Pause mode
TIMER_SLAVE_MODE_EVENT_MODE	Event mode

Enum hal_timer_input_trigger_source_enum

Table 3-476. Enum hal_timer_input_trigger_source_enum

Member name	Function description
TIMER_TRIGGER_SOURCE_ITI0	Trigger input source selection:ITI0
TIMER_TRIGGER_SOURCE_ITI1	Trigger input source selection:ITI1
TIMER_TRIGGER_SOURCE_ITI2	Trigger input source selection:ITI2
TIMER_TRIGGER_SOURCE_ITI3	Trigger input source selection:ITI3
TIMER_TRIGGER_SOURCE_CIOFE0	Trigger input source selection:CIOFE0
TIMER_TRIGGER_SOURCE_CI1FE1	Trigger input source selection:CI1FE1
TIMER_TRIGGER_SOURCE_CIOFED	Trigger input source selection:CIOFED
TIMER_TRIGGER_SOURCE_ETIFP	Trigger input source selection:ETIFP
TIMER_TRIGGER_SOURCE_DISABLE	Trigger input source selection:none

Enum hal_timer_decoder_mode_enum

Table 3-477. Enum hal_timer_decoder_mode_enum

Member name	Function description
TIMER_QUADRATURE_DECODER_MODE0	Quadrature decoder mode 0
TIMER_QUADRATURE_DECODER_MODE1	Quadrature decoder mode 1
TIMER_QUADRATURE_DECODER_MODE2	Quadrature decoder mode 2

Structure hal_timer_decoder_dma_config_struct

Table 3-478. Structure hal_timer_decoder_dma_config_struct

Member name	Function description
mem_addr0	TIMER decoder mode DMA transfer memory address 0
mem_addr1	TIMER decoder mode DMA transfer memory address 1
length	TIMER DMA transfer length

Structure hal_timer_dma_transfer_config_struct

Table 3-479. Structure hal_timer_dma_transfer_config_struct

Member name	Function description
start_addr	TIMER DMA transfer access start address
mem_addr	TIMER DMA transfer memory address
length	TIMER DMA transfer access length

Structure hal_timer_irq_struct

Table 3-480. Structure hal_timer_irq_struct

Member name	Function description
update_handle	TIMER update interrupt callback function
channelx_capture_handle	TIMER channel interrupt for input capture callback function
channelx_compare_handle	TIMER channel interrupt for output compare callback function
commutation_handle	TIMER commutation interrupt callback function
trigger_handle	TIMER trigger interrupt callback function
break_handle	TIMER break interrupt callback function

Structure hal_timer_dma_handle_cb_struct

Table 3-481. Structure hal_timer_dma_handle_cb_struct

Member name	Function description
update_dma_full_transcom_handle	TIMER DMA transfer complete interrupt handler for TIMER update DMA request

channelx_capture_dma_full_transcom_handle	TIMER DMA transfer complete interrupt handler for TIMER channel input capture DMA request
channelx_compare_dma_full_transcom_handle	TIMER DMA transfer complete interrupt handler for TIMER channel output compare DMA request
commutation_dma_full_transcom_handle	TIMER DMA transfer complete interrupt handler for TIMER commutation DMA request
trigger_dma_full_transcom_handle	TIMER DMA transfer complete interrupt handler for TIMER trigger DMA request
error_handle	TIMER DMA transfer error interrupt handler

Structure hal_timer_dev_struct

Table 3-482. Structure hal_timer_dev_struct

Member name	Function description
periph	Timerx(x=0...2,5,13..16)
service_channel	The service channel x
timer_irq	TIMER interrupt user callback function pointer structure
*p_dma_timer[7]	DMA device information structure
timer_dma	TIMER DMA interrupt user callback function pointer structure
error_state	TIMER error state
state	TIMER state
mutex	Mutex locked and unlocked state
*priv	Priv data

Structure hal_timer_init_struct

Table 3-483. Structure hal_timer_init_struct

Member name	Function description
prescaler	Prescaler value, timerx_PSC
alignedmode	Counter alignment mode
counter_direction	Counter direction
period	Period value, timerx_CAR
clock_division	Clock division value
repetition_counter	The counter repetition value, timerx_CREP
autoreload_shadow	Auto-reload shadow enable or disable, timerx_CTL0:bit7(ARSE)
master_slave_mode	Master-slave mode enable or disable, timerx_SMCFG:bit7(MSM)
trgo_selection	Master mode control, timerx_CTL1:bit4~6(MMC)

Structure hal_timer_input_capture_struct

Table 3-484. Structure hal_timer_input_capture_struct

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

ic_polarity	Channel input capture polarity
ic_selection	Channel input capture source selection
ic_prescaler	Channel input capture prescaler
ic_filter	Channel input capture filter, 0x00~0x0f

Structure hal_timer_output_compare_struct

Table 3-485. Structure hal_timer_output_compare_struct

Member name	Function description
compare_mode	Channel output compare mode control
oc_pulse_value	Channel capture or compare value
oc_polarity	Channel output polarity
ocn_polarity	Channel complementary output polarity
oc_idlestate	Idle state of channel output
ocn_idlestate	Idle state of channel complementary output
oc_shadow	Channel output compare shadow
oc_fastmode	Channel output compare fast mode
oc_clearmode	Channel output compare clear mode

Structure hal_timer_break_struct

Table 3-486. Structure hal_timer_break_struct

Member name	Function description
run_offstate	Run mode off-state
idel_offstate	Idle mode off-state
dead_time	Dead time
break_polarity	Break polarity
output_autostate	Output automatic enable
protect_mode	Complementary register protect control
break_state	Break enable

Structure hal_timer_clear_source_struct

Table 3-487. Structure hal_timer_clear_source_struct

Member name	Function description
clear_source	OCPRE clear source selection
exttrigger_polarity	External trigger polarity
exttrigger_prescaler	External trigger prescaler
exttrigger_filter	External trigger filter, 0x00~0x0f

Structure hal_timer_clock_source_struct

Table 3-488. Structure hal_timer_clock_source_struct

Member name	Function description
clock_source	Select clock input source
clock_polarity	Clock input source polarity
clock_prescaler	Clock input source prescaler
clock_filter	clock input source filter, 0x00~0x0F

Structure hal_timer_slave_mode_struct

Table 3-489. Structure hal_timer_slave_mode_struct

Member name	Function description
slavemode	Slave mode control
trigger_selection	Trigger selection
trigger_polarity	Trigger input source polarity
trigger_prescaler	Trigger input source prescaler
trigger_filter	Trigger input source filter, 0x00~0x0f

Structure hal_timer_decoder_struct

Table 3-490. Structure hal_timer_decoder_struct

Member name	Function description
decoder_mode	Decoder mode
ci0_polarity	Channel0 input capture polarity
ci0_selection	Channel0 input capture source selection
ci0_prescaler	Channel0 input capture prescaler
ci0_filter	Channel0 input capture filter
ci1_polarity	Channel1 input capture polarity
ci1_selection	Channel1 input capture source selection
ci1_prescaler	Channel1 input capture prescaler
ci1_filter	Channel1 input capture filter

Structure hal_timer_hall_sensor_struct

Table 3-491. Structure hal_timer_hall_sensor_struct

Member name	Function description
cmt_delay	Commutation delay(channel 1 compare value)
ci0_polarity	Channel0 input capture polarity
ci0_selection	Channel0 input capture source selection
ci0_prescaler	Channel0 input capture prescaler
ci0_filter	Channel0 input capture filter

Structure hal_timer_single_pulse_struct

Table 3-492. Structure hal_timer_single_pulse_struct

Member name	Function description
sp_compare_mode	Channel_out compare output control
sp_oc_pulse_value	Channel_out compare value
sp_oc_polarity	Channel_out output polarity
sp_ocn_polarity	Channel_out complementary output polarity
sp_oc_idlestate	Idle state of channel_out output
sp_ocn_idlestate	Idle state of channel_out complementary output
sp_oc_fastmode	Channel_out output compare fast mode
sp_oc_clearmode	Channel_out output compare clear mode
sp_ic_polarity	Channel_in input capture polarity
sp_ic_selection	Channel_in I/O mode selection
sp_ic_filter	Channel_in input capture filter

hal_timer_init

The description of hal_timer_init is shown as below:

Table 3-493. Function hal_timer_init

Function name	hal_timer_init
Function prototype	int32_t hal_timer_init(hal_timer_dev_struct *timer_dev, uint32_t periph, hal_timer_init_struct *timer)
Function descriptions	initialize TIMER timebase
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
periph	specify which TIMER timebase is initialized
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Input parameter{in}	
timer	pointer to hal_timer_init_struct, struct members refer to Table 3-483. Structure hal_timer_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```

/*initialize TIMER timebase */

hal_timer_dev_struct timer0_info;

hal_timer_init_struct timer0_init_parameter;

timer0_init_parameter.prescaler = 0;

timer0_init_parameter.alignedmode = TIMER_COUNTER_EDGE;

timer0_init_parameter.counter_direction = TIMER_COUNTER_UP;

timer0_init_parameter.period = 65535;

timer0_init_parameter.clock_division = TIMER_CKDIV_DIV1;

timer0_init_parameter.repetition_counter = 0;

timer0_init_parameter.autoreload_shadow = TIMER_CARL_SHADOW_DISABLE;

timer0_init_parameter.master_slave_mode = DISABLE;

timer0_init_parameter.trgo_selection = TIMER_TRI_OUT_SRC_RESET;

hal_timer_init(&timer0_info,TIMER0,&timer0_init_parameter);

```

hal_timer_input_capture_config

The description of hal_timer_input_capture_config is shown as below:

Table 3-494. Function hal_timer_input_capture_config

Function name	hal_timer_input_capture_config
Function prototype	int32_t hal_timer_input_capture_config(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_input_capture_struct *timer_inputcapture)
Function descriptions	configure TIMER input capture mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_2	Channel 2
TIMER_CH_3	Channel 3
Input parameter{in}	
timer_inputcapture	pointer to hal_timer_inputcapture_struct, struct members refer to Table 3-484. Structure hal_timer_input_capture_struct

Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT

Example:

```
/* configure TIMER input capture mode */

hal_timer_dev_struct timer0_info;

hal_timer_input_capture_struct timer0_input_capture_parameter;

timer0_input_capture_parameter.ic_polarity = TIMER_IC_POLARITY_RISING;

timer0_input_capture_parameter.ic_selection = TIMER_IC_SELECTION_DIRECTTI;

timer0_input_capture_parameter.ic_prescaler = TIMER_IC_PRESCALER_OFF;

timer0_input_capture_parameter.ic_filter = 0;

hal_timer_input_capture_config(&timer0_info,TIMER_CH_0,&timer0_input_capture_parameter);
```

hal_timer_output_compare_config

The description of hal_timer_output_compare_config is shown as below:

Table 3-495. Function hal_timer_output_compare_config

Function name	hal_timer_output_compare_config
Function prototype	int32_t hal_timer_output_compare_config(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_output_compare_struct *timer_outputcompare)
Function descriptions	configure TIMER output compare mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_2	Channel 2
TIMER_CH_3	Channel 3
Input parameter{in}	
timer_outputcompare	pointer to hal_timer_output_compare_struct, struct members refer to Table

	<u>3-485. Structure hal timer output compare struct</u>
	Output parameter{out}
-	-
	Return value
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT

Example:

```
/* configure TIMER output compare mode: TIMER_OUTPUT_TOGGLE_MODE */
hal_timer_dev_struct timer0_info;
hal_timer_output_compare_struct timer0_output_compare_parameter;
timer0_output_compare_parameter.compare_mode = TIMER_OC_MODE_TOGGLE;
timer0_output_compare_parameter.oc_pulse_value = 500;
timer0_output_compare_parameter.oc_polarity = TIMER_OC_POLARITY_HIGH;
timer0_output_compare_parameter.oc_idlestate = TIMER_OC_IDLE_STATE_LOW;
timer0_output_compare_parameter.oc_shadow = TIMER_OC_SHADOW_DISABLE;
timer0_output_compare_parameter.oc_clearmode = TIMER_OC_CLEAR_DISABLE;
hal_timer_output_compare_config(&timer0_info, TIMER_CH_0, &timer0_output_compare_parameter);
```

hal_timer_break_config

The description of hal_timer_break_config is shown as below:

Table 3-496. Function hal_timer_break_config

Function name	hal_timer_break_config
Function prototype	int32_t hal_timer_break_config(hal_timer_dev_struct *timer_dev, hal_timer_break_struct *timer_break)
Function descriptions	configure TIMER break function
Precondition	-
The called functions	-
	Input parameter{in}
timer_dev	pointer to hal_timer_dev_struct, struct members refer to <u>Table 3-482. Structure hal timer dev struct</u>
	Input parameter{in}
timer_break	pointer to hal_timer_break_struct, struct members refer to <u>Table 3-486. Structure hal timer break struct</u>
	Output parameter{out}

-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT

Example:

```
/* configure TIMER break function */

hal_timer_dev_struct timer0_info;

hal_timer_break_struct timer0_break_parameter;

timer0_break_parameter.run_offstate = TIMER_ROS_STATE_DISABLE;

timer0_break_parameter.idel_offstate = TIMER_IOS_STATE_DISABLE;

timer0_break_parameter.break_polarity = TIMER_BREAK_POLARITY_HIGH;

timer0_break_parameter.output_autostate = TIMER_OUTAUTO_DISABLE;

timer0_break_parameter.protect_mode = TIMER_CCHP_PROT_OFF;

timer0_break_parameter.break_state = TIMER_BREAK_ENABLE;

hal_timer_break_config(&timer0_info,&timer0_break_parameter);
```

hal_timer_ocpre_clear_source_config

The description of hal_timer_ocpre_clear_source_config is shown as below:

Table 3-497. Function hal_timer_ocpre_clear_source_config

Function name	hal_timer_ocpre_clear_source_config
Function prototype	int32_t hal_timer_ocpre_clear_source_config(hal_timer_dev_struct *timer_dev, hal_timer_clear_source_struct *timer_clearsource)
Function descriptions	configure TIMER OCPRE clear source
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
timer_clearsource	pointer to hal_timer_clear_source_struct, struct members refer to Table 3-487. Structure hal_timer_clear_source_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS,

	HAL_ERR_NO_SUPPORT, HAL_ERR_VAL
--	---------------------------------

Example:

```
/* configure TIMER OCPRE clear source */

hal_timer_dev_struct timer0_info;

hal_timer_clear_source_struct timer0_clear_source_parameter;

timer0_clear_source_parameter.clear_source = TIMER_OCPRE_CLEAR_SOURCE_ETIF;

timer0_clear_source_parameter.exttrigger_polarity = TIMER_EXT_TRI_POLARITY_RISING;

timer0_clear_source_parameter.exttrigger_prescaler =
TIMER_EXT_TRI_PRESCALER_OFF;

timer0_clear_source_parameter.exttrigger_filter = 0;

hal_timer_ocpre_clear_source_config(&timer0_info,&timer0_clear_source_parameter);
```

hal_timer_ci0_input_select

The description of hal_timer_ci0_input_select is shown as below:

Table 3-498. Function hal_timer_ci0_input_select

Function name	hal_timer_ci0_input_select
Function prototype	int32_t hal_timer_ci0_input_select(hal_timer_dev_struct *timer_dev, uint32_t ci0_select)
Function descriptions	configure TIMER ci0 trigger input
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
ci0_select	CI0 select
<i>TIMER_CIO_CH0IN</i>	the TIMEx_CH0 pin input is selected as channel 0 trigger input
<i>TIMER_CIO_XOR_CH0</i> 12	the result of combinational XOR of TIMEx_CH0,CH1 and CH2 pins is selected as channel 0 trigger input
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL

Example:

```
/* configure TIMER ci0 trigger input */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_ci0_input_select(&timer0_info,TIMER_CIO_XOR_CH012);
```

hal_timer_single_pulse_mode_config

The description of hal_timer_single_pulse_mode_config is shown as below:

Table 3-499. Function hal_timer_single_pulse_mode_config

Function name	hal_timer_single_pulse_mode_config
Function prototype	int32_t hal_timer_single_pulse_mode_config(hal_timer_dev_struct *timer_dev, uint32_t single_pulse)
Function descriptions	configure TIMER single pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
single_pulse	single pulse mode state
ENABLE	single pulse mode enable
DISABLE	single pulse mode disable
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL

Example:

```
/* configure TIMER single pulse mode */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_single_pulse_mode_config(&timer0_info,ENABLE);
```

hal_timer_clock_source_config

The description of hal_timer_clock_source_config is shown as below:

Table 3-500. Function hal_timer_clock_source_config

Function name	hal_timer_clock_source_config
Function prototype	int32_t hal_timer_clock_source_config(hal_timer_dev_struct *timer_dev, hal_timer_clock_source_struct *timer_clocksource)
Function descriptions	configure TIMER clock source
Precondition	-

The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
timer_clock	pointer to hal_timer_clock_source_struct, struct members refer to Table 3-488. Structure hal_timer_clock_source_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL

Example:

```

/* configure TIMER clock source */

hal_timer_dev_struct timer0_info;

hal_timer_clock_source_struct timer0_clock_source_parameter;

timer0_clock_source_parameter.clock_source = TIMER_CLOCK_SOURCE_ETIMODE1;

timer0_clock_source_parameter.clock_polarity =
TIMER_CLOCK_TRIGGER_ETI_POLARITY_RISING;

timer0_clock_source_parameter.clock_prescaler = TIMER_EXT_TRI_PRESCALER_OFF;

timer0_clock_source_parameter.clock_filter = 0;

hal_timer_clock_source_config(&timer0_info,&timer0_clock_source_parameter);

```

hal_timer_slave_mode_config

The description of hal_timer_slave_mode_config is shown as below:

Table 3-501. Function hal_timer_slave_mode_config

Function name	hal_timer_slave_mode_config
Function prototype	int32_t hal_timer_slave_mode_config(hal_timer_dev_struct *timer_dev, hal_timer_slave_mode_struct *timer_slavemode)
Function descriptions	configure TIMER slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	

timer_slavemode	pointer to hal_timer_slave_mode_struct, struct members refer to Table 3-489. Structure hal timer slave mode struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL

Example:

```

/* configure TIMER slave mode */

hal_timer_dev_struct timer0_info;

hal_timer_slave_mode_struct timer0_slave_mode_parameter;

timer0_slave_mode_parameter.slavemode = TIMER_SLAVE_MODE_RESTART_MODE;

timer0_slave_mode_parameter.trigger_selection = TIMER_TRIGGER_SOURCE_ETIFP;

timer0_slave_mode_parameter.trigger_polarity =
TIMER_CLOCK_TRIGGER_ETI_POLARITY_RISING;

timer0_slave_mode_parameter.trigger_prescaler = TIMER_EXT_TRI_PRESCALER_OFF;

timer0_slave_mode_parameter.trigger_filter = 0;

hal_timer_slave_mode_config(&timer0_info,&timer0_slave_mode_parameter);

```

hal_timer_decoder_config

The description of hal_timer_decoder_config is shown as below:

Table 3-502. Function hal_timer_decoder_config

Function name	hal_timer_decoder_config
Function prototype	int32_t hal_timer_decoder_config(hal_timer_dev_struct *timer_dev, hal_timer_decoder_struct *timer_decoder)
Function descriptions	configure TIMER decoder mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
timer_decoder	pointer to hal_timer_decoder_struct, struct members refer to Table 3-490. Structure hal timer decoder struct
Output parameter{out}	
-	-

Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT

Example:

```
/* configure TIMER decoder mode */

hal_timer_dev_struct timer0_info;

hal_timer_decoder_struct timer0_decoder_parameter;

timer0_decoder_parameter.decoder_mode = TIMER_QUADRATURE_DECODER_MODE0;
timer0_decoder_parameter.ci0_polarity = TIMER_IC_POLARITY_RISING;
timer0_decoder_parameter.ci0_selection = TIMER_IC_SELECTION_DIRECTTTI;
timer0_decoder_parameter.ci0_prescaler = TIMER_IC_PRESCALER_OFF;
timer0_decoder_parameter.ci0_filter = 0;
timer0_decoder_parameter.ci1_polarity = TIMER_IC_POLARITY_RISING;
timer0_decoder_parameter.ci1_selection = TIMER_IC_SELECTION_DIRECTTTI;
timer0_decoder_parameter.ci1_prescaler = TIMER_IC_PRESCALER_OFF;
timer0_decoder_parameter.ci1_filter = 0;

hal_timer_decoder_config(&timer0_info,&timer0_decoder_parameter);
```

hal_timer_hall_sensor_config

The description of hal_timer_hall_sensor_config is shown as below:

Table 3-503. Function hal_timer_hall_sensor_config

Function name	hal_timer_hall_sensor_config
Function prototype	int32_t hal_timer_hall_sensor_config(hal_timer_dev_struct *timer_dev, hal_timer_hall_sensor_struct *timer_hallsensor)
Function descriptions	configure TIMER hall sensor mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
timer_hallsensor	pointer to hal_timer_hall_sensor_struct, struct members refer to Table 3-491. Structure hal timer hall sensor struct
Output parameter{out}	

-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT

Example:

```
/* configure TIMER hall sensor mode */

hal_timer_dev_struct timer0_info;

hal_timer_hall_sensor_struct timer0_hall_sensor_parameter;

timer0_hall_sensor_parameter.cmt_delay = 50;

timer0_hall_sensor_parameter.ci0_polarity = TIMER_IC_POLARITY_RISING;

timer0_hall_sensor_parameter.ci0_selection = TIMER_IC_SELECTION_ITS;

timer0_hall_sensor_parameter.ci0_prescaler = TIMER_IC_PRESCALER_OFF;

timer0_hall_sensor_parameter.ci0_filter = 0;

hal_timer_hall_sensor_config(&timer0_info,&timer0_hall_sensor_parameter);
```

hal_timer_struct_init

The description of hal_timer_struct_init is shown as below:

Table 3-504. Function hal_timer_struct_init

Function name	hal_timer_struct_init
Function prototype	int32_t hal_timer_struct_init(hal_timer_struct_type_enum hal_struct_type, void *p_struct)
Function descriptions	initialize TIMER structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	type of structure to be initialized, refer to Table 3-469. Enum hal_timer_struct_type_enum
Input parameter{in}	
p_struct	pointer to TIMER structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:


```
/* initialize the TIMER hal_timer_dev_struct structure with the default values */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_struct_init(HAL_TIMER_DEV_STRUCT, &timer0_info);
```

hal_timer_deinit

The description of hal_timer_deinit is shown as below:

Table 3-505. Function hal_timer_deinit

Function name	hal_timer_deinit
Function prototype	int32_t hal_timer_deinit(hal_timer_dev_struct *timer_dev)
Function descriptions	deinitialize TIMER and device structure
Precondition	-
The called functions	hal_rcu_periph_reset_enable / hal_rcu_periph_reset_disable
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* deinitialize TIMER and device structure */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_deinit(&timer0_info);
```

hal_timer_counter_start

The description of hal_timer_counter_start is shown as below:

Table 3-506. Function hal_timer_counter_start

Function name	hal_timer_counter_start
Function prototype	int32_t hal_timer_counter_start(hal_timer_dev_struct *timer_dev)
Function descriptions	start TIMER counter
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-

Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start TIMER counter */

hal_timer_dev_struct timer0_info;

hal_timer_counter_start(&timer0_info);
```

hal_timer_counter_stop

The description of hal_timer_counter_stop is shown as below:

Table 3-507. Function hal_timer_counter_stop

Function name	hal_timer_counter_stop
Function prototype	int32_t hal_timer_counter_stop(hal_timer_dev_struct *timer_dev)
Function descriptions	stop TIMER counter
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK, HAL_ERR_NO_SUPPORT

Example:

```
/* stop TIMER counter */

hal_timer_dev_struct timer0_info;

hal_timer_counter_stop(&timer0_info);
```

hal_timer_counter_start_interrupt

The description of hal_timer_counter_start_interrupt is shown as below:

Table 3-508. Function hal_timer_counter_start_interrupt

Function name	hal_timer_counter_start_interrupt
Function prototype	int32_t hal_timer_counter_start_interrupt(hal_timer_dev_struct *timer_dev, hal_timer_irq_struct *p_irq)
Function descriptions	start TIMER counter and update interrupt
Precondition	-

The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal_timer_irq_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```

/* start TIMER counter and update interrupt */

void timer0_update_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

uint8_t update_num = 0;

timer0_irq_parameter.update_handle = timer0_update_interrupt_handle_userdefine;

hal_timer_counter_start_interrupt(&timer0_info,&timer0_irq_parameter);

void timer0_update_interrupt_handle_userdefine(void *ptr)
{
    update_num++;
}

```

hal_timer_counter_stop_interrupt

The description of hal_timer_counter_stop_interrupt is shown as below:

Table 3-509. Function hal_timer_counter_stop_interrupt

Function name	hal_timer_counter_stop_interrupt
Function prototype	int32_t hal_timer_counter_stop_interrupt(hal_timer_dev_struct *timer_dev)
Function descriptions	stop TIMER counter and update interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK, HAL_ERR_NO_SUPPORT

Example:

```
/* stop TIMER counter and update interrupt */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_counter_stop_interrupt(&timer0_info);
```

hal_timer_counter_start_dma

The description of hal_timer_counter_start_dma is shown as below:

Table 3-510. Function hal_timer_counter_start_dma

Function name	hal_timer_counter_start_dma
Function prototype	int32_t hal_timer_counter_start_dma(hal_timer_dev_struct *timer_dev, hal_timer_dma_handle_cb_struct *dmacb, uint32_t *mem_addr, uint16_t dma_length)
Function descriptions	start TIMER counter and update DMA request
Precondition	-
The called functions	hal_dma_start_interrupt
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
dmacb	pointer to hal_timer_dma_handle_cb_struct, struct members refer to Table 3-481. Structure hal timer dma handle cb struct
Input parameter{in}	
mem_addr	TIMER DMA transfer memory address
Input parameter{in}	
dma_length	TIMER DMA transfer count, 0~65535
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start TIMER counter and update DMA request */
```

```
void timer0_dmaerror(void *ptr);
```

```

void timer0_update_dmatc_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_dma_handle_cb_struct timer_dma_handle_cb;

uint32_t timer_cnt[3] = {100,200,300};

uint8_t timer0_dmaerror_num = 0;

uint8_t timer0_update_dmatc_num = 0;

timer_dma_handle_cb.          update_dma_full_transcom_handle          =          &
timer0_update_dmatc_userdefine;

timer_dma_handle_cb.error_handle = &timer0_dmaerror;

hal_timer_counter_start_dma(&timer0_info, &timer_dma_handle_cb, timer_cnt, 3);

void timer0_dmaerror(void *ptr)
{
    timer0_dmaerror_num++;
}

void timer0_update_dmatc_userdefine(void *ptr)
{
    timer0_update_dmatc_num++;
}

```

hal_timer_counter_stop_dma

The description of hal_timer_counter_stop_dma is shown as below:

Table 3-511. Function hal_timer_counter_stop_dma

Function name	hal_timer_counter_stop_dma
Function prototype	int32_t hal_timer_counter_stop_dma(hal_timer_dev_struct *timer_dev)
Function descriptions	stop TIMER counter and update DMA request
Precondition	-
The called functions	hal_dma_stop
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK, HAL_ERR_NO_SUPPORT
----------------	--

Example:

```
/* stop TIMER counter and update DMA request */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_counter_stop_dma(&timer0_info);
```

hal_timer_input_capture_start

The description of hal_timer_input_capture_start is shown as below:

Table 3-512. Function hal_timer_input_capture_start

Function name	hal_timer_input_capture_start
Function prototype	int32_t hal_timer_input_capture_start(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	start TIMER channel input capture mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
<i>TIMER_CH_3</i>	Channel 3
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start TIMER channel input capture mode */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_input_capture_start(&timer0_info, TIMER_CH_0);
```

hal_timer_input_capture_stop

The description of hal_timer_input_capture_stop is shown as below:

Table 3-513. Function hal_timer_input_capture_stop

Function name	hal_timer_input_capture_stop
Function prototype	int32_t hal_timer_input_capture_stop(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER channel input capture mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
<i>TIMER_CH_3</i>	Channel 3
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS , HAL_ERR_NO_SUPPORT , HAL_ERR_LOCK

Example:

```

/* stop TIMER channel input capture mode */

hal_timer_dev_struct timer0_info;

hal_timer_input_capture_stop(&timer0_info, TIMER_CH_0);

```

hal_timer_input_capture_start_interrupt

The description of hal_timer_input_capture_start_interrupt is shown as below:

Table 3-514. Function hal_timer_input_capture_start_interrupt

Function name	hal_timer_input_capture_start_interrupt
Function prototype	int32_t hal_timer_input_capture_start_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_irq_struct *p_irq)
Function descriptions	start TIMER channel input capture mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	

channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
<i>TIMER_CH_3</i>	Channel 3
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal_timer_irq_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```

/* start TIMER channel input capture mode and channel interrupt */
void timer0_input_capture_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

uint8_t input_capture_num = 0;

timer0_irq_parameter.channelx_capture_handle =
timer0_input_capture_interrupt_handle_userdefine;

hal_timer_input_capture_start_interrupt(&timer0_info,                TIMER_CH_0,
&timer0_irq_parameter);

void timer0_input_capture_interrupt_handle_userdefine(void *ptr)
{
    input_capture_num++;
}

```

hal_timer_input_capture_stop_interrupt

The description of hal_timer_input_capture_stop_interrupt is shown as below:

Table 3-515. Function hal_timer_input_capture_stop_interrupt

Function name	hal_timer_input_capture_stop_interrupt
Function prototype	int32_t hal_timer_input_capture_stop_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER channel input capture mode and channel interrupt

Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
<i>TIMER_CH_3</i>	Channel 3
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* stop TIMER channel input capture mode and channel interrupt */
hal_timer_dev_struct timer0_info;
hal_timer_input_capture_stop_interrupt(&timer0_info, TIMER_CH_0);
```

hal_timer_input_capture_start_dma

The description of hal_timer_input_capture_start_dma is shown as below:

Table 3-516. Function hal_timer_input_capture_start_dma

Function name	hal_timer_input_capture_start_dma
Function prototype	int32_t hal_timer_input_capture_start_dma(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_dma_handle_cb_struct *dmacb, uint32_t *mem_addr, uint16_t dma_length)
Function descriptions	start TIMER channel input capture mode and channel DMA request
Precondition	-
The called functions	hal_dma_start_interrupt
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2

<i>TIMER_CH_3</i>	Channel 3
Input parameter{in}	
dmacb	pointer to hal_timer_dma_handle_cb_struct, struct members refer to Table 3-481. Structure hal_timer_dma_handle_cb_struct
Input parameter{in}	
mem_addr	TIMER DMA transfer memory address
Input parameter{in}	
dma_length	TIMER DMA transfer count, 0~65535
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```

/* start TIMER channel input capture mode and channel DMA request */

void timer0_dmaerror(void *ptr);

void timer0_input_capture_dmatc_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_dma_handle_cb_struct timer_dma_handle_cb;

uint32_t timer_ch0val[3] = {0};

uint8_t timer0_dmaerror_num = 0;

uint8_t timer0_input_capture_dmatc_num = 0;

timer_dma_handle_cb.channelx_capture_dma_full_transcom_handle = &
timer0_input_capture_dmatc_userdefine;

timer_dma_handle_cb.error_handle = &timer0_dmaerror;

hal_timer_input_capture_start_dma(&timer0_info, TIMER_CH_0 ,&timer_dma_handle_cb,
timer_ch0val, 3);

void timer0_dmaerror(void *ptr)

{

    timer0_dmaerror_num++;

}

void timer0_input_capture_dmatc_userdefine(void *ptr)

{

```

```

        timer0_input_capture_dmatc_num++;
    }

```

hal_timer_input_capture_stop_dma

The description of hal_timer_input_capture_stop_dma is shown as below:

Table 3-517. Function hal_timer_input_capture_stop_dma

Function name	hal_timer_input_capture_stop_dma
Function prototype	int32_t hal_timer_input_capture_stop_dma(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER channel input capture mode and channel DMA request
Precondition	-
The called functions	hal_dma_stop
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
<i>TIMER_CH_3</i>	Channel 3
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```

/* stop TIMER channel input capture mode and channel DMA request */

hal_timer_dev_struct timer0_info;

hal_timer_input_capture_stop_dma (&timer0_info, TIMER_CH_0);

```

hal_timer_output_compare_start

The description of hal_timer_output_compare_start is shown as below:

Table 3-518. Function hal_timer_output_compare_start

Function name	hal_timer_output_compare_start
Function prototype	int32_t hal_timer_output_compare_start(hal_timer_dev_struct *timer_dev, uint16_t channel)

Function descriptions	start TIMER channel output compare mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
<i>TIMER_CH_3</i>	Channel 3
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK,

Example:

```
/* start TIMER channel output compare mode */
hal_timer_dev_struct timer0_info;
hal_timer_output_compare_start(&timer0_info, TIMER_CH_0);
```

hal_timer_output_compare_stop

The description of hal_timer_output_compare_stop is shown as below:

Table 3-519. Function hal_timer_output_compare_stop

Function name	hal_timer_output_compare_stop
Function prototype	int32_t hal_timer_output_compare_stop(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER channel output compare mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
<i>TIMER_CH_3</i>	Channel 3

Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_LOCK,

Example:

```
/* stop TIMER channel output compare mode */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_output_compare_stop(&timer0_info, TIMER_CH_0);
```

hal_timer_output_compare_start_interrupt

The description of hal_timer_output_compare_start_interrupt is shown as below:

Table 3-520. Function hal_timer_output_compare_start_interrupt

Function name	hal_timer_output_compare_start_interrupt
Function prototype	int32_t hal_timer_output_compare_start_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_irq_struct *p_irq)
Function descriptions	start TIMER channel output compare mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_2	Channel 2
TIMER_CH_3	Channel 3
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal timer irq struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK,

Example:

```
/* start TIMER channel output compare mode and channel interrupt */
```

```

void timer0_output_compare_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

uint8_t output_compare_num = 0;

timer0_irq_parameter.channelx_compare_handle =
timer0_output_compare_interrupt_handle_userdefine;

hal_timer_output_compare_start_interrupt(&timer0_info,                TIMER_CH_0,
&timer0_irq_parameter);

void timer0_output_compare_interrupt_handle_userdefine(void *ptr)
{
    output_compare_num++;
}

```

hal_timer_output_compare_stop_interrupt

The description of hal_timer_output_compare_stop_interrupt is shown as below:

Table 3-521. Function hal_timer_output_compare_stop_interrupt

Function name	hal_timer_output_compare_stop_interrupt
Function prototype	int32_t hal_timer_output_compare_stop_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER channel output compare mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_2	Channel 2
TIMER_CH_3	Channel 3
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* stop TIMER channel output compare mode and channel interrupt */

hal_timer_dev_struct timer0_info;

hal_timer_output_compare_stop_interrupt(&timer0_info, TIMER_CH_0);
```

hal_timer_output_compare_start_dma

The description of hal_timer_output_compare_start_dma is shown as below:

Table 3-522. Function hal_timer_output_compare_start_dma

Function name	hal_timer_output_compare_start_dma
Function prototype	int32_t hal_timer_output_compare_start_dma(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_dma_handle_cb_struct *dmacb, uint32_t *mem_addr, uint16_t dma_length)
Function descriptions	start TIMER channel output compare mode and channel DMA request
Precondition	-
The called functions	hal_dma_start_interrupt
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
<i>TIMER_CH_3</i>	Channel 3
Input parameter{in}	
dmacb	pointer to hal_timer_dma_handle_cb_struct, struct members refer to Table 3-481. Structure hal_timer_dma_handle_cb_struct
Input parameter{in}	
mem_addr	TIMER DMA transfer memory address
Input parameter{in}	
dma_length	TIMER DMA transfer count, 0~65535
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* start TIMER channel output compare mode and channel DMA request */
```

```

void timer0_dmaerror(void *ptr);

void timer0_output_compare_dmatc_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_dma_handle_cb_struct timer_dma_handle_cb;

uint32_t timer_ch0val[3] = {0};

uint8_t timer0_dmaerror_num = 0;

uint8_t timer0_output_compare_dmatc_num = 0;

timer_dma_handle_cb.channelx_compare_dma_full_transcom_handle = &
timer0_output_compare_dmatc_userdefine;

timer_dma_handle_cb.error_handle = &timer0_dmaerror;

hal_timer_output_compare_start_dma(&timer0_info, TIMER_CH_0, &timer_dma_handle_cb,
timer_ch0val, 3);

void timer0_dmaerror(void *ptr)
{
    timer0_dmaerror_num++;
}

void timer0_output_compare_dmatc_userdefine(void *ptr)
{
    timer0_output_compare_dmatc_num++;
}

```

hal_timer_output_compare_stop_dma

The description of hal_timer_output_compare_stop_dma is shown as below:

Table 3-523. Function hal_timer_output_compare_stop_dma

Function name	hal_timer_output_compare_stop_dma
Function prototype	int32_t hal_timer_output_compare_stop_dma(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER channel output compare mode and channel DMA request
Precondition	-
The called functions	hal_dma_stop
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct

Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
<i>TIMER_CH_3</i>	Channel 3
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_NO_SUPPORT, HAL_ERR_LOCK

Example:

```
/* stop TIMER channel output compare mode and channel DMA request */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_output_compare_stop_dma(&timer0_info, TIMER_CH_0);
```

hal_timer_output_compare_complementary_channel_start

The description of hal_timer_output_compare_complementary_channel_start is shown as below:

Table 3-524. Function hal_timer_output_compare_complementary_channel_start

Function name	hal_timer_output_compare_complementary_channel_start
Function prototype	int32_t hal_timer_output_compare_complementary_channel_start(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	start TIMER complementary channel output compare mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start TIMER complementary channel output compare mode */

hal_timer_dev_struct timer0_info;

hal_timer_output_compare_complementary_channel_start(&timer0_info, TIMER_CH_0);
```

hal_timer_output_compare_complementary_channel_stop

The description of hal_timer_output_compare_complementary_channel_stop is shown as below:

Table 3-525. Function hal_timer_output_compare_complementary_channel_stop

Function name	hal_timer_output_compare_complementary_channel_stop
Function prototype	int32_t hal_timer_output_compare_complementary_channel_stop(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER complementary channel output compare mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_2	Channel 2
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_LOCK

Example:

```
/* stop TIMER complementary channel output compare mode */

hal_timer_dev_struct timer0_info;

hal_timer_output_compare_complementary_channel_stop(&timer0_info, TIMER_CH_0);
```

hal_timer_output_compare_complementary_channel_start_interrupt

The description of hal_timer_output_compare_complementary_channel_start_interrupt is shown as below:

Table 3-526. Function
hal_timer_output_compare_complementary_channel_start_interrupt

Function name	hal_timer_output_compare_complementary_channel_start_interrupt
Function prototype	int32_t hal_timer_output_compare_complementary_channel_start_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_irq_struct *p_irq)
Function descriptions	start TIMER complementary channel output compare mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal timer irq struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```

/* start TIMER complementary channel output compare mode and channel interrupt */
void timer0_output_compare_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

uint8_t output_compare_num = 0;

timer0_irq_parameter.channelx_compare_handle =
timer0_output_compare_interrupt_handle_userdefine;

hal_timer_output_compare_complementary_channel_start_interrupt(&timer0_info,
TIMER_CH_0, &timer0_irq_parameter);

void timer0_output_compare_interrupt_handle_userdefine(void *ptr)

```

```
{
    output_compare_num++;
}
```

hal_timer_output_compare_complementary_channel_stop_interrupt

The description of hal_timer_output_compare_complementary_channel_stop_interrupt is shown as below:

Table 3-527. Function

hal_timer_output_compare_complementary_channel_stop_interrupt

Function name	hal_timer_output_compare_complementary_channel_stop_interrupt
Function prototype	int32_t hal_timer_output_compare_complementary_channel_stop_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER complementary channel output compare mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_2	Channel 2
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_NO_SUPPORT, HAL_ERR_LOCK

Example:

```
/* stop TIMER complementary channel output compare mode and channel interrupt */
hal_timer_dev_struct timer0_info;

hal_timer_output_compare_complementary_channel_stop_interrupt(&timer0_info,
TIMER_CH_0);
```

hal_timer_output_compare_complementary_channel_start_dma

The description of hal_timer_output_compare_complementary_channel_start_dma is shown as below:

Table 3-528. Function

hal_timer_output_compare_complementary_channel_start_dma

Function name	hal_timer_output_compare_complementary_channel_start_dma
Function prototype	<pre>int32_t hal_timer_output_compare_complementary_channel_start_dma(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_dma_handle_cb_struct *dmacb, uint32_t *mem_addr, uint16_t dma_length)</pre>
Function descriptions	start TIMER complementary channel output compare mode and channel DMA request
Precondition	-
The called functions	hal_dma_start_interrupt
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
Input parameter{in}	
dmacb	pointer to hal_timer_dma_handle_cb_struct, struct members refer to Table 3-481. Structure hal timer dma handle cb struct
Input parameter{in}	
mem_addr	TIMER DMA transfer memory address
Input parameter{in}	
dma_length	TIMER DMA transfer count, 0~65535
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* start TIMER complementary channel output compare mode and channel DMA request */
void timer0_dmaerror(void *ptr);

void timer0_output_compare_dmatc_userdefine(void *ptr);
```

```

hal_timer_dev_struct timer0_info;

hal_timer_dma_handle_cb_struct timer_dma_handle_cb;

uint32_t timer_ch0val[3] = {0};

uint8_t timer0_dmaerror_num = 0;

uint8_t timer0_output_compare_dmatc_num = 0;

timer_dma_handle_cb.channelx_compare_dma_full_transcom_handle = &
timer0_output_compare_dmatc_userdefine;

timer_dma_handle_cb.error_handle = &timer0_dmaerror;

hal_timer_output_compare_complementary_channel_start_dma(&timer0_info,
TIMER_CH_0 ,&timer_dma_handle_cb, timer_ch0val, 3);

void timer0_dmaerror(void *ptr)
{
    timer0_dmaerror_num++;
}

void timer0_output_compare_dmatc_userdefine(void *ptr)
{
    timer0_output_compare_dmatc_num++;
}

```

hal_timer_output_compare_complementary_channel_stop_dma

The description of hal_timer_output_compare_complementary_channel_stop_dma is shown as below:

Table 3-529. Function

hal_timer_output_compare_complementary_channel_stop_dma

Function name	hal_timer_output_compare_complementary_channel_stop_dma
Function prototype	<pre>int32_t hal_timer_output_compare_complementary_channel_stop_dma(hal_timer_dev_struct *timer_dev, uint16_t channel)</pre>
Function descriptions	stop TIMER complementary channel output compare mode and channel DMA request
Precondition	-
The called functions	hal_dma_stop
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482.

	<u>Structure hal_timer_dev_struct</u>
Input parameter{in}	
channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_2</i>	Channel 2
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* stop TIMER complementary channel output compare mode and channel DMA request */
hal_timer_dev_struct timer0_info;

hal_timer_output_compare_complementary_channel_stop_dma(&timer0_info,
TIMER_CH_0);
```

hal_timer_single_pulse_mode_channel_config

The description of hal_timer_single_pulse_mode_channel_config is shown as below:

Table 3-530. Function hal_timer_single_pulse_mode_channel_config

Function name	hal_timer_single_pulse_mode_channel_config
Function prototype	int32_t hal_timer_single_pulse_mode_channel_config(hal_timer_dev_struct *timer_dev, hal_timer_single_pulse_struct *timer_singlepulse, uint16_t channel_out, uint16_t channel_in)
Function descriptions	TIMER single pulse mode configure
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to <u>Table 3-482. Structure hal_timer_dev_struct</u>
Input parameter{in}	
timer_singlepulse	pointer to hal_timer_single_pulse_struct, struct members refer to <u>Table 3-492. Structure hal_timer_single_pulse_struct</u>
Input parameter{in}	
channel_out	TIMER output channel.The channel will output single pulse
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
Input parameter{in}	
channel_in	TIMER input channel.If the channel input a active signal,TIMER will start

	count.
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```

/* TIMER single pulse mode configure */

hal_timer_dev_struct timer0_info;

hal_timer_single_pulse_struct timer0_singlepulse_parameter;

timer0_singlepulse_parameter.sp_compare_mode = TIMER_OC_MODE_TOGGLE;

timer0_singlepulse_parameter.sp_oc_pulse_value = 500;

timer0_singlepulse_parameter.sp_oc_polarity = TIMER_OC_POLARITY_HIGH;

timer0_singlepulse_parameter.sp_oc_idlestate = TIMER_OC_IDLE_STATE_LOW;

timer0_singlepulse_parameter.sp_ocn_polarity = TIMER_OCN_POLARITY_HIGH;

timer0_singlepulse_parameter.sp_ocn_idlestate = TIMER_OCN_IDLE_STATE_LOW;

timer0_singlepulse_parameter.sp_oc_fastmode = TIMER_OC_FAST_DISABLE;

timer0_singlepulse_parameter.sp_oc_clearmode = TIMER_OC_CLEAR_DISABLE;

timer0_singlepulse_parameter.sp_ic_polarity = TIMER_IC_POLARITY_RISING;

timer0_singlepulse_parameter.sp_ic_selection = TIMER_IC_SELECTION_DIRECTTTI;

timer0_singlepulse_parameter.sp_ic_filter= 15;

hal_timer_single_pulse_mode_channel_config(&timer0_info,
&timer0_singlepulse_parameter, TIMER_CH_0, TIMER_CH_1);

```

hal_timer_single_pulse_start

The description of hal_timer_single_pulse_start is shown as below:

Table 3-531. Function hal_timer_single_pulse_start

Function name	hal_timer_single_pulse_start
Function prototype	int32_t hal_timer_single_pulse_start(hal_timer_dev_struct *timer_dev)
Function descriptions	start TIMER single pulse mode
Precondition	-
The called functions	-

Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start TIMER single pulse mode */
hal_timer_dev_struct timer0_info;
hal_timer_single_pulse_start (&timer0_info);
```

hal_timer_single_pulse_stop

The description of hal_timer_single_pulse_stop is shown as below:

Table 3-532. Function hal_timer_single_pulse_stop

Function name	hal_timer_single_pulse_stop
Function prototype	int32_t hal_timer_single_pulse_stop(hal_timer_dev_struct *timer_dev)
Function descriptions	stop TIMER single pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_LOCK

Example:

```
/* stop TIMER single pulse mode */
hal_timer_dev_struct timer0_info;
hal_timer_single_pulse_stop(&timer0_info);
```

hal_timer_single_pulse_start_interrupt

The description of hal_timer_single_pulse_start_interrupt is shown as below:

Table 3-533. Function hal_timer_single_pulse_start_interrupt

Function name	hal_timer_single_pulse_start_interrupt
Function prototype	int32_t hal_timer_single_pulse_start_interrupt(hal_timer_dev_struct *timer_dev, hal_timer_irq_struct *p_irq)
Function descriptions	start TIMER single pulse mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal timer irq struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```

/* start TIMER single pulse mode and channel interrupt */

void timer0_input_capture_interrupt_handle_userdefine(void *ptr);

void timer0_output_compare_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

uint8_t input_capture_num = 0;

uint8_t output_compare_num = 0;

timer0_irq_parameter.channelx_capture_handle =
timer0_input_capture_interrupt_handle_userdefine;

timer0_irq_parameter.channelx_compare_handle =
timer0_output_compare_interrupt_handle_userdefine;

hal_timer_single_pulse_start_interrupt(&timer0_info, &timer0_irq_parameter);

void timer0_input_capture_interrupt_handle_userdefine(void *ptr)
{
    input_capture_num++;
}

```

```
void timer0_output_compare_interrupt_handle_userdefine(void *ptr)
{
    output_compare_num++;
}
```

hal_timer_single_pulse_stop_interrupt

The description of hal_timer_single_pulse_stop_interrupt is shown as below:

Table 3-534. Function hal_timer_single_pulse_stop_interrupt

Function name	hal_timer_single_pulse_stop_interrupt
Function prototype	int32_t hal_timer_single_pulse_stop_interrupt(hal_timer_dev_struct *timer_dev)
Function descriptions	stop TIMER single pulse mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS , HAL_ERR_NO_SUPPORT, HAL_ERR_LOCK

Example:

```
/* stop TIMER single pulse mode and channel interrupt */
hal_timer_dev_struct timer0_info;
hal_timer_single_pulse_stop_interrupt (&timer0_info);
```

hal_timer_single_pulse_complementary_channel_start

The description of hal_timer_single_pulse_complementary_channel_start is shown as below:

Table 3-535. Function hal_timer_single_pulse_complementary_channel_start

Function name	hal_timer_single_pulse_complementary_channel_start
Function prototype	int32_t int32_t hal_timer_single_pulse_complementary_channel_start(hal_timer_dev_struct *timer_dev, uint16_t channel_out)
Function descriptions	start TIMER complementary channel single pulse mode
Precondition	-

The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel_out	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* start TIMER complementary channel single pulse mode */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_single_pulse_complementary_channel_start(&timer0_info, TIMER_CH_0);
```

hal_timer_single_pulse_complementary_channel_stop

The description of hal_timer_single_pulse_complementary_channel_stop is shown as below:

Table 3-536. Function hal_timer_single_pulse_complementary_channel_stop

Function name	hal_timer_single_pulse_complementary_channel_stop
Function prototype	int32_t hal_timer_single_pulse_complementary_channel_stop(hal_timer_dev_struct *timer_dev, uint16_t channel_out)
Function descriptions	stop TIMER complementary channel single pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel_out	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS,

	HAL_ERR_NO_SUPPORT, HAL_ERR_VAL, HAL_ERR_LOCK
--	---

Example:

```
/* stop TIMER complementary channel single pulse mode */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_single_pulse_complementary_channel_stop(&timer0_info, TIMER_CH_0);
```

hal_timer_single_pulse_complementary_channel_start_interrupt

The description of hal_timer_single_pulse_complementary_channel_start_interrupt is shown as below:

Table 3-537. Function

hal_timer_single_pulse_complementary_channel_start_interrupt

Function name	hal_timer_single_pulse_complementary_channel_start_interrupt
Function prototype	int32_t hal_timer_single_pulse_complementary_channel_start_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel_out, hal_timer_irq_struct *p_irq)
Function descriptions	start TIMER complementary channel single pulse mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
channel_out	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal timer irq struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* start TIMER complementary channel single pulse mode and channel interrupt */
```

```
void timer0_input_capture_interrupt_handle_userdefine(void *ptr);
```

```

void timer0_output_compare_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

uint8_t input_capture_num = 0;

uint8_t output_compare_num = 0;

timer0_irq_parameter.channelx_capture_handle =
timer0_input_capture_interrupt_handle_userdefine;

timer0_irq_parameter.channelx_compare_handle =
timer0_output_compare_interrupt_handle_userdefine;

hal_timer_single_pulse_complementary_channel_start_interrupt(&timer0_info,
TIMER_CH_0, &timer0_irq_parameter);

void timer0_input_capture_interrupt_handle_userdefine(void *ptr)
{
    input_capture_num++;
}

void timer0_output_compare_interrupt_handle_userdefine(void *ptr)
{
    output_compare_num++;
}

```

hal_timer_single_pulse_complementary_channel_stop_interrupt

The description of hal_timer_single_pulse_complementary_channel_stop_interrupt is shown as below:

Table 3-538. Function

hal_timer_single_pulse_complementary_channel_stop_interrupt

Function name	hal_timer_single_pulse_complementary_channel_stop_interrupt
Function prototype	int32_t hal_timer_single_pulse_complementary_channel_stop_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel_out)
Function descriptions	stop TIMER complementary channel single pulse mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	

timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
channel_out	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* stop TIMER complementary channel single pulse mode and channel interrupt */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_single_pulse_complementary_channel_stop_interrupt(&timer0_info,  
TIMER_CH_0);
```

hal_timer_slave_mode_interrupt_config

The description of hal_timer_slave_mode_interrupt_config is shown as below:

Table 3-539. Function hal_timer_slave_mode_interrupt_config

Function name	hal_timer_slave_mode_interrupt_config
Function prototype	int32_t hal_timer_slave_mode_interrupt_config(hal_timer_dev_struct *timer_dev, hal_timer_slave_mode_struct *timer_slavemode, hal_timer_irq_struct *p_irq)
Function descriptions	configure TIMER slave mode and interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
timer_slavemode	pointer to hal_timer_slave_mode_struct, struct members refer to Table 3-489. Structure hal timer slave mode struct
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal timer irq struct
Output parameter{out}	
-	-
Return value	

int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL
----------------	---

Example:

```
/* configure TIMER slave mode and interrupt */

void timer0_trigger_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

hal_timer_slave_mode_struct timer0_slave_mode_parameter;

uint8_t trigger_num = 0;

timer0_irq_parameter.trigger_handle = timer0_trigger_interrupt_handle_userdefine;

timer0_slave_mode_parameter.slavemode = TIMER_SLAVE_MODE_RESTART_MODE;

timer0_slave_mode_parameter.trigger_selection = TIMER_TRIGGER_SOURCE_ETIFP;

timer0_slave_mode_parameter.trigger_polarity =
TIMER_CLOCK_TRIGGER_ETI_POLARITY_RISING;

timer0_slave_mode_parameter.trigger_prescaler = TIMER_EXT_TRI_PRESCALER_OFF;

timer0_slave_mode_parameter.trigger_filter = 0;

hal_timer_slave_mode_interrupt_config(&timer0_info,&timer0_slave_mode_parameter,
&timer0_irq_parameter);

void timer0_trigger_interrupt_handle_userdefine(void *ptr)

{

    trigger_num++;

}
```

hal_timer_decoder_start

The description of hal_timer_decoder_start is shown as below:

Table 3-540. Function hal_timer_decoder_start

Function name	hal_timer_decoder_start
Function prototype	int32_t hal_timer_decoder_start(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	start TIMER decoder mode
Precondition	-
The called functions	-

Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_0_1	TIMER channel0 and TIMER channel1
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* start TIMER decoder mode */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_decoder_start(&timer0_info, TIMER_CH_0);
```

hal_timer_decoder_stop

The description of hal_timer_decoder_stop is shown as below:

Table 3-541. Function hal_timer_decoder_stop

Function name	hal_timer_decoder_stop
Function prototype	int32_t hal_timer_decoder_stop(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER decoder mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_0_1	TIMER channel0 and TIMER channel1
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS,

	HAL_ERR_NO_SUPPORT, HAL_ERR_VAL, HAL_ERR_LOCK
--	---

Example:

```
/* stop TIMER decoder mode */
hal_timer_dev_struct timer0_info;
hal_timer_decoder_stop(&timer0_info, TIMER_CH_0);
```

hal_timer_decoder_start_interrupt

The description of hal_timer_decoder_start_interrupt is shown as below:

Table 3-542. Function hal_timer_decoder_start_interrupt

Function name	hal_timer_decoder_start_interrupt
Function prototype	int32_t hal_timer_decoder_start_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_irq_struct *p_irq)
Function descriptions	start TIMER decoder mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_0_1	TIMER channel0 and TIMER channel1
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal_timer_irq_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* start TIMER decoder mode and channel interrupt */
void timer0_input_capture_interrupt_handle_userdefine(void *ptr);
hal_timer_dev_struct timer0_info;
hal_timer_irq_struct timer0_irq_parameter;
```

```
uint8_t input_capture_num = 0;

timer0_irq_parameter.channelx_capture_handle =
timer0_input_capture_interrupt_handle_userdefine;

hal_timer_decoder_start_interrupt(&timer0_info, TIMER_CH_0, &timer0_irq_parameter);

void timer0_input_capture_interrupt_handle_userdefine(void *ptr)

{

    input_capture_num++;

}
```

hal_timer_decoder_stop_interrupt

The description of hal_timer_decoder_stop_interrupt is shown as below:

Table 3-543. Function hal_timer_decoder_stop_interrupt

Function name	hal_timer_decoder_stop_interrupt
Function prototype	int32_t hal_timer_decoder_stop_interrupt(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER decoder mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_0_1	TIMER channel0 and TIMER channel1
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* stop TIMER decoder mode and channel interrupt */

hal_timer_dev_struct timer0_info;

hal_timer_decoder_stop_interrupt (&timer0_info, TIMER_CH_0);
```

hal_timer_decoder_start_dma

The description of hal_timer_decoder_start_dma is shown as below:

Table 3-544. Function hal_timer_decoder_start_dma

Function name	hal_timer_decoder_start_dma
Function prototype	int32_t hal_timer_decoder_start_dma(hal_timer_dev_struct *timer_dev, uint16_t channel, hal_timer_dma_handle_cb_struct *dmacb, hal_timer_decoder_dma_config_struct *decoder_dma)
Function descriptions	start TIMER decoder mode and channel DMA request
Precondition	-
The called functions	hal_dma_start_interrupt
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
channel	specify which channel is selectd
TIMER_CH_0	Channel 0
TIMER_CH_1	Channel 1
TIMER_CH_0_1	TIMER channel0 and TIMER channel1
Input parameter{in}	
dmacb	pointer to hal_timer_dma_handle_cb_struct, struct members refer to Table 3-481. Structure hal timer dma handle cb struct
Input parameter{in}	
decoder_dma	ointer to hal_timer_decoder_dma_config_struct, struct members refer to Table 3-478. Structure hal timer decoder dma config struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```

/* start TIMER decoder mode and channel DMA request */

void timer0_dmaerror(void *ptr);

void timer0_input_capture_dmatc_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_dma_handle_cb_struct timer_dma_handle_cb;

hal_timer_decoder_dma_config_struct timer_decoder_dma_config;

uint32_t timer_ch0val[3] = {0};

```

```

uint32_t timer_ch1val[3] = {0};

uint8_t timer0_dmaerror_num = 0;

uint8_t timer0_input_capture_dmatc_num = 0;

timer_decoder_dma_config.mem_addr0 = timer_ch0val;
timer_decoder_dma_config.mem_addr1 = timer_ch1val;

timer_decoder_dma_config.length = 3;

timer_dma_handle_cb.channelx_capture_dma_full_transcom_handle = &
timer0_input_capture_dmatc_userdefine;

timer_dma_handle_cb.error_handle = &timer0_dmaerror;

hal_timer_decoder_start_dma(&timer0_info,  TIMER_CH_0_1,  &timer_dma_handle_cb,
&timer_decoder_dma_config);

void timer0_dmaerror(void *ptr)
{
    timer0_dmaerror_num++;
}

void timer0_input_capture_dmatc_userdefine(void *ptr)
{
    timer0_input_capture_dmatc_num++;
}

```

hal_timer_decoder_stop_dma

The description of hal_timer_decoder_stop_dma is shown as below:

Table 3-545. Function hal_timer_decoder_stop_dma

Function name	hal_timer_decoder_stop_dma
Function prototype	int32_t hal_timer_decoder_stop_dma(hal_timer_dev_struct *timer_dev, uint16_t channel)
Function descriptions	stop TIMER decoder mode and channel DMA request
Precondition	-
The called functions	hal_dma_stop
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	

channel	specify which channel is selectd
<i>TIMER_CH_0</i>	Channel 0
<i>TIMER_CH_1</i>	Channel 1
<i>TIMER_CH_0_1</i>	TIMER channel0 and TIMER channel1
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* stop TIMER decoder mode and channel DMA request */
hal_timer_dev_struct timer0_info;
hal_timer_decoder_stop_dma(&timer0_info, TIMER_CH_0_1);
```

hal_timer_hall_sensor_start

The description of hal_timer_hall_sensor_start is shown as below:

Table 3-546. Function hal_timer_hall_sensor_start

Function name	hal_timer_hall_sensor_start
Function prototype	int32_t hal_timer_hall_sensor_start(hal_timer_dev_struct *timer_dev)
Function descriptions	start TIMER hall sensor mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start TIMER hall sensor mode */
hal_timer_dev_struct timer0_info;
hal_timer_hall_sensor_start(&timer0_info);
```

hal_timer_hall_sensor_stop

The description of hal_timer_hall_sensor_stop is shown as below:

Table 3-547. Function hal_timer_hall_sensor_stop

Function name	hal_timer_hall_sensor_stop
Function prototype	int32_t hal_timer_hall_sensor_stop(hal_timer_dev_struct *timer_dev)
Function descriptions	stop TIMER hall sensor mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_LOCK

Example:

```
/* stop TIMER hall sensor mode */
hal_timer_dev_struct timer0_info;
hal_timer_hall_sensor_stop (&timer0_info);
```

hal_timer_hall_sensor_start_interrupt

The description of hal_timer_hall_sensor_start_interrupt is shown as below:

Table 3-548. Function hal_timer_hall_sensor_start_interrupt

Function name	hal_timer_hall_sensor_start_interrupt
Function prototype	int32_t hal_timer_hall_sensor_start_interrupt(hal_timer_dev_struct *timer_dev, hal_timer_irq_struct *p_irq)
Function descriptions	start TIMER hall sensor mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal_timer_irq_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start TIMER hall sensor mode and channel interrupt */

void timer0_input_capture_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

uint8_t input_capture_num = 0;

timer0_irq_parameter.channelx_capture_handle =
timer0_input_capture_interrupt_handle_userdefine;

hal_timer_hall_sensor_start_interrupt(&timer0_info, &timer0_irq_parameter);

void timer0_input_capture_interrupt_handle_userdefine(void *ptr)
{
    input_capture_num++;
}
```

hal_timer_hall_sensor_stop_interrupt

The description of hal_timer_hall_sensor_stop_interrupt is shown as below:

Table 3-549. Function hal_timer_hall_sensor_stop_interrupt

Function name	hal_timer_hall_sensor_stop_interrupt
Function prototype	int32_t hal_timer_hall_sensor_stop_interrupt(hal_timer_dev_struct *timer_dev)
Function descriptions	stop TIMER hall sensor mode and channel interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_LOCK

Example:

```
/* stop TIMER hall sensor mode and channel interrupt */

hal_timer_dev_struct timer0_info;
```



```
hal_timer_hall_sensor_stop_interrupt(&timer0_info);
```

hal_timer_hall_sensor_start_dma

The description of hal_timer_hall_sensor_start_dma is shown as below:

Table 3-550. Function hal_timer_hall_sensor_start_dma

Function name	hal_timer_hall_sensor_start_dma
Function prototype	int32_t hal_timer_hall_sensor_start_dma(hal_timer_dev_struct *timer_dev, hal_timer_dma_handle_cb_struct *dmacb, uint32_t *mem_addr, uint16_t dma_length)
Function descriptions	start TIMER hall sensor mode and channel DMA request
Precondition	-
The called functions	hal_dma_start_interrupt
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal timer dev struct
Input parameter{in}	
dmacb	pointer to hal_timer_dma_handle_cb_struct, struct members refer to Table 3-481. Structure hal timer dma handle cb struct
Input parameter{in}	
mem_addr	TIMER DMA transfer memory address
Input parameter{in}	
dma_length	TIMER DMA transfer count, 0~65535
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_LOCK

Example:

```
/* start TIMER hall sensor mode and channel DMA request */

void timer0_dmaerror(void *ptr);

void timer0_input_capture_dmatc_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_dma_handle_cb_struct timer_dma_handle_cb;

uint32_t timer_ch0val[3] = {0};

uint8_t timer0_dmaerror_num = 0;

uint8_t timer0_input_capture_dmatc_num = 0;

timer_dma_handle_cb.channelx_capture_dma_full_transcom_handle =
```

```

&timer0_input_capture_dmatc_userdefine;

timer_dma_handle_cb.error_handle = &timer0_dmaerror;

hal_timer_hall_sensor_start_dma(&timer0_info, &timer_dma_handle_cb, timer_ch0val, 3);

void timer0_dmaerror(void *ptr)
{
    timer0_dmaerror_num++;
}

void timer0_input_capture_dmatc_userdefine(void *ptr)
{
    timer0_input_capture_dmatc_num++;
}

```

hal_timer_hall_sensor_stop_dma

The description of hal_timer_hall_sensor_stop_dma is shown as below:

Table 3-551. Function hal_timer_hall_sensor_stop_dma

Function name	hal_timer_hall_sensor_stop_dma
Function prototype	int32_t hal_timer_hall_sensor_stop_dma(hal_timer_dev_struct *timer_dev)
Function descriptions	stop TIMER hall sensor mode and channel DMA request
Precondition	-
The called functions	hal_dma_stop
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_NO_SUPPORT, HAL_ERR_LOCK

Example:

```

/* stop TIMER hall sensor mode and channel DMA request */

hal_timer_dev_struct timer0_info;

hal_timer_hall_sensor_stop_dma(&timer0_info);

```

hal_timer_dma_transfer_write_start

The description of hal_timer_dma_transfer_write_start is shown as below:

Table 3-552. Function hal_timer_dma_transfer_write_start

Function name	hal_timer_dma_transfer_write_start
Function prototype	int32_t hal_timer_dma_transfer_write_start(hal_timer_dev_struct *timer_dev, uint32_t dmareq, hal_timer_dma_transfer_config_struct *dmatcfg, hal_timer_dma_handle_cb_struct *dmacb)
Function descriptions	start TIMER DMA transfer mode for writing data to TIMER
Precondition	-
The called functions	hal_dma_start_interrupt
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
dmareq	specify which DMA request is selectd
TIMER_DMA_UPD	update DMA request
TIMER_DMA_CH0D	channel 0 DMA request
TIMER_DMA_CH1D	channel 1 DMA request
TIMER_DMA_CH2D	channel 2 DMA request
TIMER_DMA_CH3D	channel 3 DMA request
TIMER_DMA_CMTD	commutation DMA request
TIMER_DMA_TRGD	trigger DMA request
Input parameter{in}	
dmatcfg	pointer to hal_timer_dma_transfer_config_struct, struct members refer to Table 3-479. Structure hal_timer_dma_transfer_config_struct
Input parameter{in}	
dmacb	pointer to hal_timer_dma_handle_cb_struct, struct members refer to Table 3-481. Structure hal_timer_dma_handle_cb_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* start TIMER DMA transfer mode for writing data to TIMER */
void timer0_dmaerror(void *ptr);
void timer0_update_dmatc_userdefine(void *ptr);
hal_timer_dev_struct timer0_info;
```

```

hal_timer_dma_handle_cb_struct timer_dma_handle_cb;

hal_timer_dma_transfer_config_struct timer_dma_transfer_config;

uint32_t timer_chval[3] = {0};

uint8_t timer0_dmaerror_num = 0;

uint8_t timer0_update_dmatc_num = 0;

timer_dma_transfer_config.start_addr = TIMER_DMA_START_ADDRESS_CH0CV;

timer_dma_transfer_config.mem_addr = timer_chval;

timer_dma_transfer_config.length= TIMER_DMACFG_DMATC_3TRANSFER;

timer_dma_handle_cb.update_dma_full_transcom_handle           =           &
timer0_update_dmatc_userdefine;

timer_dma_handle_cb.error_handle = &timer0_dmaerror;

hal_timer_dma_transfer_write_start(&timer0_info,               TIMER_DMA_UPD,
&timer_dma_transfer_config, &timer_dma_handle_cb);

void timer0_dmaerror(void *ptr)

{

    timer0_dmaerror_num++;

}

void timer0_update_dmatc_userdefine(void *ptr)

{

    timer0_update_dmatc_num++;

}

```

hal_timer_dma_transfer_write_stop

The description of hal_timer_dma_transfer_write_stop is shown as below:

Table 3-553. Function hal_timer_dma_transfer_write_stop

Function name	hal_timer_dma_transfer_write_stop
Function prototype	int32_t hal_timer_dma_transfer_write_stop(hal_timer_dev_struct *timer_dev, uint32_t dmareq)
Function descriptions	stop TIMER DMA transfer mode for writing data to TIMER
Precondition	-
The called functions	hal_dma_stop
Input parameter{in}	

timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
dmareq	specify which DMA request is selectd
<i>TIMER_DMA_UPD</i>	update DMA request
<i>TIMER_DMA_CH0D</i>	channel 0 DMA request
<i>TIMER_DMA_CH1D</i>	channel 1 DMA request
<i>TIMER_DMA_CH2D</i>	channel 2 DMA request
<i>TIMER_DMA_CH3D</i>	channel 3 DMA request
<i>TIMER_DMA_CMTD</i>	commutation DMA request
<i>TIMER_DMA_TRGD</i>	trigger DMA request
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* stop TIMER DMA transfer mode for writing data to TIMER */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_dma_transfer_write_stop(&timer0_info, TIMER_DMA_UPD);
```

hal_timer_dma_transfer_read_start

The description of hal_timer_dma_transfer_read_start is shown as below:

Table 3-554. Function hal_timer_dma_transfer_read_start

Function name	hal_timer_dma_transfer_read_start
Function prototype	int32_t hal_timer_dma_transfer_read_start(hal_timer_dev_struct *timer_dev, uint32_t dmareq, hal_timer_dma_transfer_config_struct *dmatcfg, hal_timer_dma_handle_cb_struct *dmacb)
Function descriptions	start TIMER DMA transfer mode for read data from TIMER
Precondition	-
The called functions	hal_dma_start_interrupt
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
dmareq	specify which DMA request is selectd
<i>TIMER_DMA_UPD</i>	update DMA request
<i>TIMER_DMA_CH0D</i>	channel 0 DMA request
<i>TIMER_DMA_CH1D</i>	channel 1 DMA request

<i>TIMER_DMA_CH2D</i>	channel 2 DMA request
<i>TIMER_DMA_CH3D</i>	channel 3 DMA request
<i>TIMER_DMA_CMTD</i>	commutation DMA request
<i>TIMER_DMA_TRGD</i>	trigger DMA request
Input parameter{in}	
dmacfg	pointer to <code>hal_timer_dma_transfer_config_struct</code> , struct members refer to Table 3-479. Structure <i>hal_timer_dma_transfer_config_struct</i>
Input parameter{in}	
dmacb	pointer to <code>hal_timer_dma_handle_cb_struct</code> , struct members refer to Table 3-481. Structure <i>hal_timer_dma_handle_cb_struct</i>
Output parameter{out}	
-	-
Return value	
int32_t	error code: <code>HAL_ERR_NONE</code> , <code>HAL_ERR_ADDRESS</code> , <code>HAL_ERR_VAL</code> , <code>HAL_ERR_LOCK</code>

Example:

```

/* start TIMER DMA transfer mode for read data from TIMER */

void timer0_dmaerror(void *ptr);

void timer0_input_capture_dmatc_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_dma_handle_cb_struct timer_dma_handle_cb;

hal_timer_dma_transfer_config_struct timer_dma_transfer_config;

uint32_t timer_chval[3] = {0};

uint8_t timer0_dmaerror_num = 0;

uint8_t timer0_input_capture_dmatc_num = 0;

timer_dma_transfer_config.start_addr = TIMER_DMA_START_ADDRESS_CH0CV;

timer_dma_transfer_config.mem_addr = timer_chval;

timer_dma_transfer_config.length= TIMER_DMACFG_DMATC_3TRANSFER;

timer_dma_handle_cb.channelx_capture_dma_full_transcom_handle      =      &
timer0_input_capture_dmatc_userdefine;

timer_dma_handle_cb.error_handle = &timer0_dmaerror;

hal_timer_dma_transfer_read_start(&timer0_info,                      TIMER_DMA_CH0D,
&timer_dma_transfer_config, &timer_dma_handle_cb);

void timer0_dmaerror(void *ptr)

```

```

{
    timer0_dmaerror_num++;
}

void timer0_input_capture_dmatc_userdefine(void *ptr)
{
    timer0_input_capture_dmatc_num++;
}

```

hal_timer_dma_transfer_read_stop

The description of hal_timer_dma_transfer_read_stop is shown as below:

Table 3-555. Function hal_timer_dma_transfer_read_stop

Function name	hal_timer_dma_transfer_read_stop
Function prototype	int32_t hal_timer_dma_transfer_read_stop(hal_timer_dev_struct *timer_dev, uint32_t dmareq)
Function descriptions	stop TIMER DMA transfer mode for read data from TIMER
Precondition	-
The called functions	hal_dma_stop
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
dmareq	specify which DMA request is selectd
TIMER_DMA_UPD	update DMA request
TIMER_DMA_CH0D	channel 0 DMA request
TIMER_DMA_CH1D	channel 1 DMA request
TIMER_DMA_CH2D	channel 2 DMA request
TIMER_DMA_CH3D	channel 3 DMA request
TIMER_DMA_CMTD	commutation DMA request
TIMER_DMA_TRGD	trigger DMA request
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL, HAL_ERR_LOCK

Example:

```
/* stop TIMER DMA transfer mode for read data from TIMER */
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_dma_transfer_read_stop(&timer0_info, TIMER_DMA_CH0D);
```

hal_timer_commutation_event_config

The description of hal_timer_commutation_event_config is shown as below:

Table 3-556. Function hal_timer_commutation_event_config

Function name	hal_timer_commutation_event_config
Function prototype	int32_t hal_timer_commutation_event_config(hal_timer_dev_struct *timer_dev, uint32_t trigger_source, uint32_t com_source)
Function descriptions	configure TIMER commutation event
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
trigger_source	trigger source to select
TIMER_TRIGGER_SOURCE_ITI0	trigger source: ITI0
TIMER_TRIGGER_SOURCE_ITI1	trigger source: ITI1
TIMER_TRIGGER_SOURCE_ITI2	trigger source: ITI2
TIMER_TRIGGER_SOURCE_ITI3	trigger source: ITI3
TIMER_TRIGGER_SOURCE_DISABLE	trigger source: none
Input parameter{in}	
com_source	commutation control shadow register update control
TIMER_UPDATECTL_CCUC	the shadow registers are updated when CMTG bit is set
TIMER_UPDATECTL_CCUTRI	the shadow registers are updated when CMTG bit is set or an rising edge of TRGI occurs
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* configure TIMER commutation event */
```



```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_commutation_event_config(&timer0_info,
TIMER_TRIGGER_SOURCE_DISABLE, TIMER_UPDATECTL_CCU);
```

hal_timer_commutation_event_interrupt_config

The description of hal_timer_commutation_event_interrupt_config is shown as below:

Table 3-557. Function hal_timer_commutation_event_interrupt_config

Function name	hal_timer_commutation_event_interrupt_config
Function prototype	<pre>int32_t hal_timer_commutation_event_interrupt_config(hal_timer_dev_struct *timer_dev, uint32_t trigger_source, uint32_t com_source, hal_timer_irq_struct *p_irq)</pre>
Function descriptions	configure TIMER commutation event and enable CMT interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
trigger_source	trigger source to select
TIMER_TRIGGER_SOURCE_ITI0	trigger source: ITI0
TIMER_TRIGGER_SOURCE_ITI1	trigger source: ITI1
TIMER_TRIGGER_SOURCE_ITI2	trigger source: ITI2
TIMER_TRIGGER_SOURCE_ITI3	trigger source: ITI3
TIMER_TRIGGER_SOURCE_DISABLE	trigger source: none
Input parameter{in}	
com_source	commutation control shadow register update control
TIMER_UPDATECTL_CCU	the shadow registers are updated when CMTG bit is set
TIMER_UPDATECTL_CCUTRI	the shadow registers are updated when CMTG bit is set or an rising edge of TRGI occurs
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal_timer_irq_struct
Output parameter{out}	

-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```

/* configureTIMER commutation event and enable CMT interrupt */

void timer0_commutation_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

uint8_t commutation_num = 0;

timer0_irq_parameter.commutation_handle =
timer0_commutation_interrupt_handle_userdefine;

hal_timer_commutation_event_interrupt_config(&timer0_info,
TIMER_TRIGGER_SOURCE_DISABLE,                TIMER_UPDATECTL_CCU,
&timer0_irq_parameter);

void timer0_commutation_interrupt_handle_userdefine(void *ptr)
{
    commutation_num++;
}

```

hal_timer_commutation_event_dma_config

The description of hal_timer_commutation_event_dma_config is shown as below:

Table 3-558. Function hal_timer_commutation_event_dma_config

Function name	hal_timer_commutation_event_dma_config
Function prototype	int32_t hal_timer_commutation_event_dma_config(hal_timer_dev_struct *timer_dev, uint32_t trigger_source, uint32_t com_source, hal_timer_dma_handle_cb_struct *dmacb)
Function descriptions	Configure TIMER commutation event and enable CMT DMA request
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
trigger_source	trigger source to select
TIMER_TRIGGER_SO	trigger source: ITIO

<i>URCE_ITI0</i>	
<i>TIMER_TRIGGER_SO</i> <i>URCE_ITI1</i>	trigger source: ITI1
<i>TIMER_TRIGGER_SO</i> <i>URCE_ITI2</i>	trigger source: ITI2
<i>TIMER_TRIGGER_SO</i> <i>URCE_ITI3</i>	trigger source: ITI3
<i>TIMER_TRIGGER_SO</i> <i>URCE_DISABLE</i>	trigger source: none
Input parameter{in}	
com_source	commutation control shadow register update control
<i>TIMER_UPDATECTL_</i> <i>CCU</i>	the shadow registers are updated when CMTG bit is set
<i>TIMER_UPDATECTL_</i> <i>CCUTRI</i>	the shadow registers are updated when CMTG bit is set or an rising edge of TRGI occurs
Input parameter{in}	
dmacb	pointer to <code>hal_timer_dma_handle_cb_struct</code> , struct members refer to Table 3-481. Structure <i>hal_timer_dma_handle_cb_struct</i>
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* configure TIMER commutation event and enable CMT DMA request */
```

```
void timer0_dmaerror(void *ptr);
```

```
void timer0_commutation_dmatc_userdefine(void *ptr);
```

```
hal_timer_dev_struct timer0_info;
```

```
hal_timer_dma_handle_cb_struct timer_dma_handle_cb;
```

```
uint8_t timer0_dmaerror_num = 0;
```

```
uint8_t timer0_commutation_dmatc_num = 0;
```

```
timer_dma_handle_cb.commutation_dma_full_transcom_handle
```

```
&timer0_commutation_dmatc_userdefine;
```

```
timer_dma_handle_cb.error_handle = &timer0_dmaerror;
```

```
hal_timer_commutation_event_dma_config(&timer0_info,
```

```
TIMER_TRIGGER_SOURCE_DISABLE,
```

```
&timer_dma_handle_cb);
```

```
TIMER_UPDATECTL_CCU,
```

```
void timer0_dmaerror(void *ptr)
{
    timer0_dmaerror_num++;
}

void timer0_commutation_dmatc_userdefine(void *ptr)
{
    timer0_commutation_dmatc_num++;
}
```

hal_timer_irq_handle_set

The description of hal_timer_irq_handle_set is shown as below:

Table 3-559. Function hal_timer_irq_handle_set

Function name	hal_timer_irq_handle_set
Function prototype	int32_t hal_timer_irq_handle_set(hal_timer_dev_struct *timer_dev, hal_timer_irq_struct *p_irq)
Function descriptions	set user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Input parameter{in}	
p_irq	pointer to hal_timer_irq_struct, struct members refer to Table 3-480. Structure hal_timer_irq_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* set user-defined interrupt callback function */

void timer0_input_capture_interrupt_handle_userdefine(void *ptr);

hal_timer_dev_struct timer0_info;

hal_timer_irq_struct timer0_irq_parameter;

uint8_t input_capture_num = 0;
```

```

timer0_irq_parameter.channelx_capture_handle =
timer0_input_capture_interrupt_handle_userdefine;

hal_timer_irq_handle_set(&timer0_info, &timer0_irq_parameter);

void timer0_input_capture_interrupt_handle_userdefine(void *ptr)
{
    input_capture_num++;
}

```

hal_timer_irq_handle_all_reset

The description of hal_timer_irq_handle_all_reset is shown as below:

Table 3-560. Function hal_timer_irq_handle_all_reset

Function name	hal_timer_irq_handle_all_reset
Function prototype	int32_t hal_timer_irq_handle_all_reset(hal_timer_dev_struct *timer_dev)
Function descriptions	reset all user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	error code: HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```

/* reset all user-defined interrupt callback function */

hal_timer_dev_struct timer0_info;

hal_timer_irq_handle_all_reset(&timer0_info);

```

hal_timer_irq

The description of hal_timer_irq is shown as below:

Table 3-561. Function hal_timer_irq

Function name	hal_timer_irq
Function prototype	void hal_timer_irq(hal_timer_dev_struct *timer_dev)
Function descriptions	TIMER interrupt handler content function, which is merely used in timer_handler

Precondition	-
The called functions	-
Input parameter{in}	
timer_dev	pointer to hal_timer_dev_struct, struct members refer to Table 3-482. Structure hal_timer_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* TIMER interrupt handler content function, which is merely used in timer_handler */
hal_timer_dev_struct timer0_info;
hal_timer_irq (&timer0_info);
```

3.23. TSI

Touch Sensing Interface (TSI) provides a convenient solution for touch keys, sliders and capacitive proximity sensing applications. The TSI registers are listed in chapter [3.23.1](#), the TSI firmware functions are introduced in chapter [3.23.2](#).

3.23.1. Descriptions of Peripheral registers

TSI registers are listed in the table shown as below:

Table 3-562. TSI Registers

Registers	Descriptions
TSI_CTL0	Control register 0
TSI_INTEN	Interrupt enable register
TSI_INTC	Interrupt flag clear register
TSI_INTF	Interrupt flag register
TSI_PHM	Pin hysteresis mode register
TSI_ASW	Analog switch register
TSI_SAMPCFG	Sample configuration register
TSI_CHCFG	Channel configuration register
TSI_GCTL	Group control register
TSI_GxCYCN (x= 0..5)	Group x cycle number registers
TSI_CTL1	Control register 1

3.23.2. Descriptions of Peripheral functions

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

Table 3-563. TSI firmware function

Function name	Function description
hal_tsi_init	initialize TSI
hal_tsi_struct_init	initialize the TSI structure with the default values
hal_tsi_deinit	deinitialize TSI device structure and init structure
hal_tsi_start	start TSI module function
hal_tsi_stop	stop tsi module function
hal_tsi_start_interrupt	start TSI interrupt
hal_tsi_stop_interrupt	stop TSI interrupt
hal_tsi_irq	TSI interrupt handler content function, which is merely used in tsi_handler
hal_tsi_irq_handle_set	set user-defined interrupt callback function
hal_tsi_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_tsi_group_cycle_get	get the cycle number of specific group as soon as a charge-transfer sequence completes
hal_tsi_pins_config	configure TSI pins
hal_tsi_poll_transfer	TSI poll for charge-transfer sequence complete

Enum hal_tsi_state_enum

Table 3-564. Enum hal_tsi_state_enum

Member name	Function description
HAL_TSI_STATE_NONE	NONE(default value)
HAL_TSI_STATE_RESET	RESET
HAL_TSI_STATE_BUSY	BUSY
HAL_TSI_STATE_TIMEOUT	TIMEOUT
HAL_TSI_STATE_ERROR	ERROR
HAL_TSI_STATE_READY	READY

Enum hal_tsi_error_enum

Table 3-565. Enum hal_tsi_error_enum

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

HAL_TSI_ERROR_NONE	no error
HAL_TSI_ERROR_SYSTEM	TSI internal error
HAL_TSI_ERROR_MNERR	TSI max cycle number error
HAL_TSI_ERROR_CONFIG	configuration error occurs

Enum hal_tsi_struct_type_enum

Table 3-566. Enum hal_tsi_struct_type_enum

Member name	Function description
HAL_TSI_INIT_STRUCTURE	TSI initialization structure
HAL_TSI_IRQ_STRUCTURE	TSI irq structure
HAL_TSI_DEV_STRUCTURE	TSI device structure

Structure hal_tsi_irq_struct

Table 3-567. Structure hal_tsi_irq_struct

Member name	Function description
tsi_cctcf_handle	TSI charge-transfer complete handler function
tsi_mnerr_handle	TSI max cycle number error handler function

Structure hal_tsi_dev_struct

Table 3-568. Structure hal_tsi_dev_struct

Member name	Function description
tsi_irq	TSI device interrupt callback function pointer structure
error_state	TSI error state
state	TSI state
mutex	mutex
priv	priv data

Structure hal_tsi_init_struct

Table 3-569. hal_tsi_init_struct

Member name	Function description
charge_time	charge state duration time
transfer_time	charge transfer state duration time

ctclk_div	charge transfer clock(ctclk) division factor
seq_max	max cycle number
extend_charge_state	extend charge state
ecclk_div	extend charge clock(ecclk) division factor
extend_charge_time	extend charge state maximum duration time
pin_mode	pin mode
edge_sel	edge selection
trig_mode	trigger mode selection
sample_pins	sample pins
channel_pins	channel pins
shield_pins	shield pins

hal_tsi_init

The description of hal_tsi_init is shown as below:

Table 3-570. Function hal_tsi_init

Function name	hal_tsi_init
Function prototype	int32_t hal_tsi_init(hal_tsi_dev_struct *tsi_dev, hal_tsi_init_struct *tsi_init);
Function descriptions	initialize TSI
Precondition	-
The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Input parameter{in}	
tsi_init	points to hal_tsi_init_struct, struct members refer to Table 3-569. Structure hal_tsi_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```

/* initialize TSI */

hal_tsi_init_struct tsi_init_parameter;

hal_tsi_dev_struct tsi_info;

tsi_init_parameter.charge_time = TSI_CHARGE_2CTCLK;

tsi_init_parameter.transfer_time = TSI_TRANSFER_2CTCLK;

```

```
tsi_init_parameter.ctclk_div = TSI_CTCDIV_DIV32;

tsi_init_parameter.seq_max = TSI_MAXNUM2047;

tsi_init_parameter.extend_charge_state = TSI_EXTEND_CHARGE_DISABLE;

tsi_init_parameter.pin_mode = TSI_OUTPUT_LOW;

tsi_init_parameter.trig_mode = TSI_HW_TRIGGER_DISABLE;

tsi_init_parameter.sample_pins = TSI_G5P0;

tsi_init_parameter.channel_pins = TSI_G5P1 | TSI_G5P2 | TSI_G5P3;

hal_tsi_init(&tsi_info, &tsi_init_parameter);
```

hal_tsi_struct_init

The description of hal_tsi_struct_init is shown as below:

Table 3-571. Function hal_tsi_struct_init

Function name	hal_tsi_struct_init
Function prototype	void hal_tsi_struct_init(hal_tsi_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the TSI structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	points to hal_tsi_dev_struct, struct members refer to Table 3-566. Enum hal_tsi_struct_type_enum
Input parameter{in}	
p_struct	points to TSI structure that contains the configuration information
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TSI initialization structure */

hal_tsi_init_struct tsi_init_parameter;

hal_tsi_struct_init(HAL_TSI_INIT_STRUCT, &tsi_init_parameter);
```

hal_tsi_deinit

The description of hal_tsi_deinit is shown as below:

Table 3-572. Function hal_tsi_deinit

Function name	hal_tsi_deinit
Function prototype	int32_t hal_tsi_deinit(hal_tsi_dev_struct *tsi_dev);
Function descriptions	deinitialize TSI device structure and init structure
Precondition	-
The called functions	hal_rcu_periph_reset_enable / hal_rcu_periph_reset_disable
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* deinitialize TSI device structure and init structure */
```

```
hal_tsi_dev_struct tsi_info;
```

```
hal_tsi_deinit(&tsi_info);
```

hal_tsi_start

The description of hal_tsi_start is shown as below:

Table 3-573. Function hal_tsi_start

Function name	hal_tsi_start
Function prototype	int32_t hal_tsi_start(hal_tsi_dev_struct *tsi_dev);
Function descriptions	start TSI module function
Precondition	-
The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* start TSI module function */
```

```
hal_tsi_dev_struct tsi_info;
```

```
hal_tsi_start(&tsi_info);
```

hal_tsi_stop

The description of hal_tsi_stop is shown as below:

Table 3-574. Function hal_tsi_stop

Function name	hal_tsi_stop
Function prototype	int32_t hal_tsi_stop(hal_tsi_dev_struct *tsi_dev);
Function descriptions	stop TSI module function
Precondition	-
The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* stop tsi module function */
```

```
hal_tsi_dev_struct tsi_info;
```

```
hal_tsi_stop(&tsi_info);
```

hal_tsi_start_interrupt

The description of hal_tsi_start_interrupt is shown as below:

Table 3-575. Function hal_tsi_start_interrupt

Function name	hal_tsi_start_interrupt
Function prototype	int32_t hal_tsi_start_interrupt(hal_tsi_dev_struct *tsi_dev, hal_tsi_irq_struct *p_irq);
Function descriptions	start TSI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Input parameter{in}	
p_irq	points to hal_tsi_irq_struct, struct members refer to Table 3-567. Structure hal_tsi_irq_struct
Output parameter{out}	
-	-

Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* start TSI interrupt */
hal_tsi_dev_struct tsi_info;
hal_tsi_irq_struct tsi_irq;
hal_tsi_start_interrupt(&tsi_info, &tsi_irq);
```

hal_tsi_stop_interrupt

The description of hal_tsi_stop_interrupt is shown as below:

Table 3-576. Function hal_tsi_stop_interrupt

Function name	hal_tsi_stop_interrupt
Function prototype	int32_t hal_tsi_stop_interrupt(hal_tsi_dev_struct *tsi_dev);
Function descriptions	stop TSI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3 231. Structure hal_tsi_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_ADDRESS, HAL_ERR_NONE

Example:

```
/* stop TSI interrupt */
hal_tsi_dev_struct tsi_info;
hal_tsi_stop_interrupt(&tsi_info);
```

hal_tsi_irq

The description of hal_tsi_irq is shown as below:

Table 3-577. Function hal_tsi_irq

Function name	hal_tsi_irq
Function prototype	void hal_tsi_irq(hal_tsi_dev_struct *tsi_dev);
Function descriptions	TSI interrupt handler content function, which is merely used in tsi_handler
Precondition	-

The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the function is used in the relative interrupt routine */
```

```
hal_tsi_dev_struct tsi_info;
```

```
void TSI_IRQHandler(void)
```

```
{
```

```
    hal_tsi_irq(&tsi_info);
```

```
}
```

hal_tsi_irq_handle_set

The description of hal_tsi_irq_handle_set is shown as below:

Table 3-578. Function hal_tsi_irq_handle_set

Function name	hal_tsi_irq_handle_set
Function prototype	void hal_tsi_irq_handle_set(hal_tsi_dev_struct *tsi_dev, hal_tsi_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Input parameter{in}	
p_irq	points to hal_tsi_irq_struct, struct members refer to Table 3-567. Structure hal_tsi_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* set TSI interrupt */

hal_tsi_dev_struct tsi_info;

hal_tsi_irq_struct tsi_irq;

hal_tsi_irq_handle_set(&tsi_info, &tsi_irq);

```

hal_tsi_irq_handle_all_reset

The description of hal_tsi_irq_handle_all_reset is shown as below:

Table 3-579. Function hal_tsi_irq_handle_all_reset

Function name	hal_tsi_irq_handle_all_reset
Function prototype	void hal_tsi_irq_handle_all_reset(hal_tsi_dev_struct *tsi_dev);
Function descriptions	reset all user-defined interrupt callback function
Precondition	-
The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* reset all user-defined interrupt callback function */

hal_tsi_dev_struct tsi_info;

hal_tsi_irq_handle_all_reset (&tsi_info);

```

hal_tsi_group_cycle_get

The description of hal_tsi_group_cycle_get is shown as below:

Table 3-580. Function hal_tsi_group_cycle_get

Function name	hal_tsi_group_cycle_get
Function prototype	uint16_t hal_tsi_group_cycle_get(hal_tsi_dev_struct *tsi_dev, uint8_t group_id);
Function descriptions	get the cycle number of specific group as soon as a charge-transfer sequence completes
Precondition	-
The called functions	-
Input parameter{in}	

tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Input parameter{in}	
group_id	TSI group ID
<i>TSI_GROUP_IDX0</i>	TSI group0
<i>TSI_GROUP_IDX1</i>	TSI group1
<i>TSI_GROUP_IDX2</i>	TSI group2
<i>TSI_GROUP_IDX3</i>	TSI group3
<i>TSI_GROUP_IDX4</i>	TSI group4
<i>TSI_GROUP_IDX5</i>	TSI group5
Output parameter{out}	
-	-
Return value	
uint16_t	0x0 – 0x3FFF

Example:

```
/* get group0 cycle value */
hal_tsi_dev_struct tsi_info;
uint16_t sample_value
hal_tsi_group_cycle_get(&tsi_info, TSI_GROUP_IDX0);
```

hal_tsi_pins_config

The description of hal_tsi_pins_config is shown as below:

Table 3-581. Function hal_tsi_pins_config

Function name	hal_tsi_pins_config
Function prototype	void hal_tsi_pins_config(hal_tsi_dev_struct *tsi_dev, hal_tsi_init_struct *pins_config);
Function descriptions	configure TSI pins
Precondition	-
The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3-568. Structure hal_tsi_dev_struct
Input parameter{in}	
pins_config	points to hal_tsi_init_struct, struct members refer to Table 3-569. Structure hal_tsi_init_struct
Output parameter{out}	
-	-
Return value	

Example:

```
/* configure TSI pins */

hal_tsi_init_struct tsi_init_parameter;

hal_tsi_dev_struct tsi_info;

tsi_init_parameter.sample_pins = TSI_G4P0;

tsi_init_parameter.channel_pins = TSI_G4P1 | TSI_G4P2 | TSI_G4P3;

hal_tsi_pins_config(&tsi_info, &tsi_init_parameter);
```

hal_tsi_poll_transfer

The description of hal_tsi_poll_transfer is shown as below:

Table 3-582. Function hal_tsi_poll_transfer

Function name	hal_tsi_poll_transfer
Function prototype	void hal_tsi_poll_transfer(hal_tsi_dev_struct *tsi_dev);
Function descriptions	TSI poll for charge-transfer sequence complete
Precondition	-
The called functions	-
Input parameter{in}	
tsi_dev	points to hal_tsi_dev_struct, struct members refer to Table 3 231. Structure hal_tsi_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* TSI poll for charge-transfer sequence complete */

hal_tsi_dev_struct tsi_info;

hal_tsi_poll_transfer(&tsi_info);
```

3.24. UART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter [3.24.1](#), the UART firmware functions are introduced in chapter [3.24.2](#).

3.24.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-583. USART Registers

Registers	Descriptions
USART_CTL0	Control register 0
USART_CTL1	Control register 1
USART_CTL2	Control register 2
USART_BAUD	Baud rate register
USART_GP	Guard time and prescaler register
USART_RT	Receiver timeout register
USART_CMD	Command register
USART_STAT	Status register
USART_INTC	Status clear register
USART_RDATA	Receive data register
USART_TDATA	Transmit data register
USART_RFCS	Receive FIFO control and status register

3.24.2. Descriptions of Peripheral functions

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

Table 3-584. UART firmware function

Function name	Function description
hal_uart_struct_init	initialize the UART struct with the default values, note that this function must be called after the struct is created
hal_uart_deinit	deinitialize the UART
hal_uart_init	initialize the UART with specified values
hal_uart_irq	handle the UART interrupts
hal_uart_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_uart_irq_handle_all_reset	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_uart_transmit_poll	transmit amounts of data, poll transmit process and completed status

Function name	Function description
hal_uart_receive_poll	receive amounts of data, poll receive process and completed status
hal_uart_transmit_interrupt	transmit amounts of data by interrupt method, after the transfer is completed, the user function is called
hal_uart_receive_interrupt	receive amounts of data by interrupt method, after the transfer is completed, the user function is called
hal_uart_transmit_dma	transmit amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
hal_uart_receive_dma	receive amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
hal_uart_dma_pause	pause UART DMA transfer during transmission process
hal_uart_dma_resume	resume UART DMA transfer during transmission process
hal_uart_transmit_stop	stop UART transmit transfer
hal_uart_receive_stop	stop UART receive transfer

Enum hal_uart_struct_type_enum

Table 3-585. Enum hal_uart_struct_type_enum

Member name	Function description
HAL_UART_INIT_STRUCT	UART initialization structure
HAL_UART_DEV_STRUCT	UART DMA callback function pointer structure
HAL_UART_USER_CALLBACK_STRUCT	UART device interrupt callback function pointer structure
HAL_UART_IRQ_INIT_STRUCT	UART device information structrue

Enum hal_uart_work_mode_enum

Table 3-586. Enum hal_uart_work_mode_enum

Member name	Function description
UART_WORK_MODE_ASYNC	asynchronous communication mode
UART_WORK_MODE_SINGLE_WIRE	single wire(half-duplex) communication mode
UART_WORK_MODE_MULTIPROCESSC OR	multiprocessor communication mode
UART_WORK_MODE_LIN	LIN mode

Structure hal_uart_init_struct

Table 3-587. Structure hal_uart_init_struct

Member name	Function description
work_mode	work mode
baudrate	communication baudrate
parity	parity mode
word_length	number of data bits in a frame
stop_bit	number of stop bits
direction	communication transfer direction
over_sample	oversample mode
sample_method	one sample bit method
hardware_flow	hardware flow control
rx_fifo_en	receive FIFO enable
timeout_enable	receive timeout enable
timeout_value	the receiver timeout value
first_bit_msb	MSB is sent first on communication line
tx_rx_swap	Tx and Rx pins are swapped
rx_level_invert	the Rx pin active level is inverted
tx_level_invert	the Tx pin active level is inverted
data_bit_invert	data is inverted
overrun_disable	the reception overrun detection is disabled
rx_error_dma_stop	the DMA is disabled in case of reception error
break_frame_length	LIN break frame length
rs485_mode	rs485 mode
de_polarity	driver enable polarity
de_assertion_time	driver enable assertion time
de_deassertion_time	driver enable deassertion time
wakeup_mode	wakeup mode

Member name	Function description
address	wakeup address
addr_length	address length

Structure hal_uart_irq_struct

Table 3-588. Structure hal_uart_irq_struct

Member name	Function description
receive_complete_handle	receive complete callback function
receive_timeout_handle	receive timeout callback function
transmit_ready_handle	transmit ready callback function
transmit_complete_handle	transmit complete callback function
error_handle	error callback function
wakeup_handle	wakeup callback function
idle_line_detected_handle	idle line detected callback function
address_match_handle	address match callback function
lin_break_detected_handle	LIN break detected callback function
cts_change_handle	CTS change callback function

Enum hal_uart_state_enum

Table 3-589. Enum hal_uart_state_enum

Member name	Function description
UART_STATE_FREE	UART is free
UART_STATE_BUSY	UART is busy

Structure hal_uart_dev_struct

Table 3-590. Structure hal_uart_dev_struct

Member name	Function description
periph	usart port
uart_irq	device interrupt callback function pointer, refer to Table 3-588. Structure hal_uart_irq_struct
p_dma_rx	DMA receive pointer

Member name	Function description
p_dma_tx	DMA transmit pointer
txbuffer	transmit buffer
rxbuffer	receive buffer
work_mode	work mode
data_bit_mask	mask bit of data
last_error	the last error code
error_state	error state
tx_state	transmit state
rx_state	receive state
rx_callback	receive callback function pointer
tx_callback	transmit callback function pointer
mutex	mutex locked and unlocked state
priv	private pointer

Structure hal_uart_user_callback_struct

Table 3-591. Structure hal_uart_user_callback_struct

Member name	Function description
complete_func	transfer complete callback function
error_func	error callback function

hal_uart_struct_init

The description of hal_uart_struct_init is shown as below:

Table 3-592. Function hal_uart_struct_init

Function name	hal_uart_struct_init
Function prototype	void hal_uart_struct_init(hal_uart_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the UART struct with the default values, note that this function must be called after the struct is created
Precondition	-

The called functions	-
Input parameter{in}	
hal_struct_type	UART struct type, refer to Table 3-587. Structure hal_uart_init_struct
Input parameter{in}	
p_struct	a void pointer to structure
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_uart_irq_struct uart_irq;
```

```
/* initialize a uart irq structure */
```

```
hal_uart_struct_init(HAL_UART_IRQ_INIT_STRUCT, &uart_irq);
```

hal_uart_deinit

The description of hal_uart_deinit is shown as below:

Table 3-593. Function hal_uart_deinit

Function name	hal_uart_deinit
Function prototype	void hal_uart_deinit(hal_uart_dev_struct *uart);
Function descriptions	deinitialize the UART
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device structure, refer to Table 3-590. Structure hal_uart_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_uart_dev_struct uart0_info;

/* deinitialize the device information structure */

hal_uart_deinit(&uart0_info);
```

hal_uart_init

The description of hal_uart_init is shown as below:

Table 3-594. Function hal_uart_init

Function name	hal_uart_init
Function prototype	int32_t hal_uart_init(hal_uart_dev_struct *uart, uint32_t periph, hal_uart_init_struct *p_init);
Function descriptions	initialize the UART with specified values
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Input parameter{in}	
periph	specify which USART is initialized
USARTx	x=0,1
Input parameter{in}	
p_init	the initialization data needed to initialize UART, refer to Table 3-587. Structure hal_uart_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```
hal_uart_dev_struct uart0_info;

hal_uart_init_struct uart0_init;
```



```

/* initialize the structures */

hal_uart_struct_init(HAL_UART_DEV_STRUCT, &uart0_info);

hal_uart_struct_init(HAL_UART_INIT_STRUCT, &uart0_init);

uart0_init.work_mode = UART_WORK_MODE_ASYNC;

uart0_init.baudrate = 115200;

uart0_init.parity = UART_PARITY_NONE;

uart0_init.word_length = UART_WORD_LENGTH_8BIT;

uart0_init.stop_bit = UART_STOP_BIT_1;

uart0_init.direction = UART_DIRECTION_RX_TX;

uart0_init.over_sample = UART_OVER_SAMPLE_16;

uart0_init.hardware_flow = UART_HARDWARE_FLOW_NONE;

uart0_init.sample_method = UART_THREE_SAMPLE_BIT;

hal_uart_init(&uart0_info, USART0, &uart0_init);

```

hal_uart_irq

The description of hal_uart_irq is shown as below:

Table 3-595. Function hal_uart_irq

Function name	hal_uart_irq
Function prototype	void hal_uart_irq(hal_uart_dev_struct *uart);
Function descriptions	handle the UART interrupts
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* process the UART interrupt */
hal_uart_irq(&uart0_info);
```

hal_uart_irq_handle_set

The description of hal_uart_irq_handle_set is shown as below:

Table 3-596. Function hal_uart_irq_handle_set

Function name	hal_uart_irq_handle_set
Function prototype	void hal_uart_irq_handle_set(hal_uart_dev_struct *uart, hal_uart_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Input parameter{in}	
p_irq	a pointer to UART interrupt callback functions structure, refer to Table 3-588. Structure hal_uart_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_uart_irq_struct uart_irq;
hal_uart_struct_init(HAL_UART_IRQ_INIT_STRUCT, &uart_irq);
/* set the wakeup handle function */
uart_irq.wakeup_handle = uart_wakeup_cb;
hal_uart_irq_handle_set(&uart0_info, &uart_irq);
```

hal_uart_irq_handle_all_reset

The description of hal_uart_irq_handle_all_reset is shown as below:

Table 3-597. Function hal_uart_irq_handle_all_reset

Function name	hal_uart_irq_handle_all_reset
Function prototype	void hal_uart_irq_handle_all_reset(hal_uart_dev_struct *uart);
Function descriptions	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback */
hal_uart_irq_handle_all_reset(&uart0_info);
```

hal_uart_transmit_poll

The description of hal_uart_transmit_poll is shown as below:

Table 3-598. Function hal_uart_transmit_poll

Function name	hal_uart_transmit_poll
Function prototype	int32_t hal_uart_transmit_poll(hal_uart_dev_struct *uart, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	transmit amounts of data, poll transmit process and completed status
Precondition	-
The called functions	-
Input parameter{in}	

uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

/* transmit using polling mode */

hal_uart_transmit_poll(&uart0_info, txbuffer, TRANSMIT_SIZE, 0x1FFFFF);
```

hal_uart_receive_poll

The description of hal_uart_receive_poll is shown as below:

Table 3-599. Function hal_uart_receive_poll

Function name	hal_uart_receive_poll
Function prototype	int32_t hal_uart_receive_poll(hal_uart_dev_struct *uart, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	receive amounts of data, poll receive process and completed status
Precondition	-
The called functions	-
Input parameter{in}	

uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
#define TRANSFER_SIZE          16

uint8_t rxbuffer[TRANSFER_SIZE];

/* receive using polling mode */

hal_uart_receive_poll(&uart1_info, rxbuffer, TRANSFER_SIZE, 0x7FFFFFFF);
```

hal_uart_transmit_interrupt

The description of hal_uart_transmit_interrupt is shown as below:

Table 3-600. Function hal_uart_transmit_interrupt

Function name	hal_uart_transmit_interrupt
Function prototype	int32_t hal_uart_transmit_interrupt(hal_uart_dev_struct *uart, uint8_t *p_buffer, uint32_t length, hal_uart_user_cb p_user_func);
Function descriptions	transmit amounts of data by interrupt method, after the transfer is completed, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	

uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
p_user_func	user callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```

#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

__IO FlagStatus tx_end = RESET;

void tx_complete(hal_uart_dev_struct *uart) {
    tx_end = SET;
}

/* transmit using interrupt mode */
hal_uart_transmit_interrupt(&uart0_info, txbuffer, TRANSMIT_SIZE, tx_complete);

/* check whether the transfer is completed or not */
while(RESET == tx_end){
}

```

hal_uart_receive_interrupt

The description of hal_uart_receive_interrupt is shown as below:

Table 3-601. Function hal_uart_receive_interrupt

Function name	hal_uart_receive_interrupt
Function prototype	int32_t hal_uart_receive_interrupt(hal_uart_dev_struct *uart, uint8_t *p_buffer, uint32_t length, hal_uart_user_cb p_user_func);
Function descriptions	receive amounts of data by interrupt method, after the transfer is completed, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received
Input parameter{in}	
p_user_func	user callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE          16

uint8_t rxbuffer[TRANSFER_SIZE];

__IO FlagStatus rx_end = RESET;

void rx_complete(hal_uart_dev_struct *uart) {
    rx_end = SET;
}

/* receive data using interrupt mode */
```

```

hal_uart_receive_interrupt(&uart0_info, rxbuffer, TRANSFER_SIZE, rx_complete);

/* check whether the transfer is completed or not */

while(RESET == rx_end){

}

```

hal_uart_transmit_dma

The description of hal_uart_transmit_dma is shown as below:

Table 3-602. Function hal_uart_transmit_dma

Function name	hal_uart_transmit_dma
Function prototype	int32_t hal_uart_transmit_dma(hal_uart_dev_struct *uart, uint8_t *p_buffer, uint16_t length, hal_uart_user_callback_struct *p_func);
Function descriptions	transmit amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
p_func	pointer to callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:


```
#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

__IO FlagStatus tx_end = RESET;

hal_uart_user_callback_struct user_cb;

void tx_complete(hal_uart_dev_struct *uart) {
    tx_end = SET;
}

/* initialize the user callback structure */
hal_uart_struct_init(HAL_UART_USER_CALLBACK_STRUCT, &user_cb);

user_cb.error_func = NULL;

user_cb.complete_func = tx_complete;

/* transmit using DMA mode */
hal_uart_transmit_dma(&uart0_info, txbuffer, TRANSMIT_SIZE, &user_cb);

/* check whether the transfer is completed or not */
while(RESET == tx_end){
}
```

hal_uart_receive_dma

The description of hal_uart_receive_dma is shown as below:

Table 3-603. Function hal_uart_receive_dma

Function name	hal_uart_receive_dma
Function prototype	int32_t hal_uart_receive_dma(hal_uart_dev_struct *uart, uint8_t *p_buffer, uint16_t length, hal_uart_user_callback_struct *p_func);
Function descriptions	receive amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct

Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received
Input parameter{in}	
p_func	pointer to callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE                16

uint8_t rxbuffer[TRANSFER_SIZE];

__IO FlagStatus rx_end = RESET;

hal_uart_user_callback_struct user_cb;

void rx_complete(hal_uart_dev_struct *uart) {
    rx_end = SET;
}

/* initilize the user callback structure */

hal_uart_struct_init(HAL_UART_USER_CALLBACK_STRUCT, &user_cb);

user_cb.error_func = NULL;

user_cb.complete_func = rx_complete;

/* receive data using interrupt mode */

hal_uart_receive_dma(&uart1_info, rxbuffer, TRANSFER_SIZE, &user_cb);

/* check whether the transfer is completed or not */

while(RESET == rx_end){
}
```

hal_uart_dma_pause

The description of hal_uart_dma_pause is shown as below:

Table 3-604. Function hal_uart_dma_pause

Function name	hal_uart_dma_pause
Function prototype	int32_t hal_uart_dma_pause(hal_uart_dev_struct *uart);
Function descriptions	pause UART DMA transfer during transmission process
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* pause UART DMA transfer */
hal_uart_dma_pause(&uart0_info);
```

hal_uart_dma_resume

The description of hal_uart_dma_resume is shown as below:

Table 3-605. Function hal_uart_dma_resume

Function name	hal_uart_dma_resume
Function prototype	int32_t hal_uart_dma_resume(hal_uart_dev_struct *uart);
Function descriptions	resume UART DMA transfer during transmission process
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590.

	Structure hal_uart_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* resume UART DMA transfer */
```

```
hal_uart_dma_resume(&uart0_info);
```

hal_uart_transmit_stop

The description of hal_uart_transmit_stop is shown as below:

Table 3-606. Function hal_uart_transmit_stop

Function name	hal_uart_transmit_stop
Function prototype	int32_t hal_uart_transmit_stop(hal_uart_dev_struct *uart);
Function descriptions	stop UART transmit transfer
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* stop UART transmit transfer */
```

```
hal_uart_transmit_stop(&uart0_info);
```

hal_uart_receive_stop

The description of hal_uart_receive_stop is shown as below:

Table 3-607. Function hal_uart_receive_stop

Function name	hal_uart_receive_stop
Function prototype	int32_t hal_uart_receive_stop(hal_uart_dev_struct *uart);
Function descriptions	stop UART receive transfer
Precondition	-
The called functions	-
Input parameter{in}	
uart	a pointer to UART device information structure, refer to Table 3-590. Structure hal_uart_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* stop UART receive transfer */
hal_uart_receive_stop(&uart0_info);
```

3.25. USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter [3.25.1](#), the USRT firmware functions are introduced in chapter [3.25.2](#).

3.25.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-608. USART Registers

Registers	Descriptions
USART_CTL0	Control register 0
USART_CTL1	Control register 1
USART_CTL2	Control register 2
USART_BAUD	Baud rate register
USART_GP	Guard time and prescaler register
USART_RT	Receiver timeout register

Registers	Descriptions
USART_CMD	Command register
USART_STAT	Status register
USART_INTC	Status clear register
USART_RDATA	Receive data register
USART_TDATA	Transmit data register
USART_RFCS	Receive FIFO control and status register

3.25.2. Descriptions of Peripheral functions

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

Table 3-609. USRT firmware function

Function name	Function description
hal_usrt_struct_init	initialize the USRT struct with the default values, note that this function must be called after the struct is created
hal_usrt_deinit	deinitialize the USRT
hal_usrt_init	initialize the USRT with specified values
hal_usrt_irq	handle the USRT interrupts
hal_usrt_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_usrt_irq_handle_all_reset	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_usrt_transmit_poll	transmit amounts of data, poll transmit process and completed status
hal_usrt_receive_poll	receive amounts of data, poll receive process and completed status
hal_usrt_transmit_receive_poll	transmit & receive amounts of data, poll transfer process and completed status
hal_usrt_transmit_interrupt	transmit amounts of data by interrupt method, after the transfer is completed, the user function is called
hal_usrt_receive_interrupt	receive amounts of data by interrupt method, after the transfer is completed, the user function is called
hal_usrt_transmit_receive_interrupt	transmit & receive amounts of data by interrupt method, after

Function name	Function description
	the transfer is completed, the user function is called
hal_usrt_transmit_dma	transmit amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
hal_usrt_receive_dma	receive amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
hal_usrt_transmit_receive_dma	transmit & receive amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
hal_usrt_dma_pause	pause USRT DMA transfer during transmission process
hal_usrt_dma_resume	resume USRT DMA transfer during transmission process
hal_usrt_transfer_stop	stop USRT transmit and receive transfer

Enum hal_usrt_struct_type_enum

Table 3-610. Enum hal_usrt_struct_type_enum

Member name	Function description
HAL_USRT_INIT_STRUCT	USRT initialization structure
HAL_USRT_DEV_STRUCT	USRT DMA callback function pointer structure
HAL_USRT_USER_CALLBACK_STRUCT	USRT device interrupt callback function pointer structure

Structure hal_usrt_init_struct

Table 3-611. Structure hal_usrt_init_struct

Member name	Function description
baudrate	communication baudrate
parity	parity mode
word_length	number of data bits in a frame
stop_bit	number of stop bits
direction	communication transfer direction
rx_fifo_en	receive FIFO enable
clock_polarity	clock polarity
clock_phase	clock phase

Member name	Function description
clock_length_lastbit	clock length

Structure hal_usrt_irq_struct

Table 3-612. Structure hal_usrt_irq_struct

Member name	Function description
receive_complete_handle	receive complete callback function
transmit_ready_handle	transmit ready callback function
transmit_complete_handle	transmit complete callback function
error_handle	error callback function

Enum hal_usrt_state_enum

Table 3-613. Enum hal_usrt_state_enum

Member name	Function description
USRT_STATE_FREE	USRT is free
USRT_STATE_BUSY	USRT is busy
USRT_STATE_BUSY_TX_RX	USRT is busy for Tx Rx state

Structure hal_usrt_dev_struct

Table 3-614. Structure hal_usrt_dev_struct

Member name	Function description
periph	usart port
usrt_irq	device interrupt callback function pointer, refer to Table 3-612. Structure hal_usrt_irq_struct
p_dma_rx	DMA receive pointer
p_dma_tx	DMA transmit pointer
txbuffer	transmit buffer
rxbuffer	receive buffer
data_bit_mask	mask bit of data
last_error	the last error code
error_state	error state

Member name	Function description
tx_state	transmit state
rx_state	receive state
rx_callback	receive callback function pointer
tx_callback	transmit callback function pointer
mutex	mutex locked and unlocked state
priv	private pointer

Structure hal_usrt_user_callback_struct

Table 3-615. Structure hal_usrt_user_callback_struct

Member name	Function description
complete_func	transfer complete callback function
error_func	error callback function

hal_usrt_struct_init

The description of hal_usrt_struct_init is shown as below:

Table 3-616. Function hal_usrt_struct_init

Function name	hal_usrt_struct_init
Function prototype	void hal_usrt_struct_init(hal_usrt_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the USRT struct with the default values, note that this function must be called after the struct is created
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	USRT struct, refer to Table 3-610. Enum hal_usrt_struct_type_enum
Input parameter{in}	
p_struct	a void pointer to structure
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
hal_usrt_dev_struct usrt0_info;
```

```
/* initialize a usrt device information structure */
```

```
hal_usrt_struct_init(HAL_USRT_DEV_STRUCT, &usrt0_info);
```

hal_usrt_deinit

The description of hal_usrt_deinit is shown as below:

Table 3-617. Function hal_usrt_deinit

Function name	hal_usrt_deinit
Function prototype	void hal_usrt_deinit(hal_usrt_dev_struct *usrt);
Function descriptions	deinitialize the USRT
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_usrt_dev_struct usrt0_info;
```

```
/* deinitialize the device information structure */
```

```
hal_usrt_deinit(&usrt0_info);
```

hal_usrt_init

The description of hal_usrt_init is shown as below:

Table 3-618. Function hal_usrt_init

Function name	hal_usrt_init
Function prototype	int32_t hal_usrt_init(hal_usrt_dev_struct *usrt, uint32_t periph, hal_usrt_init_struct *p_init);
Function descriptions	initialize the USRT with specified values
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Input parameter{in}	
periph	specify which USART is initialized
USARTx	x=0,1
Input parameter{in}	
p_init	the initialization data needed to initialize USRT, refer to Table 3-611. Structure hal_usrt_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```

hal_usrt_dev_struct usrt0_info;

hal_usrt_init_struct usrt0_init;

/* initialize the structures */

hal_usrt_struct_init(HAL_USRT_DEV_STRUCT, &usrt0_info);

hal_usrt_struct_init(HAL_USRT_INIT_STRUCT, &usrt0_init);

usrt0_init.baudrate = 115200;

usrt0_init.parity = USRT_PARITY_NONE;

usrt0_init.word_length = USRT_WORD_LENGTH_8BIT;

```

```

usr0_init.stop_bit = USRT_STOP_BIT_1;

usr0_init.direction = USRT_DIRECTION_RX_TX;

usr0_init.rx_fifo_en = DISABLE;

usr0_init.clock_polarity = USRT_CLOCK_POLARITY_LOW;

usr0_init.clock_phase = USRT_CLOCK_PHASE_1CK;

usr0_init.clock_length_lastbit = USRT_LAST_BIT_NOT_OUTPUT;

hal_usrt_init(&usr0_info, USART0, &usr0_init);

```

hal_usrt_irq

The description of hal_usrt_irq is shown as below:

Table 3-619. Function hal_usrt_irq

Function name	hal_usrt_irq
Function prototype	void hal_usrt_irq(hal_usrt_dev_struct *usr);
Function descriptions	handle the USRT interrupts
Precondition	-
The called functions	-
Input parameter{in}	
usr	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* process the USRT interrupt */

hal_usrt_irq(&usr0_info);

```

hal_usrt_irq_handle_set

The description of hal_usrt_irq_handle_set is shown as below:

Table 3-620. Function `hal_usrt_irq_handle_set`

Function name	<code>hal_usrt_irq_handle_set</code>
Function prototype	<code>void hal_usrt_irq_handle_set(hal_usrt_dev_struct *usrt, hal_usrt_irq_struct *p_irq);</code>
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
<code>usrt</code>	a pointer to USRT device structure, refer to Table 3-614. Structure <code>hal_usrt_dev_struct</code>
Input parameter{in}	
<code>p_irq</code>	a pointer to USRT interrupt callback functions structure, refer to Table 3-612. Structure <code>hal_usrt_irq_struct</code>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_usrt_irq_struct usrt_irq;

/* set the error handle function */
usrt_irq.error_handle = usrt_error_cb;
hal_usrt_irq_handle_set(&usrt0_info, &usrt_irq);
```

`hal_usrt_irq_handle_all_reset`

The description of `hal_usrt_irq_handle_all_reset` is shown as below:

Table 3-621. Function `hal_usrt_irq_handle_all_reset`

Function name	<code>hal_usrt_irq_handle_all_reset</code>
Function prototype	<code>void hal_usrt_irq_handle_all_reset(hal_usrt_dev_struct *usrt);</code>
Function descriptions	reset all user-defined interrupt callback function, which will be registered

	and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure <i>hal_usrt_dev_struct</i>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback */
hal_usrt_irq_handle_all_reset(&usrt0_info);
```

hal_usrt_transmit_poll

The description of hal_usrt_transmit_poll is shown as below:

Table 3-622. Function hal_usrt_transmit_poll

Function name	hal_usrt_transmit_poll
Function prototype	int32_t hal_usrt_transmit_poll(hal_usrt_dev_struct *usrt, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	transmit amounts of data, poll transmit process and completed status
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure <i>hal_usrt_dev_struct</i>
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted

Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

/* transmit using polling mode */

hal_usrt_transmit_poll(&usrt0_info, txbuffer, TRANSMIT_SIZE, 0x1FFFFF);
```

hal_usrt_receive_poll

The description of hal_usrt_receive_poll is shown as below:

Table 3-623. Function hal_usrt_receive_poll

Function name	hal_usrt_receive_poll
Function prototype	int32_t hal_usrt_receive_poll(hal_usrt_dev_struct *usrt, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	receive amounts of data, poll receive process and completed status
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received

Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
#define TRANSFER_SIZE          16

uint8_t rxbuffer[TRANSFER_SIZE];

/* receive using polling mode */

hal_usrt_receive_poll(&usrt1_info, rxbuffer, TRANSFER_SIZE, 0x7FFFFFFF);
```

hal_usrt_transmit_receive_poll

The description of hal_usrt_transmit_receive_poll is shown as below:

Table 3-624. Function hal_usrt_transmit_receive_poll

Function name	hal_usrt_transmit_receive_poll
Function prototype	int32_t hal_usrt_transmit_receive_poll(hal_usrt_dev_struct *usrt, uint8_t *p_tx_buffer, uint8_t *p_rx_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	transmit & receive amounts of data, poll transfer process and completed status
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Input parameter{in}	
p_tx_buffer	pointer to Tx data buffer
Input parameter{in}	
p_rx_buffer	pointer to Rx data buffer

Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
#define TRANSFER_SIZE                16

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F};

uint8_t rxbuffer[TRANSFER_SIZE];

/* transmit & receive using polling mode */

hal_usrt_transmit_receive_poll(&usrt1_info,txbuffer,rxbuffer,TRANSFER_SIZE, 0x1FFFF);
```

hal_usrt_transmit_interrupt

The description of hal_usrt_transmit_interrupt is shown as below:

Table 3-625. Function hal_usrt_transmit_interrupt

Function name	hal_usrt_transmit_interrupt
Function prototype	int32_t hal_usrt_transmit_interrupt(hal_usrt_dev_struct *usrt, uint8_t *p_buffer, uint32_t length, hal_usrt_user_cb p_user_func);
Function descriptions	transmit amounts of data by interrupt method, after the transfer is completed, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct

Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
p_user_func	user callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};
__IO FlagStatus tx_end = RESET;

void tx_complete(hal_usrt_dev_struct *usrt){
    tx_end = SET;
}

/* transmit using interrupt mode */
hal_usrt_transmit_interrupt(&usrt0_info, txbuffer, TRANSMIT_SIZE, tx_complete);

/* check whether the transfer is completed or not */
while(RESET == tx_end){
}
```

hal_usrt_receive_interrupt

The description of hal_usrt_receive_interrupt is shown as below:

Table 3-626. Function hal_usrt_receive_interrupt

Function name	hal_usrt_receive_interrupt
Function prototype	int32_t hal_usrt_receive_interrupt(hal_usrt_dev_struct *usrt, uint8_t

	<code>*p_buffer, uint32_t length, hal_usrt_user_cb p_user_func);</code>
Function descriptions	receive amounts of data by interrupt method, after the transfer is completed, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received
Input parameter{in}	
p_user_func	user callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE          16

uint8_t rxbuffer[TRANSFER_SIZE];

__IO FlagStatus rx_end = RESET;

void rx_complete(hal_usrt_dev_struct *usrt){
    rx_end = SET;
}

/* receive data using interrupt mode */

hal_usrt_receive_interrupt(&usrt0_info, rxbuffer, TRANSFER_SIZE, rx_complete);

/* check whether the transfer is completed or not */
```

```
while(RESET == rx_end){  
    }  
}
```

hal_usrt_transmit_receive_interrupt

The description of hal_usrt_transmit_receive_interrupt is shown as below:

Table 3-627. Function hal_usrt_transmit_receive_interrupt

Function name	hal_usrt_transmit_receive_interrupt
Function prototype	int32_t hal_usrt_transmit_receive_interrupt(hal_usrt_dev_struct *usrt, uint8_t *p_tx_buffer, uint8_t *p_rx_buffer, uint32_t length, hal_usrt_user_cb p_user_func);
Function descriptions	transmit & receive amounts of data by interrupt method, after the transfer is completed, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Input parameter{in}	
p_tx_buffer	pointer to Tx data buffer
Input parameter{in}	
p_rx_buffer	pointer to Rx data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
p_user_func	user callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE                16

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F};

uint8_t rxbuffer[TRANSFER_SIZE];

__IO FlagStatus transfer_end = RESET;

void transfer_complete(hal_usrt_dev_struct *usrt){

    transfer_end = SET;

}

/* transmit and receive data */

hal_usrt_transmit_receive_interrupt(&usrt0_info, txbuffer, rxbuffer, TRANSFER_SIZE,
transfer_complete);

while(RESET == transfer_end){

}
```

hal_usrt_transmit_dma

The description of hal_usrt_transmit_dma is shown as below:

Table 3-628. Function hal_usrt_transmit_dma

Function name	hal_usrt_transmit_dma
Function prototype	int32_t hal_usrt_transmit_dma(hal_usrt_dev_struct *usrt, uint8_t *p_buffer, uint16_t length, hal_usrt_user_callback_struct *p_func);
Function descriptions	transmit amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	

length	number of data to be transmitted
Input parameter{in}	
p_func	pointer to callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

__IO FlagStatus tx_end = RESET;

hal_usrt_user_callback_struct user_cb;

void tx_complete(hal_usrt_dev_struct *usrt){
    tx_end = SET;
}

/* initilize the user callback structure */

hal_usrt_struct_init(HAL_USRT_USER_CALLBCAK_STRUCT, &user_cb);

user_cb.error_func = NULL;

user_cb.complete_func = tx_complete;

/* transmit using DMA mode */

hal_usrt_transmit_dma(&usrt0_info, txbuffer, TRANSMIT_SIZE, &user_cb);

/* check whether the transfer is completed or not */

while(RESET == tx_end){
}
```

hal_usrt_receive_dma

The description of hal_usrt_receive_dma is shown as below:

Table 3-629. Function hal_usrt_receive_dma

Function name	hal_usrt_receive_dma
Function prototype	int32_t hal_usrt_receive_dma(hal_usrt_dev_struct *usrt, uint8_t *p_buffer, uint16_t length, hal_usrt_user_callback_struct *p_func);
Function descriptions	receive amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received
Input parameter{in}	
p_func	pointer to callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE          16

uint8_t rxbuffer[TRANSFER_SIZE];

__IO FlagStatus rx_end = RESET;

hal_usrt_user_callback_struct user_cb;

void rx_complete(hal_usrt_dev_struct *usrt){
    rx_end = SET;
}
```

```

/* initialize the user callback structure */

hal_usrt_struct_init(HAL_USRT_USER_CALLBACK_STRUCT, &user_cb);

user_cb.error_func = NULL;

user_cb.complete_func = rx_complete;

/* receive data using interrupt mode */

hal_usrt_receive_dma(&usr1_info, rxbuffer, TRANSFER_SIZE, &user_cb);

/* check whether the transfer is completed or not */

while(RESET == rx_end){

}

```

hal_usrt_transmit_receive_dma

The description of hal_usrt_transmit_receive_dma is shown as below:

Table 3-630. Function hal_usrt_transmit_receive_dma

Function name	hal_usrt_transmit_receive_dma
Function prototype	int32_t hal_usrt_transmit_receive_dma(hal_usrt_dev_struct *usr, uint8_t *p_tx_buffer, uint8_t *p_rx_buffer, uint16_t length, hal_usrt_user_callback_struct *p_func);
Function descriptions	transmit & receive amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
usr	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Input parameter{in}	
p_tx_buffer	pointer to Tx data buffer
Input parameter{in}	
p_rx_buffer	pointer to Rx data buffer
Input parameter{in}	
length	number of data to be transmitted

Input parameter{in}	
p_func	pointer to callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE                16

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F};

uint8_t rxbuffer[TRANSFER_SIZE];

__IO FlagStatus transfer_end = RESET;

hal_usrt_user_callback_struct user_cb;

void transfer_complete(hal_usrt_dev_struct *usrt){
    transfer_end = SET;
}

hal_usrt_struct_init(HAL_USRT_USER_CALLBACK_STRUCT, &user_cb);

user_cb.error_func = NULL;

user_cb.complete_func = transfer_complete;

/* transmit and receive data */

hal_usrt_transmit_receive_dma(&usrt0_info, txbuffer, rxbuffer, TRANSFER_SIZE, &user_cb);

while(RESET == transfer_end){
}
```

hal_usrt_dma_pause

The description of hal_usrt_dma_pause is shown as below:

Table 3-631. Function hal_usrt_dma_pause

Function name	hal_usrt_dma_pause
----------------------	--------------------

Function prototype	<code>int32_t hal_usrt_dma_pause(hal_usrt_dev_struct *usrt);</code>
Function descriptions	pause USRT DMA transfer during transmission process
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* pause USRT DMA transfer */
```

```
hal_usrt_dma_pause(&usrt0_info);
```

hal_usrt_dma_resume

The description of hal_usrt_dma_resume is shown as below:

Table 3-632. Function hal_usrt_dma_resume

Function name	hal_usrt_dma_resume
Function prototype	<code>int32_t hal_usrt_dma_resume(hal_usrt_dev_struct *usrt);</code>
Function descriptions	resume USRT DMA transfer during transmission process
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Output parameter{out}	
-	-
Return value	

int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS
----------------	-------------------------------

Example:

```
/* resume USRT DMA transfer */
```

```
hal_usrt_dma_resume(&usrt0_info);
```

hal_usrt_transfer_stop

The description of hal_usrt_transfer_stop is shown as below:

Table 3-633. Function hal_usrt_transfer_stop

Function name	hal_usrt_transfer_stop
Function prototype	int32_t hal_usrt_transfer_stop(hal_usrt_dev_struct *usrt);
Function descriptions	stop USRT transmit and receive transfer
Precondition	-
The called functions	-
Input parameter{in}	
usrt	a pointer to USRT device structure, refer to Table 3-614. Structure hal_usrt_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* stop USART transmit and receive transfer data */
```

```
hal_usrt_transfer_stop(&usrt0_info);
```

3.26. IrDA

IrDA is an asynchronous half-duplex communication protocol mode, which is a working mode of USART peripheral. The USART registers are listed in chapter [3.26.1](#), the IrDA firmware functions are introduced in chapter [3.26.2](#).

3.26.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-634. USART Registers

Registers	Descriptions
USART_CTL0	Control register 0
USART_CTL1	Control register 1
USART_CTL2	Control register 2
USART_BAUD	Baud rate register
USART_GP	Guard time and prescaler register
USART_RT	Receiver timeout register
USART_CMD	Command register
USART_STAT	Status register
USART_INTC	Status clear register
USART_RDATA	Receive data register
USART_TDATA	Transmit data register
USART_RFCS	Receive FIFO control and status register

3.26.2. Descriptions of Peripheral functions

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

Table 3-635. IRDA firmware function

Function name	Function description
hal_irda_struct_init	initialize the IRDA struct with the default values, note that this function must be called after the struct is created
hal_irda_deinit	deinitialize the IRDA
hal_irda_init	initialize the IRDA with specified values
hal_irda_irq	handle the IRDA interrupts
hal_irda_irq_handle_set	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_irda_irq_handle_all_reset	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
hal_irda_transmit_poll	transmit amounts of data, poll transmit process and completed status

Function name	Function description
hal_irda_receive_poll	receive amounts of data, poll receive process and completed status
hal_irda_transmit_interrupt	transmit amounts of data by interrupt method, after the transfer is completed, the user function is called
hal_irda_receive_interrupt	receive amounts of data by interrupt method, after the transfer is completed, the user function is called
hal_irda_transmit_dma	transmit amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
hal_irda_receive_dma	receive amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
hal_irda_dma_pause	pause IRDA DMA transfer during transmission process
hal_irda_dma_resume	resume IRDA DMA transfer during transmission process
hal_irda_transmit_stop	stop IRDA transmit transfer
hal_irda_receive_stop	stop IRDA receive transfer

Enum hal_irda_struct_type_enum

Table 3-636. Enum hal_irda_struct_type_enum

Member name	Function description
HAL_IRDA_INIT_STRUCT	IRDA initialization structure
HAL_IRDA_DEV_STRUCT	IRDA DMA callback function pointer structure
HAL_IRDA_USER_CALLBACK_STRUCT	IRDA device interrupt callback function pointer structure

Structure hal_irda_init_struct

Table 3-637. Structure hal_irda_init_struct

Member name	Function description
baudrate	communication baudrate
parity	parity mode
word_length	number of data bits in a frame
direction	communication transfer direction
rx_fifo_en	receive FIFO enable
mode	power mode

Member name	Function description
prescaler	prescaler

Structure hal_irda_irq_struct

Table 3-638. Structure hal_irda_irq_struct

Member name	Function description
receive_complete_handle	receive complete callback function
transmit_ready_handle	transmit ready callback function
transmit_complete_handle	transmit complete callback function
error_handle	error callback function

Enum hal_irda_state_enum

Table 3-639. Enum hal_irda_state_enum

Member name	Function description
IRDA_STATE_FREE	IRDA is free
IRDA_STATE_BUSY	IRDA is busy

Structure hal_irda_dev_struct

Table 3-640. Structure hal_irda_dev_struct

Member name	Function description
periph	usart port
irda_irq	device interrupt callback function pointer, refer to Table 3-638. Structure hal_irda_irq_struct
p_dma_rx	DMA receive pointer
p_dma_tx	DMA transmit pointer
txbuffer	transmit buffer
rxbuffer	receive buffer
data_bit_mask	mask bit of data
last_error	the last error code
error_state	error state
tx_state	transmit state

Member name	Function description
rx_state	receive state
rx_callback	receive callback function pointer
tx_callback	transmit callback function pointer
mutex	mutex locked and unlocked state
priv	private pointer

Structure hal_irda_user_callback_struct

Table 3-641. Structure hal_irda_user_callback_struct

Member name	Function description
complete_func	transfer complete callback function
error_func	error callback function

hal_irda_struct_init

The description of hal_irda_struct_init is shown as below:

Table 3-642. Function hal_irda_struct_init

Function name	hal_irda_struct_init
Function prototype	void hal_irda_struct_init(hal_irda_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the IrDA struct with the default values, note that this function must be called after the struct is created
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	IrDA struct, refer to Table 3-636. Enum hal_irda_struct_type_enum
Input parameter{in}	
p_struct	a void pointer to structure
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
hal_irda_dev_struct irda0_info;

/* initialize a irda device information structure */

hal_irda_struct_init(HAL_IRDA_DEV_STRUCT, &irda0_info);
```

hal_irda_deinit

The description of hal_irda_deinit is shown as below:

Table 3-643. Function hal_irda_deinit

Function name	hal_irda_deinit
Function prototype	void hal_irda_deinit(hal_irda_dev_struct *irda);
Function descriptions	deinitialize the IrDA
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_irda_dev_struct irda0_info;

/* deinitialize the device information structure */

hal_irda_deinit(&irda0_info);
```

hal_irda_init

The description of hal_irda_init is shown as below:

Table 3-644. Function hal_irda_init

Function name	hal_irda_init
Function prototype	int32_t hal_irda_init(hal_irda_dev_struct *irda, uint32_t periph, hal_irda_init_struct *p_init);
Function descriptions	initialize the IrDA with specified values
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Input parameter{in}	
periph	specify which USART is initialized
USARTx	x=0
Input parameter{in}	
p_init	the initialization data needed to initialize IrDA, refer to Table 3-637. Structure hal_irda_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```

hal_irda_dev_struct irda0_info;
hal_irda_init_struct irda0_init;
/* initialize the structures */
hal_irda_struct_init(HAL_IRDA_DEV_STRUCT, &irda0_info);
hal_irda_struct_init(HAL_IRDA_INIT_STRUCT, &irda0_init);
irda0_init.baudrate = 115200;
irda0_init.parity = IRDA_PARITY_NONE;
irda0_init.word_length = IRDA_WORD_LENGTH_8BIT;

```

```

irda0_init.direction = IRDA_DIRECTION_RX_TX;

irda0_init.rx_fifo_en = DISABLE;

irda0_init.mode = IRDA_NORMAL_MODE;

irda0_init.prescaler = 1;

hal_irda_init(&irda0_info, USART0, &irda0_init);

```

hal_irda_irq

The description of hal_irda_irq is shown as below:

Table 3-645. Function hal_irda_irq

Function name	hal_irda_irq
Function prototype	void hal_irda_irq(hal_irda_dev_struct *irda);
Function descriptions	handle the IrDA interrupts
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* process the IrDA interrupt */

hal_irda_irq(&irda0_info);

```

hal_irda_irq_handle_set

The description of hal_irda_irq_handle_set is shown as below:

Table 3-646. Function hal_irda_irq_handle_set

Function name	hal_irda_irq_handle_set
Function prototype	void hal_irda_irq_handle_set(hal_irda_dev_struct *irda, hal_irda_irq_struct

	*p_irq);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Input parameter{in}	
p_irq	a pointer to IrDA interrupt callback functions structure, refer to Table 3-638. Structure hal_irda_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_irda_irq_struct irda_irq;

/* set the error handle function */

irda_irq.error_handle = irda_error_cb;

hal_irda_irq_handle_set(&irda0_info, &irda_irq);
```

hal_irda_irq_handle_all_reset

The description of hal_irda_irq_handle_all_reset is shown as below:

Table 3-647. Function hal_irda_irq_handle_all_reset

Function name	hal_irda_irq_handle_all_reset
Function prototype	void hal_irda_irq_handle_all_reset(hal_irda_dev_struct *irda);
Function descriptions	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered
Precondition	-
The called functions	-

Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback */
hal_irda_irq_handle_all_reset(&irda0_info);
```

hal_irda_transmit_poll

The description of hal_irda_transmit_poll is shown as below:

Table 3-648. Function hal_irda_transmit_poll

Function name	hal_irda_transmit_poll
Function prototype	int32_t hal_irda_transmit_poll(hal_irda_dev_struct *irda, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	transmit amounts of data, poll transmit process and completed status
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
timeout_ms	timeout duration

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

/* transmit using polling mode */

hal_irda_transmit_poll(&irda0_info, txbuffer, TRANSMIT_SIZE, 0x1FFFFF);
```

hal_irda_receive_poll

The description of hal_irda_receive_poll is shown as below:

Table 3-649. Function hal_irda_receive_poll

Function name	hal_irda_receive_poll
Function prototype	int32_t hal_irda_receive_poll(hal_irda_dev_struct *irda, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	receive amounts of data, poll receive process and completed status
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received
Input parameter{in}	
timeout_ms	timeout duration

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
#define TRANSFER_SIZE          16

uint8_t rxbuffer[TRANSFER_SIZE];

/* receive using polling mode */

hal_irda_receive_poll(&irda0_info, rxbuffer, TRANSFER_SIZE, 0x7FFFFFFF);
```

hal_irda_transmit_interrupt

The description of hal_irda_transmit_interrupt is shown as below:

Table 3-650. Function hal_irda_transmit_interrupt

Function name	hal_irda_transmit_interrupt
Function prototype	int32_t hal_irda_transmit_interrupt(hal_irda_dev_struct *irda, uint8_t *p_buffer, uint32_t length, hal_irda_user_cb p_user_func);
Function descriptions	transmit amounts of data by interrupt method, after the transfer is completed, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
p_user_func	user callback function

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

__IO FlagStatus tx_end = RESET;

void tx_complete(hal_irda_dev_struct *irda){
    tx_end = SET;
}

/* transmit using interrupt mode */

hal_irda_transmit_interrupt(&irda0_info, txbuffer, TRANSMIT_SIZE, tx_complete);

/* check whether the transfer is completed or not */

while(RESET == tx_end){
}
```

hal_irda_receive_interrupt

The description of hal_irda_receive_interrupt is shown as below:

Table 3-651. Function hal_irda_receive_interrupt

Function name	hal_irda_receive_interrupt
Function prototype	int32_t hal_irda_receive_interrupt(hal_irda_dev_struct *irda, uint8_t *p_buffer, uint32_t length, hal_irda_user_cb p_user_func);
Function descriptions	receive amounts of data by interrupt method, after the transfer is completed, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure

	<u>hal_irda_dev_struct</u>
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received
Input parameter{in}	
p_user_func	user callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE          16

uint8_t rxbuffer[TRANSFER_SIZE];
__IO FlagStatus rx_end = RESET;

void rx_complete(hal_irda_dev_struct * irda){
    rx_end = SET;
}

/* receive data using interrupt mode */
hal_irda_receive_interrupt(&irda0_info, rxbuffer, TRANSFER_SIZE, rx_complete);

/* check whether the transfer is completed or not */
while(RESET == rx_end){
}
```

hal_irda_transmit_dma

The description of hal_irda_transmit_dma is shown as below:

Table 3-652. Function hal_irda_transmit_dma

Function name	hal_irda_transmit_dma
----------------------	-----------------------

Function prototype	int32_t hal_irda_transmit_dma(hal_irda_dev_struct *irda, uint8_t *p_buffer, uint16_t length, hal_irda_user_callback_struct *p_func);
Function descriptions	transmit amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
p_func	pointer to callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

__IO FlagStatus tx_end = RESET;

hal_irda_user_callback_struct user_cb;

void tx_complete(hal_irda_dev_struct *irda){
    tx_end = SET;
}

/* initialize the user callback structure */
```

```

hal_irda_struct_init(HAL_IRDA_USER_CALLBACK_STRUCT, &user_cb);

user_cb.error_func = NULL;

user_cb.complete_func = tx_complete;

/* transmit using DMA mode */

hal_irda_transmit_dma(&irda0_info, txbuffer, TRANSMIT_SIZE, &user_cb);

/* check whether the transfer is completed or not */

while(RESET == tx_end){

}

```

hal_irda_receive_dma

The description of hal_irda_receive_dma is shown as below:

Table 3-653. Function hal_irda_receive_dma

Function name	hal_irda_receive_dma
Function prototype	int32_t hal_irda_receive_dma(hal_irda_dev_struct *irda, uint8_t *p_buffer, uint16_t length, hal_irda_user_callback_struct *p_func);
Function descriptions	receive amounts of data by DMA method, after the transfer is completed or error occurs, the user function is called
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received
Input parameter{in}	
p_func	pointer to callback function
Output parameter{out}	
-	-

Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE          16

uint8_t rxbuffer[TRANSFER_SIZE];

__IO FlagStatus rx_end = RESET;

hal_irda_user_callback_struct user_cb;

void rx_complete(hal_irda_dev_struct *irda){
    rx_end = SET;
}

/* initialize the user callback structure */

hal_irda_struct_init(HAL_IRDA_USER_CALLBACK_STRUCT, &user_cb);

user_cb.error_func = NULL;

user_cb.complete_func = rx_complete;

/* receive data using interrupt mode */

hal_irda_receive_dma(&irda0_info, rxbuffer, TRANSFER_SIZE, &user_cb);

/* check whether the transfer is completed or not */

while(RESET == rx_end){
}
```

hal_irda_dma_pause

The description of hal_irda_dma_pause is shown as below:

Table 3-654. Function hal_irda_dma_pause

Function name	hal_irda_dma_pause
Function prototype	int32_t hal_irda_dma_pause(hal_irda_dev_struct *irda);
Function descriptions	pause IrDA DMA transfer during transmission process
Precondition	-
The called functions	-

Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure <i>hal_irda_dev_struct</i>
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* pause IrDA DMA transfer */
hal_irda_dma_pause(&irda0_info);
```

hal_irda_dma_resume

The description of hal_irda_dma_resume is shown as below:

Table 3-655. Function hal_irda_dma_resume

Function name	hal_irda_dma_resume
Function prototype	int32_t hal_irda_dma_resume(hal_irda_dev_struct *irda);
Function descriptions	resume IrDA DMA transfer during transmission process
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure <i>hal_irda_dev_struct</i>
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* resume IrDA DMA transfer */
hal_irda_dma_resume(&irda0_info);
```

hal_irda_transmit_stop

The description of hal_irda_transmit_stop is shown as below:

Table 3-656. Function hal_irda_transmit_stop

Function name	hal_irda_transmit_stop
Function prototype	int32_t hal_irda_transmit_stop(hal_irda_dev_struct *irda);
Function descriptions	stop IrDA transmit transfer
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure hal_irda_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* stop USART transmit transfer */
hal_irda_transmit_stop(&irda0_info);
```

hal_irda_receive_stop

The description of hal_irda_receive_stop is shown as below:

Table 3-657. Function hal_irda_receive_stop

Function name	hal_irda_receive_stop
Function prototype	int32_t hal_irda_receive_stop(hal_irda_dev_struct *irda);
Function descriptions	stop IrDA receive transfer
Precondition	-
The called functions	-
Input parameter{in}	
irda	a pointer to IrDA device structure, refer to Table 3-640. Structure

	<i>hal_irda_dev_struct</i>
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
/* stop USART receive transfer */
hal_irda_receive_stop(&irda0_info);
```

3.27. SMARTCARD

SMARTCARD is an asynchronous half-duplex communication protocol mode, which is a working mode of USART peripheral and mainly used for SMARTCARD communication. The USART registers are listed in chapter [3.27.1](#), the USART firmware functions are introduced in chapter [3.27.2](#).

3.27.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-658. USART Registers

Registers	Descriptions
USART_CTL0	Control register 0
USART_CTL1	Control register 1
USART_CTL2	Control register 2
USART_BAUD	Baud rate register
USART_GP	Guard time and prescaler register
USART_RT	Receiver timeout register
USART_CMD	Command register
USART_STAT	Status register
USART_INTC	Status clear register
USART_RDATA	Receive data register
USART_TDATA	Transmit data register
USART_RFCS	Receive FIFO control and status register

3.27.2. Descriptions of Peripheral functions

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

Table 3-659. SMARTCARD firmware function

Function name	Function description
hal_smartcard_struct_init	initialize the smartcard structure with the default values
hal_smartcard_deinit	deinitialize smartcard
hal_smartcard_init	initialize the smartcard with specified values
hal_smartcard_irq	handle the smartcard interrupts
hal_smartcard_irq_handle_set	set user-defined interrupt callback function
hal_smartcard_irq_handle_all_reset	reset all user-defined interrupt callback function
hal_smartcard_transmit_poll	transmit amounts of data by poll method
hal_smartcard_receive_poll	receive amounts of data by poll method
hal_smartcard_transmit_interrupt	transmit amounts of data by interrupt method
hal_smartcard_receive_interrupt	receive amounts of data by interrupt method
hal_smartcard_transmit_dma	transmit amounts of data by DMA method
hal_smartcard_receive_dma	receive amounts of data by DMA method
hal_smartcard_transmit_stop	stop smartcard transmit transfer
hal_smartcard_receive_stop	stop smartcard receive transfer

Enum hal_smartcard_struct_type_enum

Table 3-660. Enum hal_smartcard_struct_type_enum

Member name	Function description
HAL_SMARTCARD_INIT_STRUCT	SMARTCARD initialization structure
HAL_SMARTCARD_DEV_STRUCT	SMARTCARD DMA callback function pointer structure
HAL_SMARTCARD_USER_CALLBACK_STRUCT	SMARTCARD device interrupt callback function pointer structure
HAL_SMARTCARD_IRQ_INIT_STRUCT	SMARTCARD device information structtrue

Structure hal_smartcard_init_struct

Table 3-661. Structure hal_smartcard_init_struct

Member name	Function description
baudrate	smartcard communication baud rate
parity	parity mode

Member name	Function description
word_length	number of data bits in a frame
stop_bit	number of stop bits
direction	communication transfer direction
clock_polarity	the state of serial colck
clock_phase	the clock transition on which the bit capture is made
clock_length_lastbit	whether the clock pulse corresponding to the last transmitted data bit
prescaler	smartcard prescaler
guard_time	smartcard guard time
nack_state	NACK transmission is enabled or disabled
early_nack	NACK occurs 1/16 bit time earlier when parity error is detected
rx_fifo_en	receive FIFO enable
timeout_enable	the receiver timeout is enabled
timeout_value	the receiver timeout value
sample_method	select sample method
block_length	the smartcard block length in T=1 reception mode
auto_retry_count	the smartcard auto-retry count
first_bit_msb	MSB is sent first on communication line
tx_rx_swap	Tx and Rx pins are swapped
rx_level_invert	the Rx pin active level is inverted
tx_level_invert	the Tx pin active level is inverted
data_bit_invert	data is inverted
overrun_disable	the reception overrun detection is disabled
rx_error_dma_stop	the DMA is disabled in case of reception error

Structure hal_smartcard_irq_struct

Table 3-662. Structure hal_smartcard_irq_struct

Member name	Function description
receive_complete_handle	smartcard receive complete callback function
transmit_ready_handle	smartcard transmit ready callback function
transmit_complete_handle	smartcard transmit complete callback function
error_handle	smartcard transfer error callback function

Enum hal_smartcard_state_enum

Table 3-663. Enum hal_smartcard_state_enum

Member name	Function description
SMARTCARD_STATE_FREE	SMARTCARD is free
SMARTCARD _STATE_BUSY	SMARTCARD is busy

Structure hal_smartcard_dev_struct

Table 3-664. Structure hal_smartcard_dev_struct

Member name	Function description
periph	usart port (usart0)
smartcard_irq	smartcard device interrupt callback function pointer
p_dma_rx	DMA receive pointer
p_dma_tx	DMA transmit pointer
txbuffer	transmit buffer
rxbuffer	receive buffer
last_error	the last error code
error_state	smartcard error state
tx_state	transmit state
rx_state	receive state
rx_callback	receive callback function pointer
tx_callback	transmit callback function pointer
mutex	mutex locked and unlocked state

Member name	Function description
priv	private pointer

Structure hal_smartcard_user_callback_struct

Table 3-665. Structure hal_smartcard_user_callback_struct

Member name	Function description
complete_func	user-defined transfer complete callback function
error_func	user-defined transfer error callback function

hal_smartcard_struct_init

The description of hal_smartcard_struct_init is shown as below:

Table 3-666. Function hal_smartcard_struct_init

Function name	hal_smartcard_struct_init
Function prototype	void hal_smartcard_struct_init(hal_smartcard_struct_type_enum hal_struct_type, void *p_struct);
Function descriptions	initialize the smartcard structure with the default values
Precondition	-
Input parameter{in}	
hal_struct_type	smartcard structure type, Table 3-660. Enum hal_smartcard_struct_type_enum
Input parameter{in}	
p_struct	init structure pointer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_smartcard_dev_struct smartcard0_info;

/* initialize SMARTCARD device information structure with the default values */

hal_smartcard_struct_init(HAL_SMARTCARD_DEV_STRUCTURE, &smartcard0_info);
```

hal_smartcard_deinit

The description of hal_smartcard_deinit is shown as below:

Table 3-667. Function hal_smartcard_deinit

Function name	hal_smartcard_deinit
Function prototype	void hal_smartcard_deinit(hal_smartcard_dev_struct *smartcard);
Function descriptions	deinitialize smartcard
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_smartcard_dev_struct smartcard0_info;

/* deinitialize SMARTCARD*/

hal_smartcard_deinit(&smartcard0_info);
```

hal_smartcard_init

The description of hal_smartcard_init is shown as below:

Table 3-668. Function hal_smartcard_init

Function name	hal_smartcard_init
Function prototype	int32_t hal_smartcard_init(hal_smartcard_dev_struct *smartcard, uint32_t periph, hal_smartcard_init_struct *p_init);
Function descriptions	initialize smartcard
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct

Input parameter{in}	
periph	specify which USART is initialized
<i>USARTx</i>	x=0
Input parameter{in}	
p_init	initialize structure, the structure members can refer to Table 3-661. Structure hal smartcard init struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```

hal_smartcard_dev_struct smartcard0_info;

hal_smartcard_init_struct smartcard0_init;

/* initialize SMARTCARD device information structure with the default values */
hal_smartcard_struct_init(HAL_SMARTCARD_INIT_STRUCT, &smartcard0_init);

smartcard0_init.baudrate = 10752;

smartcard0_init.parity = SMARTCARD_PARITY_EVEN;

smartcard0_init.word_length = SMARTCARD_WORD_LENGTH_9BIT;

smartcard0_init.stop_bit = SMARTCARD_STOP_BIT_1_5;

smartcard0_init.direction = SMARTCARD_DIRECTION_RX_TX;

smartcard0_init.clock_polarity = SMARTCARD_CLOCK_POLARITY_LOW;

smartcard0_init.clock_phase = SMARTCARD_CLOCK_PHASE_1CK;

smartcard0_init.clock_length_lastbit = SMARTCARD_LAST_BIT_NOT_OUTPUT;

smartcard0_init.prescaler = 9;

smartcard0_init.guard_time = 4;

smartcard0_init.nack_state = SMARTCARD_NACK_DISABLE;

smartcard0_init.early_nack = DISABLE;

smartcard0_init.rx_fifo_en = DISABLE;

```

```

smartcard0_init.timeout_enable = DISABLE;

smartcard0_init.timeout_value = 0;

smartcard0_init.sample_method = SMARTCARD_THREE_SAMPLE_BIT;

smartcard0_init.block_length = 0;

smartcard0_init.auto_retry_count = 0;

smartcard0_init.first_bit_msb = DISABLE;

smartcard0_init.rx_rx_swap = DISABLE;

smartcard0_init.tx_level_invert = DISABLE;

smartcard0_init.rx_level_invert = DISABLE;

smartcard0_init.data_bit_invert = DISABLE;

smartcard0_init.overrun_disable = DISABLE;

smartcard0_init.rx_error_dma_stop = DISABLE;

/* initialize SMARTCARD*/

hal_smartcard_init(&smartcard0_info, USART0, &smartcard0_init);

```

hal_smartcard_irq

The description of hal_smartcard_irq is shown as below:

Table 3-669. Function hal_smartcard_irq

Function name	hal_smartcard_irq
Function prototype	void hal_smartcard_irq(hal_smartcard_dev_struct *smartcard);
Function descriptions	smartcard interrupt handler content function
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_smartcard_dev_struct smartcard0_info;

/* SMARTCARD interrupt handler content function */

hal_smartcard_irq(&smartcard0_info);
```

hal_smartcard_irq_handle_set

The description of hal_smartcard_irq_handle_set is shown as below:

Table 3-670. Function hal_smartcard_irq_handle_set

Function name	hal_smartcard_irq_handle_set
Function prototype	void hal_smartcard_irq_handle_set(hal_smartcard_dev_struct *smartcard, hal_smartcard_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Input parameter{in}	
p_irq	interrupt callback function structure pointer, the structure members can refer to Table 3-662. Structure hal_smartcard_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_smartcard_dev_struct smartcard0_info;

void smartcard_error_handler(hal_smartcard_dev_struct *smartcard);

hal_smartcard_irq_struct smartcard0_irq;

/* initialize interrupt callback structure with the default values */

hal_smartcard_struct_init(HAL_SMARTCARD_IRQ_INIT_STRUCT, &smartcard0_irq);

smartcard0_irq.error_handle = (hal_irq_handle_cb)smartcard_error_handler;
```

```
/* set user-defined interrupt callback function */
```

```
hal_smartcard_irq_handle_set(&smartcard0_info, &smartcard0_irq);
```

hal_smartcard_irq_handle_all_reset

The description of hal_smartcard_irq_handle_all_reset is shown as below:

Table 3-671. Function hal_smartcard_irq_handle_all_reset

Function name	hal_smartcard_irq_handle_all_reset
Function prototype	void hal_smartcard_irq_handle_all_reset(hal_smartcard_dev_struct *smartcard);
Function descriptions	reset all user-defined interrupt callback function
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
hal_smartcard_dev_struct smartcard0_info;
```

```
/* reset all user-defined interrupt callback function */
```

```
hal_smartcard_irq_handle_all_reset(&smartcard0_info)
```

hal_smartcard_transmit_poll

The description of hal_smartcard_transmit_poll is shown as below:

Table 3-672. Function hal_smartcard_transmit_poll

Function name	hal_smartcard_transmit_poll
Function prototype	int32_t hal_smartcard_transmit_poll(hal_smartcard_dev_struct *smartcard, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	transmit amounts of data by poll method

Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
#define TRANSMIT_SIZE          6

uint8_t txdata[] = {0x01,0x02,0x03,0x04,0x05};

/* transmit data by poll method */

hal_smartcard_transmit_poll(&smartcard0_info, txdata, TRANSMIT_SIZE, 0x1FFF);
```

hal_smartcard_receive_poll

The description of hal_smartcard_receive_poll is shown as below:

Table 3-673. Function hal_smartcard_receive_poll

Function name	hal_smartcard_receive_poll
Function prototype	int32_t hal_smartcard_receive_poll(hal_smartcard_dev_struct *smartcard, uint8_t *p_buffer, uint32_t length, uint32_t timeout_ms);
Function descriptions	receive amounts of data by poll method
Precondition	-

Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be received
Input parameter{in}	
timeout_ms	timeout duration
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY, HAL_ERR_TIMEOUT

Example:

```
#define TRANSFER_SIZE          16

uint8_t rxbuffer[TRANSFER_SIZE];

/* receive using polling mode */

hal_smartcard_receive_poll(&smartcard0_info, rxbuffer, TRANSFER_SIZE, 0x7FFFFFFF);
```

hal_smartcard_transmit_interrupt

The description of hal_smartcard_transmit_interrupt is shown as below:

Table 3-674. Function hal_smartcard_transmit_interrupt

Function name	hal_smartcard_transmit_interrupt
Function prototype	int32_t hal_smartcard_transmit_interrupt(hal_smartcard_dev_struct *smartcard, uint8_t *p_buffer, uint32_t length, hal_smartcard_user_cb p_user_func);
Function descriptions	transmit amounts of data by interrupt method
Precondition	-
Input parameter{in}	

smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
p_user_func	user callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSMIT_SIZE          6

uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

__IO FlagStatus tx_end = RESET;

void tx_complete(hal_smartcard_dev_struct * smartcard){
    tx_end = SET;
}

/* transmit using interrupt mode */

hal_smartcard_transmit_interrupt(&smartcard0_info,    txbuffer,    TRANSMIT_SIZE,
tx_complete);

/* check whether the transfer is completed or not */

while(RESET == tx_end){
}
```

hal_smartcard_receive_interrupt

The description of hal_smartcard_receive_interrupt is shown as below:

Table 3-675. Function hal_smartcard_receive_interrupt

Function name	hal_smartcard_receive_interrupt
Function prototype	int32_t hal_smartcard_receive_interrupt(hal_smartcard_dev_struct *smartcard, uint8_t *p_buffer, uint32_t length, hal_smartcard_user_cb p_user_func);
Function descriptions	receive amounts of data by interrupt method
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be sent received
Input parameter{in}	
p_user_func	user callback function
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE                16

uint8_t rxbuffer[TRANSFER_SIZE];

__IO FlagStatus rx_end = RESET;

void rx_complete(hal_smartcard_dev_struct *smartcard){
    rx_end = SET;
}

/* receive data using interrupt mode */

hal_smartcard_receive_interrupt(&smartcard0_info,    rxbuffer,    TRANSFER_SIZE,
```

```

rx_complete);

/* check whether the transfer is completed or not */

while(RESET == rx_end){

}

```

hal_smartcard_transmit_dma

The description of hal_smartcard_transmit_dma is shown as below:

Table 3-676. Function hal_smartcard_transmit_dma

Function name	hal_smartcard_transmit_dma
Function prototype	int32_t hal_smartcard_transmit_dma(hal_smartcard_dev_struct *smartcard, uint8_t *p_buffer, uint16_t length, hal_smartcard_user_callback_struct *p_user_func);
Function descriptions	transmit amounts of data by dma method
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer
Input parameter{in}	
length	number of data to be transmitted
Input parameter{in}	
p_user_func	user callback function structure, the structure members can refer to Table 3-665. Structure hal_smartcard_user_callback_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSMIT_SIZE
```

6

```
uint8_t txbuffer[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

__IO FlagStatus tx_end = RESET;

hal_smartcard_user_callback_struct user_cb;

void tx_complete(hal_smartcard_dev_struct *smartcard){
    tx_end = SET;
}

/* initilize the user callback structure */

hal_smartcard_struct_init(HAL_SMARTCARD_USER_CALLBACK_STRUCT, &user_cb);

user_cb.error_func = NULL;

user_cb.complete_func = tx_complete;

/* transmit using DMA mode */

hal_smartcard_transmit_dma(&smartcard0_info, txbuffer, TRANSMIT_SIZE, &user_cb);

/* check whether the transfer is completed or not */

while(RESET == tx_end){
}
```

hal_smartcard_receive_dma

The description of hal_smartcard_receive_dma is shown as below:

Table 3-677. Function hal_smartcard_receive_dma

Function name	hal_smartcard_receive_dma
Function prototype	int32_t hal_smartcard_receive_dma(hal_smartcard_dev_struct *smartcard, uint8_t *p_buffer, uint16_t length, hal_smartcard_user_callback_struct *p_user_func);
Function descriptions	receive amounts of data by dma method
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Input parameter{in}	
p_buffer	pointer to data buffer

Input parameter{in}	
length	number of data to be sent received
Input parameter{in}	
p_user_func	user callback function structure, the structure members can refer to Table 3-665. Structure hal_smartcard_user_callback_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL , HAL_ERR_BUSY

Example:

```
#define TRANSFER_SIZE                16

uint8_t rxbuffer[TRANSFER_SIZE];

__IO FlagStatus rx_end = RESET;

hal_smartcard_user_callback_struct user_cb;

void rx_complete(hal_smartcard_dev_struct *smartcard){
    rx_end = SET;
}

/* initialize the user callback structure */
hal_smartcard_struct_init(HAL_SMARTCARD_USER_CALLBACK_STRUCT, &user_cb);
user_cb.error_func = NULL;
user_cb.complete_func = rx_complete;

/* receive data using interrupt mode */
hal_smartcard_receive_dma(&smartcard0_info, rxbuffer, TRANSFER_SIZE, &user_cb);

/* check whether the transfer is completed or not */
while(RESET == rx_end){
}
```

hal_smartcard_transmit_stop

The description of hal_smartcard_transmit_stop is shown as below:

Table 3-678. Function hal_smartcard_transmit_stop

Function name	hal_smartcard_transmit_stop
Function prototype	int32_t hal_smartcard_transmit_stop(hal_smartcard_dev_struct *smartcard);
Function descriptions	stop smartcard transmit transfer
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
hal_smartcard_dev_struct smartcard0_info;

/* stop SMARTCARD transmit */

hal_smartcard_transmit_stop(&smartcard0_info);
```

hal_smartcard_receive_stop

The description of hal_smartcard_receive_stop is shown as below:

Table 3-679. Function hal_smartcard_receive_stop

Function name	hal_smartcard_receive_stop
Function prototype	int32_t hal_smartcard_receive_stop(hal_smartcard_dev_struct *smartcard);
Function descriptions	stop smartcard receive transfer
Precondition	-
Input parameter{in}	
smartcard	smartcard device information structure, the structure members can refer to Table 3-664. Structure hal_smartcard_dev_struct

Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS

Example:

```
hal_smartcard_dev_struct smartcard0_info;

/* stop SMARTCARD receive */

hal_smartcard_receive_stop(&smartcard0_info);
```

3.28. WWDGT

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. The WWDGT registers are listed in chapter [3.28.1](#), the WWDGT firmware functions are introduced in chapter [3.28.2](#).

3.28.1. Descriptions of Peripheral registers

WWDGT registers are listed in the table shown as below:

Table 3-680. WWDGT Registers

Registers	Descriptions
WWDGT_CTL	Control register
WWDGT_CFG	Configuration register
WWDGT_STAT	Status register

3.28.2. Descriptions of Peripheral functions

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

Table 3-681. WWDGT firmware function

Function name	Function description
hal_wwdgt_struct_init	initialize the parameters of WWDGT struct with the default values
hal_wwdgt_init	initialize WWDGT

Function name	Function description
hal_wwdgt_deinit	deinitialize WWDGT
hal_wwdgt_irq	WWDGT interrupt handler content function
hal_wwdgt_irq_handle_set	set user-defined interrupt callback function
hal_wwdgt_irq_handle_all_reset	reset all user-defined interrupt callback function

Enum hal_wwdgt_state_enum

Table

3-682. hal_wwdgt_state_enum

enum name	enum description
HAL_WWDGT_STATE_NONE	none
HAL_WWDGT_STATE_RESET	WWDGT status reset
HAL_WWDGT_STATE_READY	WWDGT ready
HAL_WWDGT_STATE_BUSY	WWDGT busy

Enum hal_wwdgt_struct_type_enum

Table 3-683. hal_wwdgt_struct_type_enum

enum name	enum description
HAL_WWDGT_INIT_STRUCT	WWDGT initialize structure
HAL_WWDGT_DEV_STRUCT	WWDGT device information structure

Structure hal_wwdgt_dev_struct

Table 3-684. hal_wwdgt_dev_struct

Member name	Function description
state	WWDGT status
mutex	mutex locked and unlocked state

Structure hal_wwdgt_init_struct

Table 3-685. hal_wwdgt_init_struct

Member name	Function description
wwdgt_pre_select	WWDGT prescaler value
wwdgt_cnt_value	WWDGT counter window value
wwdgt_downcnt_value	WWDGT counter reload value

Structure hal_wwdgt_irq_struct

Table 3-686. hal_wwdgt_irq_struct

Member name	Function description
early_wakeup_handler	WWDGT interrupt function

hal_wwdgt_struct_init

The description of hal_wwdgt_struct_init is shown as below:

Table 3-687. Function hal_wwdgt_struct_init

Function name	hal_wwdgt_struct_init
Function prototype	void hal_wwdgt_struct_init(hal_wwdgt_struct_type_enum hal_struct_type,void *p_struct);
Function descriptions	initialize the parameters of WWDGT struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
hal_struct_type	pointer to hal_wwdgt_struct_type_enum, refer to Table 3-683. hal_wwdgt_struct_type_enum
Input parameter{in}	
*p_struct	pointer to the WWDGT structure that contains configuration information
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* initialize the WWDGT STRUCT */
```

```
hal_wwdgt_struct_init(HAL_WWDGT_INIT_STRUCT, &wwdgt_init_parameter);
```

hal_wwdgt_init

The description of hal_wwdgt_init is shown as below:

Table 3-688. Function hal_wwdgt_init

Function name	hal_wwdgt_init
Function prototype	int32_t hal_wwdgt_init(hal_wwdgt_dev_struct *wwdgt_dev, hal_wwdgt_init_struct *p_wwdgt_init);
Function descriptions	initialize WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
wwdgt_dev	pointer to WWDGT device information structure, structure member refer to Table 3-684. hal_wwdgt_dev_struct
Input parameter{in}	
p_wwdgt_init	pointer to WWDGT initialize structure, structure member refer to Table 3-685. hal_wwdgt_init_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_VAL / HAL_ERR_NONE

Example:

```
/* initialize the WWDGT */
```

```
hal_wwdgt_init(&wwdgt_info,&wwdgt_init_parameter);
```

hal_wwdgt_deinit

The description of hal_wwdgt_deinit is shown as below:

Table 3-689. Function hal_wwdgt_deinit

Function name	hal_wwdgt_deinit
Function prototype	void hal_wwdgt_deinit(hal_wwdgt_dev_struct *wwdgt_dev);
Function descriptions	deinitialize WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
wwdgt_dev	pointer to WWDGT device information structure, structure member refer to Table 3-684. hal_wwdgt_dev_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the configuration of WWDGT */
```

```
hal_wwdgt_deinit(&wwdgt_info);
```

hal_wwdgt_irq

The description of hal_wwdgt_irq is shown as below:

Table 3-690. Function hal_wwdgt_irq

Function name	hal_wwdgt_irq
Function prototype	void hal_wwdgt_irq(hal_wwdgt_dev_struct *wwdgt_dev);
Function descriptions	WWDGT interrupt handle function
Precondition	-

The called functions	-
Input parameter{in}	
wwdgt_dev	pointer to WWDGT device information structure, structure member refer to Table 3-684. hal wwdgt_dev struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* the function is used in the relative interrupt routine */

hal_wwdgt_dev_struct wwdgt_info;

WWDGT_IRQHandler(void)

```
{
    hal_wwdgt_irq(&wwdgt_info);
}
```

hal_wwdgt_irq_handle_set

The description of hal_wwdgt_irq_handle_set is shown as below:

Table 3-691. Function hal_wwdgt_irq_handle_set

Function name	hal_wwdgt_irq_handle_set
Function prototype	void hal_wwdgt_irq_handle_set(hal_wwdgt_dev_struct *wwdgt_dev, hal_wwdgt_irq_struct *p_irq);
Function descriptions	set user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered;
Precondition	-
The called functions	-
Input parameter{in}	
wwdgt_dev	pointer to WWDGT device information structure, structure member refer

	to Table 3-684. hal_wwdgt_dev_struct
Input parameter{in}	
p_irq	pointer to WWDGT interrupt callback function structure, structure member refer to Table 3-686. hal_wwdgt_irq_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set user-defined interrupt callback function */

hal_wwdgt_dev_struct wwdgt_info;

hal_wwdgt_irq_struct wwdgt_irq;

wwdgt_irq.early_wakeup_handle = wwdgt_early_wakeup_func;

hal_dac_irq_handle_set(&dac_info,&dac_irq);
```

hal_wwdgt_irq_handle_all_reset

The description of hal_wwdgt_irq_handle_all_reset is shown as below:

Table 3-692. Function hal_wwdgt_irq_handle_all_reset

Function name	hal_wwdgt_irq_handle_all_reset
Function prototype	void hal_wwdgt_irq_handle_all_reset(hal_wwdgt_dev_struct *wwdgt_dev);
Function descriptions	reset all user-defined interrupt callback function, which will be registered and called when corresponding interrupt be triggered ;
Precondition	-
The called functions	-
Input parameter{in}	
wwdgt_dev	pointer to WWDGT device information structure, structure member refer to Table 3-684. hal_wwdgt_dev_struct
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* reset all user-defined interrupt callback function */
```

```
hal_wwdgt_dev_struct wwdgt_info;
```

```
hal_wwdgt_irq_handle_all_reset(&wwdgt_info);
```

hal_wwdgt_start

The description of hal_wwdgt_start is shown as below:

Table 3-693. Function hal_wwdgt_start

Function name	hal_wwdgt_start
Function prototype	void hal_wwdgt_start(void);
Function descriptions	start WWDGT module function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start WWDGT module*/
```

```
hal_wwdgt_dev_struct wwdgt_info;
```

```
hal_wwdgt_start ();
```

hal_wwdgt_start_interrupt

The description of hal_wwdgt_start_interrupt is shown as below:

Table 3-694. Function hal_wwdgt_start_interrupt

Function name	hal_wwdgt_start_interrupt
Function prototype	int32_t hal_wwdgt_start_interrupt(hal_wwdgt_dev_struct *wwdgt_dev, hal_wwdgt_irq_struct *p_irq);
Function descriptions	start WWDGT module function by interrupt method, the function is non-blocking
Precondition	-
The called functions	-
Input parameter{in}	
wwdgt_dev	pointer to WWDGT device information structure, structure member refer to Table 3 669. hal_wwdgt_dev_struct
Input parameter{in}	
p_irq	pointer to WWDGT interrupt callback function structure, structure member refer to Table 3 167. hal_wwdgt_irq_struct
Output parameter{out}	
-	-
Return value	
int32_t	HAL_ERR_NONE, HAL_ERR_ADDRESS, HAL_ERR_VAL

Example:

```

/* set user-defined interrupt callback function */
hal_wwdgt_dev_struct wwdgt_info;
hal_wwdgt_irq_struct wwdgt_irq;
wwdgt_irq.early_wakeup_handle = wwdgt_early_wakeup_func;
hal_wwdgt_start_interrupt (&dac_info,&dac_irq);

```


hal_wwdgt_reload

The description of hal_wwdgt_reload is shown as below:

Table 3-695. Function hal_wwdgt_reload

Function name	hal_wwdgt_reload
Function prototype	int32_t hal_wwdgt_reload(hal_wwdgt_init_struct *p_wwdgt);
Function descriptions	reload the counter of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
p_wwdgt	pointer to WWDGT initialize structure , structure member refer to Table 3-670. hal_wwdgt_init_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start WWDGT module*/

hal_wwdgt_init_struct wwdgt_init_parameter;

wwdgt_init_parameter.wwdgt_pre_select = WWDGT_PSC_DIV8;

wwdgt_init_parameter.wwdgt_cnt_value = 127;

wwdgt_init_parameter.wwdgt_downcnt_value = 80;

hal_wwdgt_reload(&wwdgt_init_parameter);hal_wwdgt_start ();
```

3.29. USBFS

3.29.1. Descriptions of Peripheral registers

3.29.2. Descriptions of Peripheral functions

4. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Sep.1, 2023

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company under the intellectual property laws and treaties of the People's Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as its suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.