GigaDevice Semiconductor Inc.

GD32F1x0 ARM® Cortex™-M3 32-bit MCU

Firmware Library User Guide

Revison 1.2

(Dec. 2023)

Table of Contents

List of Figures

List of Tables

1. Introduction

This manual introduces firmware library of GD32F1x0 devices which are 32-bit microcontrollers based on the ARM processor.

The firmware library is a firmware function package, including program, data structure and macro definitions, all the performance features of peripherals of GD32F1x0 devices are involved in the package. The peripheral driving code and firmware examples on evaluation board are also included in firmware library. Users need not learn each peripherals in details and it's easy to apply a peripheral by using the firmware library. Using firmware library can greatly reduce programming time, thereby reducing development costs.

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

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

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

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

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

1.1. Rules of User Manual and Firmware Library

1.1.1. Peripherals

Table 1-1. Peripherals

Peripherals	Descriptions
ADC	Analog-to-digital converter
CAN	Controller area network
CEC	HDMI-CEC controller
CMP	Comparator
CRC	CRC calculation unit
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 I/Os
I2C	Inter-integrated circuit interface
IVREF	Programmable Current and Voltage Reference
MISC	Nested Vectored Interrupt Controller
OPA	Operational amplifiers
PMU	Power management unit
RCU	Reset and clock unit
RTC	Real-time Clock
SLCD	Segment LCD controller
SPI/I2S	Serial peripheral interface/Inter-IC sound
SYSCFG	System configuration
TIMER	TIMER
TSI	Touch sensing interface
USART	Universal synchronous/asynchronous receiver /transmitter
WWDGT	Window watchdog timer
USBD	Universal serial bus full-speed device interface

1.1.2. Naming rules

The firmware library naming rules are shown as below:

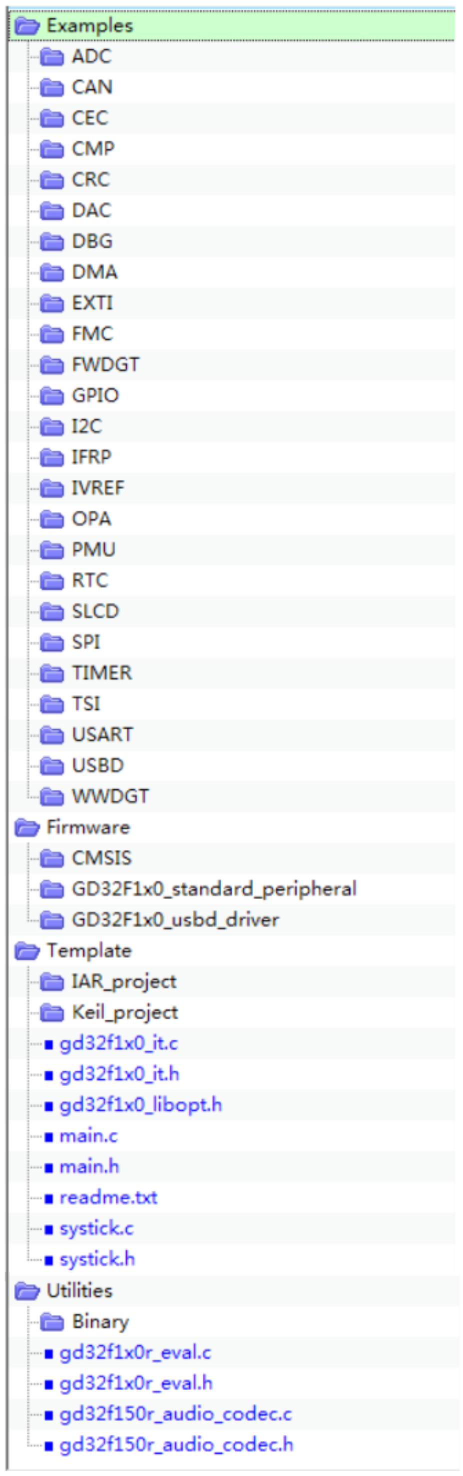
- The peripherals are shortened in XXX format, such as: ADC. More shorten information of peripherals refer to ***Peripherals***;
- The name of sourcefile and header file are started with “GD32f1x0_”, such as: GD32f1x0_adc.h;
- The constants used only in one file should be defined in the used file; the constants used in many files should be defined in corresponding header file. All the constants are written in uppercase of English letters;
- Registers are handled as constants. The naming of them are written in uppercase of English letters. In most cases, register names are shortened accord with the user manual;
- Variables are written in lowercase, when concluded by several words, underlines should be adapted among words;
- The naming of peripheral functions are started with the peripheral abbreviation added with an underline, when the function name is concluded by several words, underlines should be adapted among words, and all the peripheral functions are written in lowercase.

2. Firmware Library Overview

2.1. File Structure of Firmware Library

GD32F1x0_Firmware_Library, the file structure is shown as below:

Figure 2-1. File structure of firmware library of GD32F1x0



```
Examples
    ADC
    CAN
    CEC
    CMP
    CRC
    DAC
    DBG
    DMA
    EXTI
    FMC
    FWDGT
    GPIO
    I2C
    IFRP
    IVREF
    OPA
    PMU
    RTC
    SLCD
    SPI
    TIMER
    TSI
    USART
    USBD
    WWDGT
Firmware
    CMSIS
    GD32F1x0_standard_peripheral
    GD32F1x0_usbd_driver
Template
    IAR_project
    Keil_project
    gd32f1x0_it.c
    gd32f1x0_it.h
    gd32f1x0_libopt.h
    main.c
    main.h
    readme.txt
    systick.c
    systick.h
Utilities
    Binary
    gd32f1x0r_eval.c
    gd32f1x0r_eval.h
    gd32f150r_audio_codec.c
    gd32f150r_audio_codec.h
```

2.1.1. Examples Folder

Examples folder, each of GD32 peripheral has a subfolder. Each subfolder contains one or more examples of the peripheral, to show how to use the peripheral correctly. Each of the example subfolder includes the files shown as below:

- readme.txt: the description and using guide of the example;
- GD32f1x0_libopt.h: the header file configures all the peripherals used in the example, included by different “DEFINE” sentences (all the peripherals are enabled by default);
- GD32f1x0_it.c: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- GD32f1x0.it.h: the header file include all the prototypes of the interrupt service routines;
- systick.c: the source file include the precise time delay functions by using systick;
- systick.h: the header file include the prototype of the precise time delay functions by using systick;
- main.c: example code. Note: all the examples are not influenced by software IDEs.

2.1.2. Firmware Folder

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

- CMSIS subfolder includes the Cortex M3 kernel support files, the startup file based on the Cortex M3 kernel processor, the global header file of GD32F1x0 and system configuration file;
- GD32F1x0_standard_peripheral subfolder:
  - Include subfolder includes all the header files of firmware libray, users need not modify this folder;
  - Source subfolder includes all the source files of firmware library, users need not modify this folder;

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

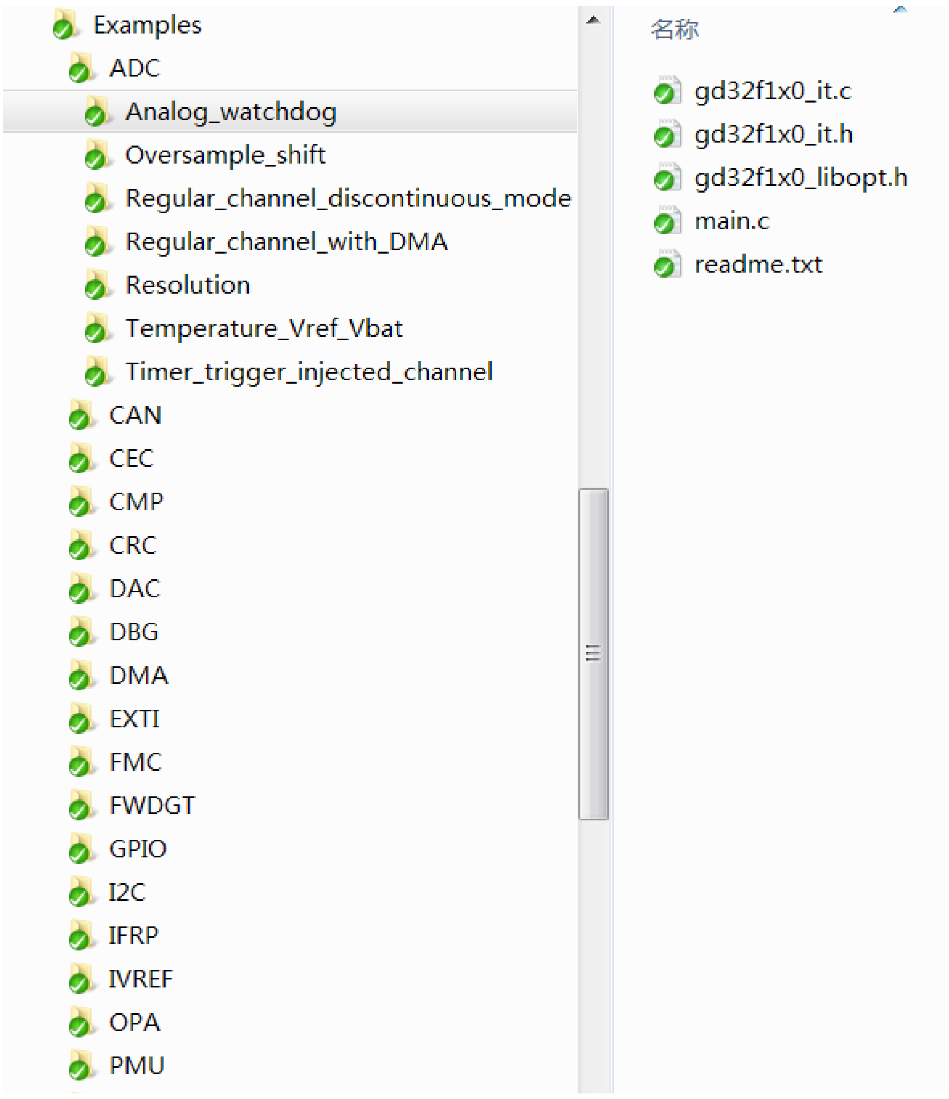
2.1.3. Template Folder

Template folder includes a simple demo of how to use LED, how to print by USART and use key to control, (IAR_project is run in IAR, and Keil_project is run in Keil4). User can use the project template to compile the formware examples, the steps are shown as below:

Select files

Open “Examples” folder, select the module to be tested, such as SPI, open ”SPI” folder, select an example of SPI, such as ”SPI_master_transmit_slave_receive_interrupt”, shown as below:

Figure 2-2. Select peripheral example files

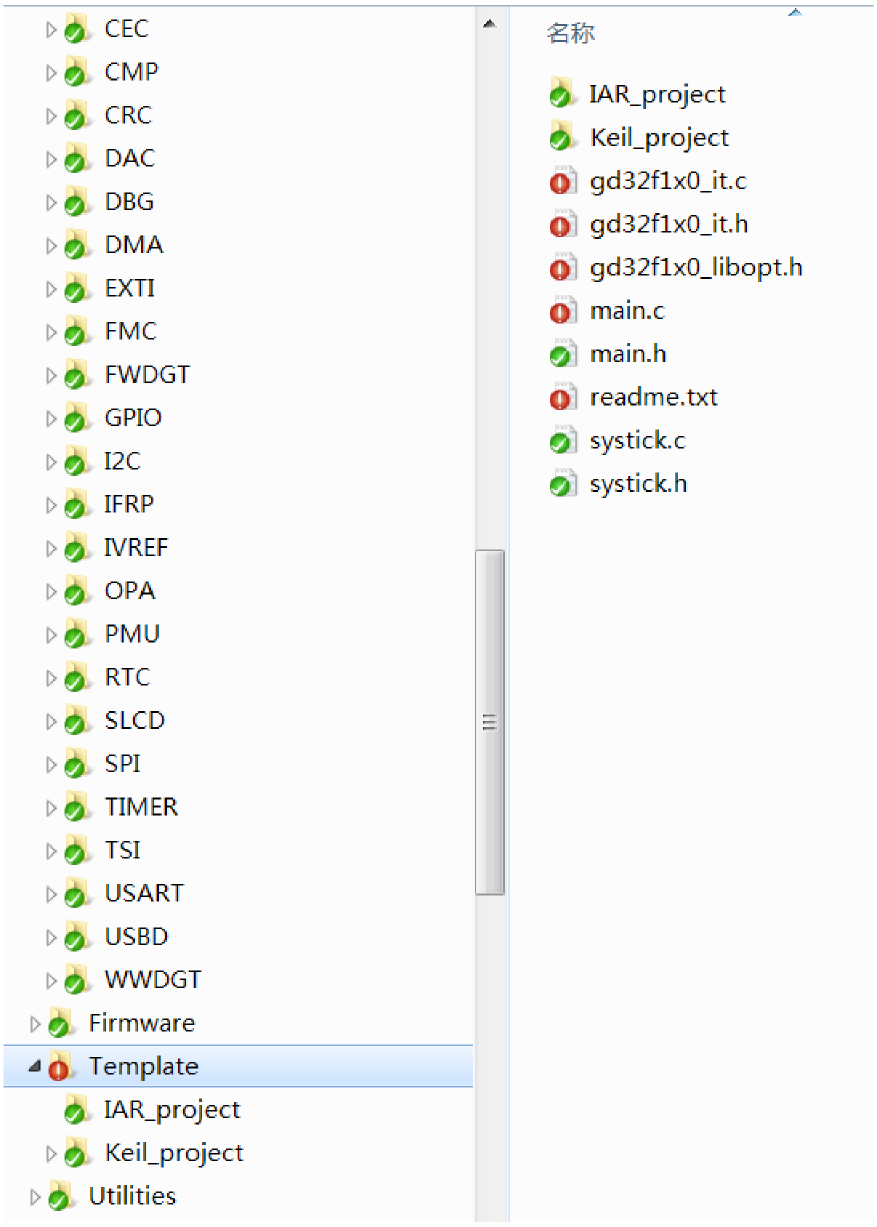

Examples
ADC
Analog_watchdog
Oversample_shift
Regular_channel_discontinuous_mode
Regular_channel_with_DMA
Resolution
Temperature_Vref_Vbat
Timer_trigger_injected_channel
CAN
CEC
CMP
CRC
DAC
DBG
DMA
EXTI
FMC
FWDGT
GPIO
I2C
IFRP
IVREF
OPA
PMU

名称
gd32f1x0_it.c
gd32f1x0_it.h
gd32f1x0_libopt.h
main.c
readme.txt

Copy files

Open “Template” folder, keep the folders of ” IAR_project” and ” Keil_project”, and delete the other files, then copy all the files in “SPI_master_transmit_slave_receive_interrupt” folder to the “Template” subfolder, shown as below:

Figure 2-3. Copy the peripheral example files



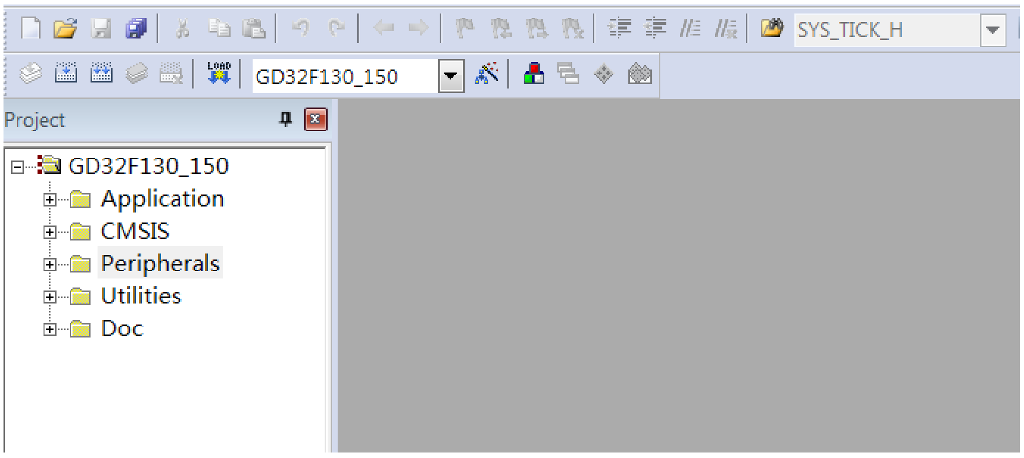
CEC
CMP
CRC
DAC
DBG
DMA
EXTI
FMC
FWDGT
GPIO
I2C
IFRP
IVREF
OPA
PMU
RTC
SLCD
SPI
TIMER
TSI
USART
USBD
WWDGT
Firmware
Template
IAR_project
Keil_project
Utilities

名称
IAR_project
Keil_project
gd32f1x0_it.c
gd32f1x0_it.h
gd32f1x0_libopt.h
main.c
main.h
readme.txt
systick.c
systick.h

Open a project

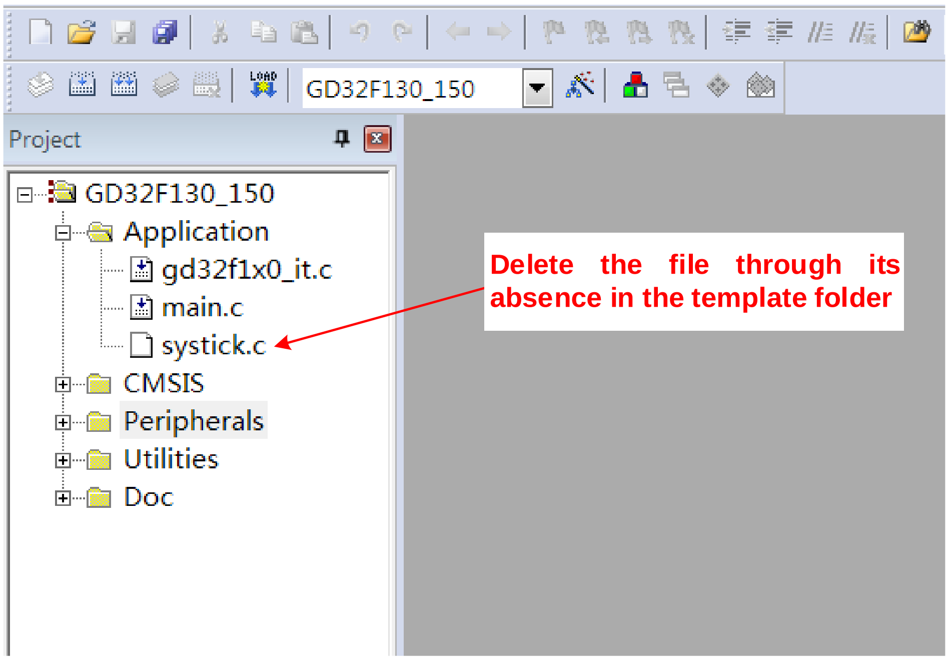
GD provides project in Keil and IAR, users can open project in different IDEs according to their need, such as ”Keil_project”, open \Template\Keil_project\Project.uvproj, shown as below:

Figure 2-4. Open the project file



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

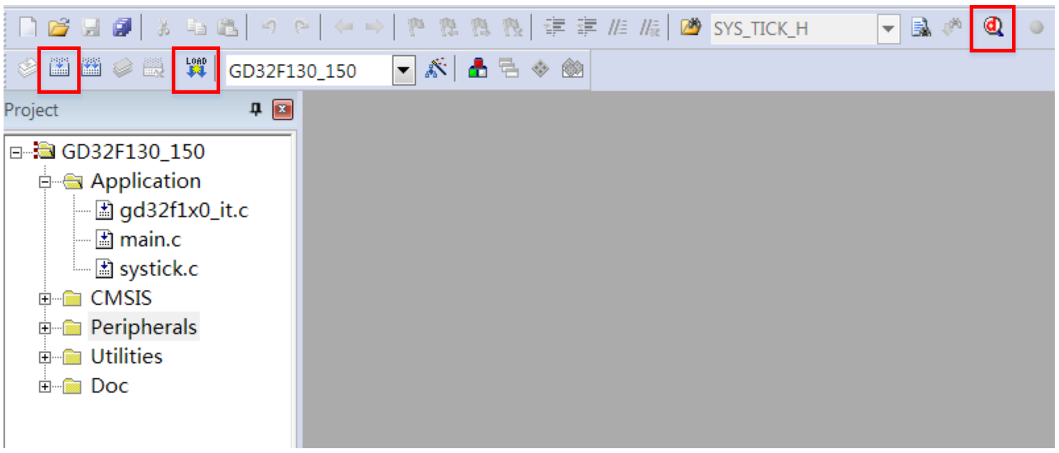
Figure 2-5. Configure project files



Compile-Debug-Download

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

Figure 2-6. Compile-debug-download



2.1.4. Utilities Folder

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

- Binary subfolders;
- gd32f1x0r_eval.h and gd32f150r_audio_codec.h are related header files of the evaluation board about running the firmware examples;
- gd32f1x0r_eval.c and gd32f150r_audio_codec.c are related source files of the evaluation board about running the firmware examples.

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

2.2. File descriptions of Firmware Library

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

Table 2-1. Function descriptions of Firmware Library

Files	Descriptions
gd32f1x0_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.
gd32f1x0_it.h	Header file, including all the prototypes of interrupt service routines.
gd32f1x0_it.c	Source files about interruput service routines of peripherals. User can written his own interrupt functions in this file. For the different interrupt service requests to the same interrupt vector, users can confirm the interrupt source by functions of judging interrupt flags of peripherals. The functions are included in the firmware library.
gd32f1x0_xxx.h	The header file of peripheral xxx, including functions about peripheral xxx, and the variables used for functions.
gd32f1x0_xxx.c	The C source file for driving peripheral xxx.

Files	Descriptions
systick.h	The header file of systick.c, including prototypes of systick configuration function and delay function.
systick.c	The source file about systick configuration function and delay function.
readme.txt	Description document about how to configure and how to use the firmware example.

3. Firmware Library of Standard Peripherals

3.1. Overview of Firmware Library of Standard Peripherals

The description format of firmware functions are shown as below:

Table 3-1. Peripheral function format of Firmware Library

Function name	Name of peripheral function
Function prototype	Declaration prototype
Function descriptions	Explain the function how to work
Precondition	Requirements should meet before calling this function
The called functions	Other firmware functions called in this functin
Input parameter{in}	
Input parameter name	Description
xxx	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	Status register
ADC_CTL0	Control register 0
ADC_CTL1	Control register 1
ADC_SAMPT0	Sample time register 0
ADC_SAMPT1	Sample time register 1
ADC_IOFFx	Inserted channel data offset register x(x=0..3)
ADC_WDHT	Watchdog high threshold register
ADC_WDLT	Watchdog low threshold register

Registers	Descriptions
ADC_RSQ0	Regular sequence register 0
ADC_RSQ1	Regular sequence register 1
ADC_RSQ2	Regular sequence register 2
ADC_ISQ	Inserted sequence register
ADC_IDATAx	Inserted data register x(x=0..3)
ADC_RDATA	Regular data register

3.2.2. Descriptions of Peripheral functions

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

Table 3-3. ADC firmware function

Function name	Function description
adc_deinit	reset ADC peripheral
adc_enable	enable ADC interface
adc_disable	disable ADC interface
adc_calibration_enable	ADC calibration and reset calibration
adc_dma_mode_enable	enable DMA request
adc_dma_mode_disable	disable DMA request
adc_tempsensor_vrefint_enable	enable the temperature sensor and Vrefint channel
adc_tempsensor_vrefint_disable	disable the temperature sensor and Vrefint channel
adc_vbat_enable	enable the Vbat channel
adc_vbat_disable	disable the Vbat channel
adc_discontinuous_mode_config	configure ADC discontinuous mode
adc_special_function_config	enable or disable ADC special function
adc_data_alignment_config	configure ADC data alignment
adc_channel_length_config	configure the length of regular channel group or inserted channel group
adc_regular_channel_config	configure ADC regular channel
adc_inserted_channel_config	configure ADC inserted channel
adc_inserted_channel_offset_config	configure ADC inserted channel offset
adc_external_trigger_config	enable ADC external trigger
adc_external_trigger_source_config	configure ADC external trigger source
adc_software_trigger_enable	enable ADC software trigger
adc_regular_data_read	read ADC regular group data register
adc_inserted_data_read	read ADC inserted group data register
adc_flag_get	get the ADC flag bits
adc_flag_clear	clear the ADC flag bits
adc_interrupt_flag_get	get the ADC interrupt bits
adc_interrupt_flag_clear	clear the ADC flag
adc_interrupt_enable	enable ADC interrupt
adc_interrupt_disable	disable ADC interrupt

Function name	Function description
adc_watchdog_single_channel_enable	configure ADC analog watchdog single channel
adc_watchdog_group_channel_enable	configure ADC analog watchdog group channel
adc_watchdog_disable	disable ADC analog watchdog
adc_watchdog_threshold_config	configure ADC analog watchdog threshold

adc_deinit

The description of adc_deinit is shown as below:

Table 3-4. Function adc_deinit

Function name	adc_deinit
Function prototype	void adc_deinit(void);
Function descriptions	reset ADC peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* reset ADC */

adc_deinit();

adc_enable

The description of adc_enable is shown as below:

Table 3-5. Function adc_enable

Function name	adc_enable
Function prototype	void adc_enable(void);
Function descriptions	enable ADC interface
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable ADC */

adc_enable();

adc_disable

The description of adc_disable is shown as below:

Table 3-6. Function adc_disable

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

Example:

/* disable ADC */

adc_disable();

adc_calibration_enable

The description of adc_calibration_enable is shown as below:

Table 3-7. Function adc_calibration_enable

Function name	adc_calibration_enable
Function prototype	void adc_calibration_enable(void);
Function descriptions	ADC calibration and reset calibration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* ADC calibration and reset calibration */

adc_calibration_enable();

adc_dma_mode_enable

The description of adc_dma_mode_enable is shown as below:

Table 3-8. Function adc_dma_mode_enable

Function name	adc_dma_mode_enable
Function prototype	void adc_dma_mode_enable(void);
Function descriptions	enable ADC DMA request
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable ADC DMA request */

adc_dma_mode_enable();

adc_dma_mode_disable

The description of adc_dma_mode_disable is shown as below:

Table 3-9. Function adc_dma_mode_disable

Function name	adc_dma_mode_disable
Function prototype	void adc_dma_mode_disable(void);
Function descriptions	disable ADC DMA request
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable ADC DMA request */

adc_dma_mode_disable();

adc_tempsensor_vrefint_enable

The description of adc_tempsensor_vrefint_enable is shown as below:

Table 3-10. Function adc_tempsensor_vrefint_enable

Function name	adc_tempsensor_vrefint_enable
Function prototype	void adc_tempsensor_vrefint_enable(void);
Function descriptions	enable the temperature sensor and Vrefint channel
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the temperature sensor and Vrefint channel */

adc_tempsensor_vrefint_enable();

adc_tempsensor_vrefint_disable

The description of adc_tempsensor_vrefint_disable is shown as below:

Table 3-11. Function adc_tempsensor_vrefint_disable

Function name	adc_tempsensor_vrefint_disable
Function prototype	void adc_tempsensor_vrefint_disable(void);
Function descriptions	disable the temperature sensor and Vrefint channel
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the temperature sensor and Vrefint channel */

adc_tempsensor_vrefint_disable();

adc_vbat_enable

The description of adc_vbat_enable is shown as below:

Table 3-12. Function adc_vbat_enable

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

Example:

/* enable the Vbat channel */

adc_vbat_enable();

adc_vbat_disable

The description of adc_vbat_disable is shown as below:

Table 3-13. Function adc_vbat_disable

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

Example:

/* disable the Vbat channel */

adc_vbat_disable();

adc_discontinuous_mode_config

The description of adc_discontinuous_mode_config is shown as below:

Table 3-14. Function adc_discontinuous_mode_config

Function name	adc_discontinuous_mode_config
Function prototype	void adc_discontinuous_mode_config(uint8_t channel_group, uint8_t length);
Function descriptions	configure ADC discontinuous mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
ADC_CHANNEL_DISCON_DISABLE	disable discontinuous mode of regular and inserted channel
Input parameter{in}	
length	number of conversions in discontinuous mode, the number can be 1..8 for regular channel, the number has no effect for inserted channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC regular channel group discontinuous mode */

adc_discontinuous_mode_config(ADC_REGULAR_CHANNEL, 6);

adc_special_function_config

The description of adc_special_function_config is shown as below:

Table 3-15. Function adc_special_function_config

Function name	adc_special_function_config
Function prototype	void adc_special_function_config(uint32_t function, ControlStatus newvalue);
Function descriptions	enable or disable ADC special function
Precondition	-
The called functions	-
Input parameter{in}	
function	the function to config

ADC_SCAN_MODE	scan mode select
ADC_INSERTED_ CHANNEL_AUTO	inserted channel group convert automatically
ADC_CONTINUOUS_ MODE	continuous mode select
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable ADC scan mode */

adc_special_function_config(ADC_SCAN_MODE, ENABLE);

adc_data_alignment_config

The description of adc_data_alignment_config is shown as below:

Table 3-16. Function adc_data_alignment_config

Function name	adc_data_alignment_config
Function prototype	void adc_data_alignment_config(uint32_t data_alignment);
Function descriptions	configure ADCx data alignment
Precondition	-
The called functions	-
Input parameter{in}	
data_alignment	data alignment select
ADC_DATAALIGN_ RIGHT	right alignment
ADC_DATAALIGN_ LEFT	left alignment
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC data alignment */

adc_data_alignment_config(ADC_DATAALIGN_RIGHT);

adc_channel_length_config

The description of adc_channel_length_config is shown as below:

Table 3-17. Function adc_channel_length_config

Function name	adc_channel_length_config
Function prototype	void adc_channel_length_config(uint8_t channel_group, uint32_t length);
Function descriptions	configure the length of regular channel group or inserted channel group
Precondition	-
The called functions	-
Input parameter{in}	
channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
length	the length of the channel, regular channel 1-16, inserted channel 1-4
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the length of ADC regular channel */

adc_channel_length_config(ADC_REGULAR_CHANNEL, 4);

adc_regular_channel_config

The description of adc_regular_channel_config is shown as below:

Table 3-18. Function adc_regular_channel_config

Function name	adc_regular_channel_config
Function prototype	void adc_regular_channel_config(uint8_t rank, uint8_t channel, uint32_t sample_time);
Function descriptions	configure ADC regular channel
Precondition	-
The called functions	-
Input parameter{in}	
rank	the regular group sequence rank, this parameter must be between 0 to 15
Input parameter{in}	
channel	the selected ADC channel
ADC_CHANNEL_x	ADC Channelx (x=0..18)
Input parameter{in}	

sample_time	the sample time value
ADC_SAMPLETIME_1POINT5	1.5 cycles
ADC_SAMPLETIME_7POINT5	7.5 cycles
ADC_SAMPLETIME_13POINT5	13.5 cycles
ADC_SAMPLETIME_28POINT5	28.5 cycles
ADC_SAMPLETIME_41POINT5	41.5 cycles
ADC_SAMPLETIME_55POINT5	55.5 cycles
ADC_SAMPLETIME_71POINT5	71.5 cycles
ADC_SAMPLETIME_239POINT5	239.5 cycles
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC regular channel */

adc_regular_channel_config(1, ADC_CHANNEL_0, ADC_SAMPLETIME_7POINT5);

adc_inserted_channel_config

The description of adc_inserted_channel_config is shown as below:

Table 3-19. Function adc_inserted_channel_config

Function name	adc_inserted_channel_config
Function prototype	void adc_inserted_channel_config(uint8_t rank, uint8_t channel, uint32_t sample_time);
Function descriptions	configure ADC inserted channel
Precondition	-
The called functions	-
Input parameter{in}	
rank	the inserted group sequencer rank, this parameter must be between 0 to 3
Input parameter{in}	
channel	the selected ADC channel
ADC_CHANNEL_x	ADC Channelx (x=0..18)
Input parameter{in}	

sample_time	the sample time value
ADC_SAMPLETIME_1POINT5	1.5 cycles
ADC_SAMPLETIME_7POINT5	7.5 cycles
ADC_SAMPLETIME_13POINT5	13.5 cycles
ADC_SAMPLETIME_28POINT5	28.5 cycles
ADC_SAMPLETIME_41POINT5	41.5 cycles
ADC_SAMPLETIME_55POINT5	55.5 cycles
ADC_SAMPLETIME_71POINT5	71.5 cycles
ADC_SAMPLETIME_239POINT5	239.5 cycles
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC0 inserted channel */

adc_inserted_channel_config(ADC0, 1, ADC_CHANNEL_0, ADC_SAMPLETIME_7POINT5);

adc_inserted_channel_offset_config

The description of adc_inserted_channel_offset_config is shown as below:

Table 3-20. Function adc_inserted_channel_offset_config

Function name	adc_inserted_channel_offset_config
Function prototype	void adc_inserted_channel_offset_config(uint8_t inserted_channel, uint16_t offset);
Function descriptions	configure ADC inserted channel offset
Precondition	-
The called functions	-
Input parameter{in}	
inserted_channel	insert channel select
ADC_INSERTED_CHANNEL_x	inserted channel, x=0,1,2,3
Input parameter{in}	
offset	the offset data, this parameter must be between 0 to 4095

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC inserted channel offset */

adc_inserted_channel_offset_config(ADC_INSERTED_CHANNEL_0, 100);

adc_external_trigger_config

The description of adc_external_trigger_config is shown as below:

Table 3-21. Function adc_external_trigger_config

Function name	adc_external_trigger_config
Function prototype	void adc_external_trigger_config(uint8_t channel_group, ControlStatus newvalue);
Function descriptions	configure ADC external trigger
Precondition	-
The called functions	-
Input parameter{in}	
channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable ADC inserted channel group external trigger */

adc_external_trigger_config(ADC_INSERTED_CHANNEL_0, ENABLE);

adc_external_trigger_source_config

The description of adc_external_trigger_source_config is shown as below:

Table 3-22. Function adc_external_trigger_source_config

Function name	adc_external_trigger_ source_config
Function prototype	void adc_external_trigger_source_config(uint8_t channel_group, uint32_t external_trigger_source);
Function descriptions	configure ADC external trigger source
Precondition	-
The called functions	-
Input parameter{in}	
channel_group	select the channel group
ADC_REGULAR_ CHANNEL	regular channel group
ADC_INSERTED_ CHANNEL	inserted channel group
Input parameter{in}	
external_trigger_ source	regular or inserted group trigger source
ADC_EXTTRIG_ REGULAR_T0_CH0	TIMER0 CH0 event select for regular channel
ADC_EXTTRIG_ REGULAR_T0_CH1	TIMER0 CH1 event select for regular channel
ADC1_EXTTRIG_ REGULAR_T0_CH2	TIMER0 CH2 event select for regular channel
ADC_EXTTRIG_ REGULAR_T1_CH1	TIMER1 CH1 event select for regular channel
ADC_EXTTRIG_ REGULAR_T2_TRGO	TIMER2 TRGO event select for regular channel
ADC_EXTTRIG_ REGULAR_T14_CH0	TIMER14 CH0 event select for regular channel
ADC_EXTTRIG_ REGULAR_EXTI_11	external interrupt line 11 for regular channel
ADC_EXTTRIG_ REGULAR_NONE	software trigger for regular channel
ADC_EXTTRIG_ INSERTED_T0_TRGO	TIMER0 TRGO event select for inserted channel
ADC_EXTTRIG_ INSERTED_T0_CH3	TIMER0 CH3 event select for inserted channel
ADC_EXTTRIG_ INSERTED_T1_TRGO	TIMER1 TRGO event select for inserted channel
ADC_EXTTRIG_ INSERTED_T1_CH0	TIMER1 CH0 event select for inserted channel
ADC_EXTTRIG_ INSERTED_T2_CH3	TIMER2 CH3 event select for inserted channel
*ADC_EXTTRIG_*	TIMER14 TRGO event select for inserted channel

INSERTED_T14_ TRGO	
ADC_EXTTRIG_ INSERTED_EXTI_15	external interrupt line 15 for inserted channel
ADC_EXTTRIG_ INSERTED_NONE	software trigger for inserted channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC regular channel external trigger source */

adc_external_trigger_source_config(ADC_REGULAR_CHANNEL, ADC_EXTTRIG_REGULAR_T0_CH0);

adc_software_trigger_enable

The description of adc_software_trigger_enable is shown as below:

Table 3-23. Function adc_software_trigger_enable

Function name	adc_software_trigger_enable
Function prototype	void adc_software_trigger_enable(uint8_t channel_group);
Function descriptions	enable ADC software trigger
Precondition	-
The called functions	-
Input parameter{in}	
channel_group	select the channel group
ADC_REGULAR_ CHANNEL	regular channel group
ADC_INSERTED_ CHANNEL	inserted channel group
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable ADC regular channel group software trigger */

adc_software_trigger_enable(ADC_REGULAR_CHANNEL);

adc_regular_data_read

The description of adc_regular_data_read is shown as below:

Table 3-24. Function adc_regular_data_read

Function name	adc_regular_data_read
Function prototype	uint16_t adc_regular_data_read(void);
Function descriptions	read ADC regular group data register
Precondition	-
The called functions	-
Output parameter{out}	
-	-
Return value	
uint16_t	ADC conversion value (0-0xFFFF)

Example:

/* read ADC regular group data register */

uint16_t adc_value = 0;

adc_value = adc_regular_data_read();

adc_inserted_data_read

The description of adc_inserted_data_read is shown as below:

Table 3-25. Function adc_inserted_data_read

Function name	adc_inserted_data_read
Function prototype	uint16_t adc_inserted_data_read(uint8_t inserted_channel);
Function descriptions	read ADC inserted group data register
Precondition	-
The called functions	-
Input parameter{in}	
inserted_channel	insert channel select
ADC_INSERTED_CHANNEL_x	inserted Channelx, x=0,1,2,3
Output parameter{out}	
-	-
Return value	
uint16_t	ADC conversion value (0-0xFFFF)

Example:

/* read ADC inserted group data register */

uint16_t adc_value = 0;

adc_value = adc_inserted_data_read (ADC_INSERTED_CHANNEL_0);

adc_flag_get

The description of adc_flag_get is shown as below:

Table 3-26. Function adc_flag_get

Function name	adc_flag_get
Function prototype	FlagStatus adc_flag_get(uint32_t flag);
Function descriptions	get the ADC flag bits
Precondition	-
The called functions	-
Input parameter{in}	
flag	the adc flag bits
ADC_FLAG_WDE	analog watchdog event flag
ADC_FLAG_EOC	end of group conversion flag
ADC_FLAG_EOIC	end of inserted group conversion flag
ADC_FLAG_STIC	start flag of inserted channel group
ADC_FLAG_STRC	start flag of regular channel group
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the ADC analog watchdog flag bits*/

FlagStatus flag_value;

flag_value = adc_flag_get(ADC_FLAG_WDE);

adc_flag_clear

The description of adc_flag_clear is shown as below:

Table 3-27. Function adc_flag_clear

Function name	adc_flag_clear
Function prototype	void adc_flag_clear(uint32_t flag);
Function descriptions	clear the ADC flag bits
Precondition	-
The called functions	-
Input parameter{in}	
flag	the adc flag bits
ADC_FLAG_WDE	analog watchdog event flag
ADC_FLAG_EOC	end of group conversion flag
ADC_FLAG_EOIC	end of inserted group conversion flag
ADC_FLAG_STIC	start flag of inserted channel group

ADC_FLAG_STRC	start flag of regular channel group
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear the ADC analog watchdog flag bits*/

adc_flag_clear(ADC_FLAG_WDE);

adc_interrupt_flag_get

The description of adc_interrupt_flag_get is shown as below:

Table 3-28. Function adc_interrupt_flag_get

Function name	adc_interrupt_flag_get
Function prototype	FlagStatus adc_interrupt_flag_get(uint32_t flag);
Function descriptions	get the ADC interrupt bits
Precondition	-
The called functions	-
Input parameter{in}	
flag	the adc interrupt bits
ADC_INT_FLAG_WDE	analog watchdog interrupt
ADC_INT_FLAG_EOC	end of group conversion interrupt
ADC_INT_FLAG_EOIC	end of inserted group conversion interrupt
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the ADC analog watchdog interrupt bits*/

FlagStatus flag_value;

flag_value = adc_interrupt_flag_get(ADC_INT_FLAG_WDE);

adc_interrupt_flag_clear

The description of adc_interrupt_flag_clear is shown as below:

Table 3-29. Function adc_interrupt_flag_clear

Function name	adc_interrupt_flag_clear
Function prototype	void adc_interrupt_flag_clear(uint32_t flag);
Function descriptions	clear the ADC interrupt bits

Precondition	-
The called functions	-
Input parameter{in}	
flag	the adc interrupt bits
ADC_INT_FLAG_WDE	analog watchdog interrupt
ADC_INT_FLAG_EOC	end of group conversion interrupt
ADC_INT_FLAG_EOIC	end of inserted group conversion interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear the ADC analog watchdog interrupt bits */

adc_interrupt_flag_clear(ADC_INT_FLAG_WDE);

adc_interrupt_enable

The description of adc_interrupt_enable is shown as below:

Table 3-30. Function adc_interrupt _enable

Function name	adc_interrupt_enable
Function prototype	void adc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the adc interrupt
ADC_INT_WDE	analog watchdog interrupt
ADC_INT_EOC	end of group conversion interrupt
ADC_INT_EOIC	end of inserted group conversion interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable ADC analog watchdog interrupt */

adc_interrupt_enable(ADC_INT_WDE);

adc_interrupt_disable

The description of adc_interrupt_disable is shown as below:

Table 3-31. Function adc_interrupt_disable

Function name	adc_interrupt_disable
Function prototype	void adc_interrupt_disable(uint32_t interrupt);
Function descriptions	Disable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the adc interrupt
ADC_INT_WDE	analog watchdog interrupt
ADC_INT_EOC	end of group conversion interrupt
ADC_INT_EOIC	end of inserted group conversion interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable ADC interrupt */

adc_interrupt_disable(ADC_INT_WDE);

adc_watchdog_single_channel_enable

The description of adc_watchdog_single_channel_enable is shown as below:

Table 3-32. Function adc_watchdog_single_channel_enable

Function name	adc_watchdog_single_channel_enable
Function prototype	void adc_watchdog_single_channel_enable(uint8_t channel);
Function descriptions	configure ADC analog watchdog single channel
Precondition	-
The called functions	-
Input parameter{in}	
channel	the selected ADC channel
ADC_CHANNEL_x	ADC Channelx(x=0..18)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC analog watchdog single channel */

adc_watchdog_single_channel_enable(ADC_CHANNEL_1);

adc_watchdog_group_channel_enable

The description of adc_watchdog_group_channel_enable is shown as below:

Table 3-33. Function adc_watchdog_group_channel_enable

Function name	adc_watchdog_group_channel_enable
Function prototype	void adc_watchdog_group_channel_enable(uint8_t channel_group);
Function descriptions	configure ADC analog watchdog group channel
Precondition	-
The called functions	-
Input parameter{in}	
channel_group	the channel group use analog watchdog
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
ADC_REGULAR_INSERTED_CHANNEL	both regular and inserted group
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC analog watchdog group channel */

adc_watchdog_group_channel_enable(ADC_REGULAR_CHANNEL);

adc_watchdog_disable

The description of adc_watchdog_disable is shown as below:

Table 3-34. Function adc_watchdog_disable

Function name	adc_watchdog_disable
Function prototype	void adc_watchdog_disable(void);
Function descriptions	disable ADC analog watchdog
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable ADC0 analog watchdog */

adc_watchdog_disable(ADC0);

adc_watchdog_threshold_config

The description of adc_watchdog_threshold_config is shown as below:

Table 3-35. Function adc_watchdog_threshold_config

Function name	adc_watchdog_threshold_config
Function prototype	void adc_watchdog_threshold_config(uint16_t low_threshold, uint16_t high_threshold);
Function descriptions	configure ADC analog watchdog threshold
Precondition	-
The called functions	-
Input parameter{in}	
low_threshold	analog watchdog low threshold, 0..4095
Input parameter{in}	
high_threshold	analog watchdog high threshold, 0..4095
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC analog watchdog threshold */

adc_watchdog_threshold_config(0x0400, 0x0A00);

3.3. CEC

Consumer Electronics Control (CEC) belongs to a part of HDMI (High-Definition Multimedia Interface) standard. CEC, as a kind of protocol, provides the advanced control functions of all kinds 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 3-36. CEC Registers

Registers	Descriptions
CEC_CTL	Control register
CEC_CFG	Configuration register
CEC_TDATA	Transmit data register

Registers	Descriptions
CEC_RDATA	Receive data register
CEC_INTF	Interrupt Flag Register
CEC_INTEN	Interrupt enable register

3.3.2. Descriptions of Peripheral functions

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

Table 3-37. CEC firmware function

Function name	Function description
cec_deinit	reset HDMI-CEC controller
cec_init	configure signal free time,the signal free time counter start option,own address
cec_error_config	configure generate Error-bit, whether stop receive message when detected bit rising error
cec_enable	enable HDMI-CEC controller
cec_disable	disable HDMI-CEC controller
cec_transmission_start	start CEC message transmission
cec_transmission_end	end CEC message transmission
cec_listen_mode_enable	enable CEC listen mode
cec_listen_mode_disable	disable CEC listen mode
cec_own_address_config	configure and clear own address
cec_sft_config	configure signal free time and the signal free time counter start option
cec_generate_errorbit_config	configure generate Error-bit when detected some abnormal situation or not
cec_stop_receive_bre_config	whether stop receive message when detected bit rising error
cec_reception_tolerance_enable	enable reception bit timing tolerance
cec_reception_tolerance_disable	disable reception bit timing tolerance
cec_data_send	send a data by the CEC peripheral
cec_data_receive	receive a data by the CEC peripheral
cec_flag_get	get CEC status
cec_flag_clear	clear CEC status
cec_interrupt_enable	enable CEC interrupt
cec_interrupt_disable	Disable CEC interrupt
cec_interrupt_flag_get	get CEC interrput flag
cec_interrupt_flag_clear	clear CEC interrput flag

cec_deinit

The description of cec_deinit is shown as below:

Table 3-38. Function cec_deinit

Function name	cec_deinit
Function prototype	void cec_deinit(void);
Function descriptions	reset HDMI-CEC controller
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* reset CEC */

cec_deinit();

cec_init

The description of cec_init is shown as below:

Table 3-39. Function cec_init

Function name	cec_init
Function prototype	void cec_init(uint32_t sftmopt, uint32_t sft, uint32_t address);
Function descriptions	configure signal free time,the signal free time counter start option,own address
Precondition	-
The called functions	-
Input parameter{in}	
sftopt	signal free time counter start option
CEC_SFT_START_STAOM	signal free time counter starts counting when STAOM is asserted
CEC_SFT_START_LAST	signal free time counter starts automatically after transmission/reception end
Input parameter{in}	
sft	signal free time
CEC_SFT_PROTOCOL_PERIOD	the signal free time will perform as HDMI-CEC protocol description
CEC_SFT_1POINT5_PERIOD	1.5 nominal data bit periods
CEC_SFT_2POINT5_PERIOD	2.5 nominal data bit periods
CEC_SFT_3POINT5_P	3.5 nominal data bit periods

ERIOD	
CEC_SFT_4POINT5_PERIOD	4.5 nominal data bit periods
CEC_SFT_5POINT5_PERIOD	5.5 nominal data bit periods
CEC_SFT_6POINT5_PERIOD	6.5 nominal data bit periods
CEC_SFT_7POINT5_PERIOD	7.5 nominal data bit periods
Input parameter{in}	
address	own address
CEC_OWN_ADDRESS_CLEAR	own address is cleared
CEC_OWN_ADDRESSx	own address is x
Output parameter{out}	
-	-
Return value	
-	-

Example：

/* init CEC*/

cec_init(CEC_SFT_START_STAOM, CEC_SFT_PROTOCOL_PERIOD, CEC_OWN_ADDRESS0);

cec_error_config

The description of cec_error_config is shown as below:

Table 3-40. Function cec_error_config

Function name	cec_error_config
Function prototype	void cec_error_config(uint32_t broadcast, uint32_t singlecast_lbpe, uint32_t singlecast_bre, uint32_t rxbrestp);
Function descriptions	configure generate Error-bit when detected some abnormal situation or not, whether stop receive message when detected bit rising error
Precondition	-
The called functions	-
Input parameter{in}	
broadcast	generate an Error-bit in broadcast message or not
CEC_BROADCAST_ERROR_BIT_ON	generate Error-bit in broadcast
CEC_BROADCAST_ERROR_BIT_OFF	do not generate Error-bit in broadcast

Input parameter{in}	
singlecast_lbpe	generate an Error-bit when detected BPLE in singlecast
CEC_LONG_PERIOD_ERROR_BIT_ON	generate Error-bit on long bit period error
CEC_LONG_PERIOD_ERROR_BIT_OFF	do not generate Error-bit on long bit period error
Input parameter{in}	
singlecast_bre	generate an Error-bit when detected BRE in singlecast
CEC_RISING_PERIOD_ERROR_BIT_ON	generate Error-bit on bit rising error
CEC_RISING_PERIOD_ERROR_BIT_OFF	do not generate Error-bit on bit rising error
Input parameter{in}	
rxbrestp	whether stop receive message when detected BRE
CEC_STOP_RISING_ERROR_BIT_ON	stop reception when detected bit rising error
CEC_STOP_RISING_ERROR_BIT_OFF	do not stop reception when detected bit rising error
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure generate Error-bit when detected some abnormal situation or not, whether stop receive message when detected bit rising error */

cec_error_config(CEC_BROADCAST_ERROR_BIT_ON, CEC_LONG_PERIOD_ERROR_BIT_ON, CEC_RISING_PERIOD_ERROR_BIT_ON, CEC_STOP_RISING_ERROR_BIT_ON);

cec_enable

The description of cec_enable is shown as below:

Table 3-41. Function cec_enable

Function name	cec_enable
Function prototype	void cec_enable (void);
Function descriptions	enable HDMI-CEC controller
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* enable HDMI-CEC controller */

cec_enable();

cec_disable

The description of cec_disable is shown as below:

Table 3-42. Function cec_disable

Function name	cec_disable
Function prototype	void cec_disable (void);
Function descriptions	disable HDMI-CEC controller
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable HDMI-CEC controller */

cec_disable ();

cec_transmission_start

The description of cec_transmission_start is shown as below:

Table 3-43. Function cec_transmission_start

Function name	cec_transmission_start
Function prototype	void cec_transmission_start (void);
Function descriptions	start CEC message transmission
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* start CEC message transmission */

cec_transmission_start ();

cec_transmission_end

The description of cec_transmission_end is shown as below:

Table 3-44. Function cec_transmission_end

Function name	cec_transmission_end
Function prototype	void cec_transmission_end (void);
Function descriptions	end CEC message transmission
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* end CEC message transmission */

cec_transmission_end ();

cec_listen_mode_enable

The description of cec_listen_mode_enable is shown as below:

Table 3-45. Function cec_listen_mode_enable

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

Example:

/* enable CEC listen mode */

cec_listen_mode_enable ();

cec_listen_mode_disable

The description of cec_listen_mode_disable is shown as below:

Table 3-46. Function cec_listen_mode_disable

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

Example:

/* disable CEC listen mode */

cec_listen_mode_disable ();

cec_own_address_config

The description of cec_own_address_config is shown as below:

Table 3-47. Function cec_own_address_config

Function name	cec_own_address_config
Function prototype	void cec_own_address_config(uint32_t address);
Function descriptions	configure and clear own address.the controller can be configured to multiple own address
Precondition	-
The called functions	-
Input parameter{in}	
address	own address
CEC_OWN_ADDRESS_CLEAR	own address is cleared
CEC_OWN_ADDRESSx	own address is x
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* configure and clear own address.the controller can be configured to multiple own address */

cec_own_address_config (CEC_OWN_ADDRESS_CLEAR);

cec_sft_config

The description of cec_sft_config is shown as below:

Table 3-48. Function cec_sft_config

Function name	cec_sft_config
Function prototype	void cec_sft_config(uint32_t sftmopt, uint32_t sft);
Function descriptions	configure signal free time and the signal free time counter start option
Precondition	-
The called functions	-
Input parameter{in}	
sftopt	signal free time counter start option
CEC_SFT_START_STAOM	signal free time counter starts counting when STAOM is asserted
CEC_SFT_START_LAST	signal free time counter starts automatically after transmission/reception end
Input parameter{in}	
sft	signal free time
CEC_SFT_PROTOCOL_PERIOD	the signal free time will perform as HDMI-CEC protocol description
CEC_SFT_1POINT5_PERIOD	1.5 nominal data bit periods
CEC_SFT_2POINT5_PERIOD	2.5 nominal data bit periods
CEC_SFT_3POINT5_PERIOD	3.5 nominal data bit periods
CEC_SFT_4POINT5_PERIOD	4.5 nominal data bit periods
CEC_SFT_5POINT5_PERIOD	5.5 nominal data bit periods
CEC_SFT_6POINT5_PERIOD	6.5 nominal data bit periods
CEC_SFT_7POINT5_PERIOD	7.5 nominal data bit periods

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure signal free time and the signal free time counter start option */

cec_sft_config (CEC_SFT_START_STAOM, CEC_SFT_PROTOCOL_PERIOD);

cec_generate_errorbit_config

The description of cec_generate_errorbit_config is shown as below:

Table 3-49. Function cec_generate_errorbit_config

Function name	cec_generate_errorbit_config
Function prototype	void cec_generate_errorbit_config(uint32_t broadcast, uint32_t singlecast_lbpe, uint32_t singlecast_bre);
Function descriptions	configure generate Error-bit when detected some abnormal situation or not
Precondition	-
The called functions	-
Input parameter{in}	
broadcast	generate Error-bit in broadcast or not
CEC_BROADCAST_ERROR_BIT_ON	generate Error-bit in broadcast
CEC_BROADCAST_ERROR_BIT_OFF	do not generate Error-bit in broadcast
Input parameter{in}	
singlecast_lbpe	generate Error-bit on long bit period error or not
CEC_LONG_PERIOD_ERROR_BIT_ON	generate Error-bit on long bit period error
CEC_LONG_PERIOD_ERROR_BIT_OFF	do not generate Error-bit on long bit period error
Input parameter{in}	
singlecast_bre	generate Error-bit on bit rising error or not
CEC_RISING_PERIOD_ERROR_BIT_ON	generate Error-bit on bit rising error
CEC_RISING_PERIOD_ERROR_BIT_OFF	do not generate Error-bit on bit rising error
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure generate Error-bit when detected some abnormal situation or not */

cec_generate_errorbit_config (CEC_BROADCAST_ERROR_BIT_ON, CEC_LONG_PERIOD_ERROR_BIT_ON, CEC_RISING_PERIOD_ERROR_BIT_ON);

cec_stop_receive_bre_config

The description of cec_stop_receive_bre_config is shown as below:

Table 3-50. Function cec_stop_receive_bre_config

Function name	cec_stop_receive_bre_config
Function prototype	void cec_stop_receive_bre_config(uint32_t rxbrestp);
Function descriptions	whether stop receive message when detected bit rising error
Precondition	-
The called functions	-
Input parameter{in}	
rxbrestp	stop reception when detected bit rising error or not
CEC_STOP_RISING_ERROR_BIT_ON	stop reception when detected bit rising error
CEC_STOP_RISING_ERROR_BIT_OFF	do not stop reception when detected bit rising error
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* whether stop receive message when detected bit rising error */

cec_stop_receive_bre_config (CEC_STOP_RISING_ERROR_BIT_ON);

cec_reception_tolerance_enable

The description of cec_reception_tolerance_enable is shown as below:

Table 3-51. Function cec_reception_tolerance_enable

Function name	cec_reception_tolerance_enable
Function prototype	void cec_reception_tolerance_enable (void);
Function descriptions	enable reception bit timing tolerance
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable reception bit timing tolerance */

cec_reception_tolerance_enable ();

cec_reception_tolerance_disable

The description of cec_reception_tolerance_disable is shown as below:

Table 3-52. Function cec_reception_tolerance_disable

Function name	cec_reception_tolerance_disable
Function prototype	void cec_reception_tolerance_disable (void);
Function descriptions	disable reception bit timing tolerance
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable reception bit timing tolerance */

cec_reception_tolerance_disable ();

cec_data_send

The description of cec_data_send is shown as below:

Table 3-53. Function cec_data_send

Function name	cec_data_send
Function prototype	void cec_data_send(uint8_t data);
Function descriptions	send a data by the CEC peripheral
Precondition	-
The called functions	-
Input parameter{in}	
data	the data to transmit
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* send a data by the CEC peripheral */

cec_data_send (0x55);

cec_data_receive

The description of cec_data_receive is shown as below:

Table 3-54. Function cec_data_receive

Function name	cec_data_receive
Function prototype	uint8_t cec_data_receive(void);
Function descriptions	receive a data by the CEC peripheral
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
Uint8_t	the data to receive

Example:

/* receive a data by the CEC peripheral */

uint8_t data = 0；

data = cec_data_receive ();

cec_flag_get

The description of cec_flag_get is shown as below:

Table 3-55. Function cec_flag_get

Function name	cec_flag_get
Function prototype	FlagStatus cec_flag_get(uint32_t flag);
Function descriptions	get CEC status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag
CEC_FLAG_BR	Rx-byte data received
CEC_FLAG_REND	end of reception
CEC_FLAG_RO	RX overrun
CEC_FLAG_BRE	bit rising error
CEC_FLAG_BPSE	short bit period error

CEC_FLAG_BPLE	long bit period error
CEC_FLAG_RAE	Rx ACK error
CEC_FLAG_ARBF	arbitration lost
CEC_FLAG_TBR	Tx-byte data request
CEC_FLAG_TEND	transmission successfully end
CEC_FLAG_TU	Tx data buffer underrun
CEC_FLAG_TERR	Tx-error
CEC_FLAG_TAERR	Tx ACK error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get CEC_FLAG_BR status */

FlagStatus flag = reset;

flag = cec_flag_get (CEC_INT_BR);

cec_flag_clear

The description of cec_flag_clear is shown as below:

Table 3-56. Function cec_flag_clear

Function name	cec_flag_clear
Function prototype	void cec_flag_clear(uint32_t flag);
Function descriptions	clear CEC status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag
CEC_FLAG_BR	Rx-byte data received
CEC_FLAG_REND	end of reception
CEC_FLAG_RO	RX overrun
CEC_FLAG_BRE	bit rising error
CEC_FLAG_BPSE	short bit period error
CEC_FLAG_BPLE	long bit period error
CEC_FLAG_RAE	Rx ACK error
CEC_FLAG_ARBF	arbitration lost
CEC_FLAG_TBR	Tx-byte data request
CEC_FLAG_TEND	transmission successfully end
CEC_FLAG_TU	Tx data buffer underrun
CEC_FLAG_TERR	Tx-error
CEC_FLAG_TAERR	Tx ACK error flag

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear CEC_FLAG_BR status */

cec_flag_get (CEC_FLAG_BR);

cec_interrupt_enable

The description of cec_interrupt_enable is shown as below:

Table 3-57. Function cec_interrupt_enable

Function name	cec_interrupt_enable
Function prototype	void cec_interrupt_enable(uint32_t flag);
Function descriptions	enable interrupt
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag
CEC_INT_BR	enable Rx-byte data received interrupt
CEC_INT_REND	enable end of reception interrupt
CEC_INT_RO	enable RX overrun interrupt
CEC_INT_BRE	enable bit rising error interrupt
CEC_INT_BPSE	enable short bit period error interrupt
CEC_INT_BPLE	enable long bit period error interrupt
CEC_INT_RAE	enable Rx ACK error interrupt
CEC_INT_ARBF	enable arbitration lost interrupt
CEC_INT_TBR	enable Tx-byte data request interrupt
CEC_INT_TEND	enable transmission successfully end interrupt
CEC_INT_TU	enable Tx data buffer underrun interrupt
CEC_INT_TERR	enable Tx-error interrupt
CEC_INT_TAERR	enable Tx ACK error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable CEC_INT_BR interrupt */

cec_interrupt_enable (CEC_INT_BR);

cec_interrupt_disable

The description of cec_interrupt_disable is shown as below:

Table 3-58. Function cec_interrupt_disable

Function name	cec_interrupt_disable
Function prototype	void cec_interrupt_disable (uint32_t flag);
Function descriptions	disable interrupt
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag
CEC_INT_BR	disable Rx-byte data received interrupt
CEC_INT_REND	disable end of reception interrupt
CEC_INT_RO	disable RX overrun interrupt
CEC_INT_BRE	disable bit rising error interrupt
CEC_INT_BPSE	disable short bit period error interrupt
CEC_INT_BPLE	disable long bit period error interrupt
CEC_INT_RAE	disable Rx ACK error interrupt
CEC_INT_ARBF	disable arbitration lost interrupt
CEC_INT_TBR	disable Tx-byte data request interrupt
CEC_INT_TEND	disable transmission successfully end interrupt
CEC_INT_TU	disable Tx data buffer underrun interrupt
CEC_INT_TERR	disable Tx-error interrupt
CEC_INT_TAERR	disable Tx ACK error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CEC_INT_BR interrupt */

cec_interrupt_disable (CEC_INT_BR);
```

cec_interrupt_flag_get

The description of cec_interrupt_flag_get is shown as below:

Table 3-59. Function cec_interrupt_flag_get

Function name	cec_interrupt_flag_get
Function prototype	FlagStatus cec_interrupt_flag_get(uint32_t flag);
Function descriptions	get CEC interrput flag
Precondition	-
The called functions	-

Input parameter{in}	
flag	specify which flag
CEC_INT_FLAG_BR	Rx-byte data received
CEC_INT_FLAG_REND	end of reception
CEC_INT_FLAG_RO	RX overrun
CEC_INT_FLAG_BRE	bit rising error
CEC_INT_FLAG_BPSE	short bit period error
CEC_INT_FLAG_BPLE	long bit period error
CEC_INT_FLAG_RAE	Rx ACK error
CEC_INT_FLAG_ARBF	arbitration lost
CEC_INT_FLAG_TBR	Tx-byte data request
CEC_INT_FLAG_TEND	transmission successfully end
CEC_INT_FLAG_TU	Tx data buffer underrun
CEC_INT_FLAG_TERR	Tx-error
CEC_INT_FLAG_TAERR	Tx ACK error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get CEC_INT_FLAG_BR status */

FlagStatus flag = reset;

flag = cec_interrupt_flag_get (CEC_INT_FLAG_BR);
```

cec_interrupt_flag_clear

The description of cec_interrupt_flag_clear is shown as below:

Table 3-60. Function cec_interrupt_flag_clear

Function name	cec_interrupt_flag_clear
Function prototype	void cec_interrupt_flag_clear(uint32_t flag);
Function descriptions	clear CEC interrput flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag
CEC_INT_FLAG_BR	Rx-byte data received
CEC_INT_FLAG_REND	end of reception
CEC_INT_FLAG_RO	RX overrun

CEC_INT_FLAG_BRE	bit rising error
CEC_INT_FLAG_BPSE	short bit period error
CEC_INT_FLAG_BPLE	long bit period error
CEC_INT_FLAG_RAE	Rx ACK error
CEC_INT_FLAG_ARBF	arbitration lost
CEC_INT_FLAG_TBR	Tx-byte data request
CEC_INT_FLAG_TEND	transmission successfully end
CEC_INT_FLAG_TU	Tx data buffer underrun
CEC_INT_FLAG_TERR	Tx-error
CEC_INT_FLAG_TAERR	Tx ACK error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear CEC_INT_FLAG_BR status */

cec_interrupt_flag_clear(CEC_INT_FLAG_BR);

3.4. CMP

The general purpose CMP can work either standalone (all terminal are available on I/Os) or together with the timers. It can 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, achieve some current control by working together with a PWM output of a timer and the DAC. 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 3-61. CMP registers

Registers	Descriptions
CMP_CS	CMP control and status register

3.4.2. Descriptions of Peripheral functions

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

Table 3-62. CMP firmware function

Function name	Function description
cmp_deinit	CMP deinit
cmp_mode_init	CMP mode init
cmp_output_init	CMP output init
cmp_enable	enable CMP
cmp_disable	disable CMP
cmp_switch_enable	enable the switch mode
cmp_switch_disable	disable the switch mode
cmp_window_enable	enable the window mode
cmp_window_disable	disable the window mode
cmp_lock_enable	lock the CMP
cmp_output_level_get	get output level

Enum cmp_enum

Table 3-63. Enum cmp_enum

Member name	Function description
CMP0	comparator 0
CMP1	comparator 1

cmp_deinit

The description of cmp_deinit is shown as below:

Table 3-64. Function cmp_deinit

Function name	cmp_deinit
Function prototype	void cmp_deinit(cmp_enum cmp_periph);
Function descriptions	CMP deinit
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum ***Table 3-63. Enum cmp_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize CMP0 */

cmp_deinit(CMP0);
```

cmp_mode_init

The description of cmp_mode_init is shown as below:

Table 3-65. Function cmp_mode_init

Function name	cmp_mode_init
Function prototype	void cmp_mode_init(cmp_enum cmp_periph, uint32_t operating_mode, uint32_t inverting_input, uint32_t output_hysteresis);
Function descriptions	CMP mode init
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum ***Table 3-63. Enum cmp_enum***
Input parameter{in}	
operating_mode	operating mode
CMP_MODE_HIGHSPEED	high speed mode
CMP_MODE_MIDDLESPEED	medium speed mode
CMP_MODE_LOWSPEED	low speed mode
CMP_MODE_VERYLOWSPEED	very-low speed mode
Input parameter{in}	
inverting_input	inverting input select
CMP_INVERTING_INPUT_1_4VREFINT	VREFINT *1/4 input
CMP_INVERTING_INPUT_1_2VREFINT	VREFINT *1/2 input
CMP_INVERTING_INPUT_3_4VREFINT	VREFINT *3/4 input
CMP_INVERTING_INPUT_VREFINT	VREFINT input
CMP_INVERTING_INPUT_DAC0_OUT0	PA4 (*DAC0_OUT0*) input
CMP_INVERTING_INPUT_PA5	PA5 input
CMP_INVERTING_INPUT_PA0_PA2	PA0 input for CMP0, PA2 input for CMP1
Input parameter{in}	
output_hysteresis	hysteresis level
CMP_HYSTERESIS_NO	output no hysteresis

CMP_HYSTERESIS_LOW	output low hysteresis
CMP_HYSTERESIS_MIDDLE	output middle hysteresis
CMP_HYSTERESIS_HIGH	output high hysteresis
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize CMP0 mode */

cmp_mode_init(CMP0, CMP_MODE_HIGHSPEED, CMP_INVERTING_INPUT_1_4VREFINT, CMP_HYSTERESIS_NO);

cmp_output_init

The description of cmp_output_init is shown as below:

Table 3-66. Function cmp_output_init

Function name	cmp_output_init
Function prototype	void cmp_output_init(cmp_enum cmp_periph, uint32_t output_selection, uint32_t output_polarity);
Function descriptions	CMP output init
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum ***Table 3-63. Enum cmp_enum***
output_selection	CMP output selection
CMP_OUTPUT_NONE	CMP output none
CMP_OUTPUT_TIMER0_BKIN	CMP output TIMER0 break input
CMP_OUTPUT_TIMER0_IC0	CMP output TIMER0_CH0 input capture
CMP_OUTPUT_TIMER0_OCPRECLR	CMP output TIMER0 OCPRE_CLR input
CMP_OUTPUT_TIMER1_IC3	CMP output TIMER1_CH3 input capture
CMP_OUTPUT_TIMER1_OCPRECLR	CMP output TIMER1 OCPRE_CLR input

CMP_OUTPUT_TIMER2_IC0	CMP output TIMER2_CH0 input capture
CMP_OUTPUT_TIMER2_OCPRECLR	CMP output TIMER2 OCPRE_CLR input
Input parameter{in}	
output_polarity	CMP output polarity
CMP_OUTPUT_POLARITY_INVERTED	output is inverted
CMP_OUTPUT_POLARITY_NONINVERTED	output is not inverted
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize CMP0 output */

cmp_output_init(CMP0, CMP_OUTPUT_TIMER0_IC0, CMP_OUTPUT_POLARITY_NONINVERTED);

cmp_enable

The description of cmp_enable is shown as below:

Table 3-67. Function cmp_enable

Function name	cmp_enable
Function prototype	void cmp_enable(cmp_enum cmp_periph);
Function descriptions	enable CMP
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum ***Table 3-63. Enum cmp_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable CMP0 */

cmp_enable(CMP0);

cmp_disable

The description of cmp_disable is shown as below:

Table 3-68. Function cmp_disable

Function name	cmp_disable
Function prototype	void cmp_disable(cmp_enum cmp_periph);
Function descriptions	disable CMP
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum ***Table 3-63. Enum cmp_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable CMP0 */

cmp_disable(CMP0);

cmp_switch_enable

The description of cmp_switch_enable is shown as below:

Table 3-69. Function cmp_switch_enable

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

Example:

/* enable the switch mode */

cmp_switch_enable();

cmp_switch_disable

The description of cmp_switch_disable is shown as below:

Table 3-70. Function cmp_switch_disable

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

Example:

```
/* disable the switch mode */
cmp_switch_disable();
```

cmp_window_enable

The description of cmp_window_enable is shown as below:

Table 3-71. Function cmp_window_enable

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

Example:

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

cmp_window_disable

The description of cmp_window_disable is shown as below:

Table 3-72. Function cmp_window_disable

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

Example:

```
/* disable the window mode */

cmp_window_disable();
```

cmp_lock_enable

The description of cmp_lock_enable is shown as below:

Table 3-73. Function cmp_lock_enable

Function name	cmp_lock_enable
Function prototype	void cmp_lock_enable(cmp_enum cmp_periph);
Function descriptions	lock the CMP
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum ***Table 3-63. Enum cmp_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock CMP0 register */

cmp_lock_enable(CMP0);
```

cmp_output_level_get

The description of cmp_output_level_get is shown as below:

Table 3-74. Function cmp_output_level_get

Function name	cmp_output_level_get
Function prototype	uint32_t cmp_output_level_get(cmp_enum cmp_periph);
Function descriptions	get output level
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum ***Table 3-63. Enum cmp_enum***
Output parameter{out}	
-	-
Return value	
uint32_t	the output level
CMP_OUTPUTLEVEL_HIGH	comparator output high
CMP_OUTPUTLEVEL_LOW	comparator output low

Example:

```
uint32_t level;

/* get CMP0 output level */

level = cmp_output_level_get(CMP0);
```

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 ***0***.

3.5.1. Descriptions of Peripheral registers

CRC registers are listed in the table shown as below:

Table 3-75. CRC Registers

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

3.5.2. Descriptions of Peripheral functions

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

Table 3-76. CRC firmware function

Function name	Function description
crc_deinit	deinit CRC calculation unit
crc_reverse_output_data_enable	enable the reverse operation of output data
crc_reverse_output_data_disable	disable the reverse operation of output data
crc_data_register_reset	reset data register to the value of initializaiton data register
crc_data_register_read	read the data register
crc_free_data_register_read	read the free data register
crc_free_data_register_write	write the free data register
crc_init_data_register_write	write the initial value register
crc_input_data_reverse_config	configure the CRC input data function
crc_single_data_calculate	CRC calculate a 32-bit data
crc_block_data_calculate	CRC calculate a 32-bit data array

crc_deinit

The description of crc_deinit is shown as below:

Table 3-77. Function crc_deinit

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

Example:

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

crc_reverse_output_data_enable

The description of crc_reverse_output_data_enable is shown as below:

Table 3-78. Function crc_reverse_output_data_enable

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

Example:

/* enable CRC reverse operation of output data */

```
crc_reverse_output_data_enable ();
```

crc_reverse_output_data_disable

The description of crc_reverse_output_data_disable is shown as below:

Table 3-79. Function crc_reverse_output_data_disable

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

Example:

/* disable crc reverse operation of output data */

```
crc_reverse_output_data_disable ();
```

crc_data_register_reset

The description of crc_data_register_reset is shown as below:

Table 3-80. Function crc_data_register_reset

Function name	crc_data_register_reset

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

Example:

```
/* reset crc data register */

crc_data_register_reset ();
```

crc_data_register_read

The description of crc_data_register_read is shown as below:

Table 3-81. Function crc_data_register_read

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

Example:

```
/* read crc data register */

uint32_t crc_value = 0;

crc_value = crc_data_register_read();
```

crc_free_data_register_read

The description of crc_free_data_register_read is shown as below:

Table 3-82. Function crc_free_data_register_read

Function name	crc_free_data_register_read

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

Example:

```
/* read crc free data register */
uint8_t crc_value = 0;
crc_value = crc_free_data_register_read();
```

crc_free_data_register_write

The description of crc_free_data_register_write is shown as below:

Table 3-83. Function crc_free_data_register_write

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

Example:

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

crc_init_data_register_write

The description of crc_init_data_register_write is shown as below:

Table 3-84. Function crc_init_data_register_write

Function name	crc_init_data_register_write

Function prototype	void crc_init_data_register_write(uint32_t init_data)
Function descriptions	write the initializaiton data register
Precondition	-
The called functions	-
Input parameter{in}	
init_data	specify 32-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* write crc initializaiton data register */

crc_init_data_register_write (0x11223344);

crc_input_data_reverse_config

The description of crc_input_data_reverse_config is shown as below:

Table 3-85. Function crc_input_data_reverse_config

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

Example:

/* configure the crc input data*/

crc_input_data_reverse_config (CRC_INPUT_DATA_WORD);

crc_single_data_calculate

The description of crc_single_data_calculate is shown as below:

Table 3-86. Function crc_single_data_calculate

Function name	crc_single_data_calculate
Function prototype	uint32_t crc_single_data_calculate(uint32_t sdata);
Function descriptions	CRC calculate a 32-bit data
Precondition	-
The called functions	-
Input parameter{in}	
sdata	specify 32-bit data
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit CRC calculate value (0-0xFFFFFFFF)

Example:

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

uint32_t val = 0, valcrc = 0;

val = (uint32_t)0xabcd1234;

rcu_periph_clock_enable(RCU_CRC);

valcrc = crc_single_data_calculate(val);
```

crc_block_data_calculate

The description of crc_block_data_calculate is shown as below:

Table 3-87. Function crc_block_data_calculate

Function name	crc_block_data_calculate
Function prototype	uint32_t crc_block_data_calculate(uint32_t array[], uint32_t size);
Function descriptions	calculate the CRC value of an array of 32-bit values
Precondition	-
The called functions	-
Input parameter{in}	
array	pointer to an array of 32 bit data words
Input parameter{in}	
size	size of the array
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit CRC calculate value (0-0xFFFFFFFF)

Example:

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

#define BUFFER_SIZE        6

uint32_t valcrc = 0;

static const uint32_t data_buffer[BUFFER_SIZE] = {

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

rcu_periph_clock_enable(RCU_CRC);

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

3.6. 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 ***3.6.1*** the DAC firmware functions are introduced in chapter ***3.6.2***.

3.6.1. Peripheral register description

DAC registers are listed in the table shown as below:

Table 3-88. DAC Registers

Register	Descriptions
DAC_CTL0	DACx control register 0
DAC_SWT	DACx software trigger register
DAC_OUT0_R12DH	DACx_OUT0 12-bit right-aligned data holding register
DAC_OUT0_L12DH	DACx_OUT0 12-bit left-aligned data holding register
DAC_OUT0_R8DH	DACx_OUT0 8-bit right-aligned data holding register
DAC_OUT0_DO	DACx_OUT0 data output register
DAC_STAT0	DACx status register 0

3.6.2. Descriptions of Peripheral functions

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

Table 3-89. DAC firmware functions

Function name	Function description
dac_deinit	deinitialize DAC
dac_enable	enable DAC
dac_disable	disable DAC
dac_dma_enable	enable DAC DMA function
dac_dma_disable	disable DAC DMA function

Function name	Function description
dac_output_buffer_enable	enable DAC output buffer
dac_output_buffer_disable	disable DAC output buffer
dac_output_value_get	get DAC output value
dac_data_set	set DAC data holding register value
dac_trigger_enable	enable DAC trigger
dac_trigger_disable	disable DAC trigger
dac_trigger_source_config	configure DAC trigger source
dac_software_trigger_enable	enable DAC software trigger
dac_flag_get	get DAC flag
dac_flag_clear	clear DAC flag
dac_interrupt_enable	enable DAC interrupt
dac_interrupt_disable	disable DAC interrupt
dac_interrupt_flag_get	get DAC interrupt flag
dac_interrupt_flag_clear	clear DAC interrupt flag

dac_deinit

The description of dac_deinit is shown as below:

Table 3-90. Function dac_deinit

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

Example:

/* deinitialize DAC0 */

dac_deinit(DAC0);

dac_enable

The description of dac_enable is shown as below:

Table 3-91. Function dac_enable

Function name	dac_enable

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

Example:

/* enable DAC0_OUT0 */

dac_enable(DAC0, DAC_OUT0);

dac_disable

The description of dac_disable is shown as below:

Table 3-92. Function dac_disable

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

Example:

/* disable DAC0_OUT0 */

dac_disable(DAC0, DAC_OUT0);

dac_dma_enable

The description of dac_dma_enable is shown as below:

Table 3-93. Function dac_dma_enable

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

Example:

/* enable DAC0_OUT0 DMA function */

dac_dma_enable(DAC0, DAC_OUT0);

dac_dma_disable

The description of dac_dma_disable is shown as below:

Table 3-94. Function dac_dma_disable

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

-	-

Example:

/* disable DAC0_OUT0 DMA function */

dac_dma_disable(DAC0, DAC_OUT0);

dac_output_buffer_enable

The description of dac_output_buffer_enable is shown as below:

Table 3-95. Function dac_output_buffer_enable

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

Example:

/* enable DAC0_OUT0 output buffer */

dac_output_buffer_enable(DAC0, DAC_OUT0);

dac_output_buffer_disable

The description of dac_output_buffer_disable is shown as below:

Table 3-96. Function dac_output_buffer_disable

Function name	dac_output_buffer_disable
Function prototype	void dac_output_buffer_disable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	disable DAC output buffer
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)

Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DAC0_OUT0 output buffer */

dac_output_buffer_disable(DAC0, DAC_OUT0);

dac_output_value_get

The description of dac_output_value_get is shown as below:

Table 3-97. Function dac_output_value_get

Function name	dac_output_value_get
Function prototype	uint16_t dac_output_value_get(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	get DAC output value
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0)
Output parameter{out}	
-	-
Return value	
uint16_t	DAC output data (0~4095)

Example:

/* get the DAC0_OUT0 last data output value */

Uint16_t data;

data = dac_output_value_get(DAC0, DAC_OUT0);

dac_data_set

The description of dac_data_set is shown as below:

Table 3-98. Function dac_data_set

Function name	dac_data_set
Function prototype	void dac_data_set(uint32_t dac_periph, uint8_t dac_out, uint32_t dac_align, uint16_t data);
Function descriptions	set DAC data holding register value
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0)
Input parameter{in}	
dac_align	DAC data alignment mode
DAC_ALIGN_12B_R	12-bit right-aligned data
DAC_ALIGN_12B_L	12-bit left-aligned data
DAC_ALIGN_8B_R	8-bit right-aligned data
Input parameter{in}	
data	data to be loaded (0~4095)
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

dac_trigger_enable

The description of dac_trigger_enable is shown as below:

Table 3-99. Function dac_trigger_enable

Function name	dac_trigger_enable
Function prototype	void dac_trigger_enable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	enable DAC trigger
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	

dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DAC0_OUT0 trigger */

dac_trigger_enable(DAC0, DAC_OUT0);

dac_trigger_disable

The description of dac_trigger_disable is shown as below:

Table 3-100. Function dac_trigger_disable

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

Example:

/* disable DAC0_OUT0 trigger */

dac_trigger_disable(DAC0, DAC_OUT0);

dac_trigger_source_config

The description of dac_trigger_source_config is shown as below:

Table 3-101. Function dac_trigger_source_config

Function name	dac_trigger_source_config
Function prototype	void dac_trigger_source_config(uint32_t dac_periph, uint8_t dac_out, uint32_t triggersource);

Function descriptions	configure DAC trigger source
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0)
Input parameter{in}	
triggersource	external trigger of DAC
DAC_TRIGGER_T5_TRGO	TIMER5 TRGO
DAC_TRIGGER_T14_TRGO	TIMER14 TRGO
DAC_TRIGGER_T2_TRGO	TIMER2 TRGO
DAC_TRIGGER_T1_TRGO	TIMER1 TRGO
DAC_TRIGGER_EXTI_9	EXTI interrupt line9 event
DAC_TRIGGER_SOFTWARE	software trigger
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DAC0_OUT0 trigger source */

dac_trigger_source_config(DAC0, DAC_OUT0, DAC_TRIGGER_T1_TRGO);
```

dac_software_trigger_enable

The description of dac_software_trigger_enable is shown as below:

Table 3-102. Function dac_software_trigger_enable

Function name	dac_software_trigger_enable
Function prototype	void dac_software_trigger_enable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	enable DAC software trigger
Precondition	-
The called functions	-
Input parameter{in}	

dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DAC0_OUT0 software trigger */

dac_software_trigger_enable(DAC0, DAC_OUT0);

dac_flag_get

The description of dac_flag_get is shown as below:

Table 3-103. Function dac_flag_get

Function name	dac_flag_get
Function prototype	FlagStatus dac_flag_get(uint32_t dac_periph, uint32_t flag);
Function descriptions	get DAC flag
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
flag	the DAC status flags
DAC_FLAG_DDUDR0	DACx_OUT0 DMA underrun flag
Output parameter{out}	
-	-
Return value	
FlagStatus	the state of DAC bit（SET or RESET）

Example:

/* get DAC0 flag */

FlagStatus flag;

flag = dac_flag_get(DAC0, DAC_FLAG_DDUDR0);

dac_flag_clear

The description of dac_flag_clear is shown as below:

Table 3-104. Function dac_flag_clear

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

Example:

/* clear DAC0 flag */

dac_flag_clear(DAC0, DAC_FLAG_DDUDR0);

dac_interrupt_enable

The description of dac_interrupt_enable is shown as below:

Table 3-105. Function dac_interrupt_enable

Function name	dac_interrupt_enable
Function prototype	void dac_interrupt_enable(uint32_t dac_periph, uint32_t interrupt);
Function descriptions	enable DAC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
interrupt	the DAC interrupt
DAC_INT_DDUDR0	DACx_OUT0 DMA underrun interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DAC0 interrupt */

dac_interrupt_enable (DAC0, DAC_INT_DDUDRIE0);

dac_interrupt_disable

The description of dac_interrupt_disable is shown as below:

Table 3-106. Function dac_interrupt_disable

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

Example:

/* disable DAC0 interrupt */

dac_interrupt_disable (DAC0, DAC_INT_DDUDRIE0);

dac_interrupt_flag_get

The description of dac_interrupt_flag_get is shown as below:

Table 3-107. Function dac_interrupt_flag_get

Function name	dac_interrupt_flag_get
Function prototype	FlagStatus dac_interrupt_flag_get(uint32_t dac_periph, uint32_t int_flag);
Function descriptions	get DAC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
int_flag	DAC interrupt flag

DAC_INT_FLAG_DDUDR0	DACx_OUT0 DMA underrun interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	the state of DAC interrupt flag(SET or RESET)

Example:

/* get DAC0 interrupt flag */

FlagStatus flag;

flag = dac_interrupt_flag_get(DAC0, DAC_INT_FLAG_DDUDR0);

dac_interrupt_flag_clear

The description of dac_interrupt_flag_clear is shown as below:

Table 3-108. Function dac_interrupt_flag_clear

Function name	dac_interrupt_flag_clear
Function prototype	void dac_interrupt_flag_clear(uint32_t dac_periph, uint32_t int_flag);
Function descriptions	clear DAC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
int_flag	DAC interrupt flag
DAC_INT_FLAG_DDUDR0	DACx_OUT0 DMA underrun interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear DAC0 interrupt flag */

dac_interrupt_flag_clear(DAC0, DAC_INT_FLAG_DDUDR0);

3.7. DBG

The DBG hold unit helps debugger to debug power saving mode. The DBG registers are listed in chapter ***3.7.1***. the DBG firmware functions are introduced in chapter ***3.7.2***.

3.7.1. Descriptions of Peripheral registers

DBG registers are listed in the table shown as below:

Table 3-109. DBG Registers

Registers	Descriptions
DBG_ID	DBG ID code register
DBG_CTL0	DBG control register0
DBG_CTL1	DBG control register1

3.7.2. Descriptions of Peripheral functions

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

Table 3-110. DBG firmware function

Function name	Function description
dbg_deinit	reset DBG register
dbg_id_get	read DBG_ID code register
dbg_low_power_enable	enable low power behavior when the MCU is in debug mode
dbg_low_power_disable	disable low power behavior when the MCU is in debug mode
dbg_periph_enable	enable peripheral behavior when the MCU is in debug mode
dbg_periph_disable	disable peripheral behavior when the MCU is in debug mode

Enum dbg_periph_enum

Table 3-111. Enum dbg_periph_enum

Member name	Function description
DBG_FWDGT_HOLD	debug FWDGT kept when core is halted
DBG_WWDGT_HOLD	debug WWDGT kept when core is halted
DBG_TIMER0_HOLD	hold TIMER0 counter when core is halted
DBG_TIMER1_HOLD	hold TIMER1 counter when core is halted
DBG_TIMER2_HOLD	hold TIMER2 counter when core is halted
DBG_TIMER5_HOLD	hold TIMER5 counter when core is halted
DBG_TIMER13_HOLD	hold TIMER13 counter when core is halted
DBG_TIMER14_HOLD	hold TIMER14 counter when core is halted
DBG_TIMER15_HOLD	hold TIMER15 counter when core is halted
DBG_TIMER16_HOLD	hold TIMER16 counter when core is halted
DBG_I2C0_HOLD	hold I2C0 smbus when core is halted
DBG_I2C1_HOLD	hold I2C1 smbus when core is halted
DBG_I2C2_HOLD	hold I2C2 smbus when core is halted
DBG_RTC_HOLD	hold RTC counter when core is halted

dbg_deinit

The description of dbg_deinit is shown as below:

Table 3-112. Function dbg_deinit

Function name	dbg_deinit
Function prototype	void dbg_deinit (void);
Function descriptions	deinitialize the DBG
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the DBG*/

dbg_deinit();
```

dbg_id_get

The description of dbg_id_get is shown as below:

Table 3-113. Function dbg_id_get

Function name	dbg_id_get
Function prototype	uint32_t dbg_id_get(void);
Function descriptions	Read DBG_ID code register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	DBG_ID code (0-0xFFFFFFFF)

Example:

```
/* read DBG_ID code register */

uint32_t id_value = 0;

id_value = dbg_id_get();
```

dbg_low_power_enable

The description of dbg_low_power_enable is shown as below:

Table 3-114. Function dbg_low_power_enable

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

Example:

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

dbg_low_power_enable(DBG_LOW_POWER_SLEEP);
```

dbg_low_power_disable

The description of dbg_low_power_disable is shown as below:

Table 3-115. Function dbg_low_power_disable

Function name	dbg_low_power_disable
Function prototype	void dbg_low_power_disable(uint32_t dbg_low_power);
Function descriptions	Disable low power behavior when the mcu is in debug mode
Precondition	-
The called functions	-
Input parameter{in}	
dbg_low_power	low power mode
DBG_LOW_POWER_SLEEP	keep debugger connection during sleep mode
DBG_LOW_POWER_DEEPSLEEP	keep debugger connection during deepsleep mode
DBG_LOW_POWER_S	keep debugger connection during standby mode

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

Example:

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

dbg_low_power_disable(DBG_LOW_POWER_SLEEP);

dbg_periph_enable

The description of dbg_periph_enable is shown as below:

Table 3-116. Function dbg_periph_enable

Function name	dbg_periph_enable
Function prototype	void dbg_periph_enable(dbg_periph_enum dbg_periph);
Function descriptions	Enable peripheral behavior when the mcu is in debug mode
Precondition	-
The called functions	-
Input parameter{in}	
dbg_periph	Peripheral refer to ***Table 3-111. Enum dbg_periph_enum***
DBG_FWDGT_HOLD	debug FWDGT kept when core is halted
DBG_WWDGT_HOLD	debug WWDGT kept when core is halted
DBG_TIMERx_HOLD	x=0,1,2,5,13,14,15,16, hold TIMERx counter when core is halted
DBG_I2Cx_HOLD	x=0,1, hold I2Cx smbus when core is halted
DBG_RTC_HOLD	hold RTC counter when core is halted
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

dbg_periph_enable(DBG_TIMER0_HOLD);

dbg_periph_disable

The description of dbg_periph_disable is shown as below:

Table 3-117. Function dbg_periph_disable

Function name	dbg_periph_disable
Function prototype	void dbg_periph_disable(dbg_periph_enum dbg_periph);
Function descriptions	Disable peripheral behavior when the mcu is in debug mode

Precondition	-
The called functions	-
Input parameter{in}	
dbg_periph	peripheral refer to ***Table 3-111. Enum dbg_periph_enum***
DBG_FWDGT_HOLD	debug FWDGT kept when core is halted
DBG_WWDGT_HOLD	debug WWDGT kept when core is halted
DBG_TIMERx_HOLD	x=0,1,2,5,13,14,15,16, hold TIMERx counter when core is halted
DBG_I2Cx_HOLD	x=0,1, hold I2Cx smbus when core is halted
DBG_RTC_HOLD	hold RTC counter when core is halted
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable peripheral behavior when the mcu is in debug mode */

dbg_periph_disable(DBG_TIMER0_HOLD);

3.8. 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 ***3.8.1***, the DMA firmware functions are introduced in chapter ***3.8.2***.

3.8.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

Table 3-118. DMA Registers

Registers	Descriptions
DMA_INTF	Interrupt flag register
DMA_INTC	Interrupt flag clear register
DMA_CHxCTL (x=0..6)	Channel x control register
DMA_CHxCNT (x=0..6)	Channel x counter register
DMA_CHxPADDR (x=0..6)	Channel x peripheral base address register
DMA_CHxMADDR (x=0..6)	Channel x memory base address register

3.8.2. Descriptions of Peripheral functions

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

Table 3-119. DMA firmware function

Function name	Function description
dma_deinit	deinitialize DMA a channel registers
dma_struct_para_init	initialize the parameters of DMA struct with the default values
dma_init	initialize DMA channel
dma_circulation_enable	enable DMA circulation mode
dma_circulation_disable	disable DMA circulation mode
dma_memory_to_memory_enable	enable memory to memory mode
dma_memory_to_memory_disable	disable memory to memory mode
dma_channel_enable	enable DMA channel
dma_channel_disable	disable DMA channel
dma_periph_address_config	set DMA peripheral base address
dma_memory_address_config	set DMA memory base address
dma_transfer_number_config	set the number of remaining data to be transferred by the DMA
dma_transfer_number_get	get the number of remaining data to be transferred by the DMA
dma_priority_config	configure priority level of DMA channel
dma_memory_width_config	configure transfer data size of memory
dma_periph_width_config	configure transfer data size of peripheral
dma_memory_increase_enable	enable next address increasement algorithm of memory
dma_memory_increase_disable	disable next address increasement algorithm of memory
dma_periph_increase_enable	enable next address increasement algorithm of peripheral
dma_periph_increase_disable	disable next address increasement algorithm of peripheral
dma_transfer_direction_config	configure the direction of data transfer on the channel
dma_flag_get	check DMA flag is set or not
dma_flag_clear	clear DMA a channel flag
dma_interrupt_enable	enable DMA interrupt
dma_interrupt_disable	disable DMA interrupt
dma_interrupt_flag_get	check DMA flag and interrupt enable bit is set or not
dma_interrupt_flag_clear	clear DMA a channel flag

dma_channel_enum

Table 3-120. Enum 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

Structure dma_parameter_struct

Table 3-121. Structure dma_parameter_struct

Member name	Function description
periph_addr	peripheral base address
periph_width	transfer data size of peripheral
periph_inc	peripheral increasing mode
memory_addr	memory base address
memory_width	transfer data size of memory
memory_inc	memory increasing mode
direction	channel data transfer direction
number	channel transfer number
priority	channel priority level

dma_deinit

The description of dma_deinit is shown as below:

Table 3-122. Function dma_deinit

Function name	dma_deinit
Function prototype	void dma_deinit(dma_channel_enum channelx);
Function descriptions	deinitialize DMA a channel registers
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA channel0 initialize */

dma_deinit(DMA_CH0);
```

dma_struct_para_init

The description of dma_struct_para_init is shown as below:

Table 3-123. Function dma_para_init

Function name	dma_struct_para_init
Function prototype	void dma_struct_para_init(dma_parameter_struct* init_struct);
Function descriptions	initialize the parameters of DMA struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to ***Table 3-121. Structure dma_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize the parameters of DMA */

dma_parameter_struct dma_init_struct;

dma_struct_para_init(&dma_init_struct);

dma_init

The description of dma_init is shown as below:

Table 3-124. Function dma_init

Function name	dma_init
Function prototype	void dma_init(dma_channel_enum channelx, dma_parameter_struct init_struct);
Function descriptions	initialize DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to ***Table 3-121. Structure dma_parameter_struct***.
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* DMA channel0 initialize */
dma_parameter_struct dma_init_struct;
dma_init_struct.direction = DMA_PERIPHERAL_TO_MEMORY;
dma_init_struct.memory_addr = (uint32_t)g_destbuf;
dma_init_struct.memory_inc = DMA_MEMORY_INCREASE_ENABLE;
dma_init_struct.memory_width = DMA_MEMORY_WIDTH_8BIT;
dma_init_struct.number = TRANSFER_NUM;
dma_init_struct.periph_addr = (uint32_t)BANK0_WRITE_START_ADDR;
dma_init_struct.periph_inc = DMA_PERIPH_INCREASE_ENABLE;
dma_init_struct.periph_width = DMA_PERIPHERAL_WIDTH_8BIT;
dma_init_struct.priority = DMA_PRIORITY_ULTRA_HIGH;
dma_init(DMA_CH0, dma_init_struct);
```

dma_circulation_enable

The description of dma_circulation_enable is shown as below:

Table 3-125. Function dma_circulation_enable

Function name	dma_circulation_enable
Function prototype	void dma_circulation_enable(dma_channel_enum channelx);
Function descriptions	enable DMA circulation mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA channel0 circulation mode */
```

dma_circulation_enable(DMA_CH0);

dma_circulation_disable

The description of dma_circulation_disable is shown as below:

Table 3-126. Function dma_circulation_disable

Function name	dma_circulation_disable
Function prototype	void dma_circulation_disable(dma_channel_enum channelx);
Function descriptions	disable DMA circulation mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DMA channel0 circulation mode */

dma_circulation_disable(DMA_CH0);

dma_memory_to_memory_enable

The description of dma_memory_to_memory_enable is shown as below:

Table 3-127. Function dma_memory_to_memory_enable

Function name	dma_memory_to_memory_enable
Function prototype	void dma_memory_to_memory_enable(dma_channel_enum channelx);
Function descriptions	enable memory to memory mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DMA channel0 memory to memory mode */

dma_memory_to_memory_enable(DMA_CH0);

dma_memory_to_memory_disable

The description of dma_memory_to_memory_disable is shown as below:

Table 3-128. Function dma_memory_to_memory_disable

Function name	dma_memory_to_memory_disable
Function prototype	void dma_memory_to_memory_disable(dma_channel_enum channelx);
Function descriptions	disable memory to memory mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*disable DMA channel0 memory to memory mode */

dma_memory_to_memory_disable(DMA_CH0);

dma_channel_enable

The description of dma_channel_enable is shown as below:

Table 3-129. Function dma_channel_enable

Function name	dma_channel_enable
Function prototype	void dma_channel_enable(dma_channel_enum channelx);
Function descriptions	enable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DMA channel0 */

dma_channel_enable(DMA_CH0);

dma_channel_disable

The description of dma_channel_disable is shown as below:

Table 3-130. Function dma_channel_disable

Function name	dma_channel_disable
Function prototype	void dma_channel_disable(dma_channel_enum channelx);
Function descriptions	disable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DMA channel0 */

dma_channel_disable(DMA_CH0);

dma_periph_address_config

The description of dma_periph_address_config is shown as below:

Table 3-131. Function dma_periph_address_config

Function name	dma_periph_address_config
Function prototype	void dma_periph_address_config(dma_channel_enum channelx, uint32_t address);
Function descriptions	set DMA peripheral base address
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	

address	peripheral base address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel0 periph address */
#define BANK0_WRITE_START_ADDR                  ((uint32_t)0x08004000)
dma_periph_address_config(DMA_CH0, BANK0_WRITE_START_ADDR);
```

dma_memory_address_config

The description of dma_memory_address_config is shown as below:

Table 3-132. Function dma_memory_address_config

Function name	dma_memory_address_config
Function prototype	void dma_memory_address_config(dma_channel_enum channelx, uint32_t address);
Function descriptions	set DMA memory base address
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
address	memory base address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel0 memory address */
uint8_t g_destbuf[TRANSFER_NUM];
dma_memory_address_config(DMA_CH0, (uint32_t) g_destbuf);
```

dma_transfer_number_config

The description of dma_transfer_number_config is shown as below:

Table 3-133. Function dma_transfer_number_config

Function name	dma_transfer_number_config
Function prototype	void dma_transfer_number_config(dma_channel_enum channelx, uint32_t number);
Function descriptions	set the number of remaining data to be transferred by the DMA
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
number	data transfer number(0x00000000-0x0000FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel0 transfer number */
#define TRANSFER_NUM                    0x400
dma_transfer_number_config(DMA_CH0, TRANSFER_NUM);
```

dma_transfer_number_get

The description of dma_transfer_number_get is shown as below:

Table 3-134. Function dma_transfer_number_get

Function name	dma_transfer_number_get
Function prototype	uint32_t dma_transfer_number_get(dma_channel_enum channelx);
Function descriptions	get the number of remaining data to be transferred by the DMA
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
uint32_t	DMA data transmission remaining quantity (0x00000000-0x0000FFFF)

Example:

/* get DMA channel0 transfer number */

uint32_t number = 0;

number = dma_transfer_number_get(DMA0, DMA_CH0);

dma_priority_config

The description of dma_priority_config is shown as below:

Table 3-135. Function dma_priority_config

Function name	dma_priority_config
Function prototype	void dma_priority_config(dma_channel_enum channelx, uint32_t priority);
Function descriptions	configure priority level of DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
priority	priority Level of this channel
DMA_PRIORITY_LOW	low priority
DMA_PRIORITY_MEDIUM	medium priority
DMA_PRIORITY_HIGH	high priority
DMA_PRIORITY_ULTRA_HIGH	ultra high priority
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure DMA channel0 priority */

dma_priority_config(DMA_CH0, DMA_PRIORITY_ULTRA_HIGH);

dma_memory_width_config

The description of dma_memory_width_config is shown as below:

Table 3-136. Function dma_memory_width_config

Function name	dma_memory_width_config
Function prototype	void dma_memory_width_config(dma_channel_enum channelx, uint32_t mwidth);

Function descriptions	configure transfer data size of memory
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
mwidth	transfer data width of memory
DMA_MEMORY_WIDTH_8BIT	transfer data width of memory is 8-bit
DMA_MEMORY_WIDTH_16BIT	transfer data width of memory is 16-bit
DMA_MEMORY_WIDTH_32BIT	transfer data width of memory is 32-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel0 memory width */
dma_memory_width_config(DMA_CH0, DMA_MEMORY_WIDTH_8BIT);
```

dma_periph_width_config

The description of dma_periph_width_config is shown as below:

Table 3-137. Function dma_periph_width_config

Function name	dma_periph_width_config
Function prototype	void dma_periph_width_config(dma_channel_enum channelx, uint32_t pwidth);
Function descriptions	configure transfer data width of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
pwidth	transfer data width of peripheral
DMA_PERIPHERAL_WIDTH_8BIT	transfer data width of peripheral is 8-bit

DMA_PERIPHERAL_WIDTH_16BIT	transfer data width of peripheral is 16-bit
DMA_PERIPHERAL_WIDTH_32BIT	transfer data width of peripheral is 32-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure DMA channel0 periph width */

dma_periph_width_config(DMA_CH0, DMA_PERIPHERAL_WIDTH_8BIT);

dma_memory_increase_enable

The description of dma_memory_increase_enable is shown as below:

Table 3-138. Function dma_memory_increase_enable

Function name	dma_memory_increase_enable
Function prototype	void dma_memory_increase_enable(dma_channel_enum channelx);
Function descriptions	enable next address increasement algorithm of memory
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DMA channel0 memory increase */

dma_memory_increase_enable(DMA_CH0);

dma_memory_increase_disable

The description of dma_memory_increase_disable is shown as below:

Table 3-139. Function dma_memory_increase_disable

Function name	dma_memory_increase_disable
Function prototype	void dma_memory_increase_disable(dma_channel_enum channelx);
Function descriptions	disable next address increasement algorithm of memory

Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DMA channel0 memory increase */

dma_memory_increase_ disable(DMA_CH0);

dma_periph_increase_enable

The description of dma_periph_increase_enable is shown as below:

Table 3-140. Function dma_periph_increase_enable

Function name	dma_periph_increase_enable
Function prototype	void dma_periph_increase_enable(dma_channel_enum channelx);
Function descriptions	enable next address increasement algorithm of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DMA channel0 periph increase */

dma_periph_increase_enable(DMA_CH0);

dma_periph_increase_disable

The description of dma_periph_increase_disable is shown as below:

Table 3-141. Function dma_periph_increase_disable

Function name	dma_periph_increase_disable

Function prototype	void dma_periph_increase_disable(dma_channel_enum channelx);
Function descriptions	disable next address increasement algorithm of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA channel0 periph increase */

dma_periph_increase_disable(DMA_CH0);
```

dma_transfer_direction_config

The description of dma_transfer_direction_config is shown as below:

Table 3-142. Function dma_transfer_direction_config

Function name	dma_transfer_direction_config
Function prototype	void dma_transfer_direction_config(dma_channel_enum channelx, uint8_t direction);
Function descriptions	configure the direction of data transfer on the channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
direction	specify the direction of data transfer
DMA_PERIPHERAL_TO_MEMORY	read from peripheral and write to memory
DMA_MEMORY_TO_PERIPHERAL	read from memory and write to peripheral
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure DMA channel0 transfer direction */

dma_transfer_direction_config(DMA_CH0, DMA_PERIPHERAL_TO_MEMORY);

dma_flag_get

The description of dma_flag_get is shown as below:

Table 3-143. Function dma_flag_get

Function name	dma_flag_get
Function prototype	FlagStatus dma_flag_get(dma_channel_enum channelx, uint32_t flag);
Function descriptions	check DMA flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
flag	specify get which flag
DMA_FLAG_G	global interrupt flag of channel
DMA_FLAG_FTF	full transfer finish flag of channel
DMA_FLAG_HTF	half transfer finish flag of channel
DMA_FLAG_ERR	error flag of channel
Output parameter{out}	
-	**-**
Return value	
FlagStatus	SET or RESET

Example:

/* get DMA channel0 flag */

FlagStatus flag = RESET;

flag = dma_flag_get(DMA_CH0, DMA_FLAG_FTF);

dma_flag_clear

The description of dma_flag_clear is shown as below:

Table 3-144. Function dma_flag_clear

Function name	dma_flag_clear
Function prototype	void dma_flag_clear(dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear DMA a channel flag
Precondition	-
The called functions	-

Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
flag	specify get which flag
DMA_FLAG_G	global interrupt flag of channel
DMA_FLAG_FTF	full transfer finish flag of channel
DMA_FLAG_HTF	half transfer finish flag of channel
DMA_FLAG_ERR	error flag of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear DMA channel0 flag */

dma_flag_clear(DMA_CH0, DMA_FLAG_FTF);

dma_interrupt_enable

The description of dma_interrupt_enable is shown as below:

Table 3-145. Function dma_interrupt_enable

Function name	dma_interrupt_enable
Function prototype	void dma_interrupt_enable(dma_channel_enum channelx, uint32_t source);
Function descriptions	enable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
source	DMA interrupt source
DMA_INT_FTF	full transfer finish interrupt of channel
DMA_INT_HTF	half transfer finish interrupt of channel
DMA_INT_ERR	error interrupt of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DMA channel0 interrupt */

dma_interrupt_enable(DMA_CH0, DMA_INT_FTF);

dma_interrupt_disable

The description of dma_interrupt_disable is shown as below:

Table 3-146. Function dma_interrupt_disable

Function name	dma_interrupt_disable
Function prototype	void dma_interrupt_disable(dma_channel_enum channelx, uint32_t source);
Function descriptions	disable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum dma_channel_enum***.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
source	DMA interrupt source
DMA_INT_FTF	full transfer finish interrupt of channel
DMA_INT_HTF	half transfer finish interrupt of channel
DMA_INT_ERR	error interrupt of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DMA channel0 interrupt */

dma_interrupt_disable(DMA_CH0, DMA_INT_FTF);

dma_interrupt_flag_get

The description of dma_interrupt_flag_get is shown as below:

Table 3-147. Function dma_interrupt_flag_get

Function name	dma_interrupt_flag_get
Function prototype	FlagStatus dma_interrupt_flag_get(dma_channel_enum channelx, uint32_t flag);
Function descriptions	check DMA flag and interrupt enable bit is set or not
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to ***Table 3-120. Enum***

	dma_channel_enum.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
flag	specify get which flag
DMA_INT_FLAG_FTF	full transfer finish interrupt flag of channel
DMA_INT_FLAG_HTF	half transfer finish interrupt flag of channel
DMA_INT_FLAG_ERR	error interrupt flag of channel
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get DMA interrupt_flag */

if(dma_interrupt_flag_get(DMA_CH3, DMA_INT_FLAG_FTF)){
        dma_interrupt_flag_clear(DMA_CH3, DMA_INT_FLAG_G);
        dma_interrupt_flag_clear(DMA_CH3, DMA_INT_FLAG_FTF);

}
```

dma_interrupt_flag_clear

The description of dma_interrupt_flag_clear is shown as below:

Table 3-148. Function dma_interrupt_flag_clear

Function name	dma_interrupt_flag_clear
Function prototype	void dma_interrupt_flag_clear(dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear DMA a channel flag
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMA channel to set, refer to Table 3-120. Enum_ dma_channel_enum.
DMA_CHx (x=0..6)	DMA channel selection
Input parameter{in}	
flag	specify get which flag
DMA_INT_FLAG_G	global interrupt flag of channel
DMA_INT_FLAG_FTF	full transfer finish interrupt flag of channel
DMA_INT_FLAG_HTF	half transfer finish interrupt flag of channel
DMA_INT_FLAG_ERR	error interrupt flag of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* get DMA interrupt_flag */

if(dma_interrupt_flag_get(DMA_CH3, DMA_INT_FLAG_FTF)){
      dma_interrupt_flag_clear(DMA_CH3, DMA_INT_FLAG_G);
      dma_interrupt_flag_clear(DMA_CH3, DMA_INT_FLAG_FTF);
}
```

3.9. EXTI

EXTI is the interrupt/event controller in the MCU. It contains up to 23 independent edge detectors and generates interrupt requests or events to the processer. The EXTI registers are listed in chapter ***3.9.1***, the EXTI firmware functions are introduced in chapter ***3.9.2***.

3.9.1. Descriptions of Peripheral registers

EXTI registers are listed in the table shown as below:

Table 3-149. 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

3.9.2. Descriptions of Peripheral functions

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

Table 3-150. EXTI firmware function

Function name	Function description
exti_deinit	deinitialize EXTI
exti_init	initialize the EXTI
exti_interrupt_enable	enable the interrupts from EXTI line x
exti_interrupt_disable	disable the interrupts from EXTI line x
exti_event_enable	enable the events from EXTI line x
exti_event_disable	disable the events from EXTI line x
exti_software_interrupt_enable	enable the software interrupt event from EXTI line x
exti_software_interrupt_disable	disable the software interrupt event from EXTI line x
exti_flag_get	get EXTI line x interrupt pending flag
exti_flag_clear	clear EXTI line x interrupt pending flag

Function name	Function description
exti_interrupt_flag_get	get EXTI line x interrupt pending flag
exti_interrupt_flag_clear	clear EXTI line x interrupt pending flag

exti_line_enum

Table 3-151. Enum exti_line_enum

Member name	Function description
EXTI_0	EXTI line 0
EXTI_1	EXTI line 1
EXTI_2	EXTI line 2
EXTI_3	EXTI line 3
EXTI_4	EXTI line 4
EXTI_5	EXTI line 5
EXTI_6	EXTI line 6
EXTI_7	EXTI line 7
EXTI_8	EXTI line 8
EXTI_9	EXTI line 9
EXTI_10	EXTI line 10
EXTI_11	EXTI line 11
EXTI_12	EXTI line 12
EXTI_13	EXTI line 13
EXTI_14	EXTI line 14
EXTI_15	EXTI line 15
EXTI_16	EXTI line 16
EXTI_17	EXTI line 17
EXTI_18	EXTI line 18
EXTI_19	EXTI line 19
EXTI_20	EXTI line 20
EXTI_21	EXTI line 21
EXTI_22	EXTI line 22
EXTI_23	EXTI line 23
EXTI_24	EXTI line 24
EXTI_25	EXTI line 25
EXTI_26	EXTI line 26
EXTI_27	EXTI line 27

exti_mode_enum

Table 3-152. Enum exti_mode_enum

Member name	Function description
EXTI_INTERRUPT	EXTI interrupt mode
EXTI_EVENT	EXTI event mode

exti_trig_type_enum

Table 3-153. Enum exti_trig_type_enum

Member name	Function description
EXTI_TRIG_RISING	EXTI rising edge trigger
EXTI_TRIG_FALLING	EXTI falling edge trigger
EXTI_TRIG_BOTH	EXTI rising and falling edge trigger
EXTI_TRIG_NONE	EXTI without rising or falling edge trigger

exti_deinit

The description of exti_deinit is shown as below:

Table 3-154. Function exti_deinit

Function name	exti_deinit
Function prototype	void exti_deinit(void);
Function descriptions	deinitialize the EXTI
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the EXTI */

exti_deinit();
```

exti_init

The description of exti_init is shown as below:

Table 3-155. Function exti_init

Function name	exti_init
Function prototype	void exti_init(exti_line_enum linex, exti_mode_enum mode, exti_trig_type_enum trig_type);
Function descriptions	initialize the EXTI
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Input parameter{in}	

mode	EXTI mode, refer to ***Table 3-152. Enum exti_mode_enum***
Input parameter{in}	
trig_type	trigger type, refer to ***Table 3-153. Enum exti_trig_type_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure EXTI_0 */

exti_init(EXTI_0, EXTI_INTERRUPT, EXTI_TRIG_BOTH);

exti_interrupt_enable

The description of exti_interrupt_enable is shown as below:

Table 3-156. Function exti_interrupt_enable

Function name	exti_interrupt_enable
Function prototype	void exti_interrupt_enable(exti_line_enum linex);
Function descriptions	enable the interrupts from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the interrupts from EXTI line 0 */

exti_interrupt_enable(EXTI_0);

exti_interrupt_disable

The description of exti_interrupt_disable is shown as below:

Table 3-157. Function exti_interrupt_disable

Function name	exti_interrupt_disable
Function prototype	void exti_interrupt_disable(exti_line_enum linex);
Function descriptions	disable the interrupt from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	

linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the interrupts from EXTI line 0 */

exti_interrupt_disable(EXTI_0);

exti_event_enable

The description of exti_event_enable is shown as below:

Table 3-158. Function exti_event_enable

Function name	exti_event_enable
Function prototype	void exti_event_enable(exti_line_enum linex);
Function descriptions	enable the events from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the events from EXTI line 0 */

exti_event_enable(EXTI_0);

exti_event_disable

The description of exti_event_disable is shown as below:

Table 3-159. Function exti_event_disable

Function name	exti_event_disable
Function prototype	void exti_event_disable(exti_line_enum linex);
Function descriptions	disable the events from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* disable the events from EXTI line 0 */

exti_event_disable(EXTI_0);

exti_software_interrupt_enable

The description of exti_software_interrupt_enable is shown as below:

Table 3-160. Function exti_software_interrupt_enable

Function name	exti_software_interrupt_enable
Function prototype	void exti_software_interrupt_enable(exti_line_enum linex);
Function descriptions	enable the software interrupt event from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable EXTI line 0 software interrupt */

exti_software_interrupt_enable(EXTI_0);

exti_software_interrupt_disable

The description of exti_software_interrupt_disable is shown as below:

Table 3-161. Function exti_software_interrupt_disable

Function name	exti_software_interrupt_disable
Function prototype	void exti_software_interrupt_disable(exti_line_enum linex);
Function descriptions	disable the software interrupt event from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* disable EXTI line 0 software interrupt */

exti_software_interrupt_disable(EXTI_0);

exti_flag_get

The description of exti_flag_get is shown as below:

Table 3-162. Function exti_flag_get

Function name	exti_flag_get
Function prototype	FlagStatus exti_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get EXTI line 0 flag status */

FlagStatus state = exti_flag_get(EXTI_0);

exti_flag_clear

The description of exti_flag_clear is shown as below:

Table 3-163. Function exti_flag_clear

Function name	exti_flag_clear
Function prototype	void exti_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear EXTI line 0 flag status */

exti_flag_clear(EXTI_0);

exti_interrupt_flag_get

The description of exti_interrupt_flag_get is shown as below:

Table 3-164. Function exti_interrupt_flag_get

Function name	exti_interrupt_flag_get
Function prototype	FlagStatus exti_interrupt_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get EXTI line 0 interrupt flag status */

FlagStatus state = exti_interrupt_flag_get(EXTI_0);

exti_interrupt_flag_clear

The description of exti_interrupt_flag_clear is shown as below:

Table 3-165. Function exti_interrupt_flag_clear

Function name	exti_interrupt_flag_clear
Function prototype	void exti_interrupt_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to ***Table 3-151. Enum exti_line_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear EXTI line 0 interrupt flag status */

exti_interrupt_flag_clear(EXTI_0);

3.10. FMC

There is flash controller and option byte for GD32F1x0 series. The FMC registers are listed in chapter 3.10.1 the FMC firmware functions are introduced in chapter 0.

3.10.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

Table 3-166. 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

3.10.2. Descriptions of Peripheral functions

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

Table 3-167. FMC firmware function

Function name	Function description
fmc_unlock	unlock the main FMC operation
fmc_lock	lock the main FMC operation
fmc_wscnt_set	set the wait state counter value
fmc_wait_state_enable	fmc wait state enable
fmc_wait_state_disable	fmc wait state disable
fmc_page_erase	FMC erase page
fmc_mass_erase	FMC erase whole chip
fmc_word_program	FMC program a word at the corresponding address
fmc_halfword_program	FMC program a half word at the corresponding address
ob_unlock	unlock the option byte operation
ob_lock	lock the option byte operation
ob_reset	reload the option byte and generate a system reset

Function name	Function description
ob_erase	erase the option byte
ob_write_protection_enable	enable option byte write protection (OB_WP)
ob_security_protection_config	configure read out protect
ob_user_write	write the FMC option byte user
ob_data_program	write the FMC option byte data
ob_user_get	get the FMC option byte OB_USER
ob_data_get	get the FMC option byte OB_DATA
ob_write_protection_get	get the FMC option byte write protection (OB_WP) in register FMC_WP
ob_obstat_plevel_get	get the state of FMC option byte security protection level
fmc_flag_get	get flag set or reset
fmc_flag_clear	clear the FMC pending flag
fmc_interrupt_enable	enable FMC interrupt
fmc_interrupt_disable	disable FMC interrupt
fmc_interrupt_flag_get	get intrrupt flag set or reset
fmc_interrupt_flag_clear	clear FMC interrupt flag state

Enum fmc_state_enum

Table 3-168. Enum fmc_state_enum

Member name	Function description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_PGERR	program error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error
FMC_OB_HSPC	option byte security protection code high

fmc_unlock

The description of fmc_unlock is shown as below:

Table 3-169. Function fmc_unlock

Function name	fmc_unlock
Function prototype	void fmc_unlock (void);
Function descriptions	unlock the main FMC operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* unlock the main FMC operation */

fmc_unlock( );

fmc_lock

The description of fmc_lock is shown as below:

Table 3-170. Function fmc_lock

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

Example:

/* lock the main FMC operation */

fmc_lock( );

fmc_wscnt_set

The description of fmc_wscnt_set is shown as below:

Table 3-171. Function fmc_wscnt_set

Function name	fmc_wscnt_set
Function prototype	void fmc_wscnt_set(uint32_t wscnt);
Function descriptions	set the wait state counter value
Precondition	-
The called functions	-
Input parameter{in}	
wscnt	wait state counter value
WS_WSCNT_0	FMC 0 wait
WS_WSCNT_1	FMC 1 wait
WS_WSCNT_2	FMC 2 wait
Output parameter{out}	
-	-

Return value	
-	-

Example:

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

fmc_wscnt_set(WS_WSCNT_1);
```

fmc_wait_state_enable

The description of fmc_wait_state_enable is shown as below:

Table 3-172. Function fmc_wscnt_set

Function name	fmc_wait_state_enable
Function prototype	void fmc_wait_state_enable(void);
Function descriptions	fmc wait state enable
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* fmc wait state enable */

fmc_wait_state_enable( );
```

fmc_wait_state_disable

The description of fmc_wait_state_disable is shown as below:

Table 3-173. Function fmc_wscnt_set

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

Example:

/* fmc wait state disable */

fmc_wait_state_disable( );

fmc_page_erase

The description of fmc_page_erase is shown as below:

Table 3-174. Function fmc_page_erase

Function name	fmc_page_erase
Function prototype	fmc_state_enum fmc_page_erase(uint32_t page_address);
Function descriptions	erase page
Precondition	fmc_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
page_address	the page address to be erased
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***
FMC_READY	the operation has been completed
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error

Example:

/* erase page */

fmc_unlock( );

fmc_state_enum state = fmc_page_erase(0x08004000);

fmc_lock( );

fmc_mass_erase

The description of fmc_mass_erase is shown as below:

Table 3-175. Function fmc_mass_erase

Function name	fmc_mass_erase
Function prototype	fmc_state_enum fmc_mass_erase(void);
Function descriptions	erase whole chip
Precondition	fmc_unlock
The called functions	fmc_ready_wait
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***
FMC_READY	the operation has been completed
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error

Example:

/* erase whole chip */

fmc_unlock( );

fmc_state_enum state = fmc_mass_erase( );

fmc_lock( );

fmc_word_program

The description of fmc_word_program is shown as below:

Table 3-176. Function fmc_word_program

Function name	fmc_word_program
Function prototype	fmc_state_enum fmc_word_program(uint32_t address, uint32_t data);
Function descriptions	program a word at the corresponding address
Precondition	fmc_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
address	the address to program
data	the data to program
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***
FMC_READY	the operation has been completed
FMC_PGERR	program error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error

Example:

/* program a word at the corresponding address */

fmc_unlock( );

fmc_state_enum fmc_state = fmc_word_program(0x08004000,0xaabbccdd);

fmc_lock( );

fmc_halfword_program

The description of fmc_halfword_program is shown as below:

Table 3-177. Function fmc_halfword_program

Function name	fmc_halfword_program
Function prototype	fmc_state_enum fmc_halfword_program(uint32_t address, uint16_t data);
Function descriptions	program a half word at the corresponding address
Precondition	fmc_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
address	the address to program
data	the data to program
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***
FMC_READY	the operation has been completed
FMC_PGERR	program error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error

Example:

/* program half word at the corresponding address */

fmc_unlock( );

fmc_state_enum fmc_state = fmc_halfword_program(0x08004000,0xaadd);

fmc_lock( );

fmc_word_reprogram

The description of fmc_word_reprogram is shown as below:

Table 3-178. Function fmc_word_reprogram

Function name	fmc_word_reprogram
Function prototype	fmc_state_enum fmc_word_reprogram(uint32_t address, uint32_t data);
Function descriptions	program a word at the corresponding address without erasing
Precondition	fmc_unlock
The called functions	fmc_ready_wait
Input parameter{in}	

address	the address to program
data	the data to program
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***
FMC_READY	the operation has been completed
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error

Example:

```
/* program a word at the corresponding address */

fmc_state_enum fmc_state;

fmc_unlock( );

fmc_state = fmc_word_program(0x08004000, 0x01234567);

ffmc_state = fmc_word_reprogram(0x08004000, 0xd583179b);

fmc_lock( );
```

ob_unlock

The description of ob_unlock is shown as below:

Table 3-179. Function ob_unlock

Function name	ob_unlock
Function prototype	void ob_unlock(void);
Function descriptions	unlock the option byte operation
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock the option byte operation */

ob_unlock( );
```

ob_lock

The description of ob_lock is shown as below:

Table 3-180. Function ob_lock

Function name	ob_lock
Function prototype	void ob_lock(void);
Function descriptions	lock the option byte operation
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the option byte operation */

ob_lock( );
```

ob_reset

The description of ob_reset is shown as below:

Table 3-181. Function ob_reset

Function name	ob_reset
Function prototype	void ob_reset(void);
Function descriptions	reload the option byte and generate a system reset
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reload the option byte and generate a system reset */

ob_reset( );
```

ob_erase

The description of ob_erase is shown as below:

Table 3-182. Function ob_erase

Function name	ob_erase
Function prototype	void ob_erase(void);
Function descriptions	erase the option byte
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***
FMC_READY	the operation has been completed
FMC_PGERR	program error
FMC_TOERR	timeout error
FMC_OB_HSPC	option byte security protection code high

Example:

```
/* erase the option byte */

fmc_unlock( );

ob_unlock( );

fmc_state_enum fmc_state = ob_erase( );

ob_lock( );

fmc_lock( );
```

ob_write_protection_enable

The description of ob_write_protection_enable is shown as below:

Table 3-183. Function ob_write_protection_enable

Function name	ob_write_protection_enable
Function prototype	fmc_state_enum ob_write_protection_enable(uint16_t ob_wp);
Function descriptions	enable option byte write protection (OB_WP)
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
ob_wp	write protection configuration data

OB_WP_NONE	disable all write protection
OB_WP_x	write protect specify sector (x=0..15)
OB_WP_ALL	write protect all sector
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***
FMC_READY	the operation has been completed
FMC_PGERR	program error
FMC_TOERR	timeout error
FMC_OB_HSPC	option byte security protection code high

Example:

```
/* enable write protection */

fmc_unlock( );

ob_unlock( );

fmc_state_enum state = ob_write_protection_enable(OB_WP_6 | OB_WP_7);

ob_lock( );

fmc_lock( );
```

ob_security_protection_config

The description of ob_security_protection_config is shown as below:

Table 3-184. Function ob_security_protection_config

Function name	ob_security_protection_config
Function prototype	fmc_state_enum ob_security_protection_config(uint8_t ob_spc);
Function descriptions	configure security protection
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
ob_spc	specify security protection
FMC_NSPC	no security protection
FMC_LSPC	low security protection
FMC_HSPC	high security protection
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***

FMC_READY	the operation has been completed
FMC_PGERR	program error
FMC_TOERR	timeout error
FMC_OB_HSPC	option byte security protection code high

Example:

```
/* enable security protection */
fmc_state_enum fmc_state;
fmc_unlock( );
ob_unlock( );
fmc_state = ob_security_protection_config (FMC_USPC);
ob_lock( );
fmc_lock( );
```

ob_user_write

The description of ob_user_write is shown as below:

Table 3-185. Function ob_user_write

Function name	ob_user_write
Function prototype	fmc_state_enum ob_user_write(uint8_t ob_user);
Function descriptions	program the FMC user option byte
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
ob_user	user option byte
OB_FWDGT_HW	hardware free watchdog timer
OB_DEEPSLEEP_RST	generate a reset when entering deepsleep mode
OB_STDBY_RST	generate a reset when entering standby mode
OB_BOOT1_SET_1	BOOT1 bit is 1
OB_VDDA_DISABLE	disable VDDA monitor
OB_SRAM_PARITY_ENABLE	enable SRAM parity check
OB_USER_RSET	reset option byte user byte
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***
FMC_READY	the operation has been completed
FMC_PGERR	program error

FMC_TOERR	timeout error
FMC_OB_HSPC	option byte security protection code high

Example:

/* program the FMC user option byte */

fmc_unlock( );

ob_unlock( );

fmc_state_enum fmc_state = ob_user_write(OB_DEEPSLEEP_RST & OB_STDBY_RST);

ob_lock( );

fmc_lock( );

ob_data_program

The description of ob_data_program is shown as below:

Table 3-186. Function ob_data_program

Function name	ob_data_program
Function prototype	fmc_state_enum ob_data_program(uint16_t ob_data);
Function descriptions	program the FMC data option byte
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
data	the byte to be programmed
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC,the enum members can refer to members of the enum ***Table 3-168. Enum fmc_state_enum***
FMC_READY	the operation has been completed
FMC_PGERR	program error
FMC_TOERR	timeout error
FMC_OB_HSPC	option byte security protection code high

Example:

/* program option bytes data */

fmc_unlock( );

ob_unlock( );

fmc_state_enum fmc_state = ob_data_program(0x5678);

ob_lock( );

fmc_lock( );

ob_user_get

The description of ob_user_get is shown as below:

Table 3-187. Function ob_user_get

Function name	ob_user_get
Function prototype	uint8_t ob_user_get(void);
Function descriptions	get OB_USER in register FMC_OBSTAT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	the FMC user option byte values(0x00 – 0xFF)

Example:

/* get the FMC user option byte */

uint8_t user = ob_user_get( );

ob_data_get

The description of ob_data_get is shown as below:

Table 3-188. Function ob_data_get

Function name	ob_data_get
Function prototype	uint16_t ob_data_get(void);
Function descriptions	get OB_DATA in register FMC_OBSTAT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	the FMC data option byte values(0x0000 – 0xFFFF)

Example:

/* get the FMC data option byte */

uint16_t data = ob_data_get( );

ob_write_protection_get

The description of ob_ write_protection_get is shown as below:

Table 3-189. Function ob_write_protection_get

Function name	ob_write_protection_get
Function prototype	uint16_t ob_write_protection_get(void);
Function descriptions	get the FMC option byte write protection (OB_WP) in register FMC_WP
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	the FMC write protection option byte value(0x0000 – 0XFFFF)

Example:

/* get the FMC option byte write protection */

uint16_t wp = ob_write_protection_get( );

ob_obstat_plevel_get

The description of ob_obstat_plevel_get is shown as below:

Table 3-190. Function ob_obstat_plevel_get

Function name	ob_obstat_plevel_get
Function prototype	uint32_t ob_obstat_plevel_get(void);
Function descriptions	get the state of FMC option byte security protection level
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	the value of PLEVEL

Example:

/* get the FMC option byte security protection level */

uint32_t obstat_plevel = ob_obstat_plevel_get( );

fmc_flag_get

The description of fmc_flag_get is shown as below:

Table 3-191. Function fmc_flag_get

Function name	fmc_flag_get
Function prototype	FlagStatus fmc_flag_get(uint32_t flag);
Function descriptions	get FMC flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	check FMC flag
FMC_FLAG_BUSY	FMC busy flag bit
FMC_FLAG_PGERR	FMC programming error flag
FMC_FLAG_WPERR	FMC write protection error flag
FMC_FLAG_END	FMC end of programming flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get FMC flag */

FlagStatus flag = fmc_flag_get(FMC_FLAG_END);

fmc_flag_clear

The description of fmc_flag_clear is shown as below:

Table 3-192. Function fmc_flag_clear

Function name	fmc_flag_clear
Function prototype	void fmc_flag_clear(uint32_t flag);
Function descriptions	clear the FMC pending flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	clear FMC flag
FMC_FLAG_PGERR	FMC operation error flag
FMC_FLAG_WPERR	FMC erase/program protection error flag
FMC_FLAG_END	FMC end of operation flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear FMC flag */

fmc_flag_clear(FMC_FLAG_END);

fmc_interrupt_enable

The description of fmc_interrupt_enable is shown as below:

Table 3-193. Function fmc_interrupt_enable

Function name	fmc_interrupt_enable
Function prototype	void fmc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable FMC interrupt
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
interrupt	the FMC interrupt source
FMC_INT_END	FMC end of program interrupt
FMC_INT_ERR	FMC error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable FMC interrupt */

fmc_unlock( );

fmc_interrupt_enable(FMC_INT_END);

fmc_lock( );

fmc_interrupt_disable

The description of fmc_interrupt_disable is shown as below:

Table 3-194. Function fmc_interrupt_disable

Function name	fmc_interrupt_disable
Function prototype	void fmc_interrupt_disable(uint32_t interrupt);
Function descriptions	disable FMC interrupt
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
interrupt	the FMC interrupt source
FMC_INT_END	FMC end of program interrupt

FMC_INT_ERR	FMC error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable FMC interrupt */

fmc_unlock( );

fmc_interrupt_disable(FMC_INT_END);

fmc_lock( );

fmc_interrupt_flag_get

The description of fmc_interrupt_flag_get is shown as below:

Table 3-195. Function fmc_interrupt_flag_get

Function name	fmc_interrupt_flag_get
Function prototype	FlagStatus fmc_interrupt_flag_get(uint32_t int_flag);
Function descriptions	get FMC interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	FMC flag
FMC_INT_FLAG_PGERR	FMC operation error flag
FMC_INT_FLAG_WPERR	FMC erase/program protection error flag
FMC_INT_FLAG_END	FMC end of operation flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get FMC interrupt flag */

FlagStatus flag = fmc_interrupt_flag_get (FMC_INT_FLAG_PGERR);

fmc_interrupt_flag_clear

The description of fmc_interrupt_flag_clear is shown as below:

Table 3-196. Function fmc_interrupt_flag_clear

Function name	fmc_interrupt_flag_ clear
Function prototype	void fmc_interrupt_flag_clear(uint32_t int_flag);
Function descriptions	clear the FMC pending interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	clear FMC flag
FMC_INT_FLAG_PGERR	FMC operation error flag
FMC_INT_FLAG_WPERR	FMC erase/program protection error flag
FMC_INT_FLAG_END	FMC end of operation flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear FMC interrupt flag */

fmc_interrupt_flag_clear(FMC_FLAG_PGERR);

3.11. 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 ***3.11.1*** the FWDGT firmware functions are introduced in chapter ***3.11.2***.

3.11.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

Table 3-197. FWDGT Registers

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

3.11.2. Descriptions of Peripheral functions

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

Table 3-198. FWDGT firmware function

Function name	Function description
fwdgt_write_enable	enable write access to FWDGT_PSC and FWDGT_RLD
fwdgt_write_disable	disable write access to FWDGT_PSC and FWDGT_RLD
fwdgt_enable	start the FWDGT counter
fwdgt_prescaler_value_config	configure the free watchdog timer counter prescaler value
fwdgt_reload_value_config	configure the free watchdog timer counter reload value
fwdgt_window_value_config	configure the FWDGT counter window value
fwdgt_counter_reload	reload the counter of FWDGT
fwdgt_config	configure counter reload value, and prescaler divider value
fwdgt_flag_get	get flag state of FWDGT

fwdgt_write_enable

The description of fwdgt_write_enable is shown as below:

Table 3-199. Function fwdgt_write_ensable

Function name	fwdgt_write_enable
Function prototype	void fwdgt_write_enable(void);
Function descriptions	enable write access to FWDGT_PSC and FWDGT_RLD
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable write access to FWDGT_PSC and FWDGT_RLD */

fwdgt_write_enable ();
```

fwdgt_write_disable

The description of fwdgt_write_disable is shown as below:

Table 3-200. Function fwdgt_write_disable

Function name	fwdgt_write_disable
Function prototype	void fwdgt_write_disable(void);
Function descriptions	disable write access to FWDGT_PSC and FWDGT_RLD

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

Example:

/* disable write access to FWDGT_PSC and FWDGT_RLD */

fwdgt_write_disable ();

fwdgt_enable

The description of fwdgt_enable is shown as below:

Table 3-201. Function fwdgt_enable

Function name	fwdgt_enable
Function prototype	void fwdgt_enable(void);
Function descriptions	start the FWDGT counter
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* start the free watchdog timer counter */

fwdgt_enable ();

fwdgt_prescaler_value_config

The description of fwdgt_prescaler_value_config is shown as below:

Table 3-202. Function fwdgt_prescaler_value_config

Function name	fwdgt_prescaler_value_config
Function prototype	ErrStatus fwdgt_prescaler_value_config(uint16_t prescaler_value);
Function descriptions	configure the free watchdog timer counter prescaler value
Precondition	-
Input parameter{in}	
prescaler_value	specify prescaler value
FWDGT_PSC_DIV4	FWDGT prescaler set to 4

FWDGT_PSC_DIV8	FWDGT prescaler set to 8
FWDGT_PSC_DIV16	FWDGT prescaler set to 16
FWDGT_PSC_DIV32	FWDGT prescaler set to 32
FWDGT_PSC_DIV64	FWDGT prescaler set to 64
FWDGT_PSC_DIV128	FWDGT prescaler set to 128
FWDGT_PSC_DIV256	FWDGT prescaler set to 256
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

/* set FWDGT prescaler to 256 */

ErrStatus flag;

flag = fwdgt_prescaler_value_config (FWDGT_PSC_DIV256);

fwdgt_reload_value_config

The description of fwdgt_reload_value_config is shown as below:

Table 3-203. Function fwdgt_reload_value_config

Function name	fwdgt_reload_value_config
Function prototype	ErrStatus fwdgt_reload_value_config(uint16_t reload_value);
Function descriptions	configure the free watchdog timer counter reload value
Precondition	-
Input parameter{in}	
reload_value	reload_value: specify window value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

/* set FWDGT reload value to 0x0FFF */

ErrStatus flag;

flag = fwdgt_reload_value_config (0x0FFF);

fwdgt_window_value_config

The description of fwdgt_window_value_config is shown as below:

Table 3-204. Function fwdgt_window_value_config

Function name	fwdgt_window_value_config

Function prototype	ErrStatus fwdgt_window_value_config(uint16_t window_value);
Function descriptions	configure the FWDGT counter window value
Precondition	-
Input parameter{in}	
window_value	window_value: specify window value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

```
/* set FWDGT window value to 0x0FFF */

ErrStatus flag;

flag = fwdgt_window_value_config (0x0FFF);
```

fwdgt_counter_reload

The description of fwdgt_counter_reload is shown as below:

Table 3-205. Function fwdgt_counter_reload

Function name	fwdgt_counter_reload
Function prototype	void fwdgt_counter_reload(void);
Function descriptions	reload the counter of FWDGT
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reload FWDGT counter */

fwdgt_counter_reload ();
```

fwdgt_config

The description of fwdgt_config is shown as below:

Table 3-206. Function fwdgt_config

Function name	fwdgt_config
Function prototype	ErrStatus fwdgt_config(uint16_t reload_value, uint8_t prescaler_div);
Function descriptions	configure counter reload value, and prescaler divider value

Precondition	-
Input parameter{in}	
reload_value	specify reload value(0x0000 - 0x0FFF)-
Input parameter{in}	
prescaler_div	FWDGT prescaler value-
FWDGT_PSC_DIV4	FWDGT prescaler set to 4
FWDGT_PSC_DIV8	FWDGT prescaler set to 8
FWDGT_PSC_DIV16	FWDGT prescaler set to 16
FWDGT_PSC_DIV32	FWDGT prescaler set to 32
FWDGT_PSC_DIV64	FWDGT prescaler set to 64
FWDGT_PSC_DIV128	FWDGT prescaler set to 128
FWDGT_PSC_DIV256	FWDGT prescaler set to 256
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

/* confiure FWDGT counter clock: 40KHz(IRC40K) / 64 = 0.625 KHz */

fwdgt_config(2*500, FWDGT_PSC_DIV64);

fwdgt_flag_get

The description of fwdgt_flag_get is shown as below:

Table 3-207. Function fwdgt_flag_get

Function name	fwdgt_flag_get
Function prototype	FlagStatus fwdgt_flag_get(uint16_t flag);
Function descriptions	get flag state of FWDGT
Precondition	-
Input parameter{in}	
flag	flag to get
FWDGT_FLAG_PUD	a write operation to FWDGT_PSC register is on going
FWDGT_FLAG_RUD	a write operation to FWDGT_RLD register is on going
FWDGT_FLAG_WUD	a write operation to FWDGT_WND register is on going
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* test if a prescaler value update is on going */

FlagStatus status;

status = fwdgt_flag_get (FWDGT_FLAG_PUD);

3.12. GPIO

GPIO is used to implement logic input/output functions for the devices. The GPIO registers are listed in chapter ***3.12.1***, the GPIO firmware functions are introduced in chapter ***3.12.2***.

3.12.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

Table 3-208. GPIO Registers

Registers	Descriptions
GPIOx_CTL	GPIO port control register
GPIOx_OMODE	GPIO port output mode register
GPIOx_OSPD	GPIO port output speed register
GPIOx_PUD	GPIO port pull-up/pull-down register
GPIOx_ISTAT	GPIO port input status register
GPIOx_OCTL	GPIO port output control register
GPIOx_BOP	GPIO port bit operation register
GPIOx_LOCK	GPIO port configuration lock register
GPIO_AFSEL0	GPIO alternate function selected register 0
GPIO_AFSEL1	GPIO alternate function selected register 1
GPIO_BC	GPIO bit clear register

3.12.2. Descriptions of Peripheral functions

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

Table 3-209. GPIO firmware function

Function name	Function description
gpio_deinit	reset GPIO port
gpio_mode_set	set GPIO mode
gpio_output_options_set	set GPIO output type and speed
gpio_bit_set	set GPIO pin bit
gpio_bit_reset	reset GPIO pin bit
gpio_bit_write	write data to the specified GPIO pin
gpio_port_write	write data to the specified GPIO port
gpio_input_bit_get	get GPIO pin input status
gpio_input_port_get	get GPIO port input status
gpio_output_bit_get	get GPIO pin output status
gpio_output_port_get	get GPIO port output status
gpio_af_set	set GPIO alternate function

Function name	Function description
gpio_pin_lock	lock GPIO pin

gpio_deinit

The description of gpio_deinit is shown as below:

Table 3-210. Function gpio_deinit

Function name	gpio_deinit
Function prototype	void gpio_deinit(uint32_t gpio_periph);
Function descriptions	reset GPIO port
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* reset GPIOA */

gpio_deinit (GPIOA);

gpio_mode_set

The description of gpio_mode_set is shown as below:

Table 3-211. Function gpio_mode_set

Function name	gpio_mode_set
Function prototype	void gpio_mode_set(uint32_t gpio_periph, uint32_t mode, uint32_t pull_up_down, uint32_t pin);
Function descriptions	set GPIO mode
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
mode	gpio pin mode
GPIO_MODE_INPUT	input mode
GPIO_MODE_OUTPUT	output mode
GPIO_MODE_AF	alternate function mode

GPIO_MODE_ANALOG	analog mode
Input parameter{in}	
pull_up_down	gpio pin with pull-up or pull-down resistor
GPIO_PUPD_NONE	floating mode, no pull-up and pull-down resistors
GPIO_PUPD_PULLUP	with pull-up resistor
GPIO_PUPD_PULLDOWN	with pull-down resistor
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* config PA0 as input mode with pullup*/

gpio_mode_set (GPIOA, GPIO_MODE_INPUT, GPIO_PUPD_PULLUP, GPIO_PIN_0);

gpio_output_options_set

The description of gpio_output_options_set is shown as below:

Table 3-212. Function gpio_output_options_set

Function name	gpio_output_options_set
Function prototype	void gpio_output_options_set(uint32_t gpio_periph, uint8_t otype, uint32_t speed, uint32_t pin);
Function descriptions	set GPIO output type and speed
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
otype	gpio pin output mode
GPIO_OTYPE_PP	push pull mode
GPIO_OTYPE_OD	open drain mode
Input parameter{in}	
speed	gpio pin output max speed
GPIO_OSPEED_2MHZ	output max speed 2MHz
GPIO_OSPEED_10MH	output max speed 10MHz

Z	
GPIO_OSPEED_50MHZ	output max speed 50MHz
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* config PA0 as push pull mode */

gpio_output_options_set (GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_2MHZ, GPIO_PIN_0);

gpio_bit_set

The description of gpio_bit_set is shown as below:

Table 3-213. Function gpio_bit_set

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

Example:

/* set PA0*/

gpio_bit_set (GPIOA, GPIO_PIN_0);

gpio_bit_reset

The description of gpio_bit_reset is shown as below:

Table 3-214. Function gpio_bit_reset

Function name	gpio_bit_reset
Function prototype	void gpio_bit_reset(uint32_t gpio_periph,uint32_t pin);
Function descriptions	reset GPIO pin
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* reset PA0*/

gpio_bit_set (GPIOA, GPIO_PIN_0);

gpio_bit_write

The description of gpio_bit_write is shown as below:

Table 3-215. Function gpio_bit_write

Function name	gpio_bit_write
Function prototype	void gpio_bit_write(uint32_t gpio_periph,uint32_t pin,bit_status bit_value);
Function descriptions	write data to the specified GPIO pin
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
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}	

bit_value	SET or RESET
RESET	clear the port pin
SET	set the port pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* write 1 to PA0 */

gpio_bit_write (GPIOA, GPIO_PIN_0, SET);

gpio_port_write

The description of gpio_port_write is shown as below:

Table 3-216. Function gpio_port_write

Function name	gpio_port_write
Function prototype	void gpio_port_write(uint32_t gpio_periph,uint16_t data);
Function descriptions	write data to the specified GPIO port
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
data	specify the value to be written to the port output data register
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*write 1010 0101 1010 0101 to Port A */

gpio_port_write (GPIOA, 0xA5A5);

gpio_input_bit_get

The description of gpio_input_bit_get is shown as below:

Table 3-217. Function gpio_input_bit_get

Function name	gpio_input_bit_get
Function prototype	FlagStatus gpio_input_bit_get(uint32_t gpio_periph,uint32_t pin);
Function descriptions	get GPIO pin input status

Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

/* get status of PA0 */

FlagStatus bit_state = gpio_input_bit_get (GPIOA, GPIO_PIN_0);

gpio_input_port_get

The description of gpio_input_port_get is shown as below:

Table 3-218. Function gpio_input_port_get

Function name	gpio_input_port_get
Function prototype	uint16_t gpio_input_port_get(uint32_t gpio_periph);
Function descriptions	get GPIO all pins input status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Output parameter{out}	
-	-
Return value	
uint16_t	0x0000-0xFFFF

Example:

/* get input value of Port A */

uint16_t port_state;

port_state = gpio_input_bit_get (GPIOA);

gpio_output_bit_get

The description of gpio_output_bit_get is shown as below:

Table 3-219. Function gpio_output_bit_get

Function name	gpio_output_bit_get
Function prototype	FlagStatus gpio_output_bit_get(uint32_t gpio_periph,uint32_t pin);
Function descriptions	get GPIO pin output status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get output status of PA0 */
FlagStatus bit_state;
bit_state = gpio_output_bit_get (GPIOA, GPIO_PIN_0);
```

gpio_output_port_get

The description of gpio_output_port_get is shown as below:

Table 3-220. Function gpio_output_port_get

Function name	gpio_output_port_get
Function prototype	uint16_t gpio_output_port_get(uint32_t gpio_periph);
Function descriptions	get GPIO all pins output status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Output parameter{out}	
-	-
Return value	

Uint16_t	0x0000-0xFFFF

Example:

/* get output value of Port A */

uint16_t port_state;

port_state = gpio_output_port_get (GPIOA);

gpio_af_set

The description of gpio_af_set is shown as below:

Table 3-221. Function gpio_af_set

Function name	gpio_af_set
Function prototype	void gpio_af_set(uint32_t gpio_periph, uint32_t alt_func_num, uint32_t pin);
Function descriptions	set GPIO alternate function
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,F)
Input parameter{in}	
alt_func_num	GPIO pin af function, please refer to specific device datasheet
GPIO_AF_0	TIMER2, TIMER13, TIMER14, TIMER16, SPI0, I2S0, SPI1, SPI2, I2S2, CK_OUT, SWDIO, SWCLK, USART0, CEC, IFRP, I2C0, I2C1, TSI, EVENTOUT
GPIO_AF_1	USART0, USART1, IFRP, CEC, TIMER2, TIMER14, I2C0, I2C1, I2C2, EVENTOUT
GPIO_AF_2	TIMER0, TIMER1, TIMER15, TIMER16, EVENTOUT
GPIO_AF_3	TSI, I2C0, TIMER14, EVENTOUT
GPIO_AF_4 (port A,B only)	TIMER13, I2C0, I2C1, I2C2, USART1
GPIO_AF_5 (port A,B only)	TIMER15, TIMER16, SPI2, I2S2, I2C0, I2C1
GPIO_AF_6 (port A,B only)	SPI1, EVENTOUT
GPIO_AF_7 (port A only)	CMP0, CMP1
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-

Return value	
-	-

Example:

/*set PA0 alternate function 0*/

gpio_af_set(GPIOA, GPIO_AF_0, GPIO_PIN_0);

gpio_pin_lock

The description of gpio_pin_lock is shown as below:

Table 3-222. Function gpio_pin_lock

Function name	gpio_pin_lock
Function prototype	void gpio_pin_lock(uint32_t gpio_periph, uint32_t pin);
Function descriptions	lock GPIO pin bit
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B)
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* lock PA0*/

gpio_pin_lock (GPIOA, GPIO_PIN_ 0);

3.13. 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 ***3.13.1***, the I2C firmware functions are introduced in chapter ***3.13.2***

3.13.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

Table 3-223. 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

3.13.2. Descriptions of Peripheral functions

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

Table 3-224. I2C firmware function

Function name	Function description
i2c_deinit	reset I2C
i2c_clock_config	configure I2C clock
i2c_mode_addr_config	configure I2C address
i2c_smbus_type_config	select SMBus type
i2c_ack_config	whether or not to send an ACK
i2c_ackpos_config	configure I2C POAP position
i2c_master_addressing	master sends slave address
i2c_dualaddr_enable	enable dual-address mode
i2c_dualaddr_disable	disable dual-address mode
i2c_enable	enable I2C
i2c_disable	disable I2C
i2c_start_on_bus	generate a START condition on I2C bus
i2c_stop_on_bus	generate a STOP condition on I2C bus
i2c_data_transmit	I2C transmit data function
i2c_data_receive	I2C receive data function
i2c_dma_config	configure I2C DMA mode
i2c_dma_last_transfer_config	configure whether next DMA EOT is DMA last transfer or not
i2c_stretch_scl_low_config	whether to stretch SCL low when data is not ready in slave mode
i2c_slave_response_to_gcall_config	whether or not to response to a general call
i2c_software_reset_config	configure software reset of I2C

Function name	Function description
i2c_pec_config	configure I2C PEC calculation
i2c_pec_transfer_config	configure whether to transfer PEC value
i2c_pec_value_get	get packet error checking value
i2c_smbus_alert_config	configure I2C alert through SMBA pin
i2c_smbus_arp_config	configure I2C ARP protocol in SMBus
i2c_flag_get	get I2C flag status
i2c_flag_clear	clear I2C flag status
i2c_interrupt_enable	enable I2C interrupt
i2c_interrupt_disable	disable I2C interrupt
i2c_interrupt_flag_get	get I2C interrupt status
i2c_interrupt_flag_clear	clear I2C interrupt status

Enum i2c_flag_enum

Table 3-225. 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 3-226. Enum i2c_interrupt_flag_enum

Member name	Function description
I2C_INT_FLAG_SBSEND	start condition sent out in master mode interrupt flag
I2C_INT_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode interrupt flag
I2C_INT_FLAG_BTC	byte transmission finishes interrupt flag
I2C_INT_FLAG_ADD10SEND	header of 10-bit address is sent in master mode interrupt flag
I2C_INT_FLAG_STPDET	stop condition detected in slave mode interrupt flag
I2C_INT_FLAG_RBNE	I2C_DATA is not empty during receiving interrupt flag
I2C_INT_FLAG_TBE	I2C_DATA is empty during transmitting interrupt flag
I2C_INT_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus interrupt flag
I2C_INT_FLAG_LOSTARB	arbitration lost in master mode interrupt flag
I2C_INT_FLAG_AERR	acknowledge error interrupt flag
I2C_INT_FLAG_OUERR	over-run or under-run situation occurs in slave mode interrupt flag
I2C_INT_FLAG_PECERR	PEC error when receiving data interrupt flag
I2C_INT_FLAG_SMBTO	timeout signal in SMBus mode interrupt flag
I2C_INT_FLAG_SMBALT	SMBus alert status interrupt flag

Enum i2c_interrupt_enum

Table 3-227. Enum i2c_interrupt_enum

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

i2c_deinit

The description of i2c_deinit is shown as below:

Table 3-228. Function i2c_deinit

Function name	i2c_deinit
Function prototype	void i2c_deinit(uint32_t i2c_periph);
Function descriptions	reset I2C
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* reset I2C0 */

i2c_deinit (I2C0);

i2c_clock_config

The description of i2c_clock_config is shown as below:

Table 3-229. Function i2c_clock_config

Function name	i2c_clock_config
Function prototype	void i2c_clock_config(uint32_t i2c_periph, uint32_t clkspeed, uint32_t dutycyc);
Function descriptions	configure I2C clock
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
clkspeed	i2c clock speed
Input parameter{in}	
dutycyc	duty cycle in fast mode
I2C_DTCY_2	T_low/T_high=2
I2C_DTCY_16_9	T_low/T_high=16/9
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure I2C0 clock speed as 100KHz */

i2c_clock_config(I2C0, 100000, I2C_DTCY_2);

i2c_mode_addr_config

The description of i2c_mode_addr_config is shown as below:

Table 3-230. Function i2c_mode_addr_config

Function name	i2c_mode_addr_config
Function prototype	void i2c_mode_addr_config(uint32_t i2c_periph, uint32_t mode, uint32_t addformat, uint32_t addr);

Function descriptions	configure I2C address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
mode	I2C mode select
I2C_I2CMODE_ENABLE	I2C mode
I2C_SMBUSMODE_ENABLE	SMBus mode
Input parameter{in}	
addformat	7bits or 10bits
I2C_ADDFORMAT_7BITS	address format is 7 bits
I2C_ADDFORMAT_10BITS	address format is 10 bits
Input parameter{in}	
addr	I2C address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure I2C0 address as 0x82, using 7 bits */

i2c_mode_addr_config(I2C0, I2C_I2CMODE_ENABLE, I2C_ADDFORMAT_7BITS, 0x82);
```

i2c_smbus_type_config

The description of i2c_smbus_type_config is shown as below:

Table 3-231. Function i2c_smbus_type_config

Function name	i2c_smbus_type_config
Function prototype	void i2c_smbus_type_config(uint32_t i2c_periph, uint32_t type);
Function descriptions	select SMBus type
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	

type	Device or host
I2C_SMBUS_DEVICE	SMBus mode device type
I2C_SMBUS_HOST	SMBus mode host type
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* config I2C0 as SMBUS host type*/

i2c_smbus_type_config (I2C0, I2C_SMBUS_HOST);

i2c_ack_config

The description of i2c_ack_config is shown as below:

Table 3-232. Function i2c_ack_config

Function name	i2c_ack_config
Function prototype	void i2c_ack_config(uint32_t i2c_periph, uint32_t ack);
Function descriptions	whether or not to send an ACK
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
ack	whether or not to send an ACK
I2C_ACK_ENABLE	ACK will be sent
I2C_ACK_DISABLE	ACK will not be sent
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 will send ACK */

i2c_ack_config (I2C0, I2C_ACK_ENABLE);

i2c_ackpos_config

The description of i2c_ackpos_config is shown as below:

Table 3-233. Function i2c_ackpos_config

Function name	i2c_ackpos_config

Function prototype	void i2c_ackpos_config(uint32_t i2c_periph, uint32_t pos);
Function descriptions	configure I2C POAP position
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
pos	ACK position
I2C_ACKPOS_CURRENT	ACKEN bit decides whether or not to send ACK or not for the current byte
I2C_ACKPOS_NEXT	ACKEN bit decides whether or not to send ACK for the next byte
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* The ACK of I2C0 is send for the current frame */

i2c_ackpos_config (I2C0, I2C_ACKPOS_CURRENT);

i2c_master_addressing

The description of i2c_master_addressing is shown as below:

Table 3-234. Function i2c_master_addressing

Function name	i2c_master_addressing
Function prototype	void i2c_master_addressing(uint32_t i2c_periph, uint32_t addr, uint32_t trandirection);
Function descriptions	master sends slave address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
addr	slave address
Input parameter{in}	
trandirection	transmitter or receiver
I2C_TRANSMITTER	transmitter
I2C_RECEIVER	receiver
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* send slave address to I2C bus and I2C0 act as receiver */

i2c_master_addressing(I2C0, 0x82, I2C_RECEIVER);

i2c_dualaddr_enable

The description of i2c_dualaddr_enable is shown as below:

Table 3-235. Function i2c_dualaddr_enable

Function name	i2c_dualaddr_enable
Function prototype	void i2c_dualaddr_enable(uint32_t i2c_periph, uint32_t addr)
Function descriptions	enable dual-address mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
addr	second address in dual-address mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable I2C0 dual-address*/

i2c_dualaddr_enable (I2C0, 0x80);

i2c_dualaddr_disable

The description of i2c_dualaddr_disable is shown as below:

Table 3-236. Function i2c_dualaddr_disable

Function name	i2c_dualaddr_disable
Function prototype	void i2c_dualaddr_disable(uint32_t i2c_periph)
Function descriptions	disable dual-address mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable I2C0 dual-address */

i2c_dualaddr_disable (I2C0);

i2c_enable

The description of i2c_enable is shown as below:

Table 3-237. Function i2c_enable

Function name	i2c_enable
Function prototype	void i2c_enable(uint32_t i2c_periph);
Function descriptions	enable I2C
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable I2C0 */

i2c_enable (I2C0);

i2c_disable

The description of i2c_disable is shown as below:

Table 3-238. Function i2c_disable

Function name	i2c_disable
Function prototype	void i2c_disable(uint32_t i2c_periph);
Function descriptions	disable I2C
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable I2C0 */

i2c_disable (I2C0);

i2c_start_on_bus

The description of i2c_start_on_bus is shown as below:

Table 3-239. Function i2c_start_on_bus

Function name	i2c_start_on_bus
Function prototype	void i2c_start_on_bus(uint32_t i2c_periph);
Function descriptions	generate a START condition on I2C bus
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 send a start condition to I2C bus */

i2c_start_on_bus (I2C0);

i2c_stop_on_bus

The description of i2c_stop_on_bus is shown as below:

Table 3-240. Function i2c_stop_on_bus

Function name	i2c_stop_on_bus
Function prototype	void i2c_stop_on_bus(uint32_t i2c_periph);
Function descriptions	generate a STOP condition on I2C bus
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 generate a STOP condition to I2C bus */

i2c_stop_on_bus (I2C0);

i2c_data_transmit

The description of i2c_data_transmit is shown as below:

Table 3-241. Function i2c_data_transmit

Function name	i2c_data_transmit
Function prototype	void i2c_data_transmit(uint32_t i2c_periph, uint8_t data);
Function descriptions	I2C transmit data function
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
data	transmit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 transmit data */

i2c_data_transmit (I2C0, 0x80);

i2c_data_receive

The description of i2c_data_receive is shown as below:

Table 3-242. Function i2c_data_receive

Function name	i2c_data_receive
Function prototype	uint8_t i2c_data_receive(uint32_t i2c_periph);
Function descriptions	I2C receive data function
Precondition	-
The called functions	-
Input parameter{in}	

i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Output parameter{out}	
-	-
Return value	
uint8_t	0x00..0xFF

Example:

/* I2C0 receive data */

uint8_t i2c_receiver;

i2c_receiver = i2c_data_receive(I2C0);

i2c_dma_config

The description of i2c_dma_config is shown as below:

Table 3-243. Function i2c_dma_config

Function name	i2c_dma_config
Function prototype	void i2c_dma_config(uint32_t i2c_periph, uint32_t dmastate);
Function descriptions	configure I2C DMA mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
dmastate	on or off
I2C_DMA_ON	enable DMA mode
I2C_DMA_OFF	disable DMA mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 DMA mode enable */

i2c_dma_config (I2C0, I2C_DMA_ON);

i2c_dma_last_transfer_config

The description of i2c_dma_last_transfer_config is shown as below:

Table 3-244. Function i2c_dma_last_transfer_config

Function name	i2c_dma_last_transfer_config
Function prototype	void i2c_dma_last_transfer_config(uint32_t i2c_periph, uint32_t dmalast);
Function descriptions	configure whether next DMA EOT is DMA last transfer or not
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
dmalast	next DMA EOT is the last transfer or not
I2C_DMALST_ON	next DMA EOT is the last transfer
I2C_DMALST_OFF	next DMA EOT is not the last transfer
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* next DMA EOT is the last transfer */

i2c_dma_last_transfer_config (I2C0, I2C_DMALST_ON);

i2c_stretch_scl_low_config

The description of i2c_stretch_scl_low_config is shown as below:

Table 3-245. Function i2c_stretch_scl_low_config

Function name	i2c_stretch_scl_low_config
Function prototype	void i2c_stretch_scl_low_config(uint32_t i2c_periph, uint32_t stretchpara);
Function descriptions	whether to stretch SCL low when data is not ready in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
stretchpara	SCL stretching enable or disable
I2C_SCLSTRETCH_ENABLE	enable SCL stretching
I2C_SCLSTRETCH_DISABLE	disable SCL stretching
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* stretch SCL low when data is not ready in slave mode */

i2c_stretch_scl_low_config (I2C0, I2C_SCLSTRETCH_ENABLE);

i2c_slave_response_to_gcall_config

The description of i2c_slave_response_to_gcall_config is shown as below:

Table 3-246. Function i2c_slave_response_to_gcall_config

Function name	i2c_slave_response_to_gcall_config
Function prototype	void i2c_slave_response_to_gcall_config(uint32_t i2c_periph, uint32_t gcallpara);
Function descriptions	whether or not to response to a general call
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
gcallpara	response to a general call or not
I2C_GCEN_ENABLE	slave will response to a general call
I2C_GCEN_DISABLE	slave will not response to a general call
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 will response to a general call */

i2c_slave_response_to_gcall_config (I2C0, I2C_GCEN_ENABLE);

i2c_software_reset_config

The description of i2c_software_reset_config is shown as below:

Table 3-247. Function i2c_software_reset_config

Function name	i2c_software_reset_config
Function prototype	void i2c_software_reset_config(uint32_t i2c_periph, uint32_t sreset);
Function descriptions	configure software reset of I2C
Precondition	-
The called functions	-

Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
sreset	reset or not
I2C_SRESET_SET	I2C is under reset
I2C_SRESET_RESET	I2C is not under reset
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* software reset I2C0 */

i2c_software_reset_config (I2C0, I2C_SRESET_SET);

i2c_pec_config

The description of i2c_pec_config is shown as below:

Table 3-248. Function i2c_pec_config

Function name	i2c_pec_config
Function prototype	void i2c_pec_config(uint32_t i2c_periph, uint32_t pecstate);
Function descriptions	configure I2C PEC calculation or not
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
pecstate	on or off
I2C_PEC_ENABLE	PEC calculation on
I2C_PEC_DISABLE	PEC calculation off
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* Enable I2C PEC calculation */

i2c_pec_config (I2C0, I2C_PEC_ENABLE);

i2c_pec_transfer_config

The description of i2c_pec_transfer_enable is shown as below:

Table 3-249. Function i2c_pec_transfer_config

Function name	i2c_pec_transfer_config
Function prototype	void i2c_pec_transfer_config(uint32_t i2c_periph, uint32_t pecpara);
Function descriptions	configure whether to transfer PEC value
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
pecpara	Transfer PEC or not
I2C_PECTRANS_ENABLE	transfer PEC
I2C_PECTRANS_DISABLE	not transfer PEC
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 transfer PEC */

i2c_pec_transfer_config (I2C0, I2C_PECTRANS_ENABLE);

i2c_pec_value_get

The description of i2c_pec_value_get is shown as below:

Table 3-250. Function i2c_pec_value_get

Function name	i2c_pec_value_get
Function prototype	uint8_t i2c_pec_value_get(uint32_t i2c_periph);
Function descriptions	get packet error checking value
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Output parameter{out}	
-	-
Return value	

uint8_t	PEC value

Example:

```
/* I2C0 get packet error checking value */

uint8_t pec_value;

pec_value = i2c_pec_value_get (I2C0);
```

i2c_smbus_alert_config

The description of i2c_smbus_alert_config is shown as below:

Table 3-251. Function i2c_smbus_alert_config

Function name	i2c_smbus_alert_config
Function prototype	void i2c_smbus_alert_config (uint32_t i2c_periph, uint32_t smbuspara);
Function descriptions	configure I2C alert through SMBA pin
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
smbuspara	issue alert through SMBA pin or not
I2C_SALTSEND_ENABLE	issue alert through SMBA pin
I2C_SALTSEND_DISABLE	not issue alert through SMBA pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 issue alert through SMBA pin enable*/

i2c_smbus_alert_config (I2C0, I2C_SALTSEND_ENABLE);
```

i2c_smbus_arp_config

The description of i2c_smbus_arp_config is shown as below:

Table 3-252. Function i2c_smbus_arp_config

Function name	i2c_smbus_arp_config
Function prototype	void i2c_smbus_arp_config(uint32_t i2c_periph, uint32_t arpstate);
Function descriptions	configure I2C ARP protocol in SMBus

Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
arpstate	ARP protocol in SMBus switch
I2C_ARP_ENABLE	enable ARP
I2C_ARP_DISABLE	disable ARP
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C0 ARP protocol in SMBus switch */

i2c_smbus_arp_config (I2C0, I2C_ARP_ENABLE);
```

i2c_flag_get

The description of i2c_flag_get is shown as below:

Table 3-253. Function i2c_flag_get

Function name	i2c_flag_get
Function prototype	FlagStatus i2c_flag_get(uint32_t i2c_periph, i2c_flag_enum flag)
Function descriptions	check I2C flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
flag	specify get which flag
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
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* check whether start condition send out */
FlagStatus flag_state = RESET;
flag_state = i2c_flag_get (I2C0, I2C_FLAG_SBSEND);
```

i2c_flag_clear

The description of i2c_flag_clear is shown as below:

Table 3-254. Function i2c_flag_clear

Function name	i2c_flag_clear
Function prototype	void i2c_flag_clear(uint32_t i2c_periph, i2c_flag_enum flag)
Function descriptions	clear I2C flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
flag	flag type
I2C_FLAG_SMBALT	SMBus Alert status
I2C_FLAG_SMBTO	timeout signal in SMBus mode
I2C_FLAG_PECERR	PEC error when receiving data
I2C_FLAG_OUERR	over-run or under-run situation occurs in slave mode

I2C_FLAG_AERR	acknowledge error
I2C_FLAG_LOSTARB	arbitration lost in master mode
I2C_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus
I2C_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear a bus error flag*/

i2c_flag_clear (I2C0, I2C_FLAG_BERR);

i2c_interrupt_enable

The description of i2c_interrupt_enable is shown as below:

Table 3-255. Function i2c_interrupt_enable

Function name	i2c_interrupt_enable
Function prototype	void i2c_interrupt_enable(uint32_t i2c_periph, i2c_interrupt_enum interrupt);
Function descriptions	enable I2C interrupt
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
inttype	interrupt type
I2C_INT_ERR	error interrupt
I2C_INT_EV	event interrupt
I2C_INT_BUF	buffer interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable I2C0 event interrupt */

i2c_interrupt_enable (I2C0, I2C_INT_EV);

i2c_interrupt_disable

The description of i2c_interrupt_disable is shown as below:

Table 3-256. Function i2c_interrupt_disable

Function name	i2c_interrupt_disable
Function prototype	void i2c_interrupt_disable(uint32_t i2c_periph, i2c_interrupt_enum interrupt);
Function descriptions	disable I2C interrupt
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
inttype	interrupt type
I2C_INT_ERR	error interrupt
I2C_INT_EV	event interrupt
I2C_INT_BUF	buffer interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable I2C0 event interrupt */

i2c_interrupt_disable (I2C0, I2C_INT_EV);

i2c_interrupt_flag_get

The description of i2c_interrupt_flag_get is shown as below:

Table 3-257. Function i2c_interrupt_flag_get

Function name	i2c_interrupt_flag_get
Function prototype	FlagStatus i2c_interrupt_flag_get(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag)
Function descriptions	get I2C interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
int_flag	interrupt flag

I2C_INT_FLAG_SBSEND	start condition sent out in master mode interrupt flag
I2C_INT_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode interrupt flag
I2C_INT_FLAG_BTC	byte transmission finishes
I2C_INT_FLAG_ADD10SEND	header of 10-bit address is sent in master mode interrupt flag
I2C_INT_FLAG_STPDET	stop condition detected in slave mode interrupt flag
I2C_INT_FLAG_RBNE	I2C_DATA is not Empty during receiving interrupt flag
I2C_INT_FLAG_TBE	I2C_DATA is empty during transmitting interrupt flag
I2C_INT_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus interrupt flag
I2C_INT_FLAG_LOSTARB	arbitration lost in master mode interrupt flag
I2C_INT_FLAG_AERR	acknowledge error interrupt flag
I2C_INT_FLAG_OUERR	over-run or under-run situation occurs in slave mode interrupt flag
I2C_INT_FLAG_PECERR	PEC error when receiving data interrupt flag
I2C_INT_FLAG_SMBTO	timeout signal in SMBus mode interrupt flag
I2C_INT_FLAG_SMBALT	SMBus alert status interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

/* check the byte transmission finishes interrupt flag is set or not*/

FlagStatus flag_state = RESET;

flag_state = i2c_interrupt_flag_get (I2C0, I2C_INT_FLAG_BTC);

i2c_interrupt_flag_clear

The description of i2c_interrupt_flag_clear is shown as below:

Table 3-258. Function i2c_interrupt_flag_clear

Function name	i2c_interrupt_flag_clear
Function prototype	void i2c_interrupt_flag_clear(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag);
Function descriptions	clear I2C interrupt flag status

Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2)
Input parameter{in}	
int_flag	interrupt flag
I2C_INT_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode interrupt flag
I2C_INT_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus interrupt flag
I2C_INT_FLAG_LOSTARB	arbitration lost in master mode interrupt flag
I2C_INT_FLAG_AERR	acknowledge error interrupt flag
I2C_INT_FLAG_OUERR	over-run or under-run situation occurs in slave mode interrupt flag
I2C_INT_FLAG_PECERR	PEC error when receiving data interrupt flag
I2C_INT_FLAG_SMBTO	timeout signal in SMBus mode interrupt flag
I2C_INT_FLAG_SMBALT	SMBus Alert status interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the acknowledge error interrupt flag */
i2c_interrupt_flag_clear (I2C0, I2C_INT_FLAG_AERR);
```

3.14. MISC

MISC is a software package that provide the interfaces for NVIC and SysTick.The NVIC and SysTick registers are listed in chapter ***3.14.1***, the MISC firmware functions are introduced in chapter ***3.14.2***.

3.14.1. Descriptions of Peripheral registers

Table 3-259. NVIC Registers

Registers	Descriptions
ISER(1)	Interrupt Set Enable Register

Registers	Descriptions
ICER[1]	Interrupt Clear Enable Register
ISPR[1]	Interrupt Set Pending Register
ICPR[1]	Interrupt Clear Pending Register
IABR[1]	Interrupt Active bit Register
IP[1]	Interrupt Priority Register
STIR[1]	Software Trigger Interrupt Register
CPUID[2]	CPUID Base Register
ICSR[2]	Interrupt Control and State Register
VTOR[2]	Vector Table Offset Register
AIRCR[2]	Application Interrupt and Reset Control Register
SCR[2]	System Control Register
CCR[2]	Configuration Control Register
SHP[2]	System Handlers Priority Registers
SHCSR[2]	System Handler Control and State Register
CFSR[2]	Configurable Fault Status Register
HFSR[2]	HardFault Status Register
DFSR[2]	Debug Fault Status Register
MMFAR[2]	MemManage Fault Address Register
BFAR[2]	BusFault Address Register
AFSR[2]	Auxiliary Fault Status Register
PFR[2]	Processor Feature Register
DFR[2]	Debug Feature Register
ADR[2]	Auxiliary Feature Register
MMFR[2]	Memory Model Feature Register
ISAR[2]	Instruction Set Attributes Register
CPACR[2]	Coprocessor Access Control Register

1. refer to the structure NVIC_Type, is defined in the core_cm3.h file
2. refer to the structure SCB_Type, is defined in the core_cm3.h file

Table 3-260. SysTick Registers

Registers	Descriptions
CTRL[1]	SysTick Control and Status Register
LOAD[1]	SysTick Reload Value Register
VAL[1]	SysTick Current Value Register
CALIB[1]	SysTick Calibration Register

1. refer to the structure SysTick_Type, is defined in the core_cm3.h file

3.14.2. Descriptions of Peripheral functions

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

Table 3-261. MISC firmware function

Function name	Function description
nvic_priority_group_set	set the priority group
nvic_irq_enable	enable NVIC interrupt request
nvic_irq_disable	disable NVIC interrupt request
nvic_vector_table_set	set the NVIC vector table address
system_lowpower_set	set the state of the low power mode
system_lowpower_reset	reset the state of the low power mode
systick_clksource_set	set the systick clock source

Enum IRQn_Type

Table 3-262. Enum IRQn_Type

Member name	Function description
WWDGT_IRQn	WWDGT interrupt
LVD_IRQn	LVD from EXTI line interrupt
RTC_IRQn	RTC global interrupt
FMC_IRQn	FMC interrupt
RCU_RQn	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
TSI_IRQn	TSI interrupt
DMA_Channel0_IRQn	DMA channel 0 global 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, CMP0 and CMP1 interrupts
TIMER0_BRK_UP_TRG_COM_IRQn	TIMER0 break, update, trigger and commutation interrupts
TIMER0_Channel_IRQn	TIMER0 channel capture compare interrupts
TIMER1_IRQn	TIMER1 interrupt
TIMER2_IRQn	TIMER2 interrupt
TIMER5_DAC_IRQn	TIMER5 and DAC interrupts
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
CEC_IRQn	CEC interrupt

Member name	Function description
I2C0_ER_IRQn	I2C0 error interrupt
I2C1_ER_IRQn	I2C1 error interrupt
I2C2_EV_IRQn	I2C2 event interrupt
I2C2_ER_IRQn	I2C2 error interrupt
USBD_LP_IRQn	USBD_LP interrupt
USBD_HP_IRQn	USBD_HP interrupt
USBDWakeUp_IRQChannel	USBD_WKUP interrupt
DMA_Channel5_6_IRQn	DMA channel 5 and channel 6 interrupts
SPI2_IRQn	SPI2 global interrupt

nvic_priority_group_set

The description of nvic_priority_group_set is shown as below:

Table 3-263. Function nvic_priority_group_set

Function name	nvic_priority_group_set
Function prototype	void nvic_priority_group_set(uint32_t nvic_prigroup);
Function descriptions	set the priority group
Precondition	-
The called functions	-
Input parameter{in}	
nvic_prigroup	priority group
NVIC_PRIGROUP_PRE0_SUB4	0 bits for pre-emption priority 4 bits for subpriority
NVIC_PRIGROUP_PRE1_SUB3	1 bits for pre-emption priority 3 bits for subpriority
NVIC_PRIGROUP_PRE2_SUB2	2 bits for pre-emption priority 2 bits for subpriority
NVIC_PRIGROUP_PRE3_SUB1	3 bits for pre-emption priority 1 bits for subpriority
NVIC_PRIGROUP_PRE4_SUB0	4 bits for pre-emption priority 0 bits for subpriority
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* priority group configuration, 0 bits for pre-emption priority 4 bits for subpriority */

nvic_priority_group_set(NVIC_PRIGROUP_PRE0_SUB4);
```

nvic_irq_enable

The description of nvic_irq_enable is shown as below:

Table 3-264. Function nvic_irq_enable

Function name	nvic_irq_enable
Function prototype	void nvic_irq_enable(uint8_t nvic_irq, uint8_t nvic_irq_pre_priority, uint8_t nvic_irq_sub_priority);
Function descriptions	enable NVIC request
Precondition	-
The called functions	nvic_priority_group_set
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to enum ***Table 3-262. Enum IRQn_Type***
Input parameter{in}	
nvic_irq_pre_priority	the pre-emption priority needed to set
Input parameter{in}	
nvic_irq_sub_priority	the subpriority needed to set
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable EXTI line 0 and 1 interrupts, nvic_irq_pre_priority is 2, nvic_irq_sub_priority is 0 */
nvic_irq_enable(EXTI0_1_IRQn, 2U, 0U);
```

nvic_irq_disable

The description of nvic_irq_disable is shown as below:

Table 3-265. Function nvic_irq_disable

Function name	nvic_irq_disable
Function prototype	void nvic_irq_disable(uint8_t nvic_irq);
Function descriptions	disable NVIC request
Precondition	-
The called functions	-
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to enum ***Table 3-262. Enum IRQn_Type***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable EXTI line 0 and 1 interrupts */
nvic_irq_disable(EXTI0_1_IRQn);
```

nvic_vector_table_set

The description of nvic_vector_table_set is shown as below:

Table 3-266. Function nvic_vector_table_set

Function name	nvic_vector_table_set
Function prototype	void nvic_vector_table_set(uint32_t nvic_vict_tab, uint32_t offset);
Function descriptions	set the NVIC vector table address
Precondition	-
The called functions	-
Input parameter{in}	
nvic_vict_tab	the RAM or FLASH base address
NVIC_VECTTAB_RAM	RAM base address
NVIC_VECTTAB_FLASH	Flash base address
Input parameter{in}	
Offset	Vector Table offset (vector table start address= base address+offset)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set vector table address = NVIC_VECTTAB_FLASH + 0x200 */
nvic_vector_table_set(NVIC_VECTTAB_FLASH, 0x200);
```

system_lowpower_set

The description of system_lowpower_set is shown as below:

Table 3-267. Function system_lowpower_set

Function name	system_lowpower_set
Function prototype	void system_lowpower_set(uint8_t lowpower_mode);
Function descriptions	set the state of the low power mode
Precondition	-
The called functions	-
Input parameter{in}	
lowpower_mode	the low power mode state
SCB_LPM_SLEEP_EXIT_ISR	if chose this para, the system always enter low power mode by exiting from ISR
SCB_LPM_DEEPSLEEP	if chose this para, the system will enter the DEEPSLEEP mode

SCB_LPM_WAKE_BY_ALL_INT	if chose this para, the lowpower mode can be woke up by all the enable and disable interrupts
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the system always enter low power mode by exiting from ISR */
system_lowpower_set(SCB_LPM_SLEEP_EXIT_ISR);
```

system_lowpower_reset

The description of system_lowpower_reset is shown as below:

Table 3-268. Function system_lowpower_reset

Function name	system_lowpower_reset
Function prototype	void system_lowpower_reset(uint8_t lowpower_mode);
Function descriptions	reset the state of the low power mode
Precondition	-
The called functions	-
Input parameter{in}	
lowpower_mode	the low power mode state
SCB_LPM_SLEEP_EXIT_ISR	if chose this para, the system will exit low power mode by exiting from ISR
SCB_LPM_DEEPSLEEP	if chose this para, the system will enter the SLEEP mode
SCB_LPM_WAKE_BY_ALL_INT	if chose this para, the lowpower mode only can be woke up by the enable interrupts
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the system will exit low power mode by exiting from ISR */
system_lowpower_reset(SCB_LPM_SLEEP_EXIT_ISR);
```

systick_clksource_set

The description of systick_clksource_set is shown as below:

Table 3-269. Function systick_clksource_set

Function name	systick_clksource_set
Function prototype	void systick_clksource_set(uint32_t systick_clksource);

Function descriptions	set the systick clock source
Precondition	-
The called functions	-
Input parameter{in}	
systick_clksource	the systick clock source needed to choose
SYSTICK_CLKSOURCE_HCLK	systick clock source is from HCLK
SYSTICK_CLKSOURCE_HCLK_DIV8	systick clock source is from HCLK/8
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* systick clock source is HCLK/8 */
systick_clksource_set(SYSTICK_CLKSOURCE_HCLK_DIV8);
```

3.15. PMU

According to 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 *3.15.1*, the PMU firmware functions are introduced in chapter *3.15.2*.

3.15.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

Table 3-270. PMU Registers

Registers	Descriptions
PMU_CTL	PMU control register
PMU_CS	PMU control and status register

3.15.2. Descriptions of Peripheral functions

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

Table 3-271. PMU firmware function

Function name	Function description
pmu_deinit	reset PMU registers
pmu_lvd_select	select low voltage detector threshold
pmu_lvd_disable	disable PMU lvd
pmu_to_sleepmode	PMU work in sleep mode

Function name	Function description
pmu_to_deepsleepmode	PMU work at deepsleep mode
pmu_to_standbymode	pmu work at standby mode
pmu_wakeup_pin_enable	enable PMU wakeup pin
pmu_wakeup_pin_disable	disable PMU wakeup pin
pmu_backup_write_enable	enable backup domain write
pmu_backup_write_disable	disable backup domain write
pmu_flag_clear	clear flag bit
pmu_flag_get	get flag state

pmu_deinit

The description of pmu_deinit is shown as below:

Table 3-272. Function pmu_deinit

Function name	pmu_deinit
Function prototype	void pmu_deinit(void);
Function descriptions	reset PMU register
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset PMU */

pmu_deinit ();
```

pmu_lvd_select

The description of pmu_lvd_select is shown as below:

Table 3-273. Function pmu_lvd_select

Function name	pmu_lvd_select
Function prototype	void pmu_lvd_select(uint32_t lvdt_n);
Function descriptions	select low voltage detector threshold
Precondition	-
The called functions	-
Input parameter{in}	
lvdt_n	voltage threshold value
PMU_LVDT_0	voltage threshold is 2.2V

PMU_LVDT_1	voltage threshold is 2.3V
PMU_LVDT_2	voltage threshold is 2.4V
PMU_LVDT_3	voltage threshold is 2.5V
PMU_LVDT_4	voltage threshold is 2.6V
PMU_LVDT_5	voltage threshold is 2.7V
PMU_LVDT_6	voltage threshold is 2.8V
PMU_LVDT_7	voltage threshold is 2.9V
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* select low voltage detector threshold as 2.9V */

pmu_lvd_select (PMU_LVDT_7);

pmu_lvd_disable

The description of pmu_lvd_disable is shown as below:

Table 3-274. Function pmu_lvd_disable

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

Example:

/* disable PMU lvd */

pmu_lvd_disable ();

pmu_to_sleepmode

The description of pmu_to_sleepmode is shown as below:

Table 3-275. Function pmu_to_sleepmode

Function name	pmu_to_sleepmode
Function prototype	void pmu_to_sleepmode(uint8_t sleepmodecmd);

Function descriptions	PMU work at sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
sleepmodecmd	command to enter sleep mode
WFI_CMD	use WFI command
WFE_CMD	use WFE command
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* PMU work at sleep mode */

pmu_to_sleepmode (WFI_CMD);

pmu_to_deepsleepmode

The description of pmu_to_deepsleepmode is shown as below:

Table 3-276. Function pmu_to_deepsleepmode

Function name	pmu_to_deepsleepmode
Function prototype	void pmu_to_deepsleepmode(uint32_t ldo,uint8_t deepsleepmodecmd);
Function descriptions	PMU work at deepsleep mode
Precondition	-
The called functions	-
Input parameter{in}	
ldo	ldo work mode
PMU_LDO_NORMAL	LDO operates normally when pmu enter deepsleep mode
PMU_LDO_LOWPOWER	LDO work at low power mode when pmu enter deepsleep mode
Input parameter{in}	
deepsleepmodecmd	command to enter deepsleep mode
WFI_CMD	use WFI command
WFE_CMD	use WFE command
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* PMU work at deepsleep mode */

pmu_to_deepsleepmode (PMU_LDO_NORMAL, WFI_CMD);

pmu_to_standbymode

The description of pmu_to_standbymode is shown as below:

Table 3-277. Function pmu_to_standbymode

Function name	pmu_to_standbymode
Function prototype	void pmu_to_standbymode(uint8_t standbymodecmd);
Function descriptions	pmu work at standby mode
Precondition	-
The called functions	-
Input parameter{in}	
standbymodecmd	command to enter standby mode
WFI_CMD	use WFI command
WFE_CMD	use WFE command
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* PMU work at standby mode */

pmu_to_standby (WFI_CMD);

pmu_wakeup_pin_enable

The description of pmu_wakeup_pin_enable is shown as below:

Table 3-278. Function pmu_wakeup_pin_enable

Function name	pmu_wakeup_pin_enable
Function prototype	void pmu_wakeup_pin_enable(uint32_t wakeup_pin);
Function descriptions	enable wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_pin	Wakeup pin
PMU_WAKEUP_PIN0	WKUP Pin 0
PMU_WAKEUP_PIN1	WKUP Pin 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable wakeup pin0 */

pmu_wakeup_pin_enable (PMU_WAKEUP_PIN0);

pmu_wakeup_pin_disable

The description of pmu_wakeup_pin_disable is shown as below:

Table 3-279. Function pmu_wakeup_pin_disable

Function name	pmu_wakeup_pin_disable
Function prototype	void pmu_wakeup_pin_disable(uint32_t wakeup_pin);
Function descriptions	disable wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_pin	Wakeup pin
PMU_WAKEUP_PIN0	WKUP Pin 0
PMU_WAKEUP_PIN1	*WKUP Pin 1*
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable wakeup pin0 */

pmu_wakeup_pin_disable (PMU_WAKEUP_PIN0);

pmu_backup_write_enable

The description of pmu_backup_write_enable is shown as below:

Table 3-280. Function pmu_backup_write_enable

Function name	pmu_backup_write_enable
Function prototype	void pmu_backup_write_enable (void);
Function descriptions	enable backup domain write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable backup domain write */

pmu_backup_write_enable ();

pmu_backup_write_disable

The description of pmu_backup_write_disable is shown as below:

Table 3-281. Function pmu_backup_write_disable

Function name	pmu_backup_write_disable
Function prototype	void pmu_backup_write_disable (void);
Function descriptions	disable backup domain write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable backup domain write */

pmu_backup_write_disable ();

pmu_flag_clear

The description of pmu_flag_clear is shown as below:

Table 3-282. Function pmu_flag_clear

Function name	pmu_flag_clear
Function prototype	void pmu_flag_clear(uint32_t flag_clear);
Function descriptions	clear flag bit
Precondition	-
The called functions	-
Input parameter{in}	
flag_clear	flag
PMU_FLAG_RESET_WAKEUP	reset wakeup flag
PMU_FLAG_RESET_STANDBY	reset standby flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear flag bit */

pmu_flag_clear (PMU_FLAG_RESET_WAKEUP);

pmu_flag_get

The description of pmu_flag_get is shown as below:

Table 3-283. Function pmu_flag_get

Function name	pmu_flag_get
Function prototype	FlagStatus pmu_flag_get(uint32_t flag);
Function descriptions	get flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag
PMU_FLAG_WAKEUP	wakeup flag
PMU_FLAG_STANDBY	standby flag
PMU_FLAG_LVD	lvd flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get flag state */

FlagStatus status;

status = pmu_flag_get (PMU_FLAG_WAKEUP);

3.16. 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 ***3.16.1***, the RCU firmware functions are introduced in chapter ***3.16.2***.

3.16.1. Descriptions of Peripheral registers

Table 3-284. RCU Registers

Registers	Descriptions
RCU_CTL0	Control register 0
RCU_CFG0	Clock configuration register 0
RCU_INT	Clock interrupt register

Registers	Descriptions
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_ADDAPB1EN	APB1 additional enable register
RCU_ADDAPB1RST	APB1 additional reset register
RCU_VKEY	Unlock voltage register
RCU_DSV	Deep-sleep mode voltage register
RCU_PDVSEL	Power down voltage select register

3.16.2. Descriptions of Peripheral functions

Table 3-285. RCU firmware function

Function name	Function description
rcu_deinit	deinitialize the RCU
rcu_periph_clock_enable	enable the peripherals clock
rcu_periph_clock_disable	disable the peripherals clock
rcu_periph_clock_sleep_enable	enable the peripherals clock when in sleep mode
rcu_periph_clock_sleep_disable	disable the peripherals clock when in sleep mode
rcu_periph_reset_enable	enable the peripherals reset
rcu_periph_reset_disable	disable the peripheral reset
rcu_bkp_reset_enable	enable the BKP domain reset
rcu_bkp_reset_disable	disable the BKP domain reset
rcu_system_clock_source_config	configure the system clock source
rcu_system_clock_source_get	get the system clock source
rcu_ahb_clock_config	configure the AHB clock prescaler selection
rcu_apb1_clock_config	configure the APB1 clock prescaler selection
rcu_apb2_clock_config	configure the APB2 clock prescaler selection
rcu_adc_clock_config	configure the ADC clock source and prescaler selection
rcu_usbd_clock_config	configure the USBD prescaler selection
rcu_ckout_config	configure the CK_OUT clock source and divider
rcu_pll_config	configure the PLL clock source selection and PLL multiply factor
rcu_usart_clock_config	configure the USART clock source selection

Function name	Function description
rcu_cec_clock_config	configure the CEC clock source selection
rcu_rtc_clock_config	configure the RTC clock source selection
rcu_hxtal_prediv_config	configure the HXTAL divider used as input of PLL
rcu_lxtal_drive_capability_config	configure the LXTAL drive capability
rcu_flag_get	get the clock stabilization and periphral reset flags
rcu_all_reset_flag_clear	clear all the reset flag
rcu_interrupt_flag_get	get the clock stabilization interrupt and ckm flags
rcu_interrupt_flag_clear	clear the interrupt flags
rcu_interrupt_enable	enable the stabilization interrupt
rcu_interrupt_disable	disable the stabilization interrupt
rcu_osci_stab_wait	wait for oscillator stabilization flags is SET or oscillator startup is timeout
rcu_osci_on	turn on the oscillator
rcu_osci_off	turn off the oscillator
rcu_osci_bypass_mode_enable	enable the oscillator bypass mode
rcu_osci_bypass_mode_disable	disable the oscillator bypass mode
rcu_hxtal_clock_monitor_enable	enable the HXTAL clock monitor
rcu_hxtal_clock_monitor_disable	disable the HXTAL clock monitor
rcu_irc8m_adjust_value_set	set the IRC8M adjust value
rcu_irc14m_adjust_value_set	set the IRC14M adjust value
rcu_voltage_key_unlock	unlock Deep-sleep mode voltage register
rcu_deepsleep_voltage_set	set the deep-sleep mode voltage value
rcu_power_down_voltage_set	set the power down voltage
rcu_clock_freq_get	get the system clock, bus and peripheral clock frequency

reg_idx

Table 3-286. Enum reg_idx

enum name	Function description
IDX_AHBEN	index of AHB enable register
IDX_APB2EN	index of APB2 enable register
IDX_APB1EN	index of APB1 enable register
IDX_ADDAPB1EN	index of APB1 additional enable register
IDX_AHBRST	index of AHB reset register
IDX_APB2RST	index of APB2 reset register
IDX_APB1RST	index of APB1 reset register
IDX_ADDAPB1RST	index of APB1 additional reset register
IDX_CTL0	index of control register0
IDX_BDCTL	index of backup domain control register
IDX_CTL1	index of control register1
IDX_RSTSCK	index of reset source / clock register
IDX_INT	index of interrupt register

enum name	Function description
IDX_CFG0	index of configuration register0
IDX_CFG2	index of configuration register2

rcu_periph_enum

Table 3-287. Enum rcu_periph_enum

enum name	Function description
RCU_DMA	DMA clock
RCU_CRC	CRC clock
RCU_GPIOA	GPIOA clock
RCU_GPIOB	GPIOB clock
RCU_GPIOC	GPIOC clock
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_TIMER5	TIMER5 clock
RCU_TIMER13	TIMER13 clock
RCU_WWDGT	WWDGT clock
RCU_SPI1	SPI1 clock
RCU_SPI2	SPI2 clock
RCU_USART1	USART1 clock
RCU_I2C0	I2C0 clock
RCU_I2C1	I2C1 clock
RCU_USBD	USBD clock
RCU_PMU	PMU clock
RCU_DAC	DAC clock
RCU_CEC	CEC clock
RCU_RTC	RTC clock
RCU_I2C2	I2C2 clock

rcu_periph_sleep_enum

Table 3-288. Enum rcu_periph_sleep_enum

enum name	Function description
RCU_SRAM_SLP	SRAM clock when sleep mode
RCU_FMC_SLP	FMC clock when sleep mode

rcu_periph_reset_enum

Table 3-289. Enum rcu_periph_reset_enum

enum name	Function description
RCU_GPIOARST	GPIOA reset
RCU_GPIOBRST	GPIOB reset
RCU_GPIOCRST	GPIOC reset
RCU_GPIODRST	GPIOD reset
RCU_GPIOFRST	GPIOF reset
RCU_TSIRST	TSI reset
RCU_CFGCMPRST	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_TIMER5RST	TIMER5 reset
RCU_TIMER13RST	TIMER13 reset
RCU_WWDGTRST	WWDGT reset
RCU_SPI1RST	SPI1 reset
RCU_SPI2RST	SPI2 reset
RCU_USART1RST	USART1 reset
RCU_I2C0RST	I2C0 reset
RCU_I2C1RST	I2C1 reset
RCU_USBDRST	USBD reset
RCU_PMURST	PMU reset
RCU_DACRST	DAC reset
RCU_CECRST	CEC reset
RCU_I2C2RST	I2C2 reset

rcu_flag_enum

Table 3-290. Enum rcu_flag_enum

enum 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
RCU_FLAG_IRC14MSTB	IRC14M stabilization flags
RCU_FLAG_V12RST	1.2V domain Power reset flags
RCU_FLAG_OBLRST	Option byte loader reset flags
RCU_FLAG_EPRST	External PIN reset flags
RCU_FLAG_PORRST	Power reset flags
RCU_FLAG_SWRST	Software reset flags
RCU_FLAG_FWDGTRST	Free watchdog timer reset flags
RCU_FLAG_WWDGTRST	Window watchdog timer reset flags
RCU_FLAG_LPRST	Low-power reset flags

rcu_int_flag_enum

Table 3-291. Enum rcu_int_flag_enum

enum name	Function description
RCU_INT_FLAG_IRC40KSTB	IRC40K stabilization interrupt flag
RCU_INT_FLAG_LXTALSTB	LXTAL stabilization interrupt flag
RCU_INT_FLAG_IRC8MSTB	IRC8M stabilization interrupt flag
RCU_INT_FLAG_HXTALSTB	HXTAL stabilization interrupt flag
RCU_INT_FLAG_PLLSTB	PLL stabilization interrupt flag
RCU_INT_FLAG_IRC14MSTB	IRC14M stabilization interrupt flag
RCU_INT_FLAG_CKM	CKM interrupt flag

rcu_int_flag_clear_enum

Table 3-292. Enum rcu_int_flag_clear_enum

enum name	Function description
RCU_INT_FLAG_IRC40KSTB_CLR	IRC40K stabilization interrupt flags clear
RCU_INT_FLAG_LXTALSTB_CLR	LXTAL stabilization interrupt flags clear
RCU_INT_FLAG_IRC8MSTB_CLR	IRC8M stabilization interrupt flags clear
RCU_INT_FLAG_HXTALSTB_CLR	HXTAL stabilization interrupt flags clear
RCU_INT_FLAG_PLLSTB_CLR	PLL stabilization interrupt flags clear
RCU_INT_FLAG_IRC14MSTB_CLR	IRC14M stabilization interrupt flags clear
RCU_INT_FLAG_CKM_CLR	CKM interrupt flags clear

rcu_int_enum

Table 3-293. Enum rcu_int_enum

enum name	Function description
RCU_INT_IRC40KSTB	IRC40K stabilization interrupt
RCU_INT_LXTALSTB	LXTAL stabilization interrupt
RCU_INT_IRC8MSTB	IRC8M stabilization interrupt
RCU_INT_HXTALSTB	HXTAL stabilization interrupt
RCU_INT_PLLSTB	PLL stabilization interrupt
RCU_INT_IRC14MSTB	IRC14M stabilization interrupt

rcu_adc_clock_enum

Table 3-294. Enum rcu_adc_clock_enum

enum name	Function description
RCU_ADCCK_IRC14M	ADC clock source select IRC14
RCU_ADCCK_APB2_DIV2	ADC clock source select APB2/2
RCU_ADCCK_APB2_DIV4	ADC clock source select APB2/4
RCU_ADCCK_APB2_DIV6	ADC clock source select APB2/6
RCU_ADCCK_APB2_DIV8	ADC clock source select APB2/8

rcu_osci_type_enum

Table 3-295. Enum rcu_osci_type_enum

enum name	Function description
RCU_HXTAL	HXTAL
RCU_LXTAL	LXTAL
RCU_IRC8M	IRC8M
RCU_IRC14M	IRC14M
RCU_IRC40K	IRC40K
RCU_PLL_CK	PLL

rcu_clock_freq_enum

Table 3-296. Enum rcu_clock_freq_enum

enum name	Function description
CK_SYS	system clock
CK_AHB	AHB clock
CK_APB1	APB1 clock
CK_APB2	APB2 clock
CK_ADC	ADC clock
CK_CEC	CEC clock
CK_USART	USART clock

rcu_deinit

The description of rcu_deinit is shown as below:

Table 3-297. Function rcu_deinit

Function name	rcu_deinit
Function prototype	void rcu_deinit(void);
Function descriptions	deinitialize the RCU, reset the value of all RCU registers into initial values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset RCU */

rcu_deinit();
```

rcu_periph_clock_enable

The description of rcu_periph_clock_enable is shown as below:

Table 3-298. Function rcu_periph_clock_enable

Function name	rcu_periph_clock_enable
Function prototype	void rcu_periph_clock_enable(rcu_periph_enum periph);
Function descriptions	enable the peripherals clock
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to ***Table 3-287. Enum rcu_periph_enum***.
RCU_GPIOx	GPIO ports clock (x=A,B,C,D,F)
RCU_DMA	DMA clock
RCU_CRC	CRC clock
RCU_TSI	TSI clock
RCU_CFGCMP	CFGCMP clock
RCU_ADC	ADC clock
RCU_TIMERx	TIMERx clock(x=0,1,2,5,13,14,15,16)
RCU_SPIx	SPIx clock (x=0,1,2)
RCU_USARTx	USARTx clock (x=0,1)
RCU_WWDGT	WWDGT clock
RCU_I2Cx	I2Cx clock (x=0,1,2)
RCU_USBD	USBD clock
RCU_PMU	PMU clock
RCU_DAC	DAC clock
RCU_CEC	CEC clock
RCU_RTC	RTC clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the USART0 clock */

rcu_periph_clock_enable(RCU_USART0);
```

rcu_periph_clock_disable

The description of rcu_periph_clock_disable is shown as below:

Table 3-299. Function rcu_periph_clock_disable

Function name	rcu_periph_clock_disable
Function prototype	void rcu_periph_clock_disable(rcu_periph_enum periph);

Function descriptions	disable the peripherals clock
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to ***Table 3-287. Enum rcu_periph_enum***.
RCU_GPIOx	GPIO ports clock (x=A,B,C,D,F)
RCU_DMA	DMA clock
RCU_CRC	CRC clock
RCU_TSI	TSI clock
RCU_CFGCMP	CFGCMP clock
RCU_ADC	ADC clock
RCU_TIMERx	TIMERx clock(x=0,1,2,5,13,14,15,16)
RCU_SPIx	SPIx clock (x=0,1,2)
RCU_USARTx	USARTx clock (x=0,1)
RCU_WWDGT	WWDGT clock
RCU_I2Cx	I2Cx clock (x=0,1,2)
RCU_USBD	USBD clock
RCU_PMU	PMU clock
RCU_DAC	DAC clock
RCU_CEC	CEC clock
RCU_RTC	RTC clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the USART0 clock */

rcu_periph_clock_disable(RCU_USART0);
```

rcu_periph_clock_sleep_enable

The description of rcu_periph_clock_sleep_enable is shown as below:

Table 3-300. Function rcu_periph_clock_sleep_enable

Function name	rcu_periph_clock_sleep_enable
Function prototype	void rcu_periph_clock_sleep_enable(rcu_periph_sleep_enum periph);
Function descriptions	enable the peripherals clock when in sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to ***Table 3-288. Enum rcu_periph_sleep_enum***.
RCU_FMC_SLP	FMC clock when sleep mode

RCU_SRAM_SLP	SRAM clock when sleep mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the FMC clock when in sleep mode */

rcu_periph_clock_sleep_enable(RCU_FMC_SLP);

rcu_periph_clock_sleep_disable

The description of rcu_periph_clock_sleep_disable is shown as below:

Table 3-301. Function rcu_periph_clock_sleep_disable

Function name	rcu_periph_clock_sleep_disable
Function prototype	void rcu_periph_clock_sleep_disable(rcu_periph_sleep_enum periph);
Function descriptions	disable the peripherals clock when in sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to ***Table 3-288. Enum rcu_periph_sleep_enum***.
RCU_FMC_SLP	FMC clock when sleep mode
RCU_SRAM_SLP	SRAM clock when sleep mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the FMC clock when in sleep mode */

rcu_periph_clock_sleep_disable(RCU_FMC_SLP);

rcu_periph_reset_enable

The description of rcu_periph_reset_enable is shown as below:

Table 3-302. Function rcu_periph_reset_enable

Function name	rcu_periph_reset_enable
Function prototype	void rcu_periph_reset_enable(rcu_periph_reset_enum periph_reset);
Function descriptions	enable the peripherals reset
Precondition	-
The called functions	-
Input parameter{in}	

periph_reset	RCU peripherals reset, refer to ***Table 3-289. Enum rcu_periph_reset_enum***.
RCU_GPIOxRST	reset GPIO ports clock (x=A,B,C,D,F)
RCU_TSIRST	reset TSI clock
RCU_CFGCMPRST	reset CFGCMP clock
RCU_ADCRST	reset ADC clock
RCU_TIMERxRST	reset TIMERx clock (x=0,1,2,5,13,14,15,16)
RCU_SPIxRST	reset SPIx clock (x=0,1,2)
RCU_USARTxRST	reset USARTx clock (x=0,1)
RCU_WWDGTRST	reset WWDGT clock
RCU_I2CxRST	reset I2Cx clock (x=0,1,2)
RCU_USBDRST	reset USBD clock
RCU_PMURST	reset PMU clock
RCU_DACRST	reset DAC
RCU_CECRST	reset CEC
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable SPI0 reset */

rcu_periph_reset_enable(RCU_SPI0RST);

rcu_periph_reset_disable

The description of rcu_periph_reset_disable is shown as below:

Table 3-303. Function rcu_periph_reset_disable

Function name	rcu_periph_reset_disable
Function prototype	void rcu_periph_reset_disable(rcu_periph_reset_enum periph_reset);
Function descriptions	disable the peripheral reset
Precondition	-
The called functions	-
Input parameter{in}	
periph_reset	RCU peripherals reset, refer to ***Table 3-289. Enum rcu_periph_reset_enum***.
RCU_GPIOxRST	reset GPIO ports clock (x=A,B,C,D,F)
RCU_TSIRST	reset TSI clock
RCU_CFGCMPRST	reset CFGCMP clock
RCU_ADCRST	reset ADC clock
RCU_TIMERxRST	reset TIMERx clock (x=0,1,2,5,13,14,15,16)
RCU_SPIxRST	reset SPIx clock (x=0,1,2)

RCU_USARTxRST	reset USARTx clock (x=0,1)
RCU_WWDGTRST	reset WWDGT clock
RCU_I2CxRST	reset I2Cx clock (x=0,1,2)
RCU_USBDRST	reset USBD clock
RCU_PMURST	reset PMU clock
RCU_DACRST	reset DAC
RCU_CECRST	reset CEC
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable SPI0 reset */

rcu_periph_reset_disable(RCU_SPI0RST);

rcu_bkp_reset_enable

The description of rcu_bkp_reset_enable is shown as below:

Table 3-304. Function rcu_bkp_reset_enable

Function name	rcu_bkp_reset_enable
Function prototype	void rcu_bkp_reset_enable(void);
Function descriptions	enable the BKP domain reset
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* reset the BKP domain */

rcu_bkp_reset_enable();

rcu_bkp_reset_disable

The description of rcu_bkp_reset_disable is shown as below:

Table 3-305. Function rcu_bkp_reset_disable

Function name	rcu_bkp_reset_disable
Function prototype	void rcu_bkp_reset_disable(void);

Function descriptions	disable the BKP domain reset
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the BKP domain reset */

rcu_bkp_reset_disable();

rcu_system_clock_source_config

The description of rcu_system_clock_source_config is shown as below:

Table 3-306. Function rcu_system_clock_source_config

Function name	rcu_system_clock_source_config
Function prototype	void rcu_system_clock_source_config(uint32_t ck_sys);
Function descriptions	configure the system clock source
Precondition	-
The called functions	-
Input parameter{in}	
ck_sys	system clock source select
RCU_CKSYSSRC_IRC8M	select CK_IRC8M as the CK_SYS source
RCU_CKSYSSRC_HXTAL	select CK_HXTAL as the CK_SYS source
RCU_CKSYSSRC_PLL	select CK_PLL as the CK_SYS source
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the CK_HXTAL as the CK_SYS source */

rcu_system_clock_source_config(RCU_CKSYSSRC_HXTAL);

rcu_system_clock_source_get

The description of rcu_system_clock_source_get is shown as below:

Table 3-307. Function rcu_system_clock_source_get

Function name	rcu_system_clock_source_get
Function prototype	uint32_t rcu_system_clock_source_get(void);
Function descriptions	get the system clock source
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	RCU_SCSS_IRC8M/RCU_SCSS_HXTAL/RCU_SCSS_PLL

Example:

```
uint32_t temp_cksys_status;

/* get the CK_SYS source */

temp_cksys_status = rcu_system_clock_source_get();
```

rcu_ahb_clock_config

The description of rcu_ahb_clock_config is shown as below:

Table 3-308. Function rcu_ahb_clock_config

Function name	rcu_ahb_clock_config
Function prototype	void rcu_ahb_clock_config(uint32_t ck_ahb);
Function descriptions	configure the AHB clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_ahb	AHB clock prescaler selection
RCU_AHB_CKSYS_DIVx	select CK_SYS / x, (x=1, 2, 4, 8, 16, 64, 128, 256, 512)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_SYS/128 */

rcu_ahb_clock_config(RCU_AHB_CKSYS_DIV128);
```

rcu_apb1_clock_config

The description of rcu_apb1_clock_config is shown as below:

Table 3-309. Function rcu_apb1_clock_config

Function name	rcu_apb1_clock_config
Function prototype	void rcu_apb1_clock_config(uint32_t ck_apb1);
Function descriptions	configure the APB1 clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_apb1	APB1 clock prescaler selection
RCU_APB1_CKAHB_DIVx	select (CK_AHB / x) as CK_APB1 (x=1,2,4,8,16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/16 as CK_APB1 */

rcu_apb1_clock_config(RCU_APB1_CKAHB_DIV16);
```

rcu_apb2_clock_config

The description of rcu_apb2_clock_config is shown as below:

Table 3-310. Function rcu_apb2_clock_config

Function name	rcu_apb2_clock_config
Function prototype	void rcu_apb2_clock_config(uint32_t ck_apb2);
Function descriptions	configure the APB2 clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_apb2	APB2 clock prescaler selection
RCU_APB2_CKAHB_DIVx	select (CK_AHB / x) as CK_APB2 clock (x=1,2,4,8,16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB2 */
```

rcu_apb2_clock_config(RCU_APB2_CKAHB_DIV8);

rcu_adc_clock_config

The description of rcu_adc_clock_config is shown as below:

Table 3-311. Function rcu_adc_clock_config

Function name	rcu_adc_clock_config
Function prototype	void rcu_adc_clock_config(rcu_adc_clock_enum ck_adc);
Function descriptions	configure the ADC clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_adc	ADC clock prescaler selection, refer to ***Table 3-294. Enum rcu_adc_clock_enum***.
RCU_ADCCK_IRC14M	select CK_IRC14M as CK_ADC
RCU_ADCCK_IRC28M	select CK_IRC28M as CK_ADC
RCU_ADCCK_APB2_DIVx	select (CK_APB2 / x) as CK_ADC(x=2,4,6,8)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the ADC prescaler factor */

rcu_adc_clock_config(RCU_CKADC_CKAPB2_DIV8);

rcu_usbd_clock_config

The description of rcu_usbd_clock_config is shown as below:

Table 3-312. Function rcu_usbd_clock_config

Function name	rcu_usbd_clock_config
Function prototype	void rcu_usbd_clock_config(uint32_t ck_usbd);
Function descriptions	configure the USBD clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_usbd	USBD clock prescaler selection
RCU_USBD_CKPLL_DIV1	select CK_PLL as CK_USBD
RCU_USBD_CKPLL_DIV1_5	select CK_PLL / 1.5 as CK_USBD

RCU_USBD_CKPLL_DIV2	select CK_PLL / 2 as CK_USBD
RCU_USBD_CKPLL_DIV2_5	select CK_PLL / 2.5 as CK_USBD
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the CK_PLL as CK_USBD */

rcu_usbd_clock_config(RCU_USBD_CKPLL_DIV1);

rcu_ckout_config

The description of rcu_ckout_config is shown as below:

Table 3-313. Function rcu_ckout_config

Function name	rcu_ckout_config
Function prototype	void rcu_ckout_config(uint32_t ckout_src, uint32_t ckout_div);
Function descriptions	configure the CK_OUT clock source and divoision factor
Precondition	-
The called functions	-
Input parameter{in}	
ckout_src	CK_OUT clock source selection
RCU_CKOUTSRC_NONE	no clock selected
RCU_CKOUTSRC_IRC14M	select high speed 14M internal oscillator clock
RCU_CKOUTSRC_IRC40K	select high speed 40K internal oscillator clock
RCU_CKOUTSRC_LXTAL	select LXTAL clock
RCU_CKOUTSRC_CKSYS	select system clock CK_SYS
RCU_CKOUTSRC_IRC8M	select high speed 8M internal oscillator clock
RCU_CKOUTSRC_HXTAL	select HXTAL clock
RCU_CKOUTSRC_CKPLL_DIV1	select CK_PLL clock
RCU_CKOUTSRC_CKPLL_DIV2	Select (CK_PLL / 2)clock

Input parameter{in}	
ckout_div	CK_OUT divider
RCU_CKOUT_DIVx	CK_OUT is divided by x(x=1,2,4,8,16,32,64,128)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the HXTAL as CK_OUT clock source */

rcu_ckout_config(RCU_CKOUTSRC_HXTAL, RCU_CKOUT_DIV1);

rcu_ckout0_config

The description of rcu_ckout0_config is shown as below:

Table 3-314. Function rcu_ckout0_config

Function name	rcu_ckout0_config
Function prototype	void rcu_ckout0_config(uint32_t ckout0_src, uint32_t ckout0_div);
Function descriptions	configure the CK_OUT0 clock source and divoision factor
Precondition	-
The called functions	-
Input parameter{in}	
ckout0_src	CK_OUT0 clock source selection
RCU_CKOUT0SRC_NONE	no clock selected
RCU_CKOUT0SRC_IRC28M	select high speed 28M internal oscillator clock
RCU_CKOUT0SRC_IRC40K	select high speed 40K internal oscillator clock
RCU_CKOUT0SRC_LXTAL	select LXTAL clock
RCU_CKOUT0SRC_CKSYS	select system clock CK_SYS
RCU_CKOUT0SRC_IRC8M	select high speed 8M internal oscillator clock
RCU_CKOUT0SRC_HXTAL	select HXTAL clock
RCU_CKOUT0SRC_CKPLL_DIV1	select CK_PLL clock
RCU_CKOUT0SRC_CKPLL_DIV2	Select (CK_PLL / 2) clock
Input parameter{in}	

ckout0_div	CK_OUT0 divider
RCU_CKOUT0_DIVx	CK_OUT0 is divided by x(x=1,2,4,8,16,32,64,128)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the HXTAL as CK_OUT0 clock source */

rcu_ckout0_config(RCU_CKOUT0SRC_HXTAL, RCU_CKOUT0_DIV1);

rcu_ckout1_config

The description of rcu_ckout1_config is shown as below:

Table 3-315. Function rcu_ckout1_config

Function name	rcu_ckout1_config
Function prototype	void rcu_ckout1_config(uint32_t ckout1_src, uint32_t ckout1_div);
Function descriptions	configure the CK_OUT1 clock source and divoision factor
Precondition	-
The called functions	-
Input parameter{in}	
ckout1_src	CK_OUT1 clock source selection
RCU_CKOUT1SRC_NONE	no clock selected
RCU_CKOUT1SRC_IRC28M	select high speed 28M internal oscillator clock
RCU_CKOUT1SRC_IRC40K	select high speed 40K internal oscillator clock
RCU_CKOUT1SRC_LXTAL	select LXTAL clock
RCU_CKOUT1SRC_CKSYS	select system clock CK_SYS
RCU_CKOUT1SRC_IRC8M	select high speed 8M internal oscillator clock
RCU_CKOUT1SRC_HXTAL	select HXTAL clock
RCU_CKOUT1SRC_CKPLL_DIV1	select CK_PLL clock
RCU_CKOUT1SRC_CKPLL_DIV2	Select（CK_PLL / 2）clock
Input parameter{in}	
ckout1_div	CK_OUT1 divider

RCU_CKOUT1_DIVx	CK_OUT1 is divided by x(x=1,2,…,64)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the HXTAL as CK_OUT1 clock source */

rcu_ckout1_config(RCU_CKOUT1SRC_HXTAL, RCU_CKOUT1_DIV1);

rcu_pll_config

The description of rcu_pll_config is shown as below:

Table 3-316. Function rcu_pll_config

Function name	rcu_pll_config
Function prototype	void rcu_pll_config(uint32_t pll_src, uint32_t pll_mul);
Function descriptions	configure the PLL clock source selection and PLL multiply factor
Precondition	-
The called functions	-
Input parameter{in}	
pll_src	PLL clock source selection
RCU_PLLSRC_IRC8M_DIV2	select CK_IRC8M/2 as PLL source clock
RCU_PLLSRC_HXTAL	select HXTAL as PLL source clock
Input parameter{in}	
pll_mul	PLL clock multiplication factor
RCU_PLL_MULx	PLL source clock * x (x = 2..32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the PLL */

rcu_pll_config(RCU_PLLSRC_IRC8M_DIV2, RCU_PLL_MUL10);

rcu_usart_clock_config

The description of rcu_usart_clock_config is shown as below:

Table 3-317. Function rcu_usart_clock_config

Function name	rcu_usart_clock_config
Function prototype	void rcu_usart_clock_config(uint32_t ck_usart);

Function descriptions	configure the USART clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_usart	USART clock source selection
RCU_USART0SRC_CKAPB2	CK_USART0 select CK_APB2
RCU_USART0SRC_CKSYS	CK_USART0 select CK_SYS
RCU_USART0SRC_LXTAL	CK_USART0 select CK_LXTAL
RCU_USART0SRC_IRC8M	CK_USART0 select CK_IRC8M
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the LXTAL as USART0 clock */
rcu_usart_clock_config(RCU_USART0SRC_LXTAL);
```

rcu_cec_clock_config

The description of rcu_cec_clock_config is shown as below:

Table 3-318. Function rcu_cec_clock_config

Function name	rcu_cec_clock_config
Function prototype	void rcu_cec_clock_config(uint32_t ck_cec);
Function descriptions	configure the CEC clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_cec	CEC clock source selection
RCU_CECSRC_IRC8M_DIV244	CK_CEC select CK_IRC8M/244
RCU_CECSRC_LXTAL	CK_CEC select CK_LXTAL
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the CEC clock source selection */

rcu_cec_clock_config(RCU_CECSRC_LXTAL);

rcu_rtc_clock_config

The description of rcu_rtc_clock_config is shown as below:

Table 3-319. Function rcu_rtc_clock_config

Function name	rcu_rtc_clock_config
Function prototype	void rcu_rtc_clock_config(uint32_t rtc_clock_source);
Function descriptions	configure the RTC clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
rtc_clock_source	RTC clock source selection
RCU_RTCSRC_NONE	no clock selected
RCU_RTCSRC_LXTAL	select CK_LXTAL as RTC source clock
RCU_RTCSRC_IRC40K	select CK_IRC40K as RTC source clock
RCU_RTCSRC_HXTAL_DIV32	select (CK_HXTAL / 32) as RTC source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the RTC clock source selection */

rcu_rtc_clock_config(RCU_RTCSRC_IRC40K);

rcu_slcd_clock_config

The description of rcu_slcd_clock_config is shown as below:

Table 3-320. Function rcu_slcd_clock_config

Function name	rcu_slcd_clock_config
Function prototype	void rcu_slcd_clock_config(uint32_t slcd_clock_source);
Function descriptions	configure the SLCD clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
slcd_clock_source	SLCD clock source selection
RCU_SLCDSRC_NONE	no clock selected

RCU_SLCDSRC_LXTAL	select CK_LXTAL as SLCD source clock
RCU_SLCDSRC_IRC40K	select CK_IRC40K as SLCD source clock
RCU_SLCDSRC_HXTAL_DIV32	select (CK_HXTAL / 32) as SLCD source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the CK_SLCD clock source selection */

rcu_slcd_clock_config(RCU_SLCDSRC_IRC40K);

rcu_hxtal_prediv_config

The description of rcu_hxtal_prediv_config is shown as below:

Table 3-321. Function rcu_hxtal_prediv_config

Function name	rcu_hxtal_prediv_config
Function prototype	void rcu_hxtal_prediv_config(uint32_t hxtal_prediv);
Function descriptions	configure the HXTAL divider used as input of PLL
Precondition	-
The called functions	-
Input parameter{in}	
hxtal_prediv	HXTAL divider used as input of PLL
RCU_PLL_PREDVx	HXTAL divided x used as input of PLL (x=1..16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the HXTAL divider */

rcu_hxtal_prediv_config(RCU_PLL_PREDV2);

rcu_lxtal_drive_capability_config

The description of rcu_lxtal_drive_capability_config is shown as below:

Table 3-322. Function rcu_lxtal_drive_capability_config

Function name	rcu_lxtal_drive_capability_config
Function prototype	void rcu_lxtal_drive_capability_config(uint32_t lxtal_dricap);

Function descriptions	configure the LXTAL drive capability
Precondition	-
The called functions	-
Input parameter{in}	
lxtal_dricap	drive capability of LXTAL
RCU_LXTAL_LOWDRI	lower driving capability
RCU_LXTAL_MED_LOWDRI	medium low driving capability
RCU_LXTAL_MED_HIGHDRI	medium high driving capability
RCU_LXTAL_HIGHDRI	higher driving capability
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the LXTAL drive capability */

rcu_lxtal_drive_capability_config (RCU_LXTAL_LOWDRI);

rcu_flag_get

The description of rcu_flag_get is shown as below:

Table 3-323. Function rcu_flag_get

Function name	rcu_flag_get
Function prototype	FlagStatus rcu_flag_get(rcu_flag_enum flag);
Function descriptions	get the clock stabilization and periphral reset flags
Precondition	-
The called functions	-
Input parameter{in}	
flag	the clock stabilization and periphral reset flags, refer to ***Table 3-290. Enum rcu_flag_enum***.
RCU_FLAG_IRC40KSTB	IRC40K stabilization flag
RCU_FLAG_LXTALSTB	LXTAL stabilization flag
RCU_FLAG_IRC8MSTB	IRC8M stabilization flag
RCU_FLAG_HXTALSTB	HXTAL stabilization flag
RCU_FLAG_PLLSTB	PLL stabilization flag
RCU_FLAG_IRC14MS	IRC14M stabilization flag

TB	
RCU_FLAG_V12RST	1.2V domain power reset flag
RCU_FLAG_OBLRST	option byte loader reset flag
RCU_FLAG_EPRST	external PIN reset flag
RCU_FLAG_PORRST	power reset flag
RCU_FLAG_SWRST	software reset flag
RCU_FLAG_FWDGTRST	free watchdog timer reset flag
RCU_FLAG_WWDGTRST	window watchdog timer reset flag
RCU_FLAG_LPRST	low-power reset flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the clock stabilization flag */
if(RESET != rcu_flag_get(RCU_FLAG_LXTALSTB)){
}
```

rcu_all_reset_flag_clear

The description of rcu_all_reset_flag_clear is shown as below:

Table 3-324. Function rcu_all_reset_flag_clear

Function name	rcu_all_reset_flag_clear
Function prototype	void rcu_all_reset_flag_clear(void);
Function descriptions	clear all the reset flag
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear all the reset flag */
rcu_all_reset_flag_clear();
```

rcu_interrupt_flag_get

The description of rcu_interrupt_flag_get is shown as below:

Table 3-325. Function rcu_interrupt_flag_get

Function name	rcu_interrupt_flag_get
Function prototype	FlagStatus rcu_interrupt_flag_get(rcu_int_flag_enum int_flag);
Function descriptions	get the clock stabilization interrupt and ckm flags
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt and ckm flags, refer to ***Table 3-291. Enum rcu_int_flag_enum***.
RCU_INT_FLAG_IRC40KSTB	IRC40K stabilization interrupt flag
RCU_INT_FLAG_LXTALSTB	LXTAL stabilization interrupt flag
RCU_INT_FLAG_IRC8MSTB	IRC8M stabilization interrupt flag
RCU_INT_FLAG_HXTALSTB	HXTAL stabilization interrupt flag
RCU_INT_FLAG_PLLSTB	PLL stabilization interrupt flag
RCU_INT_FLAG_IRC14MSTB	IRC14M stabilization interrupt flag
RCU_INT_FLAG_CKM	HXTAL clock stuck interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the clock stabilization interrupt flag */
if(SET == rcu_interrupt_flag_get(RCU_INT_FLAG_HXTALSTB)){
}
```

rcu_interrupt_flag_clear

The description of rcu_interrupt_flag_clear is shown as below:

Table 3-326. Function rcu_interrupt_flag_clear

Function name	rcu_interrupt_flag_clear
Function prototype	void rcu_interrupt_flag_clear(rcu_int_flag_clear_enum int_flag_clear)
Function descriptions	clear the interrupt flags

Precondition	-
The called functions	-
Input parameter{in}	
int_flag_clear	clock stabilization and stuck interrupt flags clear, refer to ***Table 3-292. Enum rcu_int_flag_clear_enum***.
RCU_INT_FLAG_IRC40KSTB_CLR	IRC40K stabilization interrupt flag clear
RCU_INT_FLAG_LXTALSTB_CLR	LXTAL stabilization interrupt flag clear
RCU_INT_FLAG_IRC8MSTB_CLR	IRC8M stabilization interrupt flag clear
RCU_INT_FLAG_HXTALSTB_CLR	HXTAL stabilization interrupt flag clear
RCU_INT_FLAG_PLLSTB_CLR	PLL stabilization interrupt flag clear
RCU_INT_FLAG_IRC14MSTB_CLR	IRC14M stabilization interrupt flag clear
RCU_INT_FLAG_CKM_CLR	clock stuck interrupt flag clear
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the interrupt HXTAL stabilization interrupt flag */

rcu_interrupt_flag_clear(RCU_INT_FLAG_HXTALSTB_CLR);
```

rcu_interrupt_enable

The description of rcu_interrupt_enable is shown as below:

Table 3-327. Function rcu_interrupt_enable

Function name	rcu_interrupt_enable
Function prototype	void rcu_interrupt_enable(rcu_int_enum stab_int);
Function descriptions	enable the stabilization interrupt
Precondition	-
The called functions	-
Input parameter{in}	
stab_int	clock stabilization interrupt, refer to ***Table 3-293. Enum rcu_int_enum***.
RCU_INT_IRC40KSTB	IRC40K stabilization interrupt enable
RCU_INT_LXTALSTB	LXTAL stabilization interrupt enable
RCU_INT_IRC8MSTB	IRC8M stabilization interrupt enable

RCU_INT_HXTALSTB	HXTAL stabilization interrupt enable
RCU_INT_PLLSTB	PLL stabilization interrupt enable
RCU_INT_IRC14MSTB	IRC14M stabilization interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the HXTAL stabilization interrupt */

rcu_interrupt_enable(RCU_INT_HXTALSTB);

rcu_interrupt_disable

The description of rcu_interrupt_disable is shown as below:

Table 3-328. Function rcu_interrupt_disable

Function name	rcu_interrupt_disable
Function prototype	void rcu_interrupt_disable(rcu_int_enum stab_int);
Function descriptions	disable the stabilization interrupt
Precondition	-
The called functions	-
Input parameter{in}	
stab_int	clock stabilization interrupt, refer to ***Table 3-293. Enum rcu_int_enum***.
RCU_INT_IRC40KSTB	IRC40K stabilization interrupt disable
RCU_INT_LXTALSTB	LXTAL stabilization interrupt disable
RCU_INT_IRC8MSTB	IRC8M stabilization interrupt disable
RCU_INT_HXTALSTB	HXTAL stabilization interrupt disable
RCU_INT_PLLSTB	PLL stabilization interrupt disable
RCU_INT_IRC14MSTB	IRC14M stabilization interrupt disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the HXTAL stabilization interrupt */

rcu_interrupt_disable(RCU_INT_HXTALSTB);

rcu_osci_stab_wait

The description of rcu_osci_stab_wait is shown as below:

Table 3-329. Function rcu_osci_stab_wait

Function name	rcu_osci_stab_wait
Function prototype	ErrStatus rcu_osci_stab_wait(rcu_osci_type_enum osci);
Function descriptions	wait for oscillator stabilization flags is SET or oscillator startup is timeout
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to ***Table 3-295. Enum rcu_osci_type_enum***.
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_LXTAL	low speed crystal oscillator(LXTAL)
RCU_IRC8M	internal 8M RC oscillators(IRC8M)
RCU_IRC14M	internal 14M RC oscillators(IRC14M)
RCU_IRC40K	internal 40K RC oscillator(IRC40K)
RCU_PLL_CK	phase locked loop(PLL)
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* wait for oscillator stabilization flag */

if(SUCCESS == rcu_osci_stab_wait(RCU_HXTAL)){

}
```

rcu_osci_on

The description of rcu_osci_on is shown as below:

Table 3-330. Function rcu_osci_on

Function name	rcu_osci_on
Function prototype	void rcu_osci_on(rcu_osci_type_enum osci);
Function descriptions	turn on the oscillator
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to ***Table 3-295. Enum rcu_osci_type_enum***.
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_LXTAL	low speed crystal oscillator(LXTAL)
RCU_IRC8M	internal 8M RC oscillators(IRC8M)
RCU_IRC14M	internal 14M RC oscillators(IRC14M)
RCU_IRC40K	internal 40K RC oscillator(IRC40K)
RCU_PLL_CK	phase locked loop(PLL)
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* turn on the high speed crystal oscillator */

rcu_osci_on(RCU_HXTAL);

rcu_osci_off

The description of rcu_osci_off is shown as below:

Table 3-331. Function rcu_osci_off

Function name	rcu_osci_off
Function prototype	void rcu_osci_off(rcu_osci_type_enum osci);
Function descriptions	turn off the oscillator
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to ***Table 3-295. Enum rcu_osci_type_enum***.
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_LXTAL	low speed crystal oscillator(LXTAL)
RCU_IRC8M	internal 8M RC oscillators(IRC8M)
RCU_IRC14M	internal 14M RC oscillators(IRC14M)
RCU_IRC40K	internal 40K RC oscillator(IRC40K)
RCU_PLL_CK	phase locked loop(PLL)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* turn off the high speed crystal oscillator */

rcu_osci_off(RCU_HXTAL);

rcu_osci_bypass_mode_enable

The description of rcu_osci_bypass_mode_enable is shown as below:

Table 3-332. Function rcu_osci_bypass_mode_enable

Function name	rcu_osci_bypass_mode_enable
Function prototype	void rcu_osci_bypass_mode_enable(rcu_osci_type_enum osci);
Function descriptions	enable the oscillator bypass mode
Precondition	HXTALEN or LXTALEN must be reset before it

The called functions	-
Input parameter{in}	
osci	oscillator types, refer to ***Table 3-295. Enum rcu_osci_type_enum***.
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_LXTAL	low speed crystal oscillator(LXTAL)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the high speed crystal oscillator bypass mode */

rcu_osci_bypass_mode_enable(RCU_HXTAL);

rcu_osci_bypass_mode_disable

The description of rcu_osci_bypass_mode_disable is shown as below:

Table 3-333. Function rcu_osci_bypass_mode_disable

Function name	rcu_osci_bypass_mode_disable
Function prototype	void rcu_osci_bypass_mode_disable(rcu_osci_type_enum osci);
Function descriptions	disable the oscillator bypass mode
Precondition	HXTALEN or LXTALEN must be reset before it
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to ***Table 3-295. Enum rcu_osci_type_enum***.
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_LXTAL	low speed crystal oscillator(LXTAL)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the high speed crystal oscillator bypass mode */

rcu_osci_bypass_mode_disable(RCU_HXTAL);

rcu_hxtal_clock_monitor_enable

The description of rcu_hxtal_clock_monitor_enable is shown as below:

Table 3-334. Function rcu_hxtal_clock_monitor_enable

Function name	rcu_hxtal_clock_monitor_enable
Function prototype	void rcu_hxtal_clock_monitor_enable(void);

Function descriptions	enable the HXTAL clock monitor
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the HXTAL clock monitor */

rcu_hxtal_clock_monitor_enable();

rcu_hxtal_clock_monitor_disable

The description of rcu_hxtal_clock_monitor_disable is shown as below:

Table 3-335. Function rcu_hxtal_clock_monitor_disable

Function name	rcu_hxtal_clock_monitor_disable
Function prototype	void rcu_hxtal_clock_monitor_disable(void);
Function descriptions	disable the HXTAL clock monitor
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the HXTAL clock monitor */

rcu_hxtal_clock_monitor_disable();

rcu_irc8m_adjust_value_set

The description of rcu_irc8m_adjust_value_set is shown as below:

Table 3-336. Function rcu_irc8m_adjust_value_set

Function name	rcu_irc8m_adjust_value_set
Function prototype	void rcu_irc8m_adjust_value_set(uint32_t irc8m_adjval);
Function descriptions	set the IRC8M adjust value
Precondition	-

The called functions	-
Input parameter{in}	
irc8m_adjval	IRC8M adjust value, must be between 0 and 0x1F
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set the IRC8M adjust value */

rcu_irc8m_adjust_value_set(0x10);

rcu_irc14m_adjust_value_set

The description of rcu_irc14m_adjust_value_set is shown as below:

Table 3-337. Function rcu_irc14m_adjust_value_set

Function name	rcu_irc14m_adjust_value_set
Function prototype	void rcu_irc14m_adjust_value_set(uint32_t irc14m_adjval);
Function descriptions	set the IRC14M adjust value
Precondition	-
The called functions	-
Input parameter{in}	
irc14m_adjval	IRC14M adjust value, must be between 0 and 0x1F
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set the IRC14M adjust value */

rcu_irc14m_adjust_value_set(0x10);

rcu_irc28m_adjust_value_set

The description of rcu_irc28m_adjust_value_set is shown as below:

Table 3-338. Function rcu_irc28m_adjust_value_set

Function name	rcu_irc28m_adjust_value_set
Function prototype	void rcu_irc28m_adjust_value_set(uint32_t irc28m_adjval);
Function descriptions	set the IRC28M adjust value
Precondition	-
The called functions	-
Input parameter{in}	

irc28m_adjval	IRC28M adjust value, must be between 0 and 0x1F
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set the IRC28M adjust value */

rcu_irc28m_adjust_value_set(0x10);

rcu_voltage_key_unlock

The description of rcu_voltage_key_unlock is shown as below:

Table 3-339. Function rcu_voltage_key_unlock

Function name	rcu_voltage_key_unlock
Function prototype	void rcu_voltage_key_unlock (void);
Function descriptions	unlock the voltage key
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* unlock the voltage key */

rcu_voltage_key_unlock();

rcu_deepsleep_voltage_set

The description of rcu_deepsleep_voltage_set is shown as below:

Table 3-340. Function rcu_deepsleep_voltage_set

Function name	rcu_deepsleep_voltage_set
Function prototype	void rcu_deepsleep_voltage_set(uint32_t dsvol);
Function descriptions	set voltage in deep sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
dsvol	deep sleep mode voltage
*RCU_DEEPSLEEP_V_*	the core voltage is 1.2V in deep-sleep mode

1_2	
RCU_DEEPSLEEP_V_1_1	the core voltage is 1.1V in deep-sleep mode
RCU_DEEPSLEEP_V_1_0	the core voltage is 1.0V in deep-sleep mode
RCU_DEEPSLEEP_V_0_9	the core voltage is 0.9V in deep-sleep mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set the deep-sleep mode voltage */

rcu_deepsleep_voltage_set(RCU_DEEPSLEEP_V_1_0);

rcu_power_down_voltage_set

The description of rcu_power_down_voltage_set is shown as below:

Table 3-341. Function rcu_power_down_voltage_set

Function name	rcu_power_down_voltage_set
Function prototype	void rcu_power_down_voltage_set(uint32_t pdvol);
Function descriptions	set the power down voltage
Precondition	-
The called functions	-
Input parameter{in}	
pdvol	power down voltage select
RCU_PDR_V_2_6	power down voltage is 2.6V
RCU_PDR_V_1_8	power down voltage is 1.8V
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure power down voltage */

rcu_power_down_voltage_set(RCU_PDR_V_2_6);

rcu_clock_freq_get

The description of rcu_clock_freq_get is shown as below:

Table 3-342. Function rcu_clock_freq_get

Function name	rcu_clock_freq_get
Function prototype	uint32_t rcu_clock_freq_get(rcu_clock_freq_enum clock);
Function descriptions	get the system clock, bus clock and peripheral clock frequency
Precondition	-
The called functions	-
Input parameter{in}	
clock	the clock frequency which to get
CK_SYS	system clock frequency
CK_AHB	AHB clock frequency
CK_APB1	APB1 clock frequency
CK_APB2	APB2 clock frequency
CK_ADC	ADC clock frequency
CK_CEC	CEC clock frequency
CK_USART	USART clock frequency
Output parameter{out}	
-	-
Return value	
uint32_t	clock frequency of system, AHB, APB1, APB2, ADC or USART

Example:

```
uint32_t temp_freq;

/* get the system clock frequency */

temp_freq = rcu_clock_freq_get(CK_SYS);
```

3.17. 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 ***3.17.1***, the FWDGT firmware functions are introduced in chapter ***3.17.2***.

3.17.1. Descriptions of Peripheral registers

RTC registers are listed in the table shown as below:

Table 3-343. RTC Registers

Registers	Descriptions
RTC_TIME	RTC time of day register
RTC_DATE	RTC date register
RTC_CTL	RTC control register
RTC_STAT	RTC status register

Registers	Descriptions
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 registor
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

3.17.2. Descriptions of Peripheral functions

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

Table 3-344. RTC firmware function

Function name	Function description
rtc_deinit	reset most of the RTC registers
rtc_init	initialize RTC registers
rtc_init_mode_enter	enter RTC init mode
rtc_init_mode_exit	exit RTC init mode
rtc_register_sync_wait	wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated
rtc_current_time_get	get current time and date
rtc_subsecond_get	get current subsecond value
rtc_alarm_config	configure RTC alarm
rtc_alarm_subsecond_config	configure subsecond of RTC alarm
rtc_alarm_get	get RTC alarm
rtc_alarm_subsecond_get	get RTC alarm subsecond
rtc_alarm_enable	enable RTC alarm
rtc_alarm_disable	disable RTC alarm
rtc_timestamp_enable	enable RTC time-stamp
rtc_timestamp_disable	disable RTC time-stamp
rtc_timestamp_get	get RTC timestamp time and date
rtc_timestamp_subsecond_get	get RTC time-stamp subsecond

Function name	Function description
rtc_tamper_enable	enable RTC tamper
rtc_tamper_disable	disable RTC tamper
rtc_interrupt_enable	enable specified RTC interrupt
rtc_interrupt_disable	disble specified RTC interrupt
rtc_flag_get	check specified flag
rtc_flag_clear	clear specified flag
rtc_alter_output_config	configure RTC alternate output source
rtc_calibration_config	configure RTC calibration register
rtc_hour_adjust	ajust the daylight saving time by adding or substracting one hour from the current time
rtc_second_adjust	ajust RTC second or subsecond value of current time
rtc_bypass_shadow_enable	enable RTC bypass shadow registers function
rtc_bypass_shadow_disable	disable RTC bypass shadow registers function
rtc_refclock_detection_enable	enable RTC reference clock detection function
rtc_refclock_detection_disable	disable RTC reference clock detection function

Structure rtc_parameter_struct

Table 3-345. Structure rtc_parameter_struct

Member name	Function description
rtc_year	RTC year value: 0x0 - 0x99(BCD format)
rtc_month	RTC month value
rtc_date	RTC date value: 0x1 - 0x31(BCD format)
rtc_day_of_week	RTC weekday value
rtc_hour	RTC hour value
rtc_minute	RTC minute value: 0x0 - 0x59(BCD format)
rtc_second	RTC second value: 0x0 - 0x59(BCD format)
rtc_factor_asyn	RTC asynchronous prescaler value: 0x0 - 0x7F
rtc_factor_syn	RTC synchronous prescaler value: 0x0 - 0x7FFF
rtc_am_pm	RTC AM/PM value
rtc_display_format	RTC time notation

Structure rtc_alarm_struct

Table 3-346. Structure rtc_alarm_struct

Member name	Function description
rtc_alarm_mask	RTC alarm 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: 0x0 - 0x59(BCD format)
rtc_alarm_second	RTC alarm second value: 0x0 - 0x59(BCD format)

rtc_am_pm	RTC alarm AM/PM value

Structure rtc_timestamp_struct

Table 3-347. Structure rtc_timestamp_struct

Member name	Function description
rtc_timestamp_month	RTC time-stamp month value
rtc_timestamp_date	RTC time-stamp date value: 0x1 - 0x31(BCD format)
rtc_timestamp_day	RTC time-stamp weekday value
rtc_timestamp_hour	RTC time-stamp hour value
rtc_timestamp_minute	RTC time-stamp minute value: 0x0 - 0x59(BCD format)
rtc_timestamp_second	RTC time-stamp second value: 0x0 - 0x59(BCD format)
rtc_am_pm	RTC time-stamp AM/PM value

Structure rtc_tamper_struct

Table 3-348. Structure rtc_tamper_struct

Member name	Function description
rtc_tamper_source	RTC tamper source
rtc_tamper_trigger	RTC tamper trigger
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_enable	RTC tamper precharge feature during a voltage level detection
rtc_tamper_precharge_time	RTC tamper precharge duration if precharge feature is enabled
rtc_tamper_with_timestamp	RTC tamper time-stamp feature

rtc_deinit

The description of rtc_deinit is shown as below:

Table 3-349. Function rtc_deinit

Function name	rtc_deinit
Function prototype	ErrStatus rtc_deinit(void);
Function descriptions	reset most of the RTC registers
Precondition	-
The called functions	rcu_periph_reset_enable/ rcu_periph_reset_disable -
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
ErrStatus	ERROR or SUCCESS

Example:

/* reset most of the RTC registers*/

ErrStatus error_status = ERROR;

error_status = rtc_deinit();

rtc_init

The description of rtc_init is shown as below:

Table 3-350. Function rtc_init

Function name	rtc_init
Function prototype	ErrStatus rtc_init(rtc_parameter_struct* rtc_initpara_struct);
Function descriptions	initialize RTC registers
Precondition	-
Input parameter{in}	
rtc_initpara_struct	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure ***Table 3-345. Structure rtc_parameter_struct***.
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

/* reset most of the RTC registers*/

ErrStatus error_status = ERROR;

error_status **=** rtc_init ();

rtc_init_mode_enter

The description of rtc_init_mode_enter is shown as below:

Table 3-351. Function rtc_init_mode_enter

Function name	rtc_init_mode_enter
Function prototype	ErrStatus rtc_init_mode_enter(void);
Function descriptions	enter RTC init mode
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/*enter RTC init mode*/

ErrStatus error_status = ERROR;

error_status = rtc_init_mode_enter ();
```

rtc_init_mode_exit

The description of rtc_init_mode_exit is shown as below:

Table 3-352. Function rtc_init_mode_exit

Function name	rtc_init_mode_exit
Function prototype	void rtc_init_mode_exit(void);
Function descriptions	exit RTC init mode
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*exit RTC init mode*/

rtc_init_mode_exit ();
```

rtc_register_sync_wait

The description of rtc_register_sync_wait is shown as below:

Table 3-353. Function rtc_register_sync_wait

Function name	rtc_register_sync_wait
Function prototype	ErrStatus rtc_register_sync_wait(void);
Function descriptions	wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
ErrStatus	ERROR or SUCCESS

Example:

/*wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated*/

ErrStatus error_status = ERROR;

error_status = rtc_register_sync_wait ();

rtc_current_time_get

The description of rtc_current_time_get is shown as below:

Table 3-354. Function rtc_current_time_get

Function name	rtc_current_time_get
Function prototype	void rtc_current_time_get(rtc_parameter_struct* rtc_initpara_struct);
Function descriptions	get current time and date
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
rtc_initpara_struct	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure ***Table 3-345. Structure rtc_parameter_struct***.
Return value	
-	-

Example:

/*get current time and date*/

rtc_parameter_struct rtc_initpara_struct;

rtc_current_time_get (&rtc_initpara_struct);

rtc_subsecond_get

The description of rtc_subsecond_get is shown as below:

Table 3-355. Function rtc_subsecond_get

Function name	rtc_subsecond_get
Function prototype	uint32_t rtc_subsecond_get(void);
Function descriptions	get current subsecond value
Precondition	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
uint32_t	current subsecond value(0x00-0xFFFF)

Example:

/*get current subsecond value*/

uint32_t sub_second = 0;

sub_second = rtc_subsecond_get();

rtc_alarm_config

The description of rtc_alarm_config is shown as below:

Table 3-356. Function rtc_alarm_config

Function name	rtc_alarm_config
Function prototype	void rtc_alarm_config(rtc_alarm_struct* rtc_alarm_time)
Function descriptions	configure RTC alarm
Precondition	-
Input parameter{in}	
rtc_alarm_time	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to members of the structure ***Table 3-346. Structure rtc_alarm_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*rtc_alarm_config*/

rtc_alarm_struct rtc_alarm_time;

rtc_alarm_config(&rtc_alarm_time);

rtc_alarm_subsecond_config

The description of rtc_alarm_subsecond_config is shown as below:

Table 3-357. Function rtc_alarm_subsecond_config

Function name	rtc_alarm_subsecond_config
Function prototype	void rtc_alarm_subsecond_config(uint32_t mask_subsecond, uint32_t subsecond);
Function descriptions	configure subsecond of RTC alarm
Precondition	-

Input parameter{in}	
mask_subsecond	alarm subsecond mask
RTC_MASKSSC_0_14	mask alarm subsecond configuration
RTC_MASKSSC_1_14	mask RTC_ALRM0SS_SSC[14:1], and RTC_ALRM0SS_SSC[0] is to be compared
RTC_MASKSSC_2_14	mask RTC_ALRM0SS_SSC[14:2], and RTC_ALRM0SS_SSC[1:0] is to be compared
RTC_MASKSSC_3_14	mask RTC_ALRM0SS_SSC[14:3], and RTC_ALRM0SS_SSC[2:0] is to be compared
RTC_MASKSSC_4_14	mask RTC_ALRM0SS_SSC[14:4], and RTC_ALRM0SS_SSC[3:0] is to be compared
RTC_MASKSSC_5_14	mask RTC_ALRM0SS_SSC[14:5], and RTC_ALRM0SS_SSC[4:0] is to be compared
RTC_MASKSSC_6_14	mask RTC_ALRM0SS_SSC[14:6], and RTC_ALRM0SS_SSC[5:0] is to be compared
RTC_MASKSSC_7_14	mask RTC_ALRM0SS_SSC[14:7], and RTC_ALRM0SS_SSC[6:0] is to be compared
RTC_MASKSSC_8_14	mask RTC_ALRM0SS_SSC[14:8], and RTC_ALRM0SS_SSC[7:0] is to be compared
RTC_MASKSSC_9_14	mask RTC_ALRM0SS_SSC[14:9], and RTC_ALRM0SS_SSC[8:0] is to be compared
RTC_MASKSSC_10_14	mask RTC_ALRM0SS_SSC[14:10], and RTC_ALRM0SS_SSC[9:0] is to be compared
RTC_MASKSSC_11_14	mask RTC_ALRM0SS_SSC[14:11], and RTC_ALRM0SS_SSC[10:0] is to be compared
RTC_MASKSSC_12_14	mask RTC_ALRM0SS_SSC[14:12], and RTC_ALRM0SS_SSC[11:0] is to be compared
RTC_MASKSSC_13_14	mask RTC_ALRM0SS_SSC[14:13], and RTC_ALRM0SS_SSC[12:0] is to be compared
RTC_MASKSSC_14	mask RTC_ALRM0SS_SSC[14], and RTC_ALRM0SS_SSC[13:0] is to be compared
RTC_MASKSSC_NONE	mask none, and RTC_ALRM0SS_SSC[14:0] is to be compared
Input parameter{in}	
subsecond	alarm subsecond value(0x000 - 0x7FFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*configure subsecond of RTC alarm*/

rtc_subsecond_config (RTC_MASKSSC_9_14, 0x7FFF);
```

rtc_alarm_get

The description of rtc_alarm_get is shown as below:

Table 3-358. Function rtc_alarm_get

Function name	rtc_alarm_get
Function prototype	void rtc_alarm_get(rtc_alarm_struct* rtc_alarm_time);
Function descriptions	get RTC alarm
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
rtc_alarm_time	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to members of the structure ***Table 3-346. Structure rtc_alarm_struct***.
Return value	
-	-

Example:

rtc_alarm_struct rtc_alarm_time;

rtc_alarm_get (&rtc_alarm_time);

rtc_alarm_subsecond_get

The description of rtc_alarm_subsecond_get is shown as below:

Table 3-359. Function rtc_alarm_subsecond_get

Function name	rtc_alarm_subsecond_get
Function prototype	uint32_t rtc_alarm_subsecond_get(void);
Function descriptions	get RTC alarm subsecond
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	RTC alarm subsecond value(0x0-0x7FFF)

Example:

/*get RTC alarm subsecond*/

uint32_t subsecond = 0;

subsecond = rtc_alarm_subsecond_get();

rtc_alarm_enable

The description of rtc_alarm_enable is shown as below:

Table 3-360. Function rtc_alarm_enable

Function name	rtc_alarm_enable
Function prototype	void rtc_alarm_enable(void);
Function descriptions	enable RTC alarm
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*enable RTC alarm*/

rtc_alarm_enable();
```

rtc_alarm_disable

The description of rtc_alarm_disable is shown as below:

Table 3-361. Function rtc_alarm_disable

Function name	rtc_alarm_disable
Function prototype	ErrStatus rtc_alarm_disable(void);
Function descriptions	disable RTC alarm
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/*disable RTC alarm*/

ErrStatus error_status = ERROR;

error_status = rtc_alarm_disable();
```

rtc_timestamp_enable

The description of rtc_timestamp_enable is shown as below:

Table 3-362. Function rtc_timestamp_enable

Function name	rtc_timestamp_enable
Function prototype	void rtc_timestamp_enable(uint32_t edge);
Function descriptions	enable RTC time-stamp
Precondition	-
Input parameter{in}	
edge	specify which edge to detect of time-stamp
RTC_TIMESTAMP_RISING_EDGE	rising edge is valid event edge for timestamp event
RTC_TIMESTAMP_FALLING_EDGE	falling edge is valid event edge for timestamp event
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*enable RTC time-stamp*/

rtc_timestamp_enable (RTC_TIMESTAMP_RISING_EDGE);

rtc_timestamp_disable

The description of rtc_timestamp_disable is shown as below:

Table 3-363. Function rtc_timestamp_disable

Function name	rtc_timestamp_disable
Function prototype	void rtc_timestamp_disable(void);
Function descriptions	disable RTC time-stamp
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*disable RTC time-stamp*/

rtc_timestamp_disable ();

rtc_timestamp_get

The description of rtc_timestamp_get is shown as below:

Table 3-364. Function rtc_timestamp_get

Function name	rtc_timestamp_get
Function prototype	void rtc_timestamp_get(rtc_timestamp_struct* rtc_timestamp);
Function descriptions	get RTC timestamp time and date
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
rtc_timestamp	Pointer to a rtc_timestamp_struct structure which contains parameters for RTC time-stamp configuration，the structure members can refer to members of the structure ***Table 3-348. Structure rtc_tamper_struct***.
Return value	
-	-

Example:

/* get RTC timestamp time and date */

rtc_timestamp_struct rtc_timestamp;

rtc_timestamp_get(& rtc_timestamp);

rtc_timestamp_subsecond_get

The description of rtc_timestamp_subsecond_get is shown as below:

Table 3-365. Function rtc_timestamp_subsecond_get

Function name	rtc_timestamp_subsecond_get
Function prototype	uint32_t rtc_timestamp_subsecond_get(void);
Function descriptions	get RTC time-stamp subsecond
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	RTC time-stamp subsecond value

Example:

/* get RTC time-stamp subsecond */

uint32_t subsecond = 0;

subsecond = rtc_timestamp_subsecond_get();

rtc_tamper_enable

The description of rtc_tamper_enable is shown as below:

Table 3-366. Function rtc_tamper_enable

Function name	rtc_tamper_enable
Function prototype	void rtc_tamper_enable(rtc_tamper_struct* rtc_tamper);
Function descriptions	enable RTC tamper
Precondition	-
Input parameter{in}	
rtc_tamper	pointer to a rtc_tamper_struct structure which contains parameters for RTC tamper configuration, the structure members can refer to members of the structure ***Table 3-348. Structure rtc_tamper_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable RTC tamper */

rtc_tamper_struct rtc_tamper

rtc_tamper_enable(& rtc_tamper);

rtc_tamper_disable

The description of rtc_tamper_disable is shown as below:

Table 3-367. Function rtc_tamper_disable

Function name	rtc_tamper_disable
Function prototype	void rtc_tamper_disable(uint32_t source);
Function descriptions	disable RTC tamper
Precondition	-
Input parameter{in}	
source	specify which tamper source to be disabled
RTC_TAMPER0	RTC tamper0
RTC_TAMPER1	RTC tamper1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable RTC tamper */

rtc_tamper_disable(RTC_TAMPER0);

rtc_interrupt_enable

The description of rtc_interrupt_enable is shown as below:

Table 3-368. Function rtc_interrupt_enable

Function name	rtc_interrupt_enable
Function prototype	void rtc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable specified RTC interrupt
Precondition	-
Input parameter{in}	
interrupt	specify which interrupt source to be enabled
RTC_INT_TIMESTAMP	timestamp interrupt
RTC_INT_ALARM	alarm interrupt
RTC_INT_TAMP	tamp interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable specified RTC interrupt*/

rtc_interrupt_enable(RTC_INT_TAMP);

rtc_interrupt_disable

The description of rtc_interrupt_disable is shown as below:

Table 3-369. Function rtc_interrupt_disable

Function name	rtc_interrupt_disable
Function prototype	void rtc_interrupt_disable(uint32_t interrupt);
Function descriptions	disble specified RTC interrupt
Precondition	-
Input parameter{in}	
interrupt	specify which RTC interrupt to disable
RTC_INT_TIMESTAMP	second interrupt
RTC_INT_ALARM	alarm interrupt
RTC_INT_TAMP	tamp interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disble specified RTC interrupt */

rtc_interrupt_disable(RTC_INT_TAMP);

rtc_flag_get

The description of rtc_flag_get is shown as below:

Table 3-370. Function rtc_flag_get

Function name	rtc_flag_get
Function prototype	FlagStatus rtc_flag_get(uint32_t flag);
Function descriptions	check specified flag
Precondition	-
Input parameter{in}	
flag	specify which flag to check
RTC_FLAG_RECALIBRATION	recalibration pending flag
RTC_FLAG_TAMP1	tamper 1 event flag
RTC_FLAG_TAMP0	tamper 0 event flag
RTC_FLAG_TIMESTAMP_OVERFLOW	time-stamp overflow event flag
RTC_FLAG_TIMESTAMP	time-stamp event flag
RTC_FLAG_ALARM0	alarm event flag
RTC_FLAG_INIT	init mode event flag
RTC_FLAG_RSYN	time and date registers synchronized event flag
RTC_FLAG_YCM	year parameter configured event flag
RTC_FLAG_SHIFT	shift operation pending flag
RTC_FLAG_ALARM0_WRITTEN	alarm writen available flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* check time-stamp event flag */

FlagStatus flag = RESET;

flag = rtc_flag_get(RTC_FLAG_TIMESTAMP);

rtc_flag_clear

The description of rtc_flag_clear is shown as below:

Table 3-371. Function rtc_flag_clear

Function name	rtc_flag_clear
Function prototype	void rtc_flag_clear(uint32_t flag);
Function descriptions	clear specified flag
Precondition	-
Input parameter{in}	
flag	specify which flag to clear
RTC_FLAG_TAMP1	tamper 1 event flag
RTC_FLAG_TAMP0	tamper 0 event flag
RTC_FLAG_TIMESTAMP_OVERFLOW	time-stamp overflow event flag
RTC_FLAG_TIMESTAMP	time-stamp event flag
RTC_FLAG_ALARM0	alarm event flag
RTC_FLAG_RSYN	time and date registers synchronized event flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear time-stamp event flag */
rtc_flag_clear (RTC_FLAG_TIMESTAMP);
```

rtc_alter_output_config

The description of rtc_alter_output_config is shown as below:

Table 3-372. Function rtc_alter_output_config

Function name	rtc_alter_output_config
Function prototype	void rtc_alter_output_config(uint32_t source, uint32_t mode);
Function descriptions	configure rtc alternate output source
Precondition	-
Input parameter{in}	
source	specify signal to output
RTC_CALIBRATION_512HZ	when the LSE freqency is 32768Hz and the RTC_PSC is the default value, output 512Hz signal
RTC_CALIBRATION_1HZ	when the LSE freqency is 32768Hz and the RTC_PSC is the default value, output 1Hz signal
RTC_ALARM_HIGH	when the alarm flag is set, the output pin is high
RTC_ALARM_LOW	when the Alarm flag is set, the output pin is low
Input parameter{in}	
mode	specify the output pin (PC13) mode when output alarm signal

RTC_ALARM_OUTPUT_OD	open drain mode
RTC_ALARM_OUTPUT_PP	push pull mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure rtc alternate output source */
rtc_alter_output_config(RTC_ALARM_LOW, RTC_ALARM_OUTPUT_PP);
```

rtc_calibration_config

The description of rtc_calibration_config is shown as below:

Table 3-373. Function rtc_calibration_config

Function name	rtc_calibration_config
Function prototype	ErrStatus rtc_calibration_config(uint32_t window, uint32_t plus, uint32_t minus);
Function descriptions	configure RTC calibration register
Precondition	-
Input parameter{in}	
window	select calibration window
RTC_CALIBRATION_WINDOW_32S	2exp20 RTCCLK cycles, 32s if RTCCLK = 32768 Hz
RTC_CALIBRATION_WINDOW_16S	2exp19 RTCCLK cycles, 16s if RTCCLK = 32768 Hz
RTC_CALIBRATION_WINDOW_8S	2exp18 RTCCLK cycles, 8s if RTCCLK = 32768 Hz
Input parameter{in}	
plus	add RTC clock or not
RTC_CALIBRATION_PLUS_SET	add one RTC clock every 2048 rtc clock
RTC_CALIBRATION_PLUS_RESET	no effect
Input parameter{in}	
minus	the RTC clock to minus during the calibration window(0x0 - 0x1FF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

/* configure RTC calibration register*/

ErrStatus error_status = ERROR;

error_status = rtc_calibration_config(RTC_CALIBRATION_WINDOW_32S, RTC_CALIBRATION_PLUS_SET, 0x1FF);

rtc_hour_adjust

The description of rtc_hour_adjust is shown as below:

Table 3-374. Function rtc_hour_adjust

Function name	rtc_hour_adjust
Function prototype	void rtc_hour_adjust(uint32_t operation);
Function descriptions	adjust the daylight saving time by adding or substracting one hour from the current time
Precondition	-
Input parameter{in}	
operation	hour ajustment operation
RTC_CTL_A1H	add one hour
RTC_CTL_S1H	substract one hour
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* adjust the daylight saving time by adding one hour from the current time */

rtc_hour_adjust(RTC_CTL_A1H);

rtc_second_adjust

The description of rtc_second_adjust is shown as below:

Table 3-375. Function rtc_second_adjust

Function name	rtc_second_adjust
Function prototype	ErrStatus rtc_second_adjust(uint32_t add, uint32_t minus);
Function descriptions	adjust RTC second or subsecond value of current time
Precondition	-
Input parameter{in}	
add	add 1s to current time or not
RTC_SHIFT_ADD1S_RESET	no effect
RTC_SHIFT_ADD1S_S	add 1s to current time

ET	
Input parameter{in}	
minus	number of subsecond to minus from current time(0x0 - 0x7FFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* adjust RTC second or subsecond value of current time */

ErrStatus error_status = ERROR;

error_status = rtc_second_adjust(RTC_SHIFT_ADD1S_SET, 0);

rtc_bypass_shadow_enable

The description of rtc_bypass_shadow_enableis shown as below:

Table 3-376. Function rtc_bypass_shadow_enable

Function name	rtc_bypass_shadow_enable
Function prototype	void rtc_bypass_shadow_enable(void);
Function descriptions	enable RTC bypass shadow registers function
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable RTC bypass shadow registers function*/

rtc_bypass_shadow_enable();

rtc_bypass_shadow_disable

The description of rtc_bypass_shadow_disable shown as below:

Table 3-377. Function rtc_bypass_shadow_disable

Function name	rtc_bypass_shadow_disable
Function prototype	void rtc_bypass_shadow_disable (void);
Function descriptions	disable RTC bypass shadow registers function
Precondition	-
Input parameter{in}	

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

Example:

/* disable RTC bypass shadow registers function*/

rtc_bypass_shadow_disable ();

rtc_refclock_detection_enable

The description of rtc_refclock_detection_enable shown as below:

Table 3-378. Function rtc_refclock_detection_enable

Function name	rtc_refclock_detection_enable
Function prototype	ErrStatus rtc_refclock_detection_enable(void);
Function descriptions	enable RTC reference clock detection function
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

/* enable RTC reference clock detection function*/

ErrStatus error_status = ERROR;

error_status = rtc_refclock_detection_enable();

rtc_refclock_detection_disable

The description of rtc_refclock_detection_disableshown as below:

Table 3-379. Function rtc_refclock_detection_disable

Function name	rtc_refclock_detection_disable
Function prototype	ErrStatus rtc_refclock_detection_disable(void);
Function descriptions	disable RTC reference clock detection function
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

/* disableRTC reference clock detection function*/

ErrStatus error_status = ERROR;

error_status = rtc_refclock_detection_disable ();

3.18. SLCD

The SLCD controller directly drives LCD displays by creating the AC segment and commonvoltage signals automatically. The SLCD registers are listed in chapter ***3.18.1***, the SLCD firmware functions are introduced in chapter ***3.18.2***.

3.18.1. Descriptions of Peripheral registers

SLCD registers are listed in the table shown as below:

Table 3-380. SLCD Registers

Registers	Descriptions
SLCD_CTL	Control register
SLCD_CFG	Configuration register
SLCD_STAT	Status register
SLCD_STATC	Status flag clear register
SLCD_DATA0	Display data register 0
SLCD_DATA1	Display data register 1
SLCD_DATA2	Display data register 2
SLCD_DATA3	Display data register 3
SLCD_DATA4	Display data register 4
SLCD_DATA5	Display data register 5
SLCD_DATA6	Display data register 6
SLCD_DATA7	Display data register 7

3.18.2. Descriptions of Peripheral functions

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

Table 3-381. SLCD firmware function

Function name	Function description
slcd_deinit	SLCD reset
slcd_enable	enable SLCD interface

Function name	Function description
slcd_disable	disable SLCD interface
slcd_bias_voltage_select	SLCD bias voltage select
slcd_duty_select	SLCD duty cycle select
slcd_clock_config	config the prescaler and the divider of SLCD clock
slcd_blink_mode_config	SLCD blink mode config
slcd_contrast_ratio_config	SLCD contrast ratio config
slcd_dead_time_config	SLCD dead time duration config
slcd_pulse_on_duration_config	SLCD pulse on duration config
slcd_com_seg_remap	SLCD common/segment pad select
slcd_voltage_source_select	SLCD voltage source select
slcd_high_drive_config	enable or disable permanent high drive
slcd_data_register_write	write SLCD display data registers
slcd_data_update_request	SLCD data update request
slcd_interrupt_config	the SLCD interrupt config
slcd_flag_get	get the SLCD status flag
slcd_flag_clear	clear the SLCD flag
slcd_interrupt_flag_get	get the SLCD interrupt flag
slcd_interrupt_flag_clear	clear the SLCD interrupt flag

Enum slcd_data_register_enum

Table 3-382. slcd_data_register_enum

Member name	Function description
SLCD_DATA_REG0	SLCD display data register 0
SLCD_DATA_REG1	SLCD display data register 1
SLCD_DATA_REG2	SLCD display data register 2
SLCD_DATA_REG3	SLCD display data register 3
SLCD_DATA_REG4	SLCD display data register 4
SLCD_DATA_REG5	SLCD display data register 5
SLCD_DATA_REG6	SLCD display data register 6
SLCD_DATA_REG7	SLCD display data register 7

slcd_deinit

The description of slcd_deinit is shown as below:

Table 3-383. Function slcd_deinit

Function name	slcd_deinit
Function prototype	void slcd_deinit(void);
Function descriptions	SLCD reset
Precondition	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* reset the SLCD */

slcd_deinit ( );

slcd_enable

The description of slcd_enable is shown as below:

Table 3-384. Function slcd_enable

Function name	slcd_enable
Function prototype	void slcd_enable(void);
Function descriptions	enable SLCD interface
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable SLCD */

slcd_enable ( );

slcd_disable

The description of slcd_disable is shown as below:

Table 3-385. Function slcd_disable

Function name	slcd_disable
Function prototype	void slcd_disable(void);
Function descriptions	disable SLCD interface
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable SLCD */

slcd_disable ( );

slcd_bias_voltage_select

The description of slcd_bias_voltage_select is shown as below:

Table 3-386. Function slcd_bias_voltage_select

Function name	slcd_bias_voltage_select
Function prototype	void slcd_bias_voltage_select(uint32_t bias_voltage);
Function descriptions	SLCD bias voltage select
Precondition	-
Input parameter{in}	
bias_voltage	the SLCD voltage bias
SLCD_BIAS_1_4	1/4 voltage bias
SLCD_BIAS_1_2	1/2 voltage bias
SLCD_BIAS_1_3	1/3 voltage bias
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set the SLCD 1/4 bias voltage */

slcd_bias_voltage_select(SLCD_BIAS_1_4);

slcd_duty_select

The description of slcd_duty_select is shown as below:

Table 3-387. Function slcd_duty_select

Function name	slcd_duty_select
Function prototype	void slcd_duty_select(uint32_t duty);
Function descriptions	SLCD duty cycle select
Precondition	-
Input parameter{in}	
duty	duty select
SLCD_DUTY_STATIC	static duty
SLCD_DUTY_1_2	1/2 duty
SLCD_DUTY_1_3	1/3 duty
SLCD_DUTY_1_4	1/4 duty
SLCD_DUTY_1_6	1/6 duty

SLCD_DUTY_1_8	1/8 duty
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SLCD duty cycle select */

slcd_duty_select (SLCD_DUTY_1_2);

slcd_clock_config

The description of slcd_clock_config is shown as below:

Table 3-388. Function slcd_clock_config

Function name	slcd_clock_config
Function prototype	void slcd_clock_config(uint32_t prescaler,uint32_t divider);
Function descriptions	config the prescaler and the divider of SLCD clock
Precondition	-
Input parameter{in}	
prescaler	the prescaler factor
SLCD_PRESCALER_1	$f_{PSC} = f_{in\_clk}$
SLCD_PRESCALER_2	$f_{PSC} = f_{in\_clk}/2$
SLCD_PRESCALER_4	$f_{PSC} = f_{in\_clk}/4$
SLCD_PRESCALER_8	$f_{PSC} = f_{in\_clk}/8$
SLCD_PRESCALER_16	$f_{PSC} = f_{in\_clk}/4$
SLCD_PRESCALER_32	$f_{PSC} = f_{in\_clk}/32$
SLCD_PRESCALER_64	$f_{PSC} = f_{in\_clk}/64$
SLCD_PRESCALER_128	$f_{PSC} = f_{in\_clk}/128$
SLCD_PRESCALER_256	$f_{PSC} = f_{in\_clk}/256$
SLCD_PRESCALER_512	$f_{PSC} = f_{in\_clk}/512$
SLCD_PRESCALER_1024	$f_{PSC} = f_{in\_clk}/1024$
SLCD_PRESCALER_2048	$f_{PSC} = f_{in\_clk}/2048$
SLCD_PRESCALER_4096	$f_{PSC} = f_{in\_clk}/4096$

SLCD_PRESCALER_8192	$f_{PSC} = f_{in\_clk}/8192$
SLCD_PRESCALER_16384	$f_{PSC} = f_{in\_clk}/16384$
SLCD_PRESCALER_32768	$f_{PSC} = f_{in\_clk}/32768$
Input parameter{in}	
divider	the divider factor
SLCD_DIVIDER_16	$f_{SLCD} = f_{PSC}/16$
SLCD_DIVIDER_17	$f_{SLCD} = f_{PSC}/17$
SLCD_DIVIDER_18	$f_{SLCD} = f_{PSC}/18$
SLCD_DIVIDER_19	$f_{SLCD} = f_{PSC}/19$
SLCD_DIVIDER_20	$f_{SLCD} = f_{PSC}/20$
SLCD_DIVIDER_21	$f_{SLCD} = f_{PSC}/21$
SLCD_DIVIDER_22	$f_{SLCD} = f_{PSC}/22$
SLCD_DIVIDER_23	$f_{SLCD} = f_{PSC}/23$
SLCD_DIVIDER_24	$f_{SLCD} = f_{PSC}/24$
SLCD_DIVIDER_25	$f_{SLCD} = f_{PSC}/25$
SLCD_DIVIDER_26	$f_{SLCD} = f_{PSC}/26$
SLCD_DIVIDER_27	$f_{SLCD} = f_{PSC}/27$
SLCD_DIVIDER_28	$f_{SLCD} = f_{PSC}/28$
SLCD_DIVIDER_29	$f_{SLCD} = f_{PSC}/29$
SLCD_DIVIDER_30	$f_{SLCD} = f_{PSC}/30$
SLCD_DIVIDER_31	$f_{SLCD} = f_{PSC}/31$
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config the prescaler and the divider of SLCD clock */

slcd_clock_config(SLCD_PRESCALER_4, SLCD_DIVIDER_19);
```

slcd_blink_mode_config

The description of slcd_blink_mode_config is shown as below:

Table 3-389. Function slcd_blink_mode_config

Function name	slcd_blink_mode_config
Function prototype	void slcd_blink_mode_config(uint32_t mode,uint32_t blink_divider);
Function descriptions	SLCD blink mode config
Precondition	-
Input parameter{in}	

mode	blink mode
SLCD_BLINKMODE_OFF	blink disabled
SLCD_BLINKMODE_SEG0_COM0	blink enabled on SEG[0], COM[0]
SLCD_BLINKMODE_SEG0_ALLCOM	blink enabled on SEG[0], all COM
SLCD_BLINKMODE_ALLSEG_ALLCOM	blink enabled on all SEG and all COM
Input parameter{in}	
divider	the divider factor
SLCD_BLINK_FREQUENCY_DIV8	blink frequency = f_{SLCD}/8
SLCD_BLINK_FREQUENCY_DIV16	blink frequency = f_{SLCD}/16
SLCD_BLINK_FREQUENCY_DIV32	blink frequency = f_{SLCD}/32
SLCD_BLINK_FREQUENCY_DIV64	blink frequency = f_{SLCD}/64
SLCD_BLINK_FREQUENCY_DIV128	blink frequency = f_{SLCD}/128
SLCD_BLINK_FREQUENCY_DIV256	blink frequency = f_{SLCD}/256
SLCD_BLINK_FREQUENCY_DIV512	blink frequency = f_{SLCD}/512
SLCD_BLINK_FREQUENCY_DIV1024	blink frequency = f_{SLCD}/1024
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SLCD blink mode config */

slcd_blink_mode_config(SLCD_BLINKMODE_SEG0_COM0,
SLCD_BLINK_FREQUENCY_DIV8);
```

slcd_contrast_ratio_config

The description of slcd_contrast_ratio_config is shown as below:

Table 3-390. Function slcd_contrast_ratio_config

Function name	slcd_contrast_ratio_config

Function prototype	void slcd_contrast_ratio_config(uint32_t contrast_ratio);
Function descriptions	SLCD contrast ratio config
Precondition	-
Input parameter{in}	
contrast_ratio	specify the VSLCD voltage
SLCD_CONTRAST_LEVEL_0	maximum SLCD Voltage = 2.60V
SLCD_CONTRAST_LEVEL_1	maximum SLCD Voltage = 2.73V
SLCD_CONTRAST_LEVEL_2	maximum SLCD Voltage = 2.86V
SLCD_CONTRAST_LEVEL_3	maximum SLCD Voltage = 2.99V
SLCD_CONTRAST_LEVEL_4	maximum SLCD Voltage = 3.12V
SLCD_CONTRAST_LEVEL_5	maximum SLCD Voltage = 3.25V
SLCD_CONTRAST_LEVEL_6	maximum SLCD Voltage = 3.38V
SLCD_CONTRAST_LEVEL_7	maximum SLCD Voltage = 3.51V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SLCD contrast ratio config */

slcd_contrast_ratio_config(SLCD_CONTRAST_LEVEL_0);
```

slcd_dead_time_config

The description of slcd_dead_time_config is shown as below:

Table 3-391. Function slcd_dead_time_config

Function name	slcd_dead_time_config
Function prototype	void slcd_dead_time_config(uint32_t dead_time);
Function descriptions	SLCD dead time duration config
Precondition	-
Input parameter{in}	
dead_time	the length of the dead time between frames
SLCD_DEADTIME_PERIOD_0	no dead time

SLCD_DEADTIME_PERIOD_1	1 phase inserted between couple of frame
SLCD_DEADTIME_PERIOD_2	2 phase inserted between couple of frame
SLCD_DEADTIME_PERIOD_3	3 phase inserted between couple of frame
SLCD_DEADTIME_PERIOD_4	4 phase inserted between couple of frame
SLCD_DEADTIME_PERIOD_5	5 phase inserted between couple of frame
SLCD_DEADTIME_PERIOD_6	6 phase inserted between couple of frame
SLCD_DEADTIME_PERIOD_7	7 phase inserted between couple of frame
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* slcd_dead_time_config */

slcd_dead_time_config (SLCD_DEADTIME_PERIOD_1);

slcd_pulse_on_duration_config

The description of slcd_pulse_on_duration_config is shown as below:

Table 3-392. Function slcd_pulse_on_duration_config

Function name	slcd_pulse_on_duration_config
Function prototype	void slcd_pulse_on_duration_config(uint32_t duration);
Function descriptions	SLCD pulse on duration config
Precondition	-
Input parameter{in}	
duration	the pulse duration in terms of PSC pulses
SLCD_PULSEON_DURATION_0	pulse on duration = 0
SLCD_PULSEON_DURATION_1	pulse on duration = $1/f_{psc}$
SLCD_PULSEON_DURATION_2	pulse on duration = $2/f_{psc}$
SLCD_PULSEON_DURATION_3	pulse on duration = $3/f_{psc}$
SLCD_PULSEON_DU	pulse on duration = $4/f_{psc}$

RATION_4	
SLCD_PULSEON_DURATION_5	pulse on duration = $5/f_{psc}$
SLCD_PULSEON_DURATION_6	pulse on duration = $6/f_{psc}$
SLCD_PULSEON_DURATION_7	pulse on duration = $7/f_{psc}$
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SLCD pulse on duration config */

slcd_pulse_on_duration_config(SLCD_PULSEON_DURATION_7);

slcd_com_seg_remap

The description of slcd_com_seg_remap is shown as below:

Table 3-393. Function slcd_com_seg_remap

Function name	slcd_com_seg_remap
Function prototype	void slcd_com_seg_remap(ControlStatus newvalue);
Function descriptions	SLCD common/segment pad select
Precondition	-
Input parameter{in}	
NewValue	ENABLE or DISABLE
ENABLE	LCD_COM[7:4] pad select LCD_SEG[31:28]
DISABLE	LCD_COM[7:4] pad select LCD_COM[7:4]
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SLCD common/segment pad select */

slcd_com_seg_remap(ENABLE);

slcd_voltage_source_select

The description of slcd_voltage_source_select is shown as below:

Table 3-394. Function slcd_voltage_source_select

Function name	slcd_voltage_source_select

Function prototype	void slcd_voltage_source_select(uint8_t voltage_source);
Function descriptions	SLCD voltage source select
Precondition	-
Input parameter{in}	
voltage_source	the SLCD voltage source
SLCD_VOLTAGE_INTERNAL	internal source
SLCD_VOLTAGE_EXTERNAL	external source (VSLCD pin)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SLCD voltage source select */

slcd_voltage_source_select(SLCD_VOLTAGE_EXTERNAL);

slcd_high_drive_config

The description of slcd_high_drive_config is shown as below:

Table 3-395. Function slcd_high_drive_config

Function name	slcd_high_drive_config
Function prototype	void slcd_high_drive_config(ControlStatus newvalue);
Function descriptions	enable or disable permanent high drive
Precondition	-
Input parameter{in}	
NewValue	ENABLE or DISABLE
ENABLE	Permanent high drive enabled
DISABLE	Permanent high drive disabled
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable permanent high drive */

slcd_high_drive_config(ENABLE);

slcd_data_register_write

The description of slcd_data_register_write is shown as below:

Table 3-396. Function slcd_high_drive_config

Function name	slcd_data_register_write
Function prototype	void slcd_data_register_write(slcd_data_register_enum register_number, uint32_t data);
Function descriptions	write slcd display data registers
Precondition	-
Input parameter{in}	
register_number	refer to ***Table 3-382. slcd_data_register_enum***.
SLCD_DATA_REGx(x= 0, 1, …, 7)	SLCD_DATAx寄存器
Input parameter{in}	
data	the data write to the register
0-0xffffffff	data value
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* write slcd display data registers */

slcd_data_register_write (SLCD_DATA_REG0, 0xffff);

slcd_data_update_request

The description of slcd_data_update_request is shown as below:

Table 3-397. Function slcd_data_update_request

Function name	slcd_data_update_request
Function prototype	void slcd_data_update_request(void);
Function descriptions	SLCD data update request
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SLCD data update request */

slcd_data_update_request();

slcd_interrupt_config

The description of slcd_interrupt_config is shown as below:

Table 3-398. Function slcd_interrupt_config

Function name	slcd_interrupt_config
Function prototype	void slcd_interrupt_config(uint8_t slcd_interrupt,ControlStatus newvalue);
Function descriptions	the SLCD interrupt config
Precondition	-
Input parameter{in}	
slcd_interrupt	interrupt source
SLCD_INT_SOF	start of frame interrupt
SLCD_INT_UPD	SLCD update done interrupt
Input parameter{in}	
newvalue	ENABLE or DISABLE interrupt
ENABLE	interrupt enabled
DISABLE	interrupt disabled
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the SLCD interrupt config */

slcd_interrupt_config(SLCD_INT_SOF, ENABLE);
```

slcd_flag_get

The description of slcd_flag_get is shown as below:

Table 3-399. Function slcd_flag_get

Function name	slcd_flag_get
Function prototype	FlagStatus slcd_flag_get(uint8_t slcd_flag);
Function descriptions	get the SLCD status flag
Precondition	-
Input parameter{in}	
slcd_flag	status flag
SLCD_FLAG_ON	SLCD controller on flag
SLCD_FLAG_SOF	start of frame flag
SLCD_FLAG_UPR	SLCD data update request flag
SLCD_FLAG_UPD	update SLCD data done flag
SLCD_FLAG_VRDY	SLCD voltage ready flag
SLCD_FLAG_SYN	SLCD CFG register synchronization flag
Output parameter{out}	

-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the SLCD status flag */

slcd_flag_get(SLCD_FLAG_ON);

slcd_flag_clear

The description of slcd_flag_clear is shown as below:

Table 3-400. Function slcd_flag_clear

Function name	slcd_flag_clear
Function prototype	void slcd_flag_clear (uint8_t slcd_flag);
Function descriptions	Clear the SLCD status flag
Precondition	-
Input parameter{in}	
slcd_flag	status flag
SLCD_FLAG_SOF	start of frame flag
SLCD_FLAG_UPD	update SLCD data done flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear the SLCD status flag */

slcd_flag_clear(SLCD_FLAG_SOF);

slcd_interrupt_flag_get

The description of slcd_interrupt_flag_get is shown as below:

Table 3-401. Function slcd_interrupt_flag_get

Function name	slcd_interrupt_flag_get
Function prototype	FlagStatus slcd_interrupt_flag_get (uint8_t slcd_interrupt);
Function descriptions	get the SLCD interrupt flag
Precondition	-
Input parameter{in}	
slcd_interrupt	interrupt source
SLCD_INT_FLAG_SOF	start of frame interrupt
SLCD_INT_FLAG_UPD	SLCD update done interrupt
Output parameter{out}	

-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the SLCD interrupt flag */

slcd_interrupt_flag_get(SLCD_INT_FLAG_SOF);

slcd_interrupt_flag_clear

The description of slcd_interrupt_flag_clear is shown as below:

Table 3-402. Function slcd_interrupt_flag_clear

Function name	slcd_interrupt_flag_clear
Function prototype	FlagStatus slcd_interrupt_flag_clear (uint8_t slcd_interrupt);
Function descriptions	clear the SLCD interrupt flag
Precondition	-
Input parameter{in}	
slcd_interrupt	interrupt source
SLCD_INT_FLAG_SOF	start of frame interrupt
SLCD_INT_FLAG_UPD	SLCD update done interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear the SLCD interrupt flag */

slcd_interrupt_flag_clear(SLCD_INT_FLAG_SOF);

3.19. SPI/I2S

The SPI/I2S module can communicate with external devices using the SPI protocol or the I2S audio protocol. The SPI/I2S registers are listed in chapter ***3.19.1***, the SPI/I2S firmware functions are introduced in chapter ***3.19.2***.

3.19.1. Descriptions of Peripheral registers

SPI/I2S registers are listed in the table shown as below: :

Table 3-403. SPI/I2S registers

Registers	Descriptions
SPI_CTL0	SPI control register 0

Registers	Descriptions
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

3.19.2. Descriptions of Peripheral functions

SPI/I2S firmware functions are listed in the table shown as below:

Table 3-404. SPI/I2S firmware function

Function name	Function description
spi_i2s_deinit	reset SPI and I2S peripheral
spi_struct_para_init	initialize the parameters of SPI struct with the default values
spi_init	initialize SPI peripheral parameter
spi_enable	enable SPI
spi_disable	disable SPI
i2s_init	initialize I2S peripheral parameter
i2s_psc_config	configure I2S peripheral prescaler
i2s_enable	enable I2S
i2s_disable	disable I2S
spi_nss_output_enable	enable SPI NSS output function
spi_nss_output_disable	disable SPI NSS output function
spi_nss_internal_high	pull NSS pin high in software mode
spi_nss_internal_low	pull NSS pin low in software mode
spi_dma_enable	enable SPI DMA function
spi_dma_disable	disable SPI DMA function
spi_i2s_data_frame_format_config	configure SPI/I2S data frame format
spi_bidirectional_transfer_config	configure SPI bidirectional transfer direction
spi_i2s_data_transmit	SPI transmit data
spi_i2s_data_receive	SPI receive data
i2s_format_error_clear	clear I2S format error flag status
spi_crc_polynomial_set	set SPI CRC polynomial
spi_crc_polynomial_get	get SPI CRC polynomial
spi_crc_on	turn on SPI CRC function
spi_crc_off	turn off SPI CRC function
spi_crc_next	SPI next data is CRC value
spi_crc_get	get SPI CRC send value or receive value
spi_crc_error_clear	clear SPI CRC error flag status

Function name	Function description
spi_i2s_flag_get	get SPI and I2S flag status
spi_i2s_interrupt_enable	enable SPI and I2S interrupt
spi_i2s_interrupt_disable	disable SPI and I2S interrupt
spi_i2s_interrupt_flag_get	get SPI and I2S interrupt status

Structure spi_parameter_struct

Table 3-405. Structure spi_parameter_struct

Member name	Function description
device_mode	SPI master or slave (SPI_MASTER, SPI_SLAVE)
trans_mode	SPI transtype (SPI_TRANSMODE_FULLDUPLEX, SPI_TRANSMODE_RECEIVEONLY, SPI_TRANSMODE_BDRECEIVE, SPI_TRANSMODE_BDTRANSMIT)
frame_size	SPI frame size (SPI_FRAMESIZE_8BIT, SPI_FRAMESIZE_16BIT)
nss	SPI NSS control by hardware or software (SPI_NSS_SOFT, SPI_NSS_HARD)
endian	SPI big endian or little endian (SPI_ENDIAN_MSB, SPI_ENDIAN_LSB)
clock_polarity_phase	SPI clock phase and polarity (SPI_CK_PL_LOW_PH_1EDGE, SPI_CK_PL_HIGH_PH_1EDGE,SPI_CK_PL_LOW_PH_2EDGE, SPI_CK_PL_HIGH_PH_2EDGE)
prescale	SPI prescale factor (SPI_PSC_n (n=2,4,8,16,32,64,128,256))

spi_i2s_deinit

The description of spi_i2s_deinit is shown as below:

Table 3-406. Function spi_i2s_deinit

Function name	spi_i2s_deinit
Function prototype	void spi_i2s_deinit(uint32_t spi_periph);
Function descriptions	Reset SPIx and I2Sx peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
spi_periph	SPI/I2S peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* reset SPI0 */

spi_i2s_deinit(SPI0);

spi_struct_para_init

The description of spi_struct_para_init is shown as below:

Table 3-407. Function spi_struct_para_init

Function name	spi_struct_para_init
Function prototype	void spi_ struct_para_init(spi_parameter_struct* spi_struct);
Function descriptions	Initialize the parameters of SPI struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
spi_struct	SPI init parameter struct, the structure members can refer to ***Table 3-405. Structure spi_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize the parameters of SPI */

spi_parameter_struct spi_init_struct;

spi_struct_para_init(&spi_init_struct);

spi_init

The description of spi_init is shown as below:

Table 3-408. Function spi_init

Function name	spi_init
Function prototype	void spi_init(uint32_t spi_periph, spi_parameter_struct* spi_struct);
Function descriptions	Initialize SPIx peripheral
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	

spi_struct	SPI parameter initialization stuct, the structure members can refer to members of the structure ***Table 3-405. Structure spi_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize SPI0 */
spi_parameter_struct spi_init_struct;
spi_init_struct.trans_mode                  = SPI_TRANSMODE_BDTRANSMIT;
spi_init_struct.device_mode                 = SPI_MASTER;
spi_init_struct.frame_size                  = SPI_FRAMESIZE_8BIT;
spi_init_struct.clock_polarity_phase = SPI_CK_PL_HIGH_PH_2EDGE;
spi_init_struct.nss                         = SPI_NSS_SOFT;
spi_init_struct.prescale                    = SPI_PSC_8;
spi_init_struct.endian                      = SPI_ENDIAN_MSB;
spi_init(SPI0, &spi_init_struct);
```

spi_enable

The description of spi_enable is shown as below:

Table 3-409. Function spi_enable

Function name	spi_enable
Function prototype	void spi_enable(uint32_t spi_periph);
Function descriptions	Enable SPIx
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 */
```

spi_enable(SPI0);

spi_disable

The description of spi_disable is shown as below:

Table 3-410. Function spi_disable

Function name	spi_disable
Function prototype	void spi_disable(uint32_t spi_periph);
Function descriptions	Disable SPIx
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable SPI0 */

spi_disable(SPI0);

i2s_init

The description of i2s_init is shown as below:

Table 3-411. Function i2s_init

Function name	i2s_init
Function prototype	void i2s_init(uint32_t spi_periph, uint32_t mode, uint32_t standard, uint32_t ckpl);
Function descriptions	Initialize I2Sx peripheral
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2Sx peripheral
SPIx	x=0,2
Input parameter{in}	
mode	I2S operation mode
I2S_MODE_SLAVETX	I2S slave transmit mode
I2S_MODE_SLAVERX	I2S slave receive mode
I2S_MODE_MASTERTX	I2S master transmit mode

I2S_MODE_MASTERRX	I2S master receive mode
Input parameter{in}	
standard	I2S standard
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
Input parameter{in}	
ckpl	I2S idle state clock polarity
I2S_CKPL_LOW	I2S clock polarity low level
I2S_CKPL_HIGH	I2S clock polarity high level
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize I2S0 */

i2s_init(SPI0, I2S_MODE_MASTERTX, I2S_STD_PHILLIPS, I2S_CKPL_LOW);

i2s_psc_config

The description of i2s_psc_config is shown as below:

Table 3-412. Function i2s_psc_config

Function name	i2s_psc_config
Function prototype	void i2s_psc_config(uint32_t spi_periph, uint32_t audiosample, uint32_t frameformat, uint32_t mckout);
Function descriptions	Configure I2Sx prescaler
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
spi_periph	I2Sx peripheral
SPIx	x=0,2
Input parameter{in}	
audiosample	I2S audio sample rate
I2S_AUDIOSAMPLE_8K	audio sample rate is 8KHz
I2S_AUDIOSAMPLE_11K	audio sample rate is 11KHz
I2S_AUDIOSAMPLE_1	audio sample rate is 16KHz

6K	
I2S_AUDIOSAMPLE_22K	audio sample rate is 22KHz
I2S_AUDIOSAMPLE_32K	audio sample rate is 32KHz
I2S_AUDIOSAMPLE_44K	audio sample rate is 44KHz
I2S_AUDIOSAMPLE_48K	audio sample rate is 48KHz
I2S_AUDIOSAMPLE_96K	audio sample rate is 96KHz
I2S_AUDIOSAMPLE_192K	audio sample rate is 192KHz
Input parameter{in}	
frameformat	I2S data length and channel length
I2S_FRAMEFORMAT_DT16B_CH16B	I2S data length is 16 bit and channel length is 16 bit
I2S_FRAMEFORMAT_DT16B_CH32B	I2S data length is 16 bit and channel length is 32 bit
I2S_FRAMEFORMAT_DT24B_CH32B	I2S data length is 24 bit and channel length is 32 bit
I2S_FRAMEFORMAT_DT32B_CH32B	I2S data length is 32 bit and channel length is 32 bit
Input parameter{in}	
mckout	I2S master clock output
I2S_MCKOUT_ENABLE	I2S master clock output enable
I2S_MCKOUT_DISABLE	I2S master clock output disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure I2S0 prescaler */

i2s_psc_config(SPI0, I2S_AUDIOSAMPLE_44K, I2S_FRAMEFORMAT_DT16B_CH16B, I2S_MCKOUT_DISABLE);

i2s_enable

The description of i2s_enable is shown as below:

Table 3-413. Function i2s_enable

Function name	i2s_enable
Function prototype	void i2s_enable(uint32_t spi_periph);
Function descriptions	Enable I2Sx
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2Sx peripheral
SPIx	x=0,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2S0 */

i2s_enable(SPI0);
```

i2s_disable

The description of i2s_disable is shown as below:

Table 3-414. Function i2s_disable

Function name	i2s_disable
Function prototype	void i2s_disable(uint32_t spi_periph);
Function descriptions	Disable I2Sx
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2Sx peripheral
SPIx	x=0,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2S0 */

i2s_disable(SPI0);
```

spi_nss_output_enable

The description of spi_nss_output_enable is shown as below:

Table 3-415. Function spi_nss_output_enable

Function name	spi_nss_output_enable
Function prototype	void spi_nss_output_enable(uint32_t spi_periph);
Function descriptions	Enable SPIx NSS output function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPIx peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable SPI0 NSS output */

spi_nss_output_enable(SPI0);

spi_nss_output_disable

The description of spi_nss_output_disable is shown as below:

Table 3-416. Function spi_nss_output_disable

Function name	spi_nss_output_disable
Function prototype	void spi_nss_output_disable(uint32_t spi_periph);
Function descriptions	Disable SPIx NSS output function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPIx peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable SPI0 NSS output */

spi_nss_output_disable(SPI0);

spi_nss_internal_high

The description of spi_nss_internal_high is shown as below:

Table 3-417. Function spi_nss_internal_high

Function name	spi_nss_internal_high
Function prototype	void spi_nss_internal_high(uint32_t spi_periph);
Function descriptions	pull NSS pin high in software mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SPI0 NSS pin is pulled high level in software mode */

spi_nss_internal_high(SPI0);

spi_nss_internal_low

The description of spi_nss_internal_low is shown as below:

Table 3-418. Function spi_nss_internal_low

Function name	spi_nss_internal_low
Function prototype	void spi_nss_internal_low(uint32_t spi_periph);
Function descriptions	pull NSS pin low in software mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SPI0 NSS pin is pulled low level in software mode */

spi_nss_internal_low(SPI0);

spi_dma_enable

The description of spi_dma_enable is shown as below:

Table 3-419. Function spi_dma_enable

Function name	spi_dma_enable
Function prototype	void spi_dma_enable(uint32_t spi_periph, uint8_t dma);
Function descriptions	Enable SPIx DMA function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
dma	SPI DMA mode
SPI_DMA_TRANSMIT	SPI transmit data use DMA
SPI_DMA_RECEIVE	SPI receive data use DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable SPI0 transmit data DMA function */

spi_dma_enable(SPI0, SPI_DMA_TRANSMIT);

spi_dma_disable

The description of spi_dma_disable is shown as below:

Table 3-420. Function spi_dma_disable

Function name	spi_dma_disable
Function prototype	void spi_dma_disable(uint32_t spi_periph, uint8_t dma);
Function descriptions	Disable SPIx DMA function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
dma	SPI DMA mode
SPI_DMA_TRANSMIT	SPI transmit data use DMA
SPI_DMA_RECEIVE	SPI receive data use DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable SPI0 transmit data DMA function */

spi_dma_disable(SPI0, SPI_DMA_TRANSMIT);

spi_i2s_data_frame_format_config

The description of spi_i2s_data_frame_format_config is shown as below:

Table 3-421. Function spi_i2s_data_frame_format_config

Function name	spi_i2s_data_frame_format_config
Function prototype	void spi_i2s_data_frame_format_config(uint32_t spi_periph, uint16_t frame_format);
Function descriptions	Configure SPIx/I2Sx data frame format
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
frame_format	SPI frame size
SPI_FRAMESIZE_8BIT	SPI frame size is 8 bits
SPI_FRAMESIZE_16BIT	SPI frame size is 16 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure SPI0/I2S0 data frame format size is 16 bits */

spi_i2s_data_frame_format_config(SPI0, SPI_FRAMESIZE_16BIT);

spi_bidirectional_transfer_config

The description of spi_bidirectional_transfer_config is shown as below:

Table 3-422. Function spi_bidirectional_transfer_config

Function name	spi_bidirectional_transfer_config
Function prototype	void spi_bidirectional_transfer_config(uint32_t spi_periph, uint32_t transfer_direction);
Function descriptions	Configure SPIx bidirectional transfer direction
Precondition	-
The called functions	-

Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
transfer_direction	SPI transfer direction
SPI_BIDIRECTIONAL_TRANSMIT	SPI work in transmit-only mode
SPI_BIDIRECTIONAL_RECEIVE	SPI work in receive-only mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SPI0 works in transmit-only mode */

spi_bidirectional_transfer_config(SPI0, SPI_BIDIRECTIONAL_TRANSMIT);

spi_i2s_data_transmit

The description of spi_i2s_data_transmit is shown as below:

Table 3-423. Function spi_i2s_data_transmit

Function name	spi_i2s_data_transmit
Function prototype	void spi_i2s_data_transmit(uint32_t spi_periph, uint16_t data);
Function descriptions	SPI transmit data
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
data	16-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SPI0 transmit data */

spi_i2s_data_transmit(SPI0, spi0_send_array[send_n]);

spi_i2s_data_receive

The description of spi_i2s_data_receive is shown as below:

Table 3-424. Function spi_i2s_data_receive

Function name	spi_i2s_data_receive
Function prototype	uint16_t spi_i2s_data_receive(uint32_t spi_periph);
Function descriptions	SPI receive data
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
uint16_t	16-bit data

Example:

/* SPI0 receive data */

spi0_receive_array[receive_n] = spi_i2s_data_receive(SPI0);

i2s_format_error_clear

The description of i2s_format_error_clear is shown as below:

Table 3-425. Function i2s_format_error_clear

Function name	i2s_format_error_clear
Function prototype	void i2s_format_error_clear(uint32_t spi_periph);
Function descriptions	clear I2S format error flag status
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear I2S0 format error flag status */

i2s_format_error_clear(SPI0);

spi_crc_polynomial_set

The description of spi_crc_polynomial_set is shown as below:

Table 3-426. Function spi_crc_polynomial_set

Function name	spi_crc_polynomial_set
Function prototype	void spi_crc_polynomial_set(uint32_t spi_periph, uint16_t crc_poly);
Function descriptions	Set SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
crc_poly	CRC polynomial value
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set SPI0 CRC polynomial */

spi_crc_polynomial_set(SPI0, CRC_VALUE);

spi_crc_polynomial_get

The description of spi_crc_polynomial_get is shown as below:

Table 3-427. Function spi_crc_polynomial_get

Function name	spi_crc_polynomial_get
Function prototype	uint16_t spi_crc_polynomial_get(uint32_t spi_periph);
Function descriptions	Get SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
uint16_t	16 bit CRC polynomial value

Example:

/* get SPI0 CRC polynomial */

uint16_t crc_val;

crc_val = spi_crc_polynomial_get(SPI0);

spi_crc_on

The description of spi_crc_on is shown as below:

Table 3-428. Function spi_crc_on

Function name	spi_crc_on
Function prototype	void spi_crc_on(uint32_t spi_periph);
Function descriptions	Turn on CRC function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* turn on SPI0 CRC function */

spi_crc_on(SPI0);

spi_crc_off

The description of spi_crc_off is shown as below:

Table 3-429. Function spi_crc_off

Function name	spi_crc_off
Function prototype	void spi_crc_off(uint32_t spi_periph);
Function descriptions	Turn off CRC function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* turn off SPI0 CRC function */

spi_crc_off(SPI0);

spi_crc_next

The description of spi_crc_next is shown as below:

Table 3-430. Function spi_crc_next

Function name	spi_crc_next
Function prototype	void spi_crc_next(uint32_t spi_periph);
Function descriptions	SPI next data is CRC value
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SPI0 next data is CRC value */

spi_crc_next(SPI0);

spi_crc_get

The description of spi_crc_get is shown as below:

Table 3-431. Function spi_crc_get

Function name	spi_crc_get
Function prototype	uint16_t spi_crc_get(uint32_t spi_periph,uint8_t crc);
Function descriptions	Get SPI CRC send value or receive value
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
crc	SPI crc value
SPI_CRC_TX	get transmit crc value
SPI_CRC_RX	get receive crc value

Output parameter{out}	
-	-
Return value	
uint16_t	16-bit CRC value

Example:

/* get SPI0 CRC send value */

uint16_t crc_val;

crc_val = spi_crc_get(SPI0,SPI_CRC_TX);

spi_crc_error_clear

The description of spi_crc_error_clear is shown as below:

Table 3-432. Function spi_crc_error_clear

Function name	spi_crc_error_clear
Function prototype	void spi_crc_error_clear(uint32_t spi_periph);
Function descriptions	Clear SPI CRC error flag status
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear SPI0 CRC error flag status */

spi_crc_error_clear(SPI0);

spi_i2s_flag_get

The description of spi_i2s_flag_get is shown as below:

Table 3-433. Function spi_i2s_flag_get

Function name	spi_i2s_flag_get
Function prototype	FlagStatus spi_i2s_flag_get(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	Get SPIx and I2Sx flag status
Precondition	-
The called functions	-
Input parameter{in}	

spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
flag	SPI/I2S flag status
SPI _FLAG_TBE	transmit buffer empty flag
SPI _FLAG_RBNE	receive buffer not empty flag
SPI_FLAG_TRANS	transmit on-going flag
SPI_FLAG_RXORERR	receive overrun error flag
SPI_FLAG_CONFERR	mode config error flag
SPI_FLAG_CRCERR	CRC error flag
SPI_FLAG_FERR	SPI format error interrupt flag
I2S_FLAG_TBE	transmit buffer empty flag
I2S_FLAG_RBNE	receive buffer not empty flag
I2S_FLAG_TRANS	transmit on-going flag
I2S_FLAG_RXORERR	overrun error flag
I2S_FLAG_TXURERR	underrun error flag
I2S_FLAG_CH	channel side flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SPI0 transmit buffer empty flag status */

while(RESET == spi_i2s_flag_get(SPI0, SPI_FLAG_TBE));

spi_i2s_data_transmit(SPI0, spi0_send_array[send_n++]);
```

spi_i2s_interrupt_enable

The description of spi_i2s_interrupt_enable is shown as below:

Table 3-434. Function spi_i2s_interrupt_enable

Function name	spi_i2s_interrupt_ enable
Function prototype	void spi_i2s_interrupt_enable(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	Enable SPIx and I2Sx interrupt
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
interrupt	SPI/I2S interrupt
SPI_I2SINT_TBE	transmit buffer empty interrupt

SPI_I2S_INT_RBNE	receive buffer not empty interrupt
SPI_I2S_INT_ERR	CRC error,configuration error,reception overrun error and transmission underrun error interrupt
Output parameter{out}	
-	-
Return value	

Example:

/* enable SPI0 transmit buffer empty interrupt */

spi_i2s_interrupt_enable(SPI0, SPI_I2S_INT_TBE);

spi_i2s_interrupt_disable

The description of spi_i2s_interrupt_disable is shown as below:

Table 3-435. Function spi_i2s_interrupt_disable

Function name	spi_i2s_interrupt_ disable
Function prototype	void spi_i2s_interrupt_disable(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	Disable SPIx and I2Sx interrupt
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
interrupt	SPI/I2S interrupt
SPI_I2SINT_TBE	transmit buffer empty interrupt
SPI_I2S_INT_RBNE	receive buffer not empty interrupt
SPI_I2S_INT_ERR	CRC error,configuration error,reception overrun error and transmission underrun error interrupt
Output parameter{out}	
-	-
Return value	

Example:

/* disable SPI0 transmit buffer empty interrupt */

spi_i2s_interrupt_disable(SPI0, SPI_I2S_INT_TBE);

spi_i2s_interrupt_flag_get

The description of spi_i2s_interrupt_flag_get is shown as below:

Table 3-436. Function spi_i2s_interrupt_flag_get

Function name	spi_i2s_interrupt_flag_get
Function prototype	FlagStatus spi_i2s_interrupt_flag_get(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	Get SPIx and I2Sx interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
interrupt	SPI/I2S interrupt
SPI_I2S_INT_FLAG_TBE	transmit buffer empty interrupt
SPI_I2S_INT_FLAG_RBNE	receive buffer not empty interrupt
SPI_I2S_INT_FLAG_RXORERR	overrun interrupt
SPI_INT_FLAG_CONFERR	config error interrupt
SPI_INT_FLAG_CRCERR	CRC error interrupt
I2S_INT_FLAG_TXURERR	underrun error interrupt
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SPI0 transmit buffer empty interrupt status */
If(RESET != spi_i2s_interrupt_flag_get(SPI0, SPI_I2S_INT_FLAG_TBE)){
    while(RESET == spi_i2s_flag_get(SPI0, SPI_FLAG_TBE));
    spi_i2s_data_transmit(SPI0, spi0_send_array[send_n++]);
}
```

3.20. SYSCFG

The SYSCFG registers are listed in chapter ***3.20.1***, the SYSCFG firmware functions are introduced in chapter ***3.20.2***.

3.20.1. Descriptions of Peripheral registers

SYSCFG registers are listed in the table shown as below:

Table 3-437. SYSCFG Registers

Registers	Descriptions
SYSCFG_CFG0	system configuration register 0
SYSCFG_EXTISS0	EXTI sources selection register 0
SYSCFG_EXTISS1	EXTI sources selection register 1
SYSCFG_EXTISS2	EXTI sources selection register 2
SYSCFG_EXTISS3	EXTI sources selection register 3
SYSCFG_CFG2	system configuration register 2
SYSCFG_CPSCTL	system I/O compensation control register

3.20.2. Descriptions of Peripheral functions

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

Table 3-438. SYSCFG firmware function

Function name	Function description
syscfg_deinit	deinit syscfg module
syscfg_dma_remap_enable	enable the DMA channels remapping
syscfg_dma_remap_disable	disable the DMA channels remapping
syscfg_high_current_enable	enable PB9 high current capability
syscfg_high_current_disable	disable PB9 high current capability
syscfg_exti_line_config	configure the GPIO pin as EXTI Line
syscfg_lock_config	connect TIMER0/14/15/16 break input to the selected parameter
syscfg_flag_get	check if the specified flag in SYSCFG_CFG2 is set or not
syscfg_flag_clear	clear the flag in SYSCFG_CFG2 by writing 1
syscfg_compensation_config	configure the I/O compensation cell
syscfg_cps_rdy_flag_get	check if the I/O compensation cell ready flag is set or not

syscfg_deinit

The description of syscfg_deinit is shown as below:

Table 3-439. Function syscfg_deinit

Function name	syscfg_deinit
Function prototype	void syscfg_deinit(void);
Function descriptions	reset the SYSCFG registers
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

/* reset SYSCFG registers */

syscfg_deinit();

syscfg_dma_remap_enable

The description of syscfg_dma_remap_enable is shown as below:

Table 3-440. Function syscfg_dma_remap_enable

Function name	syscfg_dma_remap_enable
Function prototype	void syscfg_dma_remap_enable (void);
Function descriptions	enable the DMA channels remapping
Precondition	-
The called functions	-
Input parameter{in}	
syscfg_dma_remap	specify the DMA channels to remap
SYSCFG_DMA_REMAP_TIMER16	remap TIMER16 channel0 and UP DMA requests to channel1(defaut channel0)
SYSCFG_DMA_REMAP_TIMER15	remap TIMER15 channel2 and UP DMA requests to channel3(defaut channel2)
SYSCFG_DMA_REMAP_USART0RX	remap USART0 Rx DMA request to channel4(default channel2)
SYSCFG_DMA_REMAP_USART0TX	remap USART0 Tx DMA request to channel3(default channel1)
SYSCFG_DMA_REMAP_ADC	remap ADC DMA requests from channel0 to channel1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DMA channel remap*/

syscfg_dma_remap_enable(SYSCFG_DMA_REMAP_TIMER16);

syscfg_dma_remap_disable

The description of syscfg_dma_remap_disable is shown as below:

Table 3-441. Function syscfg_dma_remap_disable

Function name	syscfg_dma_remap_disable
Function prototype	void syscfg_dma_remap_disable (void);
Function descriptions	disable the DMA channels remapping
Precondition	-
The called functions	-
Input parameter{in}	
syscfg_dma_remap	specify the DMA channels to remap
SYSCFG_DMA_REMAP_TIMER16	remap TIMER16 channel0 and UP DMA requests to channel1(defaut channel0)
SYSCFG_DMA_REMAP_TIMER15	remap TIMER15 channel2 and UP DMA requests to channel3(defaut channel2)
SYSCFG_DMA_REMAP_USART0RX	remap USART0 Rx DMA request to channel4(default channel2)
SYSCFG_DMA_REMAP_USART0TX	remap USART0 Tx DMA request to channel3(default channel1)
SYSCFG_DMA_REMAP_ADC	remap ADC DMA requests from channel0 to channel1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DMA channel remap*/

syscfg_dma_remap_disable(SYSCFG_DMA_REMAP_TIMER16);

syscfg_high_current_enable

The description of syscfg_high_current_enable is shown as below:

Table 3-442. Function syscfg_ high_current_enable

Function name	syscfg_high_current_enable
Function prototype	void syscfg_high_current_enable(void);
Function descriptions	enable PB9 high current capability
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* enable PB9 high current capability */

syscfg_high_current_enable();

syscfg_high_current_disable

The description of syscfg_high_current_disable is shown as below:

Table 3-443. Function syscfg_high_current_disable

Function name	syscfg_high_current_disable
Function prototype	void syscfg_high_current_disable(void);
Function descriptions	disable PB9 high current capability
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable PB9 high current capability */

syscfg_high_current_disable();

syscfg_exti_line_config

The description of syscfg_exti_line_config is shown as below:

Table 3-444. Function syscfg_exti_line_config

Function name	syscfg_exti_line_config
Function prototype	void syscfg_exti_line_config(uint8_t exti_port, uint8_t exti_pin);
Function descriptions	configure the GPIO pin as EXTI Line
Precondition	-
The called functions	-
Input parameter{in}	
exti_port	specify the GPIO port used in EXTI
EXTI_SOURCE_GPIOx	x=A,B,C,D,F
exti_pin	specify the EXTI line
EXTI_SOURCE_PINx	x=0..15(GPIOA, GPIOB, GPIOC, GPIOD, GPIOF)

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the GPIO pin as EXTI Line */

syscfg_exti_line_config(EXTI_SOURCE_GPIOA, EXTI_SOURCE_PIN0);

syscfg_lock_config

The description of syscfg_lock_config is shown as below:

Table 3-445. Function syscfg_lock_config

Function name	syscfg_lock_config
Function prototype	void syscfg_lock_config (uint32_t syscfg_lock);
Function descriptions	connect TIMER0/14/15/16 break input to the selected parameter
Precondition	-
The called functions	-
Input parameter{in}	
syscfg_lock	specify the parameter to be connected
SYSCFG_LOCK_LOCKUP	Cortex-M3 lockup output connected to the break input
SYSCFG_LOCK_SRAM_PARITY_ERROR	SRAM_PARITY check error connected to the break input
SYSCFG_LOCK_LVD	LVD interrupt connected to the break input
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure syscfg lock*/

syscfg_lock_config(SYSCFG_LOCK_LOCKUP);

syscfg_flag_get

The description of syscfg_flag_get is shown as below:

Table 3-446. Function syscfg_flag_get

Function name	syscfg_flag_get
Function prototype	FlagStatus syscfg_flag_get(uint32_t syscfg_flag);
Function descriptions	check if the specified flag in SYSCFG_CFG2 is set or not
Precondition	-

The called functions	-
Input parameter{in}	
syscfg_flag	specify the flag in SYSCFG_CFG2 to check
SYSCFG_SRAM_PCEF	SRAM parity check error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get syscfg flag */

FlagStatus status;

status = syscfg_flag_get(SYSCFG_SRAM_PCEF);

syscfg_flag_clear

The description of syscfg_flag_clear is shown as below:

Table 3-447. Function syscfg_flag_ clear

Function name	syscfg_flag_clear
Function prototype	void syscfg_flag_clear (uint32_t syscfg_flag);
Function descriptions	clear the flag in SYSCFG_CFG2 by writing 1
Precondition	-
The called functions	-
Input parameter{in}	
syscfg_flag	specify the flag in SYSCFG_CFG2 to check
SYSCFG_SRAM_PCEF	SRAM parity check error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear syscfg flag */

syscfg_flag_clear(SYSCFG_SRAM_PCEF);

syscfg_compensation_config

The description of syscfg_compensation_config is shown as below:

Table 3-448. Function syscfg_compensation_config

Function name	syscfg_compensation_config
Function prototype	void syscfg_compensation_config(uint32_t syscfg_compensation);
Function descriptions	configure the I/O compensation cell
Precondition	-
The called functions	-
Input parameter{in}	
syscfg_compensation	specifies the I/O compensation cell mode
SYSCFG_COMPENSATION_ENABLE	I/O compensation cell is enabled
SYSCFG_COMPENSATION_DISABLE	I/O compensation cell is disabled
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the I/O compensation cell */

syscfg_compensation_config(SYSCFG_COMPENSATION_ENABLE);
```

syscfg_cps_rdy_flag_get

The description of syscfg_cps_rdy_flag_get is shown as below:

Table 3-449. Function syscfg_cps_rdy_flag_get

Function name	syscfg_cps_rdy_flag_get
Function prototype	FlagStatus syscfg_cps_rdy_flag_get(void);
Function descriptions	check if the I/O compensation cell ready flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear syscfg flag */

FlagStatus ready_flag;

ready_flag = syscfg_cps_rdy_flag_get();
```

3.21. 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 ***3.21.1***, the TIMER firmware functions are introduced in chapter ***3.21.2.***

3.21.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

Table 3-450. 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, 13, 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
TIMERx_CNT (TIMERx, x=0, 1, 2, 5, 13, 14, 15, 16)	Counter register

Registers	Descriptions
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, 5, 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_IRMP (TIMERx, x=13)	Channel complementary protection register
TIMERx_CCHP (TIMERx, x=0, 1, 2, 14, 15, 16)	TIMER complementary channel protection register
TIMERx_DMACFG (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

3.21.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

Table 3-451. TIMERx firmware function

Function name	Function description
timer_deinit	deinit a timer
timer_struct_para_init	initialize the parameters of TIMER init parameter struct with the default values
timer_init	initialize TIMER counter
timer_enable	enable a timer
timer_disable	disable a timer
timer_auto_reload_shadow_enable	enable the auto reload shadow function

Function name	Function description
timer_auto_reload_shadow_disable	disable the auto reload shadow function
timer_update_event_enable	enable the update event
timer_update_event_disable	disable the update event
timer_counter_alignment	set TIMER counter alignment mode
timer_counter_up_direction	set TIMER counter up direction
timer_counter_down_direction	set TIMER counter down direction
timer_prescaler_config	configure TIMER prescaler
timer_repetition_value_config	configure TIMER repetition register value
timer_autoreload_value_config	configure TIMER autoreload register value
timer_counter_value_config	configure TIMER counter register value
timer_counter_read	read TIMER counter value
timer_prescaler_read	read TIMER prescaler value
timer_single_pulse_mode_config	configure TIMER single pulse mode
timer_update_source_config	configure TIMER update source
timer_ocpre_clear_source_config	configure TIMER OCPRE clear source selection
timer_interrupt_enable	enable the TIMER interrupt
timer_interrupt_disable	disable the TIMER interrupt
timer_interrupt_flag_get	get timer interrupt flag
timer_interrupt_flag_clear	clear TIMER interrupt flag
timer_flag_get	get TIMER flags
timer_flag_clear	clear TIMER flags
timer_dma_enable	enable the TIMER DMA
timer_dma_disable	disable the TIMER DMA
timer_channel_dma_request_ source_select	channel DMA request source selection
timer_dma_transfer_config	configure the TIMER DMA transfer
timer_event_software_generate	software generate events
timer_break_struct_para_init	initialize the parameters of TIMER break parameter struct with the default values
timer_break_config	configure TIMER break function
timer_break_enable	enable TIMER break function
timer_break_disable	disable TIMER break function
timer_automatic_output_enable	enable TIMER output automatic function
timer_automatic_output_disable	disable TIMER output automatic function
timer_primary_output_config	configure TIMER primary output function
timer_channel_control_shadow_ config	channel capture/compare control shadow register enable
timer_channel_control_shadow_ update_config	configure TIMER channel control shadow register update control
timer_channel_output_struct _para_init	initialize the parameters of TIMER channel output parameter struct with the default values

Function name	Function description
timer_channel_output_config	configure TIMER channel output function
timer_channel_output_mode_config	configure TIMER channel output compare mode
timer_channel_output_pulse_ value_config	configure TIMER channel output pulse value
timer_channel_output_shadow_ config	configure TIMER channel output shadow function
timer_channel_output_fast_config	configure TIMER channel output fast function
timer_channel_output_clear_config	configure TIMER channel output clear function
timer_channel_output_polarity_ config	configure TIMER channel output polarity
timer_channel_complementary_ output_polarity_config	configure TIMER channel complementary output polarity
timer_channel_output_state_config	configure TIMER channel enable state
timer_channel_complementary_ output_state_config	configure TIMER channel complementary output enable state
timer_channel_input_struct_ para_init	initialize the parameters of TIMER channel input parameter struct with the default values
timer_input_capture_config	configure TIMER input capture parameter
timer_channel_input_capture_ prescaler_config	configure TIMER channel input capture prescaler value
timer_channel_capture_value_ register_read	read TIMER channel capture compare register valu
timer_input_pwm_capture_config	configure TIMER input pwm capture function
timer_hall_mode_config	configure TIMER hall sensor mode
timer_input_trigger_source_select	select TIMER input trigger source
timer_master_output_trigger_ source_select	select TIMER master mode output trigger source
timer_slave_mode_select	select TIMER slave mode
timer_master_slave_mode_config	configure TIMER master slave mode
timer_external_trigger_config	configure TIMER external trigger input
timer_quadrature_decoder_ mode_config	configure TIMER quadrature decoder mode
timer_internal_clock_config	configure TIMER internal clock mode
timer_internal_trigger_as_external_clo ck_config	configure TIMER the internal trigger as external clock input
timer_external_trigger_as_external_cl ock_config	configure TIMER the external trigger as external clock input
timer_external_clock_mode0_config	configure TIMER the external clock mode 0
timer_external_clock_mode1_config	configure TIMER the external clock mode 1
timer_external_clock_mode1_ disable	disable TIMER the external clock mode 1
timer_channel_remap_config	configure TIMER channel remap function

Function name	Function description
timer_write_chxval_register_config	configure TIMER write CHxVAL register selection
timer_output_value_selection_ config	configure TIMER output value selection

Structure timer_parameter_struct

Table 3-452. Structure timer_parameter_struct

Member name	Function description
prescaler	prescaler value（0~65535）
alignedmode	aligned mode（TIMER_COUNTER_EDGE, TIMER_COUNTER_CENTER_DOWN, TIMER_COUNTER_CENTER_UP, TIMER_COUNTER_CENTER_BOTH）
counterdirection	counter direction（TIMER_COUNTER_UP, TIMER_COUNTER_DOWN）
period	period value（0~65535）
clockdivision	clock division value（TIMER_CKDIV_DIV1, TIMER_CKDIV_DIV2, TIMER_CKDIV_DIV4）
repetitioncounter	the counter repetition value（0~255）

Structure timer_break_parameter_struct

Table 3-453. Structure timer_break_parameter_struct

Member name	Function description
runoffstate	run mode off-state（TIMER_ROS_STATE_ENABLE, TIMER_ROS_STATE_DISABLE）
ideloffstate	idle mode off-state（TIMER_IOS_STATE_ENABLE, TIMER_IOS_STATE_DISABLE）
deadtime	dead time（0~255）
breakpolarity	break polarity（TIMER_BREAK_POLARITY_LOW, TIMER_BREAK_POLARITY_HIGH）
outputautostate	output automatic enable （TIMER_OUTAUTO_ENABLE, TIMER_OUTAUTO_DISABLE）
protectmode	complementary register protect control（TIMER_CCHP_PROT_OFF, TIMER_CCHP_PROT_0, TIMER_CCHP_PROT_1, TIMER_CCHP_PROT_2）
breakstate	break enable（TIMER_BREAK_ENABLE, TIMER_BREAK_DISABLE）

Structure timer_oc_parameter_struct

Table 3-454. Structure timer_oc_parameter_struct

Member name	Function description
outputstate	channel output state（TIMER_CCX_ENABLE, TIMER_CCX_DISABLE）
outputnstate	channel complementary output state（TIMER_CCXN_ENABLE, TIMER_CCXN_DISABLE）

Member name	Function description
ocpolarity	channel output polarity（TIMER_OC_POLARITY_HIGH, TIMER_OC_POLARITY_LOW）
ocnpolarity	channel complementary output polarity（TIMER_OCN_POLARITY_HIGH, TIMER_OCN_POLARITY_LOW）
ocidlestate	idle state of channel output（TIMER_OC_IDLE_STATE_LOW, TIMER_OC_IDLE_STATE_HIGH）
ocnidlestate	idle state of channel complementary output（TIMER_OCN_IDLE_STATE_LOW, TIMER_OCN_IDLE_STATE_HIGH）

Structure timer_ic_parameter_struct

Table 3-455. Structure timer_ic_parameter_struct

Member name	Function description
icpolarity	channel input polarity（TIMER_IC_POLARITY_RISING, TIMER_IC_POLARITY_FALLING, TIMER_IC_POLARITY_BOTH_EDGE）
icselection	channel input mode selection（TIMER_IC_SELECTION_DIRECTTI, TIMER_IC_SELECTION_INDIRECTTI, TIMER_IC_SELECTION_ITS）
icprescaler	channel input capture prescaler（TIMER_IC_PSC_DIV1, TIMER_IC_PSC_DIV2, TIMER_IC_PSC_DIV4, TIMER_IC_PSC_DIV8）
icfilter	channel input capture filter control（0~15）

timer_deinit

The description of timer_deinit is shown as below:

Table 3-456. Function timer_deinit

Function name	timer_deinit
Function prototype	void timer_deinit(uint32_t timer_periph);
Function descriptions	deinit a TIMER
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset TIMER0 */

timer_deinit (TIMER0);
```

timer_struct_para_init

The description of timer_struct_para_init is shown as below:

Table 3-457. Function timer_struct_para_init

Function name	timer_struct_para_init
Function prototype	void timer_struct_para_init(timer_parameter_struct* initpara);
Function descriptions	initialize the parameters of TIMER init parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to***Table 3-452. Structure timer_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER init parameter struct with a default value */

timer_parameter_struct timer_initpara;

timer_struct_para_init(timer_initpara);
```

timer_init

The description of timer_init is shown as below:

Table 3-458. Function timer_init

Function name	timer_init
Function prototype	void timer_init(uint32_t timer_periph, timer_parameter_struct* initpara);
Function descriptions	initialize TIMER counter
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to***Table 3-452. Structure timer_parameter_struct***.
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* initialize TIMER0 */
timer_parameter_struct timer_initpara;
timer_initpara.prescaler            = 107;
timer_initpara.alignedmode          = TIMER_COUNTER_EDGE;
timer_initpara.counterdirection     = TIMER_COUNTER_UP;
timer_initpara.period               = 999;
timer_initpara.clockdivision        = TIMER_CKDIV_DIV1;
timer_initpara.repetitioncounter = 1;
timer_init(TIMER0,&timer_initpara);
```

timer_enable

The description of timer_enable is shown as below:

Table 3-459. Function timer_enable

Function name	timer_enable
Function prototype	void timer_enable(uint32_t timer_periph);
Function descriptions	enable a timer
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 */
timer_enable (TIMER0);
```

timer_disable

The description of timer_disable is shown as below:

Table 3-460. Function timer_disable

Function name	timer_disable
Function prototype	void timer_disable(uint32_t timer_periph);
Function descriptions	disable a timer
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable TIMER0 */

timer_disable (TIMER0);

timer_auto_reload_shadow_enable

The description of timer_auto_reload_shadow_enable is shown as below:

Table 3-461. Function timer_auto_reload_shadow_enable

Function name	timer_auto_reload_shadow_enable
Function prototype	void timer_auto_reload_shadow_enable(uint32_t timer_periph);
Function descriptions	enable the auto reload shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the TIMER0 auto reload shadow function */

timer_auto_reload_shadow_enable (TIMER0);

timer_auto_reload_shadow_disable

The description of timer_auto_reload_shadow_disable is shown as below:

Table 3-462. Function timer_auto_reload_shadow_disable

Function name	timer_auto_reload_shadow_ disable
Function prototype	void timer_auto_reload_shadow_ disable (uint32_t timer_periph);
Function descriptions	disable the auto reload shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the TIMER0 auto reload shadow function */

timer_auto_reload_shadow_disable (TIMER0);

timer_update_event_enable

The description of timer_update_event_enable is shown as below:

Table 3-463. Function timer_update_event_enable

Function name	timer_update_event_enable
Function prototype	void timer_update_event_enable(uint32_t timer_periph);
Function descriptions	enable the update event
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable TIMER0 the update event */

timer_update_event_enable (TIMER0);

timer_update_event_disable

The description of timer_update_event_disable is shown as below:

Table 3-464. Function timer_update_event_disable

Function name	timer_update_event_ disable
Function prototype	void timer_update_event_ disable (uint32_t timer_periph);
Function descriptions	disable the update event
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable TIMER0 the update event */

timer_update_event_disable (TIMER0);

timer_counter_alignment

The description of timer_counter_alignment is shown as below:

Table 3-465. Function timer_counter_alignment

Function name	timer_counter_alignment
Function prototype	void timer_counter_alignment(uint32_t timer_periph, uint16_t aligned);
Function descriptions	set TIMER counter alignment mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2)	TIMER peripheral selection
Input parameter{in}	
aligned	alignment mode
TIMER_COUNTER_EDGE	No center-aligned mode (edge-aligned mode). The direction of the counter isspecified by the DIR bit.
TIMER_COUNTER_CENTER_DOWN	Center-aligned and counting down assert mode. The counter counts under center aligned and channel is configured in output mode (CHxMS=00 in

	TIMERx_CHCTL0register). Only when the counter is counting down, compare interrupt flag of channels can be set.
TIMER_COUNTER_CENTER_UP	Center-aligned and counting up assert mode. The counter counts under center aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0register). Only when the counter is counting up, compare interrupt flag of channels can be set.
TIMER_COUNTER_CENTER_BOTH	Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Both when the counter is counting up and counting down, compare interrupt flag of channels can be set.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set TIMER0 counter center-aligned and counting up assert mode */

timer_counter_alignment (TIMER0, TIMER_COUNTER_CENTER_UP);

timer_counter_up_direction

The description of timer_counter_up_direction is shown as below:

Table 3-466. Function timer_counter_up_direction

Function name	timer_counter_up_direction
Function prototype	void timer_counter_up_direction(uint32_t timer_periph);
Function descriptions	set TIMER counter up direction
Precondition	set TIMER counter no center-aligned mode (edge-aligned mode)
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set TIMER0 counter up direction */

timer_counter_up_direction (TIMER0);

timer_counter_down_direction

The description of timer_counter_down_direction is shown as below:

Table 3-467. Function timer_counter_down_direction

Function name	timer_counter_ down _direction
Function prototype	void timer_counter_ down _direction(uint32_t timer_periph);
Function descriptions	set TIMER counter down direction
Precondition	set TIMER counter no center-aligned mode (edge-aligned mode)
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set TIMER0 counter down direction */

timer_counter_down_direction (TIMER0);

timer_prescaler_config

The description of timer_prescaler_config is shown as below:

Table 3-468. Function timer_prescaler_config

Function name	timer_prescaler_config
Function prototype	void timer_prescaler_config(uint32_t timer_periph, uint16_t prescaler, uint8_t pscreload);
Function descriptions	configure TIMER prescaler
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Input parameter{in}	
prescaler	prescaler value (0~65535)
Input parameter{in}	
pscreload	prescaler reload mode
TIMER_PSC_RELOAD_NOW	the prescaler is loaded right now
TIMER_PSC_RELOAD	the prescaler is loaded at the next update event

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

Example:

/* configure TIMER0 prescaler */

timer_prescaler_config (TIMER0, 3000, TIMER_PSC_RELOAD_NOW);

timer_repetition_value_config

The description of timer_repetition_value_config is shown as below:

Table 3-469. Function timer_repetition_value_config

Function name	timer_repetition_value_config
Function prototype	void timer_repetition_value_config(uint32_t timer_periph, uint16_t repetition);
Function descriptions	configure TIMER repetition register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,15,16)	TIMER peripheral selection
Input parameter{in}	
repetition	the counter repetition value (0~255)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 repetition register value */

timer_repetition_value_config (TIMER0, 98);

timer_autoreload_value_config

The description of timer_autoreload_value_config is shown as below:

Table 3-470. Function timer_autoreload_value_config

Function name	timer_autoreload_value_config
Function prototype	void timer_autoreload_value_config(uint32_t timer_periph, uint16_t autoreload);
Function descriptions	configure TIMER autoreload register value

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Input parameter{in}	
autoreload	the counter auto-reload value (0-65535)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER autoreload register value */

timer_autoreload_value_config (TIMER0, 3000);

timer_counter_value_config

The description of timer_counter_value_config is shown as below:

Table 3-471. Function timer_counter_value_config

Function name	timer_counter_value_config
Function prototype	void timer_counter_value_config(uint32_t timer_periph, uint16_t counter);
Function descriptions	configure TIMER counter register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Input parameter{in}	
counter	the counter value（0-65535）
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 counter register value */

timer_counter_value_config (TIMER0, 3000);

timer_counter_read

The description of timer_counter_read is shown as below:

Table 3-472. Function timer_counter_read

Function name	timer_counter_read
Function prototype	uint32_t timer_counter_read(uint32_t timer_periph);
Function descriptions	read TIMER counter value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint32_t	counter value（0~65535）

Example:

```
/* read TIMER0 counter value */
uint32_t i = 0;
i = timer_counter_read (TIMER0);
```

timer_prescaler_read

The description of timer_prescaler_read is shown as below:

Table 3-473. Function timer_prescaler_read

Function name	timer_prescaler_read
Function prototype	uint16_t timer_prescaler_read(uint32_t timer_periph);
Function descriptions	read TIMER prescaler value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint16_t	prescaler register value (0~65535)

Example:

/* read TIMER0 prescaler value */

uint16_t i = 0;

i = timer_prescaler_read (TIMER0);

timer_single_pulse_mode_config

The description of timer_single_pulse_mode_config is shown as below:

Table 3-474. Function timer_single_pulse_mode_config

Function name	timer_single_pulse_mode_config
Function prototype	void timer_single_pulse_mode_config(uint32_t timer_periph, uint8_t spmode);
Function descriptions	configure TIMER single pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 14..16)	TIMER peripheral selection
Input parameter{in}	
spmode	pulse mode
TIMER_SP_MODE_SINGLE	single pulse mode
TIMER_SP_MODE_REPETITIVE	repetitive pulse mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 single pulse mode */

timer_single_pulse_mode_config (TIMER0, TIMER_SP_MODE_SINGLE);

timer_update_source_config

The description of timer_update_source_config is shown as below:

Table 3-475. Function timer_update_source_config

Function name	timer_update_source_config
Function prototype	void timer_update_source_config(uint32_t timer_periph, uint32_t update);
Function descriptions	configure TIMER update source
Precondition	-

The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 13..16)	TIMER peripheral selection
Input parameter{in}	
update	update source
TIMER_UPDATE_SRC_GLOBAL	Any of the following events generate an update interrupt or DMA request: – The UPG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event
TIMER_UPDATE_SRC_REGULAR	Only counter overflow/underflow generates an update interrupt or DMA request.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER update only by counter overflow/underflow */

timer_update_source_config (TIMER0, TIMER_UPDATE_SRC_REGULAR);

timer_ocpre_clear_source_config

The description of timer_ocpre_clear_source_config is shown as below:

Table 3-476. Function t timer_ocpre_clear_source_config

Function name	timer_ocpre_clear_source_config
Function prototype	void timer_ocpre_clear_source_config (uint32_t timer_periph, uint8_t ocpreclear);
Function descriptions	configure TIMER OCPRE clear source selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2)	TIMER peripheral selection
Input parameter{in}	
ocpreclear	clear source
TIMER_OCPRE_CLEAR_SOURCE_CLR	OCPRE_CLR_INT is connected to the OCPRE_CLR input

TIMER_OCPRE_CLEAR_SOURCE_ETIF	OCPRE_CLR_INT is connected to ETIF
Output parameter{out}	
-	-
Return value	
-	-

例如：

/* configure TIMER0 OCPRE_CLR_INT is connected to the OCPRE_CLR input */

timer_ocpre_clear_source_config(TIMER0, TIMER_OCPRE_CLEAR_SOURCE_CLR);

timer_interrupt_enable

The description of timer_interrupt_enable is shown as below:

Table 3-477. Function timer_interrupt_enable

Function name	timer_interrupt_enable
Function prototype	void timer_interrupt_enable(uint32_t timer_periph, uint32_t interrupt);
Function descriptions	enable the TIMER interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
interrupt	timer interrupt enable source
TIMER_INT_UP	update interrupt enable, TIMERx (x=0..2, 5, 13..16)
TIMER_INT_CH0	channel 0 interrupt enable, TIMERx(x=0..2, 13..16)
TIMER_INT_CH1	channel 1 interrupt enable, TIMERx(x=0..2, 14)
TIMER_INT_CH2	channel 2 interrupt enable, TIMERx(x=0..2)
TIMER_INT_CH3	channel 3 interrupt enable , TIMERx(x=0..2)
TIMER_INT_CMT	commutation interrupt enable, TIMERx (x=0, 14..16)
TIMER_INT_TRG	trigger interrupt enable, TIMERx(x=0..2, 14)
TIMER_INT_BRK	break interrupt enable, TIMERx (x=0, 14..16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the TIMER0 update interrupt */

timer_interrupt_enable (TIMER0, TIMER_INT_UP);

timer_interrupt_disable

The description of timer_interrupt_disable is shown as below:

Table 3-478. Function timer_interrupt_disable

Function name	timer_interrupt_ disable
Function prototype	void timer_interrupt_ disable (uint32_t timer_periph, uint32_t interrupt);
Function descriptions	disable the TIMER interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
interrupt	timer interrupt disable source
TIMER_INT_UP	update interrupt disable, TIMERx (x=0..2, 5, 13..16)
TIMER_INT_CH0	channel 0 interrupt disable, TIMERx(x=0..2, 13..16)
TIMER_INT_CH1	channel 1 interrupt disable, TIMERx(x=0..2, 14)
TIMER_INT_CH2	channel 2 interrupt disable, TIMERx(x=0..2)
TIMER_INT_CH3	channel 3 interrupt disable, TIMERx(x=0..2)
TIMER_INT_CMT	commutation interrupt disable, TIMERx (x=0, 14..16)
TIMER_INT_TRG	trigger interrupt disable, TIMERx(x=0..2, 14)
TIMER_INT_BRK	break interrupt disable, TIMERx(x=0, 14..16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMER0 update interrupt */

timer_interrupt_disable (TIMER0, TIMER_INT_UP);
```

timer_interrupt_flag_get

The description of timer_interrupt_flag_get is shown as below:

Table 3-479. Function timer_interrupt_flag_get

Function name	timer_interrupt_flag_get
Function prototype	FlagStatus timer_interrupt_flag_get(uint32_t timer_periph, uint32_t interrupt);
Function descriptions	get timer interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	

timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
interrupt	the timer interrupt bits
TIMER_INT_FLAG_UP	update interrupt flag,TIMERx(x=0..2, 5, 13..16)
TIMER_INT_FLAG_CH0	channel 0 interrupt flag,TIMERx(x=0..2, 13..16)
TIMER_INT_FLAG_CH1	channel 1 interrupt flag,TIMERx(x=0..2, 14)
TIMER_INT_FLAG_CH2	channel 2 interrupt flag,TIMERx TIMERx(x=0..2)
TIMER_INT_FLAG_CH3	channel 3 interrupt flag,TIMERx TIMERx(x=0..2)
TIMER_INT_FLAG_CMT	channel commutation interrupt flag, TIMERx (x=0, 14..16)
TIMER_INT_FLAG_TRG	trigger interrupt flag, TIMERx(x=0..2, 14)
TIMER_INT_FLAG_BRK	break interrupt flag, TIMERx(x=0, 14..16)
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get TIMER0 update interrupt flag */

FlagStatus Flag_ interrupt = RESET;

Flag_ interrupt = timer_interrupt_flag_get (TIMER0, TIMER_INT_FLAG_UP);

timer_interrupt_flag_clear

The description of timer_interrupt_flag_clear is shown as below:

Table 3-480. Function timer_interrupt_flag_clear

Function name	timer_interrupt_flag_clear
Function prototype	void timer_interrupt_flag_clear(uint32_t timer_periph, uint32_t interrupt);
Function descriptions	clear TIMER interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
interrupt	the timer interrupt bits
TIMER_INT_FLAG_UP	update interrupt flag,TIMERx(x=0..2, 5, 13..16)
TIMER_INT_FLAG_CH0	channel 0 interrupt flag,TIMERx(x=0..2, 13..16)
TIMER_INT_FLAG_CH1	channel 1 interrupt flag,TIMERx(x=0..2, 14)
TIMER_INT_FLAG_CH2	channel 2 interrupt flag,TIMERx TIMERx(x=0..2)
TIMER_INT_FLAG_CH3	channel 3 interrupt flag,TIMERx TIMERx(x=0..2)

TIMER_INT_FLAG_CMT	channel commutation interrupt flag, TIMERx (x=0, 14..16)
TIMER_INT_FLAG_TRG	trigger interrupt flag, TIMERx(x=0..2, 14)
TIMER_INT_FLAG_BRK	break interrupt flag, TIMERx(x=0, 14..16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear TIMER0 update interrupt flag */

timer_interrupt_flag_clear (TIMER0, TIMER_INT_FLAG_UP);

timer_flag_get

The description of timer_flag_get is shown as below:

Table 3-481. Function timer_flag_get

Function name	timer_flag_get
Function prototype	FlagStatus timer_flag_get(uint32_t timer_periph, uint32_t flag);
Function descriptions	get TIMER flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
flag	the timer interrupt flags
TIMER_FLAG_UP	update flag,TIMERx(x=0.. 2, 5, 13..16)
TIMER_FLAG_CH0	channel 0 flag, TIMERx(x=0..2, 13..16)
TIMER_FLAG_CH1	channel 1 flag, TIMERx(x=0..2, 14)
TIMER_FLAG_CH2	channel 2 flag, TIMERx(x=0..2)
TIMER_FLAG_CH3	channel 3 flag, TIMERx(x=0..2)
TIMER_FLAG_CMT	channel commutation flag, TIMERx(x=0, 14..16)
TIMER_FLAG_TRG	trigger flag, TIMERx(x=0..2, 14)
TIMER_FLAG_BRK	break flag, TIMERx(x=0, 14..16)
TIMER_FLAG_CH0O	channel 0 overcapture flag, TIMERx(x=0..2, 3..16)
TIMER_FLAG_CH1O	channel 1 overcapture flag, TIMERx(x=0..2, 14)
TIMER_FLAG_CH2O	channel 2 overcapture flag, TIMERx(x=0..2)
TIMER_FLAG_CH3O	channel 3 overcapture flag, TIMERx(x=0..2)
Output parameter{out}	
-	-
Return value	

FlagStatus	SET or RESET

Example:

/* get TIMER0 update flags */

FlagStatus Flag_status = RESET;

Flag_status = timer_flag_get (TIMER0, TIMER_FLAG_UP);

timer_flag_clear

The description of timer_flag_clear is shown as below:

Table 3-482. Function timer_flag_clear

Function name	timer_flag_clear
Function prototype	void timer_flag_clear(uint32_t timer_periph, uint32_t flag);
Function descriptions	clear TIMER flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
flag	the timer interrupt flags
TIMER_FLAG_UP	update flag,TIMERx(x=0..2, 5, 13..16)
TIMER_FLAG_CH0	channel 0 flag, TIMERx(x=0..2, 13..16)
TIMER_FLAG_CH1	channel 1 flag, TIMERx(x=0..2, 14)
TIMER_FLAG_CH2	channel 2 flag, TIMERx(x=0..2)
TIMER_FLAG_CH3	channel 3 flag, TIMERx(x=0..2)
TIMER_FLAG_CMT	channel commutation flag, TIMERx(x=0, 14..16)
TIMER_FLAG_TRG	trigger flag, TIMERx(x=0..2, 14)
TIMER_FLAG_BRK	break flag, TIMERx(x=0, 14..16)
TIMER_FLAG_CH0O	channel 0 overcapture flag, TIMERx(x=0..2, 13..16)
TIMER_FLAG_CH1O	channel 1 overcapture flag, TIMERx(x=0..2, 14)
TIMER_FLAG_CH2O	channel 2 overcapture flag, TIMERx(x=0..2)
TIMER_FLAG_CH3O	channel 3 overcapture flag, TIMERx(x=0..2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear TIMER0 update flags */

timer_flag_clear (TIMER0, TIMER_FLAG_UP);

timer_dma_enable

The description of timer_dma_enable is shown as below:

Table 3-483. Function timer_dma_enable

Function name	timer_dma_enable
Function prototype	void timer_dma_enable(uint32_t timer_periph, uint16_t dma);
Function descriptions	enable the TIMER DMA
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
dma	timer DMA source enable
TIMER_DMA_UPD	update DMA enable, TIMERx(x=0..2, 5, 14..16)
TIMER_DMA_CH0D	channel 0 DMA enable, TIMERx(x=0..2, 14..16)
TIMER_DMA_CH1D	channel 1 DMA enable, TIMERx(x=0..2, 4)
TIMER_DMA_CH2D	channel 2 DMA enable, TIMERx(x=0..2)
TIMER_DMA_CH3D	channel 3 DMA enable, TIMERx(x=0..2)
TIMER_DMA_CMTD	commutation DMA request enable, TIMERx(x=0, 14)
TIMER_DMA_TRGD	trigger DMA enable, TIMERx(x=0..2, 14)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the TIMER0 update DMA */

timer_dma_enable (TIMER0, TIMER_DMA_UPD);

timer_dma_disable

The description of timer_dma_disable is shown as below:

Table 3-484. Function timer_dma_disable

Function name	timer_dma_disable
Function prototype	void timer_dma_disable (uint32_t timer_periph, uint16_t dma);
Function descriptions	disable the TIMER DMA
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters

Input parameter{in}	
dma	timer DMA source disable
TIMER_DMA_UPD	update DMA enable, TIMERx(x=0..2, 5, 14..16)
TIMER_DMA_CH0D	channel 0 DMA enable, TIMERx(x=0..2, 14..16)
TIMER_DMA_CH1D	channel 1 DMA enable, TIMERx(x=0..2, 14)
TIMER_DMA_CH2D	channel 2 DMA enable, TIMERx(x=0..2)
TIMER_DMA_CH3D	channel 3 DMA enable, TIMERx(x=0..2)
TIMER_DMA_CMTD	commutation DMA request enable, TIMERx(x=0, 14)
TIMER_DMA_TRGD	trigger DMA enable, TIMERx(x=0..2, 14)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the TIMER0 update DMA */

timer_dma_disable (TIMER0, TIMER_DMA_UPD);

timer_channel_dma_request_source_select

The description of timer_channel_dma_request_source_select is shown as below:

Table 3-485. Function timer_channel_dma_request_source_select

Function name	timer_channel_dma_request_source_select
Function prototype	void timer_channel_dma_request_source_select(uint32_t timer_periph, uint32_t dma_request);
Function descriptions	channel DMA request source selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 14,..16)	TIMER peripheral selection
Input parameter{in}	
dma_request	channel DMA request source selection
TIMER_DMAREQUEST_CHANNELEVENT	DMA request of channel n is sent when channel y event occurs
TIMER_DMAREQUEST_UPDATEEVENT	DMA request of channel n is sent when update event occurs
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* TIMER0 channel DMA request of channel n is sent when channel y event occurs */

timer_channel_dma_request_source_select(TIMER0, TIMER_DMAREQUEST_CHANNELEVENT);

timer_dma_transfer_config

The description of timer_dma_transfer_config is shown as below:

Table 3-486. Function timer_dma_transfer_config

Function name	timer_dma_transfer_config
Function prototype	void timer_dma_transfer_config(uint32_t timer_periph, uint32_t dma_baseaddr, uint32_t dma_lenth);
Function descriptions	configure the TIMER DMA transfer
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 14,..16)	TIMER peripheral selection
Input parameter{in}	
dma_baseaddr	DMA transfer access start address
TIMER_DMACFG_DMATA_CTL0	DMA transfer address is TIMER_CTL0, TIMERx(x=0..2, 14..16)
TIMER_DMACFG_DMATA_CTL1	DMA transfer address is TIMER_CTL1, TIMERx(x=0..2, 14..16)
TIMER_DMACFG_DMATA_SMCFG	DMA transfer address is TIMER_SMCFG, TIMERx(x=0..2, 14)
TIMER_DMACFG_DMATA_DMAINTEN	DMA transfer address is TIMER_DMAINTEN, TIMERx(x=0..2, 14..16)
TIMER_DMACFG_DMATA_INTF	DMA transfer address is TIMER_INTF, TIMERx(x=0..2, 14..16)
TIMER_DMACFG_DMATA_SWEVG	DMA transfer address is TIMER_SWEVG, TIMERx(x=0..2, 14..16)
TIMER_DMACFG_DMATA_CHCTL0	DMA transfer address is TIMER_CHCTL0, TIMERx(x=0..2, 14..16)
TIMER_DMACFG_DMATA_CHCTL1	DMA transfer address is TIMER_CHCTL1, TIMERx(x=0..2)
TIMER_DMACFG_DMATA_CHCTL2	DMA transfer address is TIMER_CHCTL2, TIMERx (x=0..2, 14..16)
TIMER_DMACFG_DMATA_CNT	DMA transfer address is TIMER_CNT, TIMERx (x=0..2, 14..16)
TIMER_DMACFG_DMATA_PSC	DMA transfer address is TIMER_PSC, TIMERx (x=0..2, 14..16)
TIMER_DMACFG_DMA	MA transfer address is TIMER_CAR, TIMERx (x=0..2, 14..16)

TA_CAR	
TIMER_DMACFG_DMATA_CREP	DMA transfer address is TIMER_CREP, TIMERx (x=0, 14..16)
TIMER_DMACFG_DMATA_CH0CV	DMA transfer address is TIMER_CH0CV, TIMERx (x=0..2, 14..16)
TIMER_DMACFG_DMATA_CH1CV	DMA transfer address is TIMER_CH1CV, TIMERx(x=0..2, 14)
TIMER_DMACFG_DMATA_CH2CV	DMA transfer address is TIMER_CH2CV, TIMERx(x=0..2)
TIMER_DMACFG_DMATA_CH3CV	DMA transfer address is TIMER_CH3CV, TIMERx(x=0..2)
TIMER_DMACFG_DMATA_CCHP	DMA transfer address is TIMER_CCHP, TIMERx (x=0, 14..16)
TIMER_DMACFG_DMATA_DMACFG	DMA transfer address is TIMER_DMACFG, TIMERx (x=0..2, 14..16)
Input parameter{in}	
dma_lenth	DMA transfer count
TIMER_DMACFG_DMATC_xTRANSFER	x=1..18, DMA transfer x time
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the TIMER0 DMA transfer */

timer_dma_transfer_config(TIMER0, TIMER_DMACFG_DMATA_CTL0, TIMER_DMACFG_DMATC_5TRANSFER);

timer_event_software_generate

The description of timer_event_software_generate is shown as below:

Table 3-487. Function timer_event_software_generate

Function name	timer_event_software_generate
Function prototype	void timer_event_software_generate(uint32_t timer_periph, uint16_t event);
Function descriptions	software generate events
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters

Input parameter{in}	
event	the timer software event generation sources
TIMER_EVENT_SRC_UPG	update event, TIMERx(x=0..2, 5, 13..16)
TIMER_EVENT_SRC_CH0G	channel 0 capture or compare event generation, TIMERx(x=0..2, 13..16)
TIMER_EVENT_SRC_CH1G	channel 1 capture or compare event generation, TIMERx(x=0..2, 14)
TIMER_EVENT_SRC_CH2G	channel 2 capture or compare event generation, TIMERx(x=0..2)
TIMER_EVENT_SRC_CH3G	channel 3 capture or compare event generation, TIMERx(x=0..2)
TIMER_EVENT_SRC_CMTG	channel commutation event generation, TIMERx(x=0, 14..16)
TIMER_EVENT_SRC_TRGG	trigger event generation, TIMERx(x=0..2, 14)
TIMER_EVENT_SRC_BRKG	break event generation, TIMERx(x=0, 14..16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* software generate update event*/

timer_event_software_generate (TIMER0, TIMER_EVENT_SRC_UPG);

timer_break_struct_para_init

The description of timer_break_struct_para_init is shown as below:

Table 3-488. Function timer_break_struct_para_init

Function name	timer_break_struct_para_init
Function prototype	void timer_break_struct_para_init(timer_break_parameter_struct* breakpara);
Function descriptions	initialize the parameters of TIMER break parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
breakpara	TIMER break parameter struct, the structure members can refer to ***Table 3-453. Structure timer_break_parameter_struct***.
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* initialize TIMER break parameter struct with a default value */

timer_break_parameter_struct timer_breakpara;

timer_break_struct_para_init(timer_breakpara);

timer_break_config

The description of timer_break_config is shown as below:

Table 3-489. Function timer_break_config

Function name	timer_break_config
Function prototype	void timer_break_config(uint32_t timer_periph, timer_break_parameter_struct* breakpara);
Function descriptions	configure TIMER break function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0, 14..16)	TIMER peripheral selection
Input parameter{in}	
breakpara	TIMER break parameter struct, the structure members can refer to***Table 3-453. Structure timer_break_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 break function */

timer_break_parameter_struct timer_breakpara;

timer_breakpara.runoffstate          = TIMER_ROS_STATE_DISABLE;

timer_breakpara.ideloffstate         = TIMER_IOS_STATE_DISABLE ;

timer_breakpara.deadtime             = 255;

timer_breakpara.breakpolarity        = TIMER_BREAK_POLARITY_LOW;

timer_breakpara.outputautostate      = TIMER_OUTAUTO_ENABLE;

timer_breakpara.protectmode          = TIMER_CCHP_PROT_0;
timer_breakpara.breakstate            = TIMER_BREAK_ENABLE;

timer_break_config(TIMER0, &timer_breakpara);
```

timer_break_enable

The description of timer_break_enable is shown as below:

Table 3-490. Function timer_break_enable

Function name	timer_break_enable
Function prototype	void timer_break_enable(uint32_t timer_periph);
Function descriptions	enable TIMER break function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0, 14..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 break function*/

timer_break_enable (TIMER0);
```

timer_break_disable

The description of timer_break_disable is shown as below:

Table 3-491. Function timer_break_disable

Function name	timer_break_disable
Function prototype	void timer_break_disable(uint32_t timer_periph);
Function descriptions	disable TIMER break function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0, 14..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable TIMER0 break function*/

timer_break_ disable (TIMER0);

timer_automatic_output_enable

The description of timer_automatic_output_enable is shown as below:

Table 3-492. Function timer_automatic_output_enable

Function name	timer_automatic_output_enable
Function prototype	void timer_automatic_output_enable(uint32_t timer_periph);
Function descriptions	enable TIMER output automatic function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0, 14..16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable TIMER0 output automatic function */

timer_automatic_output_enable (TIMER0);

timer_automatic_output_disable

The description of timer_automatic_output_disable is shown as below:

Table 3-493. Function timer_automatic_output_disable

Function name	timer_automatic_output_ disable
Function prototype	void timer_automatic_output_ disable (uint32_t timer_periph);
Function descriptions	disable TIMER output automatic function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0, 14..16)	TIMER peripheral selection
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* disable TIMER0 output automatic function */

timer_automatic_output_disable (TIMER0);

timer_primary_output_config

The description of timer_primary_output_config is shown as below:

Table 3-494. Function timer_primary_output_config

Function name	timer_primary_output_config
Function prototype	void timer_primary_output_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	configure TIMER primary output function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,14..16)	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable TIMER0 primary output function */

timer_primary_output_config (TIMER0, ENABLE);

timer_channel_control_shadow_config

The description of timer_channel_control_shadow_config is shown as below:

Table 3-495. Function timer_channel_control_shadow_config

Function name	timer_channel_control_shadow_config
Function prototype	void timer_channel_control_shadow_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	channel commutation control shadow register enable
Precondition	-

The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0, 14..16)	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* channel capture/compare control shadow register enable */

timer_channel_control_shadow_config (TIMER0, ENABLE);

timer_channel_control_shadow_update_config

The description of timer_channel_control_shadow_update_config is shown as below:

Table 3-496. Function timer_channel_control_shadow_update_config

Function name	timer_channel_control_shadow_update_config
Function prototype	void timer_channel_control_shadow_update_config(uint32_t timer_periph, uint8_t ccuctl);
Function descriptions	configure commutation control shadow register update control
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0, 14..16)	TIMER peripheral selection
Input parameter{in}	
ccuctl	channel control shadow register update control
TIMER_UPDATECTL_CCU	the shadow registers update by when CMTG bit is set
TIMER_UPDATECTL_CCUTRI	the shadow registers update by when CMTG bit is set or an rising edge of TRGI occurs
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel control shadow register update when CMTG bit is set */

timer_channel_control_shadow_update_config (TIMER0, TIMER_UPDATECTL_CCU);

timer_channel_output_struct_para_init

The description of timer_channel_output_struct_para_init is shown as below:

Table 3-497. Function timer_channel_output_struct_para_init

Function name	timer_channel_output_struct_para_init
Function prototype	void timer_channel_output_struct_para_init(timer_oc_parameter_struct* ocpara);
Function descriptions	initialize the parameters of TIMER channel output parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
ocpara	TIMER channel output parameter struct, the structure members can refer to*Table 3-454. Structure timer_oc_parameter_struct*.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize TIMER channel output parameter struct with a default value */

timer_oc_parameter_struct timer_ocinitpara;

timer_channel_output_struct_para_init(timer_ocinitpara);

timer_channel_output_config

The description of timer_channel_output_config is shown as below:

Table 3-498. Function timer_channel_output_config

Function name	timer_channel_output_config
Function prototype	void timer_channel_output_config(uint32_t timer_periph, uint16_t channel, timer_oc_parameter_struct* ocpara);
Function descriptions	configure TIMER channel output function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	

channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..2, 14))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..2))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..2))
Input parameter{in}	
ocpara	TIMER channel output parameter struct, the structure members can refer to*Table 3-454. Structure timer_oc_parameter_struct*.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output function */
timer_oc_parameter_struct timer_ocinitpara;
timer_ocinitpara.outputstate   = TIMER_CCX_ENABLE;
timer_ocinitpara.outputnstate = TIMER_CCXN_ENABLE;
timer_ocinitpara.ocpolarity    = TIMER_OC_POLARITY_HIGH;
timer_ocinitpara.ocnpolarity   = TIMER_OCN_POLARITY_HIGH;
timer_ocinitpara.ocidlestate   = TIMER_OC_IDLE_STATE_HIGH;
timer_ocinitpara.ocnidlestate = TIMER_OCN_IDLE_STATE_LOW;
timer_channel_output_config(TIMER0, TIMER_CH_0, &timer_ocinitpara);
```

timer_channel_output_mode_config

The description of timer_channel_output_mode_config is shown as below:

Table 3-499. Function timer_channel_output_mode_config

Function name	timer_channel_output_mode_config
Function prototype	void timer_channel_output_mode_config(uint32_t timer_periph, uint16_t channel, uint16_t ocmode);
Function descriptions	configure TIMER channel output compare mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured

TIMER_CH_0	TIMER channel 0 (TIMERx (x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1 (TIMERx (x=0..2, 14))
TIMER_CH_2	TIMER channel 2 (TIMERx (x=0..2))
TIMER_CH_3	IMER channel 3 (TIMERx (x=0..2))
Input parameter{in}	
ocmode	channel output compare mode
TIMER_OC_MODE_TIMING	timing mode
TIMER_OC_MODE_ACTIVE	set the channel output
TIMER_OC_MODE_INACTIVE	clear the channel output
TIMER_OC_MODE_TOGGLE	toggle on match
TIMER_OC_MODE_LOW	force low mode
TIMER_OC_MODE_HIGH	force high mode
TIMER_OC_MODE_PWM0	PWM mode 0
TIMER_OC_MODE_PWM1	PWM mode 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel PWM 0 mode */

timer_channel_output_mode_config(TIMER0, TIMER_CH_0, TIMER_OC_MODE_PWM0);

timer_channel_output_pulse_value_config

The description of timer_channel_output_pulse_value_config is shown as below:

Table 3-500. Function timer_channel_output_pulse_value_config

Function name	timer_channel_output_pulse_value_config
Function prototype	void timer_channel_output_pulse_value_config(uint32_t timer_periph, uint16_t channel, uint32_t pulse);
Function descriptions	configure TIMER channel output pulse value
Precondition	-
The called functions	-
Input parameter{in}	

timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx (x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1 (TIMERx TIMERx(x=0..2, 14))
TIMER_CH_2	TIMER channel 2 (TIMERx (x=0..2))
TIMER_CH_3	IMER channel 3 (TIMERx (x=0..2))
Input parameter{in}	
pulse	channel output pulse value（0~65535）
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 output pulse value */

timer_channel_output_pulse_value_config(TIMER0, TIMER_CH_0, 399);

timer_channel_output_shadow_config

The description of timer_channel_output_shadow_config is shown as below:

Table 3-501. Function timer_channel_output_shadow_config

Function name	timer_channel_output_shadow_config
Function prototype	void timer_channel_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t ocshadow);
Function descriptions	configure TIMER channel output shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx (x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1 (TIMERx (x=0..2, 14))
TIMER_CH_2	TIMER channel 2 (TIMERx (x=0..2))
TIMER_CH_3	IMER channel 3 (TIMERx (x=0..2))
Input parameter{in}	
ocshadow	channel output shadow state
TIMER_OC_SHADOW_ENABLE	channel output shadow state enable

TIMER_OC_SHADOW_DISABLE	channel output shadow state disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*configure TIMER0 channel 0 output shadow function */

timer_channel_output_shadow_config (TIMER0, TIMER_CH_0, TIMER_OC_SHADOW_ENABLE);

timer_channel_output_fast_config

The description of timer_channel_output_fast_config is shown as below:

Table 3-502. Function timer_channel_output_fast_config

Function name	timer_channel_output_fast_config
Function prototype	void timer_channel_output_fast_config(uint32_t timer_periph, uint16_t channel, uint16_t ocfast);
Function descriptions	configure TIMER channel output fast function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx (x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1 (TIMERx (x=0..2, 14))
TIMER_CH_2	TIMER channel 2 (TIMERx (x=0..2))
TIMER_CH_3	TIMER channel 3 (TIMERx (x=0..2))
Input parameter{in}	
ocfast	channel output fast function
TIMER_OC_FAST_ENABLE	channel output fast function enable
TIMER_OC_FAST_DISABLE	channel output fast function disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 output fast function */

timer_channel_output_fast_config (TIMER0, TIMER_CH_0, TIMER_OC_FAST_ENABLE);

timer_channel_output_clear_config

The description of timer_channel_output_clear_config is shown as below:

Table 3-503. Function timer_channel_output_clear_config

Function name	timer_channel_output_clear_config
Function prototype	void timer_channel_output_clear_config(uint32_t timer_periph, uint16_t channel, uint16_t occlear);
Function descriptions	configure TIMER channel output clear function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER periphera
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx (x=0..2))
TIMER_CH_1	TIMER channel 1 (TIMERx (x=0..2))
TIMER_CH_2	TIMER channel 2 (TIMERx (x=0..2))
TIMER_CH_3	TIMER channel 3 (TIMERx (x=0..2))
Input parameter{in}	
occlear	channel output clear function
TIMER_OC_CLEAR_ENABLE	channel output clear function enable
TIMER_OC_CLEAR_DISABLE	channel output clear function disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 output clear function */

timer_channel_output_clear_config (TIMER0, TIMER_CH_0,
TIMER_OC_CLEAR_ENABLE);

timer_channel_output_polarity_config

The description of timer_channel_output_polarity_config is shown as below:

Table 3-504. Function timer_channel_output_polarity_config

Function name	timer_channel_output_polarity_config
Function prototype	void timer_channel_output_polarity_config(uint32_t timer_periph, uint16_t channel, uint16_t ocpolarity);
Function descriptions	configure TIMER channel output polarity
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx (x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1 (TIMERx (x=0..2, 14))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..2))
TIMER_CH_3	IMER channel 3 (TIMERx (x=0..2))
Input parameter{in}	
ocpolarity	channel output polarity
TIMER_OC_POLARITY_HIGH	channel output polarity is high
TIMER_OC_POLARITY_LOW	channel output polarity is low
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 output polarity */

timer_channel_output_polarity_config (TIMER0, TIMER_CH_0, TIMER_OC_POLARITY_HIGH);

timer_channel_complementary_output_polarity_config

The description of timer_channel_complementary_output_polarity_config is shown as below:

Table 3-505. Function timer_channel_complementary_output_polarity_config

Function name	timer_channel_complementary_output_polarity_config
Function prototype	void timer_channel_complementary_output_polarity_config(uint32_t timer_periph, uint16_t channel, uint16_t ocnpolarity);
Function descriptions	configure TIMER channel complementary output polarity
Precondition	-
The called functions	-

Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0(TIMERx (x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1(TIMERx (x=0..2, 14))
TIMER_CH_2	TIMER channel 2(TIMERx (x=0..2))
TIMER_CH_3	TIMER channel 3(TIMERx (x=0..2))
Input parameter{in}	
ocpolarity	channel complementary output polarity
TIMER_OCN_POLARITY_HIGH	channel complementary output polarity is high
TIMER_OCN_POLARITY_LOW	channel complementary output polarity is low
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 complementary output polarity */

timer_channel_complementary_output_polarity_config (TIMER0, TIMER_CH_0, TIMER_OCN_POLARITY_HIGH);

timer_channel_output_state_config

The description of timer_channel_output_state_config is shown as below:

Table 3-506. Function timer_channel_output_state_config

Function name	timer_channel_output_state_config
Function prototype	void timer_channel_output_state_config(uint32_t timer_periph, uint16_t channel, uint32_t state);
Function descriptions	configure TIMER channel enable state
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0(TIMERx (x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1(TIMERx (x=0..2, 14))

TIMER_CH_2	TIMER channel 2(TIMERx (x=0..2))
TIMER_CH_3	TIMER channel 3(TIMERx (x=0..2))
Input parameter{in}	
state	TIMER channel enable state
TIMER_CCX_ENABLE	channel enable
TIMER_CCX_DISABLE	channel disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 enable state */

timer_channel_output_state_config (TIMER0, TIMER_CH_0, TIMER_CCX_ENABLE);

timer_channel_complementary_output_state_config

The description of timer_channel_complementary_output_state_config is shown as below:

Table 3-507. Function timer_channel_complementary_output_state_config

Function name	timer_channel_complementary_output_state_config
Function prototype	void timer_channel_complementary_output_state_config(uint32_t timer_periph, uint16_t channel, uint16_t ocnstate);
Function descriptions	configure TIMER channel complementary output enable state
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0, 14,..16)	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0, *TIMERx (x=0, 14..16)*
TIMER_CH_1	TIMER channel 1, *TIMERx (x=0)*
TIMER_CH_2	TIMER channel 2, *TIMERx (x=0)*
Input parameter{in}	
state	TIMER channel complementary output enable state
TIMER_CCXN_ENABLE	channel complementary enable
TIMER_CCXN_DISABLE	channel complementary disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 complementary output enable state */

timer_channel_complementary_output_state_config (TIMER0, TIMER_CH_0, TIMER_CCXN_ENABLE);

timer_channel_input_struct_para_init

The description of timer_channel_input_struct_para_init is shown as below:

Table 3-508. Function timer_channel_input_struct_para_init

Function name	timer_channel_input_struct_para_init
Function prototype	void timer_channel_input_struct_para_init(timer_ic_parameter_struct* icpara);
Function descriptions	initialize the parameters of TIMER channel input parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
icpara	TIMER channel intput parameter struct, the structure members can refer to*Table 3-455. Structure timer_ic_parameter_struct*.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize TIMER channel input parameter struct with a default value */

timer_ic_parameter_struct timer_icinitpara;

timer_channel_input_struct_para_init(&timer_icinitpara);

timer_input_capture_config

The description of timer_input_capture_config is shown as below:

Table 3-509. Function timer_input_capture_config

Function name	timer_input_capture_config
Function prototype	void timer_input_capture_config(uint32_t timer_periph, uint16_t channel, timer_ic_parameter_struct* icpara);
Function descriptions	configure TIMER input capture parameter
Precondition	-
The called functions	timer_channel_input_capture_prescaler_config
Input parameter{in}	
timer_periph	TIMER peripheral

TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0(TIMERx (x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1(TIMERx (x=0..2, 14))
TIMER_CH_2	TIMER channel 2(TIMERx (x=0..2))
TIMER_CH_3	TIMER channel 3(TIMERx (x=0..2))
Input parameter{in}	
icpara	TIMER channel intput parameter struct, the structure members can refer to***Table 3-455. Structure timer_ic_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 input capture parameter    */

timer_ic_parameter_struct timer_icinitpara;

timer_icinitpara.icpolarity    = TIMER_IC_POLARITY_RISING;

timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTI;

timer_icinitpara.icprescaler = TIMER_IC_PSC_DIV1;

timer_icinitpara.icfilter          = 0x0;

timer_input_capture_config(TIMER0, TIMER_CH_0, &timer_icinitpara);
```

timer_channel_input_capture_prescaler_config

The description of timer_channel_input_capture_prescaler_config is shown as below:

Table 3-510. Function timer_channel_input_capture_prescaler_config

Function name	timer_channel_input_capture_prescaler_config
Function prototype	void timer_channel_input_capture_prescaler_config(uint32_t timer_periph, uint16_t channel, uint16_t prescaler);
Function descriptions	configure TIMER channel input capture prescaler value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0(TIMERx (x=0..2, 13..16))

TIMER_CH_1	TIMER channel 1(TIMERx (x=0..2, 14))
TIMER_CH_2	TIMER channel 2(TIMERx (x=0..2))
TIMER_CH_3	TIMER channel 3(TIMERx (x=0..2))
Input parameter{in}	
prescaler	channel input capture prescaler value
TIMER_IC_PSC_DIV1	no prescaler
TIMER_IC_PSC_DIV2	divided by 2
TIMER_IC_PSC_DIV4	divided by 4
TIMER_IC_PSC_DIV8	divided by 8
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 input capture prescaler value */

timer_channel_input_capture_prescaler_config (TIMER0, TIMER_CH_0, TIMER_IC_PSC_DIV2);

timer_channel_capture_value_register_read

The description of timer_channel_capture_value_register_read is shown as below:

Table 3-511. Function timer_channel_capture_value_register_read

Function name	timer_channel_capture_value_register_read
Function prototype	uint32_t timer_channel_capture_value_register_read(uint32_t timer_periph, uint16_t channel);
Function descriptions	read TIMER channel capture compare register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0(TIMERx (x=0..2, 13..16))
TIMER_CH_1	TIMER channel 1(TIMERx (x=0..2, 14))
TIMER_CH_2	TIMER channel 2(TIMERx (x=0..2))
TIMER_CH_3	TIMER channel 3(TIMERx (x=0..2))
Output parameter{out}	
-	-
Return value	
uint32_t	channel capture compare register value (0~65535)

Example:

/* read TIMER0 channel 0 capture compare register value */

uint32_t ch0_value = 0;

ch0_value = timer_channel_capture_value_register_read (TIMER0, TIMER_CH_0);

timer_input_pwm_capture_config

The description of timer_input_pwm_capture_config is shown as below:

Table 3-512. Function timer_input_pwm_capture_config

Function name	timer_input_pwm_capture_config
Function prototype	void timer_input_pwm_capture_config(uint32_t timer_periph, uint16_t channel, timer_ic_parameter_struct* icpwm);
Function descriptions	configure TIMER input pwm capture function
Precondition	-
The called functions	timer_channel_input_capture_prescaler_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 14)	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0
TIMER_CH_1	TIMER channel 1
Input parameter{in}	
icpwm	TIMER channel intput pwm parameter struct, the structure members can refer to***Table 3-455. Structure timer_ic_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 input pwm capture parameter */

timer_ic_parameter_struct timer_icinitpara;

timer_icinitpara.icpolarity = TIMER_IC_POLARITY_RISING;

timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTI;

timer_icinitpara.icprescaler = TIMER_IC_PSC_DIV1;

timer_icinitpara.icfilter = 0x0;

timer_input_pwm_capture_config (TIMER0, TIMER_CH_0, &timer_icinitpara);

timer_hall_mode_config

The description of timer_hall_mode_config is shown as below:

Table 3-513. Function timer_hall_mode_config

Function name	timer_hall_mode_config
Function prototype	void timer_hall_mode_config(uint32_t timer_periph, uint8_t hallmode);
Function descriptions	configure TIMER hall sensor mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2)	TIMER peripheral selection
Input parameter{in}	
hallmode	TIMER hall sensor mode state
TIMER_HALLINTERFACE_ENABLE	TIMER hall sensor mode enable
TIMER_HALLINTERFACE_DISABLE	TIMER hall sensor mode disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 hall sensor mode */

timer_hall_mode_config (TIMER0, TIMER_HALLINTERFACE_ENABLE);

timer_input_trigger_source_select

The description of timer_input_trigger_source_select is shown as below:

Table 3-514. Function timer_input_trigger_source_select

Function name	timer_input_trigger_source_select
Function prototype	void timer_input_trigger_source_select(uint32_t timer_periph, uint32_t intrigger);
Function descriptions	select TIMER input trigger source
Precondition	SMC[2:0] = 000
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 14)	please refer to the following parameters
Input parameter{in}	
intrigger	trigger selection

TIMER_SMCFG_TRGSEL_ITI0	Internal trigger input 0(ITI0，TIMERx(x=0..2, 14))
TIMER_SMCFG_TRGSEL_ITI1	Internal trigger input 0 (ITI1，TIMERx(x=0..2, 14))
TIMER_SMCFG_TRGSEL_ITI2	Internal trigger input 0 (ITI2, TIMERx(ITI2，TIMERx(x=0..2))
TIMER_SMCFG_TRGSEL_ITI3	Internal trigger input 0(ITI3，TIMERx(x=0..2, 14))
TIMER_SMCFG_TRGSEL_CI0F_ED	CI0 edge flag (CI0F_ED，TIMERx(x=0..2, 14))
TIMER_SMCFG_TRGSEL_CI0FE0	channel 0 input Filtered output(CI0FE0，TIMERx(x=0..2, 14))
TIMER_SMCFG_TRGSEL_CI1FE1	channel 1 input Filtered output(CI1FE1，TIMERx(x=0..2, 14))
TIMER_SMCFG_TRGSEL_ETIFP	External trigger input filter output(ETIFP，TIMERx(x=0..2))
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* select TIMER0 input trigger source */

timer_input_trigger_source_select (TIMER0, TIMER_SMCFG_TRGSEL_ITI0);

timer_master_output_trigger_source_select

The description of timer_master_output_trigger_source_select is shown as below:

Table 3-515. Function timer_master_output_trigger_source_select

Function name	timer_master_output_trigger_source_select
Function prototype	void timer_master_output_trigger_source_select(uint32_t timer_periph, uint32_t outrigger);
Function descriptions	select TIMER master mode output trigger source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 5, 14)	TIMER peripheral selection
Input parameter{in}	
outrigger	master mode control
TIMER_TRI_OUT_SRC_RESET	Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in

	the latter case, the signal on TRGO is delayed compared to the actual reset
TIMER_TRI_OUT_SRC_ENABLE	Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected.
TIMER_TRI_OUT_SRC_UPDATE	Update. In this mode the master mode controller selects the update event as TRGO.
TIMER_TRI_OUT_SRC_CH0	Capture/compare pulse. In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred in channel 0.
TIMER_TRI_OUT_SRC_O0CPRE	Compare. In this mode the master mode controller selects the O0CPRE signal is used as TRGO.
TIMER_TRI_OUT_SRC_O1CPRE	Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO.
TIMER_TRI_OUT_SRC_O2CPRE	Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO.
TIMER_TRI_OUT_SRC_O3CPRE	Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO.
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* select TIMER0 master mode output trigger source */

timer_master_output_trigger_source_select (TIMER0, TIMER_TRI_OUT_SRC_RESET);

timer_slave_mode_select

The description of timer_slave_mode_select is shown as below:

Table 3-516. Function timer_slave_mode_select

Function name	timer_slave_mode_select
Function prototype	void timer_slave_mode_select(uint32_t timer_periph, uint32_t slavemode);
Function descriptions	select TIMER slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
*TIMERx(*x=0..2, 14*)*	TIMER peripheral selection
Input parameter{in}	

slavemode	slave mode
TIMER_SLAVE_MODE_DISABLE	slave mode disable, TIMERx(x=0..2, 14)
TIMER_QUAD_DECODER_MODE0	quadrature decoder mode 0 TIMERx(x=0..2)
TIMER_QUAD_DECODER_MODE1	quadrature decoder mode 1, TIMERx(x=0..2)
TIMER_QUAD_DECODER_MODE2	quadrature decoder mode 2, TIMERx(x=0..2)
TIMER_SLAVE_MODE_RESTART	restart mode, TIMERx(x=0..2, 14)
TIMER_SLAVE_MODE_PAUSE	pause mode, TIMERx(x=0..2, 14)
TIMER_SLAVE_MODE_EVENT	event mode, TIMERx(x=0..2, 14)
TIMER_SLAVE_MODE_EXTERNAL0	external clock mode 0, TIMERx(x=0..2, 14)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* select TIMER0 slave mode */

timer_slave_mode_select (TIMER0, TIMER_QUAD_DECODER_MODE0);

timer_master_slave_mode_config

The description of timer_master_slave_mode_config is shown as below:

Table 3-517. Function timer_master_slave_mode_config

Function name	timer_master_slave_mode_config
Function prototype	void timer_master_slave_mode_config(uint32_t timer_periph, uint8_t masterslave);
Function descriptions	configure TIMER master slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2, 14)	TIMER peripheral selection
Input parameter{in}	
masterslave	master slave mode state
TIMER_MASTER_SLAV	master slave mode enable

E_MODE_ENABLE	
TIMER_MASTER_SLAVE_MODE_DISABLE	master slave mode disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 master slave mode    */
timer_master_slave_mode_config (TIMER0, TIMER_MASTER_SLAVE_MODE_ENABLE);
```

timer_external_trigger_config

The description of timer_external_trigger_config is shown as below:

Table 3-518. Function timer_external_trigger_config

Function name	timer_external_trigger_config
Function prototype	void timer_external_trigger_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER external trigger input
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2)	TIMER peripheral selection
Input parameter{in}	
extprescaler	external trigger prescaler
TIMER_EXT_TRI_PSC_OFF	no divided
TIMER_EXT_TRI_PSC_DIV2	divided by 2
TIMER_EXT_TRI_PSC_DIV4	divided by 4
TIMER_EXT_TRI_PSC_DIV8	divided by 8
Input parameter{in}	
expolarity	external trigger polarity
TIMER_ETP_FALLING	active low or falling edge active
TIMER_ETP_RISING	active high or rising edge active
Input parameter{in}	
extfilter	external trigger filter control（0~15）
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* configure TIMER0 external trigger input */

timer_external_trigger_config (TIMER0, TIMER_EXT_TRI_PSC_DIV2, TIMER_ETP_FALLING, 10);

timer_quadrature_decoder_mode_config

The description of timer_quadrature_decoder_mode_config is shown as below:

Table 3-519. Function timer_quadrature_decoder_mode_config

Function name	timer_quadrature_decoder_mode_config
Function prototype	void timer_quadrature_decoder_mode_config(uint32_t timer_periph, uint32_t decomode, uint16_t ic0polarity, uint16_t ic1polarity);
Function descriptions	configure TIMER quadrature decoder mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2)	TIMER peripheral selection
Input parameter{in}	
decomode	quadrature decoder mode
TIMER_QUAD_DECODER_MODE0	counter counts on CI0FE0 edge depending on CI1FE1 level
TIMER_QUAD_DECODER_MODE1	counter counts on CI1FE1 edge depending on CI0FE0 level
TIMER_QUAD_DECODER_MODE2	counter counts on both CI0FE0 and CI1FE1 edges depending on the level of the other input
Input parameter{in}	
ic0polarity	IC0 polarity
TIMER_IC_POLARITY_RISING	capture rising edge
TIMER_IC_POLARITY_FALLING	capture falling edge
Input parameter{in}	
ic1polarity	IC1 polarity
TIMER_IC_POLARITY_RISING	capture rising edge
TIMER_IC_POLARITY_FALLING	capture falling edge

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 quadrature decoder mode */

timer_quadrature_decoder_mode_config (TIMER0, TIMER_QUAD_DECODER_MODE0, TIMER_IC_POLARITY_RISING, TIMER_IC_POLARITY_RISING);

timer_internal_clock_config

The description of timer_internal_clock_config is shown as below:

Table 3-520. Function timer_internal_clock_config

Function name	timer_internal_clock_config
Function prototype	void timer_internal_clock_config(uint32_t timer_periph);
Function descriptions	configure TIMER internal clock mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
*TIMERx(*x=0..2, 14*)*	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 internal clock mode */

timer_internal_clock_config (TIMER0);

timer_internal_trigger_as_external_clock_config

The description of timer_internal_trigger_as_external_clock_config is shown as below:

Table 3-521. Function timer_internal_trigger_as_external_clock_config

Function name	timer_internal_trigger_as_external_clock_config
Function prototype	void timer_internal_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t intrigger);
Function descriptions	configure TIMER the internal trigger as external clock input
Precondition	-
The called functions	timer_input_trigger_source_select
Input parameter{in}	

timer_periph	TIMER peripheral
*TIMERx(*x=0..2, 14*)*	TIMER peripheral selection
Input parameter{in}	
intrigger	trigger selection
TIMER_SMCFG_TRGSEL_ITI0	Internal trigger input 0 (ITI0), TIMERx(x=0..2, 14)
TIMER_SMCFG_TRGSEL_ITI1	Internal trigger input 0 (ITI1) , TIMERx(x=0..2, 14)
TIMER_SMCFG_TRGSEL_ITI2	Internal trigger input 0 (ITI2) , TIMERx(x=0..2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 the internal trigger ITI0 as external clock input */

timer_internal_trigger_as_external_clock_config (TIMER0, TIMER_SMCFG_TRGSEL_ITI0);

timer_external_trigger_as_external_clock_config

The description of timer_external_trigger_as_external_clock_config is shown as below:

Table 3-522. Function timer_external_trigger_as_external_clock_config

Function name	timer_external_trigger_as_external_clock_config
Function prototype	void timer_external_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t extrigger, uint16_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER the external trigger as external clock input
Precondition	-
The called functions	timer_input_trigger_source_select
Input parameter{in}	
timer_periph	TIMER peripheral
*TIMERx(*x=0..2, 14*)*	TIMER peripheral selection
Input parameter{in}	
extrigger	external trigger selection
TIMER_SMCFG_TRGSEL_CI0F_ED	CI0 edge flag (CI0F_ED)
TIMER_SMCFG_TRGSEL_CI0FE0	channel 0 input Filtered output (CI0FE0)
TIMER_SMCFG_TRGSEL_CI1FE1	channel 1 input Filtered output (CI1FE1)
Input parameter{in}	
expolarity	external trigger polarity

TIMER_IC_POLARITY_RISING	active high or rising edge active
TIMER_IC_POLARITY_FALLING	active low or falling edge active
TIMER_IC_POLARITY_BOTH_EDGE	falling edge or rising edge active
Input parameter{in}	
extfilter	external trigger filter control（0~15）
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 the external trigger CI0FE0 as external clock input */

timer_external_trigger_as_external_clock_config (TIMER0, TIMER_SMCFG_TRGSEL_CI0FE0, TIMER_IC_POLARITY_RISING, 0);

timer_external_clock_mode0_config

The description of timer_external_clock_mode0_config is shown as below:

Table 3-523. Function timer_external_clock_mode0_config

Function name	timer_external_clock_mode0_config
Function prototype	void timer_external_clock_mode0_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER the external clock mode0
Precondition	-
The called functions	timer_external_trigger_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2)	TIMER peripheral selection
Input parameter{in}	
extprescaler	ETI external trigger prescaler
TIMER_EXT_TRI_PSC_OFF	no divided
TIMER_EXT_TRI_PSC_DIV2	divided by 2
TIMER_EXT_TRI_PSC_DIV4	divided by 4
TIMER_EXT_TRI_PSC_DIV8	divided by 8
Input parameter{in}	

expolarity	ETI external trigger polarity
TIMER_ETP_FALLING	active low or falling edge active
TIMER_ETP_RISING	active high or rising edge active
Input parameter{in}	
extfilter	ETI external trigger filter control（0~15）
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 the external clock mode0 */

timer_external_clock_mode0_config (TIMER0, TIMER_EXT_TRI_PSC_DIV2, TIMER_ETP_FALLING, 0);

timer_external_clock_mode1_config

The description of timer_external_clock_mode1_config is shown as below:

Table 3-524. Function timer_external_clock_mode1_config

Function name	timer_external_clock_mode1_config
Function prototype	void timer_external_clock_mode1_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER the external clock mode1
Precondition	-
The called functions	timer_external_trigger_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2)	TIMER peripheral selection
Input parameter{in}	
extprescaler	ETI external trigger prescaler
TIMER_EXT_TRI_PSC_OFF	no divided
TIMER_EXT_TRI_PSC_DIV2	divided by 2
TIMER_EXT_TRI_PSC_DIV4	divided by 4
TIMER_EXT_TRI_PSC_DIV8	divided by 8
Input parameter{in}	
expolarity	ETI external trigger polarity
TIMER_ETP_FALLING	active low or falling edge active
TIMER_ETP_RISING	active high or rising edge active

Input parameter{in}	
extfilter	ETI external trigger filter control（0~15）
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 the external clock mode1 */

timer_external_clock_mode1_config (TIMER0, TIMER_EXT_TRI_PSC_DIV2, TIMER_ETP_FALLING, 0);

timer_external_clock_mode1_disable

The description of timer_external_clock_mode1_disable is shown as below:

Table 3-525. Function timer_external_clock_mode1_disable

Function name	timer_external_clock_mode1_disable
Function prototype	void timer_external_clock_mode1_disable(uint32_t timer_periph);
Function descriptions	disable TIMER the external clock mode1
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..2*)*	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable TIMER0 the external clock mode1 */

timer_external_clock_mode1_disable (TIMER0);

timer_channel_remap_config

The description of timer_channel_remap_config is shown as below:

Table 3-526. Function timer_channel_remap_config

Function name	timer_channel_remap_config
Function prototype	void timer_channel_remap_config (uint32_t timer_periph, uint32_t remap);
Function descriptions	configure TIMER channel remap function
Precondition	-
The called functions	timer_external_trigger_config

Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=13)	TIMER peripheral selection
Input parameter{in}	
remap	remap function selection
TIMER13_CI0_RMP_GPIO	timer13 channel 0 input is connected to GPIO(TIMER13_CH0)
TIMER13_CI0_RMP_RTCCLK	timer13 channel 0 input is connected to the RTCCLK
TIMER13_CI0_RMP_HXTAL_DIV32	timer13 channel 0 input is connected to HXTAL/32 clock
TIMER13_CI0_RMP_CKOUTSEL	timer13 channel 0 input is connected to CKOUTSEL
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER13 channel 0 input is connected to GPIO */

timer_channel_remap_config (TIMER13, TIMER13_CI0_RMP_GPIO);
```

3.22. 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 ***3.22.1***, the TSI firmware functions are introduced in chapter ***3.22.2***.

3.22.1. Descriptions of Peripheral registers

TSI registers are listed in the table shown as below:

Table 3-527. TSI Registers

Registers	Descriptions
TSI_CTL	Control register0
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

Registers	Descriptions
TSI_GCTL	Group control register
TSI_GxCYCN (x= 0..5)	Group x cycle number registers

3.22.2. Descriptions of Peripheral functions

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

Table 3-528. TSI firmware function

Function name	Function description
tsi_deinit	reset TSI peripheral
tsi_init	initialize TSI plus prescaler,charge plus,transfer plus,max cycle number
tsi_enable	enable TSI module
tsi_disable	disable TSI module
tsi_sample_pin_enable	enable sample pin
tsi_sample_pin_disable	disable sample pin
tsi_channel_pin_enable	enable channel pin
tsi_channel_pin_disable	disable channel pin
tsi_sofeware_mode_config	configure TSI triggering by software
tsi_software_start	start a charge-transfer sequence when TSI is in software trigger mode
tsi_software_stop	stop a charge-transfer sequence when TSI is in software trigger mode
tsi_hardware_mode_config	configure TSI triggering by hardware
tsi_pin_mode_config	configure TSI pin mode when charge-transfer sequence is IDLE
tsi_extend_charge_config	configure extend charge state
tsi_plus_config	configure charge plus and transfer plus
tsi_max_number_config	configure the max cycle number of a charge-transfer sequence
tsi_hysteresis_on	switch on hysteresis pin
tsi_hysteresis_off	switch off hysteresis pin
tsi_analog_on	switch on analog pin
tsi_analog_off	switch off analog pin
tsi_flag_get	get flag
tsi_flag_clear	clear flag
tsi_interrupt_enable	enable TSI interrupt
tsi_interrupt_disable	disable TSI interrupt
tsi_interrupt_flag_get	get TSI interrupt flag
tsi_interrupt_flag_clear	Clear TSI interrupt flag
tsi_group_enable	enbale group
tsi_group_disable	disbale group

Function name	Function description
tsi_group_status_get	get group complete status
tsi_group0_cycle_get	get the cycle number for group0 as soon as a charge-transfer sequence completes
tsi_group1_cycle_get	get the cycle number for group1 as soon as a charge-transfer sequence completes
tsi_group2_cycle_get	get the cycle number for group2 as soon as a charge-transfer sequence completes
tsi_group3_cycle_get	get the cycle number for group3 as soon as a charge-transfer sequence completes
tsi_group4_cycle_get	get the cycle number for group4 as soon as a charge-transfer sequence completes
tsi_group5_cycle_get	get the cycle number for group5 as soon as a charge-transfer sequence completes

tsi_deinit

The description of tsi_deinit is shown as below:

Table 3-529. Function tsi_deinit

Function name	tsi_deinit
Function prototype	void tsi_deinit(void);
Function descriptions	reset TSI peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* reset TSI*/

tsi_deinit();

tsi_init

The description of tsi_init is shown as below:

Table 3-530. Function tsi_init

Function name	tsi_init
Function prototype	void tsi_init(uint32_t prescaler,uint32_t charge_duration,uint32_t transfer_duration,uint32_t max_number);

Function descriptions	initialize TSI plus prescaler,charge plus,transfer plus,max cycle number
Precondition	-
The called functions	-
Input parameter{in}	
prescaler	CTCLK clock division factor
TSI_CTCDIV_DIV1	$f_{CTCLK} = f_{HCLK}$
TSI_CTCDIV_DIV2	$f_{CTCLK} = f_{HCLK}/2$
TSI_CTCDIV_DIV4	$f_{CTCLK} = f_{HCLK}$ /4
TSI_CTCDIV_DIV8	$f_{CTCLK} = f_{HCLK}$ /8
TSI_CTCDIV_DIV16	$f_{CTCLK} = f_{HCLK}$ /16
TSI_CTCDIV_DIV32	$f_{CTCLK} = f_{HCLK}$ /32
TSI_CTCDIV_DIV64	$f_{CTCLK} = f_{HCLK}$ /64
TSI_CTCDIV_DIV128	$f_{CTCLK} = f_{HCLK}$ /128
Input parameter{in}	
charge_duration	charge state duration time
TSI_CHARGE_1CTCLK(x=1..16)	the duration time of charge state is x CTCLK
Input parameter{in}	
transfer_duration	charge transfer state duration time
TSI_TRANSFER_xCTCLK(x=1..16)	the duration time of transfer state is x CTCLK
Input parameter{in}	
max_number	max cycle number
TSI_MAXNUM255	the max cycle number of a sequence is 255
TSI_MAXNUM511	the max cycle number of a sequence is 511
TSI_MAXNUM1023	the max cycle number of a sequence is 1023
TSI_MAXNUM2047	the max cycle number of a sequence is 2047
TSI_MAXNUM4095	the max cycle number of a sequence is 4095
TSI_MAXNUM8191	the max cycle number of a sequence is 8191
TSI_MAXNUM16383	the max cycle number of a sequence is 16383
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* init TSI*/

tsi_init (TSI_CTCDIV_DIV128, TSI_CHARGE_10CTCLK, TSI_TRANSFER_8CTCLK,
TSI_MAXNUM511);
```

tsi_enable

The description of tsi_enable is shown as below:

Table 3-531. Function tsi_enable

Function name	tsi_enable
Function prototype	void tsi_enable (void);
Function descriptions	enable TSI module
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TSI module */

tsi_enable();
```

tsi_disable

The description of tsi_disable is shown as below:

Table 3-532. Function tsi_disable

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

Example:

```
/* disable TSI module */

tsi_disable ();
```

tsi_sample_pin_enable

The description of tsi_sample_pin_enable is shown as below:

Table 3-533. Function tsi_sample_pin_enable

Function name	tsi_sample_pin_enable

Function prototype	void tsi_sample_pin_enable(uint32_t sample);
Function descriptions	enable sample pin
Precondition	-
The called functions	-
Input parameter{in}	
sample	sample pin
TSI_SAMPCFG_GxPy(x=0..5,y=0..3)	pin y of group x is sample pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable G5P3 sample pin */

tsi_sample_pin_enable (TSI_SAMPCFG_G5P3);

tsi_sample_pin_disable

The description of tsi_sample_pin_disable is shown as below:

Table 3-534. Function tsi_sample_pin_disable

Function name	tsi_sample_pin_disable
Function prototype	void tsi_sample_pin_disable(uint32_t sample);
Function descriptions	disable sample pin
Precondition	-
The called functions	-
Input parameter{in}	
sample	sample pin
TSI_SAMPCFG_GxPy(x=0..5,y=0..3)	pin y of group x is sample pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable G5P3 sample pin */

tsi_sample_pin_disable (TSI_SAMPCFG_G5P3);

tsi_channel_pin_enable

The description of tsi_channel_pin_enable is shown as below:

Table 3-535. Function tsi_channel_pin_enable

Function name	tsi_channel_pin_enable
Function prototype	void tsi_channel_pin_enable(uint32_t channel);
Function descriptions	enable channel pin
Precondition	-
The called functions	-
Input parameter{in}	
channel	channel pin
TSI_CHCFG_GxPy(x= 0..5,y=0..3)	pin y of group x
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable G5P3 channel pin */

tsi_channel_pin_enable (TSI_CHCFG_G5P3);

tsi_channel_pin_disable

The description of tsi_channel_pin_disable is shown as below:

Table 3-536. Function tsi_channel_pin_disable

Function name	tsi_channel_pin_disable
Function prototype	void tsi_channel_pin_disable(uint32_t channel);
Function descriptions	disable channel pin
Precondition	-
The called functions	-
Input parameter{in}	
channel	channel pin
TSI_CHCFG_GxPy(x= 0..5,y=0..3)	pin y of group x
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable G5P3 channel pin */

tsi_channel_pin_disable (TSI_CHCFG_G5P3);

tsi_sofeware_mode_config

The description of tsi_sofeware_mode_config is shown as below:

Table 3-537. Function tsi_sofeware_mode_config

Function name	tsi_sofeware_mode_config
Function prototype	void tsi_sofeware_mode_config (void);
Function descriptions	configure TSI triggering by software
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TSI triggering by software */

tsi_sofeware_mode_config ();
```

tsi_software_start

The description of tsi_software_start is shown as below:

Table 3-538. Function tsi_software_start

Function name	tsi_software_start
Function prototype	void tsi_software_start (void);
Function descriptions	start a charge-transfer sequence when TSI is in software trigger mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start a charge-transfer sequence when TSI is in software trigger mode */

tsi_software_start ();
```

tsi_software_stop

The description of tsi_software_stop is shown as below:

Table 3-539. Function tsi_software_stop

Function name	tsi_software_stop
Function prototype	void tsi_software_stop (void);
Function descriptions	stop a charge-transfer sequence when TSI is in software trigger mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* stop a charge-transfer sequence when TSI is in software trigger mode */

tsi_software_stop ();

tsi_hardware_mode_config

The description of tsi_hardware_mode_config is shown as below:

Table 3-540. Function tsi_hardware_mode_config

Function name	tsi_hardware_mode_config
Function prototype	void tsi_hardware_mode_config(uint8_t trigger_edge);
Function descriptions	configure TSI triggering by hardware
Precondition	-
The called functions	-
Input parameter{in}	
trigger_edge	the edge type in hardware trigger mode
TSI_FALLING_TRIGGER	falling edge trigger TSI charge transfer sequence
TSI_RISING_TRIGGER	rising edge trigger TSI charge transfer sequence
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TSI triggering by hardware */

tsi_hardware_mode_config (TSI_FALLING_TRIGGER);

tsi_pin_mode_config

The description of tsi_pin_mode_config is shown as below:

Table 3-541. Function tsi_pin_mode_config

Function name	tsi_pin_mode_config
Function prototype	void tsi_pin_mode_config(uint8_t pin_mode);
Function descriptions	configure TSI pin mode when charge-transfer sequence is IDLE
Precondition	-
The called functions	-
Input parameter{in}	
pin_mode	pin mode when charge-transfer sequence is IDLE
TSI_OUTPUT_LOW	TSI pin will output low when IDLE
TSI_INPUT_FLOATING	TSI pin will keep input_floating when IDLE
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TSI pin mode when charge-transfer sequence is IDLE */

tsi_pin_mode_config (TSI_OUTPUT_LOW);

tsi_extend_charge_config

The description of tsi_extend_charge_config is shown as below:

Table 3-542. Function tsi_extend_charge_config

Function name	tsi_extend_charge_config
Function prototype	void tsi_extend_charge_config(ControlStatus extend,uint8_t prescaler,uint32_t max_duration);
Function descriptions	configure extend charge state
Precondition	-
The called functions	-
Input parameter{in}	
extend	enable or disable extend charge state
ENABLE	enable extend charge state
DISABLE	disable extend charge state
Input parameter{in}	
prescaler	ECCLK clock division factor
TSI_EXTEND_DIV1	$f_{ECCLK} = f_{HCLK}$
TSI_EXTEND_DIV2	$f_{ECCLK} = f_{HCLK} /2$
Input parameter{in}	
max_duration	extend charge state maximum duration time

value range 1...128	extend charge state maximum duration time is 1*tECCLK~128*tECCLK
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure extend charge state */

tsi_extend_charge_config (ENABLE, TSI_EXTEND_DIV2, 10);

tsi_plus_config

The description of tsi_plus_config is shown as below:

Table 3-543. Function tsi_plus_config

Function name	tsi_plus_config
Function prototype	void tsi_plus_config(uint32_t prescaler,uint32_t charge_duration,uint32_t transfer_duration);
Function descriptions	configure charge plus and transfer plus
Precondition	-
The called functions	-
Input parameter{in}	
prescaler	CTCLK clock division factor
TSI_CTCDIV_DIV1	$f_{CTCLK} = f_{HCLK}$
TSI_CTCDIV_DIV2	$f_{CTCLK} = f_{HCLK} /2$
TSI_CTCDIV_DIV4	$f_{CTCLK} = f_{HCLK} /4$
TSI_CTCDIV_DIV8	$f_{CTCLK} = f_{HCLK} /8$
TSI_CTCDIV_DIV16	$f_{CTCLK} = f_{HCLK} /16$
TSI_CTCDIV_DIV32	$f_{CTCLK} = f_{HCLK} /32$
TSI_CTCDIV_DIV64	$f_{CTCLK} = f_{HCLK} /64$
TSI_CTCDIV_DIV128	$f_{CTCLK} = f_{HCLK} /128$
Input parameter{in}	
charge_duration	charge state duration time
TSI_CHARGE_1CTCLK(x=1..16)	the duration time of charge state is x CTCLK
Input parameter{in}	
transfer_duration	charge transfer state duration time
TSI_TRANSFER_xCTCLK(x=1..16)	the duration time of transfer state is x CTCLK
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure charge plus and transfer plus */

tsi_plus_config (TSI_CTCDIV_DIV128, TSI_CHARGE_10CTCLK, TSI_TRANSFER_8CTCLK);

tsi_max_number_config

The description of tsi_max_number_config is shown as below:

Table 3-544. Function tsi_max_number_config

Function name	tsi_max_number_config
Function prototype	void tsi_max_number_config(uint32_t max_number);
Function descriptions	configure the max cycle number of a charge-transfer sequence
Precondition	-
The called functions	-
Input parameter{in}	
max_number	max cycle number
TSI_MAXNUM255	the max cycle number of a sequence is 255
TSI_MAXNUM511	the max cycle number of a sequence is 511
TSI_MAXNUM1023	the max cycle number of a sequence is 1023
TSI_MAXNUM2047	the max cycle number of a sequence is 2047
TSI_MAXNUM4095	the max cycle number of a sequence is 4095
TSI_MAXNUM8191	the max cycle number of a sequence is 8191
TSI_MAXNUM16383	the max cycle number of a sequence is 16383
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the max cycle number of a charge-transfer sequence */

tsi_max_number_config (TSI_MAXNUM1023);

tsi_hysteresis_on

The description of tsi_hysteresis_on is shown as below:

Table 3-545. Function tsi_hysteresis_on

Function name	tsi_hysteresis_on
Function prototype	void tsi_hysteresis_on(uint32_t group_pin);
Function descriptions	switch on hysteresis pin
Precondition	-
The called functions	-

Input parameter{in}	
group_pin	select pin which will be switched on hysteresis
TSI_PHM_GxPy(x=0..5, y=0..3)	pin y of group x switch on hysteresis
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* switch on hysteresis pin */

tsi_hysteresis_on (TSI_PHM_G5P3);

tsi_hysteresis_off

The description of tsi_hysteresis_off is shown as below:

Table 3-546. Function tsi_hysteresis_off

Function name	tsi_hysteresis_off
Function prototype	void tsi_hysteresis_off(uint32_t group_pin);
Function descriptions	switch off hysteresis pin
Precondition	-
The called functions	-
Input parameter{in}	
group_pin	select pin which will be switched off hysteresis
TSI_PHM_GxPy(x=0..5, y=0..3)	pin y of group x switch off hysteresis
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* switch off hysteresis pin */

tsi_hysteresis_off (TSI_PHM_G5P3);

tsi_analog_on

The description of tsi_analog_on is shown as below:

Table 3-547. Function tsi_analog_on

Function name	tsi_analog_on
Function prototype	void tsi_analog_on(uint32_t group_pin);
Function descriptions	switch on analog pin

Precondition	-
The called functions	-
Input parameter{in}	
group_pin	select pin which will be switched on analog
TSI_ASW_GxPy (x=0..5,y=0..3)	pin y of group x switch on analog
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* switch on analog pin */

tsi_analog_on (TSI_ASW_G5P3);

tsi_analog_off

The description of tsi_analog_off is shown as below:

Table 3-548. Function tsi_analog_off

Function name	tsi_analog_off
Function prototype	void tsi_analog_off(uint32_t group_pin);
Function descriptions	switch off analog pin
Precondition	-
The called functions	-
Input parameter{in}	
group_pin	select pin which will be switched off analog
TSI_ASW_GxPy (x=0..5,y=0..3)	pin y of group x switch off analog
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* switch off analog pin */

tsi_analog_off (TSI_ASW_G5P3);

tsi_flag_get

The description of tsi_flag_get is shown as below:

Table 3-549. Function tsi_flag_get

Function name	tsi_flag_get

Function prototype	FlagStatus tsi_flag_get(uint32_t flag);
Function descriptions	get flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag
TSI_FLAG_CTCF	charge-transfer complete flag
TSI_FLAG_MNERR	max cycle number error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TSI_FLAG_CTCF flag */

FlagStatus flag = RESET;

flag = tsi_flag_get (TSI_FLAG_CTCF);
```

tsi_flag_clear

The description of tsi_flag_clear is shown as below:

Table 3-550. Function tsi_flag_clear

Function name	tsi_flag_clear
Function prototype	void tsi_flag_clear(uint32_t flag);
Function descriptions	clear flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	select flag which will be cleared
TSI_FLAG_CTCF	charge-transfer complete flag
TSI_FLAG_MNERR	max cycle number error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear TSI_FLAG_CTCF flag */

tsi_flag_clear (TSI_FLAG_CTCF);
```

tsi_interrupt_enable

The description of tsi_interrupt_enable is shown as below:

Table 3-551. Function tsi_interrupt_enable

Function name	tsi_interrupt_enable
Function prototype	void tsi_interrupt_enable(uint32_t source);
Function descriptions	enable TSI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
source	select interrupt which will be enabled
TSI_INT_CCTCF	charge-transfer complete flag interrupt enable
TSI_INT_MNERR	max cycle number error interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable TSI_INT_CCTCF interrupt */

tsi_interrupt_enable (TSI_INT_CCTCF);

tsi_interrupt_disable

The description of tsi_interrupt_disable is shown as below:

Table 3-552. Function tsi_interrupt_disable

Function name	tsi_interrupt_disable
Function prototype	void tsi_interrupt_disable(uint32_t source);
Function descriptions	disable TSI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
source	select interrupt which will be disabled
TSI_INT_CCTCF	charge-transfer complete flag interrupt disable
TSI_INT_MNERR	max cycle number error interrupt disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable TSI_INT_CCTCF interrupt */

tsi_interrupt_disable (TSI_INT_CCTCF);

tsi_interrupt_flag_get

The description of tsi_interrupt_flag_get is shown as below:

Table 3-553. Function tsi_interrupt_flag_get

Function name	tsi_interrupt_flag_get
Function prototype	FlagStatus tsi_interrupt_flag_get(uint32_t flag);
Function descriptions	get TSI interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag
TSI_INT_FLAG_CTCF	charge-transfer complete interrupt flag
TSI_INT_FLAG_MNERR	max cycle number error interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get TSI_INT_FLAG_CTCF interrupt flag */

FlagStatus flag = RESET;

flag = tsi_interrupt_flag_get (TSI_INT_FLAG_CTCF);

tsi_interrupt_flag_clear

The description of tsi_interrupt_flag_clear is shown as below:

Table 3-554. Function tsi_interrupt_flag_clear

Function name	tsi_interrupt_flag_clear
Function prototype	void tsi_interrupt_flag_clear(uint32_t flag);
Function descriptions	clear TSI interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	select flag which will be cleared
TSI_INT_FLAG_CTCF	charge-transfer complete interrupt flag
TSI_INT_FLAG_MNERR	max cycle number error interrupt flag
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* clear TSI_INT_FLAG_CTCF interrupt flag */

tsi_interrupt_flag_clear (TSI_INT_FLAG_CTCF);

tsi_group_enable

The description of tsi_group_enable is shown as below:

Table 3-555. Function tsi_group_enable

Function name	tsi_group_enable
Function prototype	void tsi_group_enable(uint32_t group);
Function descriptions	enbale group
Precondition	-
The called functions	-
Input parameter{in}	
group	select group to be enabled
TSI_GCTL_GEx(x=0..5)	the x group will be enabled
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable group 5 */

tsi_group_enable (TSI_GCTL_GE5);

tsi_group_disable

The description of tsi_group_disable is shown as below:

Table 3-556. Function tsi_group_disable

Function name	tsi_group_disable
Function prototype	void tsi_group_disable(uint32_t group);
Function descriptions	disbale group
Precondition	-
The called functions	-
Input parameter{in}	
group	select group to be disabled
TSI_GCTL_GEx(x=0..5)	the x group will be disabled
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* disable group 5 */

tsi_group_disable (TSI_GCTL_GE5);

tsi_group_status_get

The description of tsi_group_status_get is shown as below:

Table 3-557. Function tsi_group_status_get

Function name	tsi_group_status_get
Function prototype	FlagStatus tsi_group_status_get(uint32_t group);
Function descriptions	get group complete status
Precondition	-
The called functions	-
Input parameter{in}	
group	select group
TSI_GCTL_GCx(x=0..5)	get the complete status of group x
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get group complete status */

FlagStatus flag = RESET;

flag = tsi_flag_get (TSI_GCTL_GC5);

tsi_group0_cycle_get

The description of tsi_group0_cycle_get is shown as below:

Table 3-558. Function tsi_group0_cycle_get

Function name	tsi_group0_cycle_get
Function prototype	uint16_t tsi_group0_cycle_get(void);
Function descriptions	get the cycle number for group0 as soon as a charge-transfer sequence completes
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
uint16_t	0-16383

Example:

/* get the cycle number for group0 as soon as a charge-transfer sequence completes */

uint16_t flag = 0;

flag = tsi_group0_cycle_get ();

tsi_group1_cycle_get

The description of tsi_group1_cycle_get is shown as below:

Table 3-559. Function tsi_group1_cycle_get

Function name	tsi_group1_cycle_get
Function prototype	uint16_t tsi_group1_cycle_get(void);
Function descriptions	get the cycle number for group1 as soon as a charge-transfer sequence completes
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	0-16383

Example:

/* get the cycle number for group1 as soon as a charge-transfer sequence completes */

uint16_t flag = 0;

flag = tsi_group1_cycle_get ();

tsi_group2_cycle_get

The description of tsi_group2_cycle_get is shown as below:

Table 3-560. Function tsi_group2_cycle_get

Function name	tsi_group2_cycle_get
Function prototype	uint16_t tsi_group2_cycle_get(void);
Function descriptions	get the cycle number for group2 as soon as a charge-transfer sequence completes
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	0-16383

Example:

/* get the cycle number for group2 as soon as a charge-transfer sequence completes */

uint16_t flag = 0;

flag = tsi_group2_cycle_get ();

tsi_group3_cycle_get

The description of tsi_group3_cycle_get is shown as below:

Table 3-561. Function tsi_group3_cycle_get

Function name	tsi_group3_cycle_get
Function prototype	uint16_t tsi_group3_cycle_get(void);
Function descriptions	get the cycle number for group3 as soon as a charge-transfer sequence completes
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	0-16383

Example:

/* get the cycle number for group3 as soon as a charge-transfer sequence completes */

uint16_t flag = 0;

flag = tsi_group3_cycle_get ();

tsi_group4_cycle_get

The description of tsi_group4_cycle_get is shown as below:

Table 3-562. Function tsi_group4_cycle_get

Function name	tsi_group4_cycle_get
Function prototype	uint16_t tsi_group4_cycle_get(void);

Function descriptions	get the cycle number for group4 as soon as a charge-transfer sequence completes
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	0-16383

Example:

/* get the cycle number for group4 as soon as a charge-transfer sequence completes */

uint16_t flag = 0;

flag = tsi_group4_cycle_get ();

tsi_group5_cycle_get

The description of tsi_group5_cycle_get is shown as below:

Table 3-563. Function tsi_group5_cycle_get

Function name	tsi_group5_cycle_get
Function prototype	uint16_t tsi_group5_cycle_get(void);
Function descriptions	get the cycle number for group5 as soon as a charge-transfer sequence completes
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	0-16383

Example:

/* get the cycle number for group5 as soon as a charge-transfer sequence completes */

uint16_t flag = 0;

flag = tsi_group5_cycle_get ();

3.23. USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter ***3.23.1***, the USART firmware functions are introduced in chapter ***3.23.2***.

3.23.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-564. 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

3.23.2. Descriptions of Peripheral functions

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

Table 3-565. USART firmware function

Function name	Function description
usart_deinit	reset USART
usart_baudrate_set	configure USART baud rate value
usart_parity_config	configure USART parity function
usart_word_length_set	configure USART word length
usart_stop_bit_set	configure USART stop bit length
usart_enable	enable USART
usart_disable	disable USART
usart_transmit_config	configure USART transmitter
usart_receive_config	configure USART receiver
usart_data_first_config	data is transmitted/received with the LSB/MSB first
usart_invert_config	configure USART inverted
usart_overrun_enable	enable the USART overrun function
usart_overrun_disable	disable the USART overrun function

Function name	Function description
usart_oversample_config	configure the USART oversample mode
usart_sample_bit_config	configure sample bit method
usart_receiver_timeout_enable	enable receiver timeout
usart_receiver_timeout_disable	disable receiver timeout
usart_receiver_timeout_threshold_config	configure receiver timeout threshold
usart_data_transmit	USART transmit data function
usart_data_receive	USART receive data function
usart_address_config	configure the address of the USART in wake up by address match mode
usart_address_detection_mode_config	configure address detection mode
usart_mute_mode_enable	enable mute mode
usart_mute_mode_disable	disable mute mode
usart_mute_mode_wakeup_config	configure wakeup method in mute mode
usart_lin_mode_enable	enable LIN mode
usart_lin_mode_disable	disable LIN mode
usart_lin_break_dection_length_config	configure LIN break frame length
usart_halfduplex_enable	enable half duplex mode
usart_halfduplex_disable	disable half duplex mode
usart_clock_enable	enable USART clock
usart_clock_disable	disable USART clock
usart_synchronous_clock_config	configure USART synchronous mode parameters
usart_guard_time_config	configure guard time value in smartcard mode
usart_smartcard_mode_enable	enable smartcard mode
usart_smartcard_mode_disable	disable smartcard mode
usart_smartcard_mode_nack_enable	enable NACK in smartcard mode
usart_smartcard_mode_nack_disable	disable NACK in smartcard mode
usart_smartcard_autoretry_config	configure smartcard auto-retry number
usart_block_length_config	configure block length
usart_irda_mode_enable	enable IrDA mode
usart_irda_mode_disable	disable IrDA mode
usart_prescaler_config	configure the peripheral clock prescaler in USART IrDA low-power mode or SmartCard mode
usart_irda_lowpower_config	configure IrDA low-power
usart_hardware_flow_rts_config	configure hardware flow control RTS
usart_hardware_flow_cts_config	configure hardware flow control CTS
usart_rs485_driver_enable	enable RS485 driver
usart_rs485_driver_disable	disable RS485 driver
usart_driver_assertime_config	configure driver enable assertion time

Function name	Function description
usart_driver_deassertime_config	configure driver enable de-assertion time
usart_depolarity_config	configure driver enable polarity mode
usart_dma_receive_config	configure USART DMA for reception
usart_dma_transmit_config	configure USART DMA for transmission
usart_reception_error_dma_disable	disable DMA on reception error
usart_reception_error_dma_enable	enable DMA on reception error
usart_wakeup_enable	enable USART to wakeup the mcu from deep-sleep mode
usart_wakeup_disable	disable USART to wakeup the mcu from deep-sleep mode
usart_wakeup_mode_config	configure the USART wakeup mode from deep-sleep mode
usart_command_enable	enable USART command
usart_flag_get	get flag in STAT/RFCS register
usart_flag_clear	clear flag in STAT register
usart_interrupt_enable	enable USART interrupt
usart_interrupt_disable	disable USART interrupt
usart_interrupt_flag_get	get USART interrupt and flag status
usart_interrupt_flag_clear	clear USART interrupt flag

Enum usart_flag_enum

Table 3-566. Enum usart_flag_enum

Member name	Function description
USART_FLAG_REA	receive enable acknowledge flag
USART_FLAG_TEA	transmit enable acknowledge flag
USART_FLAG_WU	wakeup from deep-sleep mode flag
USART_FLAG_RWU	receiver wakeup from mute mode
USART_FLAG_SB	send break flag
USART_FLAG_AM	ADDR match flag
USART_FLAG_BSY	busy flag
USART_FLAG_EB	end of block flag
USART_FLAG_RT	receiver timeout flag
USART_FLAG_CTS	CTS level
USART_FLAG_CTSF	CTS change flag
USART_FLAG_LBD	LIN break detected flag
USART_FLAG_TBE	transmit data buffer empty
USART_FLAG_TC	transmission complete
USART_FLAG_RBNE	read data buffer not empty
USART_FLAG_IDLE	IDLE line detected flag
USART_FLAG_ORERR	overrun error
USART_FLAG_NERR	noise error flag
USART_FLAG_FERR	frame error flag
USART_FLAG_PERR	parity error flag

Enum usart_interrupt_flag_enum

Table 3-567. Enum usart_interrupt_flag_enum

Member name	Function description
USART_INT_FLAG_EB	end of block interrupt flag
USART_INT_FLAG_RT	receiver timeout interrupt flag
USART_INT_FLAG_AM	address match interrupt flag
USART_INT_FLAG_PERR	parity error interrupt flag
USART_INT_FLAG_TBE	transmitter buffer empty interrupt flag
USART_INT_FLAG_TC	transmission complete interrupt flag
USART_INT_FLAG_RBNE	read data buffer not empty interrupt flag
USART_INT_FLAG_RBNE_ORERR	overrun error interrupt flag
USART_INT_FLAG_IDLE	IDLE line detected interrupt flag
USART_INT_FLAG_LBD	LIN break detected interrupt flag
USART_INT_FLAG_WU	wakeup from deep-sleep mode interrupt flag
USART_INT_FLAG_CTS	CTS interrupt flag
USART_INT_FLAG_ERR_NERR	noise error interrupt flag
USART_INT_FLAG_ERR_ORERR	overrun error interrupt flag
USART_INT_FLAG_ERR_FERR	frame error interrupt flag

Enum usart_interrupt_enum

Table 3-568. Enum usart_interrupt_enum

Member name	Function description
USART_INT_EB	end of block interrupt
USART_INT_RT	receiver timeout interrupt
USART_INT_AM	address match interrupt
USART_INT_PERR	parity error interrupt
USART_INT_TBE	transmitter buffer empty interrupt
USART_INT_TC	transmission complete interrupt
USART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt
USART_INT_IDLE	IDLE line detected interrupt
USART_INT_LBD	LIN break detected interrupt
USART_INT_WU	wakeup from deep-sleep mode interrupt
USART_INT_CTS	CTS interrupt
USART_INT_ERR	error interrupt

Enum usart_invert_enum

Table 3-569. Enum usart_invert_enum

Member name	Function description
USART_DINV_ENABLE	data bit level inversion

Member name	Function description
USART_DINV_DISABLE	data bit level not inversion
USART_TXPIN_ENABLE	TX pin level inversion
USART_TXPIN_DISABLE	TX pin level not inversion
USART_RXPIN_ENABLE	RX pin level inversion
USART_RXPIN_DISABLE	RX pin level not inversion
USART_SWAP_ENABLE	swap TX/RX pins
USART_SWAP_DISABLE	not swap TX/RX pins

usart_deinit

The description of usart_deinit is shown as below:

Table 3-570. Function usart_deinit

Function name	usart_deinit
Function prototype	void usart_deinit(uint32_t usart_periph);
Function descriptions	reset USART
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset USART0 */

usart_deinit(USART0);
```

usart_baudrate_set

The description of usart_baudrate_set is shown as below:

Table 3-571. Function usart_baudrate_set

Function name	usart_baudrate_set
Function prototype	void usart_baudrate_set(uint32_t usart_periph, uint32_t baudval);
Function descriptions	configure USART baud rate value
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
baudval	baud rate value

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 baud rate value */

usart_baudrate_set(USART0, 115200);

usart_parity_config

The description of usart_parity_config is shown as below:

Table 3-572. Function usart_parity_config

Function name	usart_parity_config
Function prototype	void usart_parity_config(uint32_t usart_periph, uint32_t paritycfg);
Function descriptions	configure USART parity
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
paritycfg	configure USART parity
USART_PM_NONE	no parity
USART_PM_ODD	odd parity
USART_PM_EVEN	even parity
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 parity */

usart_parity_config(USART0, USART_PM_EVEN);

usart_word_length_set

The description of usart_word_length_set is shown as below:

Table 3-573. Function usart_word_length_set

Function name	usart_word_length_set
Function prototype	void usart_word_length_set(uint32_t usart_periph, uint32_t wlen);
Function descriptions	configure USART word length
Precondition	-

Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
wlen	USART word length configure
USART_WL_8BIT	8 bits
USART_WL_9BIT	9 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 word length */

usart_word_length_set(USART0, USART_WL_9BIT);

usart_stop_bit_set

The description of usart_stop_bit_set is shown as below:

Table 3-574. Function usart_stop_bit_set

Function name	usart_stop_bit_set
Function prototype	void usart_stop_bit_set(uint32_t usart_periph, uint32_t stblen);
Function descriptions	configure USART stop bit length
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
stblen	USART stop bit configure
USART_STB_1BIT	1 bit
USART_STB_0_5BIT	0.5 bit
USART_STB_2BIT	2 bits
USART_STB_1_5BIT	1.5 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 stop bit length */

usart_stop_bit_set(USART0, USART_STB_1_5BIT);

usart_enable

The description of usart_enable is shown as below:

Table 3-575. Function usart_enable

Function name	usart_enable
Function prototype	void usart_enable(uint32_t usart_periph);
Function descriptions	enable USART
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable USART0 */

usart_enable(USART0);

usart_disable

The description of usart_disable is shown as below:

Table 3-576. Function usart_disable

Function name	usart_disable
Function prototype	void usart_disable(uint32_t usart_periph);
Function descriptions	disable USART
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 */

usart_disable(USART0);

usart_transmit_config

The description of usart_transmit_config is shown as below:

Table 3-577. Function usart_transmit_config

Function name	usart_transmit_config
Function prototype	void usart_transmit_config(uint32_t usart_periph, uint32_t txconfig);
Function descriptions	configure USART transmitter
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
txconfig	enable or disable USART transmitter
USART_TRANSMIT_ENABLE	enable USART transmission
USART_TRANSMIT_DISABLE	disable USART transmission
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 transmitter */

usart_transmit_config(USART0, USART_TRANSMIT_ENABLE);
```

usart_receive_config

The description of usart_receive_config is shown as below:

Table 3-578. Function usart_receive_config

Function name	usart_receive_config
Function prototype	void usart_receive_config(uint32_t usart_periph, uint32_t rxconfig);
Function descriptions	configure USART receiver
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
rxconfig	enable or disable USART receiver
USART_RECEIVE_ENABLE	enable USART reception
USART_RECEIVE_DIS	disable USART reception

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

Example:

/* configure USART0 receiver */

usart_receive_config(USART0, USART_RECEIVE_ENABLE);

usart_data_first_config

The description of usart_data_first_config is shown as below:

Table 3-579. Function usart_ data_first_config

Function name	usart_data_first_config
Function prototype	void usart_data_first_config(uint32_t usart_periph, uint32_t msbf);
Function descriptions	data is transmitted/received with the LSB/MSB first
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
msbf	LSB/MSB
USART_MSBF_LSB	LSB first
USART_MSBF_MSB	MSB first
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure LSB of data first */

usart_data_first_config(USART0, USART_MSBF_LSB);

usart_invert_config

The description of usart_invert_config is shown as below:

Table 3-580. Function usart_invert_config

Function name	usart_invert_config
Function prototype	void usart_invert_config(uint32_t usart_periph, usart_invert_enum invertpara);
Function descriptions	USART inverted configure

Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
invertpara	refer to usart_invert_enum
USART_DINV_ENABLE	data bit level inversion
USART_DINV_DISABLE	data bit level not inversion
USART_TXPIN_ENABLE	TX pin level inversion
USART_TXPIN_DISABLE	TX pin level not inversion
USART_RXPIN_ENABLE	RX pin level inversion
USART_RXPIN_DISABLE	RX pin level not inversion
USART_SWAP_ENABLE	swap TX/RX pins
USART_SWAP_DISABLE	not swap TX/RX pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 inversion */

usart_invert_config(USART0, USART_DINV_ENABLE);
```

usart_overrun_enable

The description of usart_overrun_enable is shown as below:

Table 3-581. Function usart_overrun_enable

Function name	usart_overrun_enable
Function prototype	void usart_overrun_enable(uint32_t usart_periph);
Function descriptions	enable the USART overrun function
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable USART0 overrun */

usart_overrun_enable(USART0);

usart_overrun_disable

The description of usart_overrun_disable is shown as below:

Table 3-582. Function usart_overrun_disable

Function name	usart_overrun_disable
Function prototype	void usart_overrun_disable(uint32_t usart_periph);
Function descriptions	disable the USART overrun function
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 overrun */

usart_overrun_disable(USART0);

usart_oversample_config

The description of usart_oversample_config is shown as below:

Table 3-583. Function usart_oversample_config

Function name	usart_oversample_config
Function prototype	void usart_oversample_config(uint32_t usart_periph, uint32_t oversamp);
Function descriptions	configure the USART oversample mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
oversamp	oversample value

USART_OVSMOD_8	oversampling by 8
USART_OVSMOD_16	oversampling by 16
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 oversampling by 8 */

usart_oversample_config(USART0,USART_OVSMOD_8);

usart_sample_bit_config

The description of usart_sample_bit_config is shown as below:

Table 3-584. Function usart_sample_bit_config

Function name	usart_sample_bit_config
Function prototype	void usart_sample_bit_config(uint32_t usart_periph, uint32_t osb);
Function descriptions	configure the sample bit method
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
osb	sample bit method
USART_OSB_1BIT	One sample bit method
USART_OSB_3BIT	Three sample bit method
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* config USART0 1 bit sample mode */

usart_sample_bit_config(USART0, USART_OSB_1BIT);

usart_receiver_timeout_enable

The description of usart_receiver_timeout_enable is shown as below:

Table 3-585. Function usart_receiver_timeout_enable

Function name	usart_receiver_timeout_enable
Function prototype	void usart_receiver_timeout_enable(uint32_t usart_periph);
Function descriptions	enable receiver timeout

Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable USART0 receiver timeout */

usart_receiver_timeout_enable(USART0);

usart_receiver_timeout_disable

The description of usart_receiver_timeout_disable is shown as below:

Table 3-586. Function usart_receiver_timeout_disable

Function name	usart_receiver_timeout_disable
Function prototype	void usart_receiver_timeout_disable(uint32_t usart_periph);
Function descriptions	disable receiver timeout
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 receiver timeout */

usart_receiver_timeout_disable(USART0);

usart_receiver_timeout_threshold_config

The description of usart_receiver_timeout_threshold_config is shown as below:

Table 3-587. Function usart_receiver_timeout_threshold_config

Function name	usart_receiver_timeout_threshold_config
Function prototype	void usart_receiver_timeout_threshold_config(uint32_t usart_periph, uint32_t rtimeout);
Function descriptions	configure receiver timeout threshold
Precondition	-

Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Input parameter{in}	
rtimeout	receiver timeout
0x00000000-0x00FFFFFF	receiver timeout value
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set the receiver timeout threshold of USART0*/

usart_receiver_timeout_threshold_config(USART0, 115200*3);

usart_data_transmit

The description of usart_data_transmit is shown as below:

Table 3-588. Function usart_data_transmit

Function name	usart_data_transmit
Function prototype	void usart_data_transmit(uint32_t usart_periph, uint32_t data);
Function descriptions	USART transmit data function
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
data	data of transmission
0-0x1FF	data of transmission
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 transmit data */

usart_data_transmit(USART0, 0xAA);

usart_data_receive

The description of usart_data_receive is shown as below:

Table 3-589. Function usart_data_receive

Function name	usart_data_receive
Function prototype	void usart_data_receive(uint32_t usart_periph);
Function descriptions	USART receive data function
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
uint32_t	data of received（0-0x1FF）

Example:

/* USART0 receive data */

uint16_t temp;

temp = usart_data_receive(USART0);

usart_address_config

The description of usart_address_config is shown as below:

Table 3-590. Function usart_address_config

Function name	usart_address_config
Function prototype	void usart_address_config(uint32_t usart_periph, uint8_t addr);
Function descriptions	configure the address of the USART terminal
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
addr	address of USART
0-0xFF	address of USART
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure address of the USART0 */

usart_address_config(USART0, 0x00);

usart_address_detection_mode_config

The description of usart_address_detection_mode_config is shown as below:

Table 3-591. Function usart_address_detection_mode_config

Function name	usart_address_detection_mode_config
Function prototype	void usart_address_detection_mode_config(uint32_t usart_periph, uint32_t addmod);
Function descriptions	configure address detection mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
addmod	address detection mode
USART_ADDM_4BIT	4 bits
USART_ADDM_FULLBIT	full bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*configure address detection mode */

usart_address_config(USART0, USART_ADDM_4BIT);

usart_mute_mode_enable

The description of usart_mute_mode_enable is shown as below:

Table 3-592. Function usart_mute_mode_enable

Function name	usart_mute_mode_enable
Function prototype	void usart_mute_mode_enable(uint32_t usart_periph);
Function descriptions	enable mute mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable USART0 receiver in mute mode */

usart_mute_mode_enable(USART0);

usart_mute_mode_disable

The description of usart_mute_mode_disable is shown as below:

Table 3-593. Function usart_mute_mode_disable

Function name	usart_mute_mode_disable
Function prototype	void usart_mute_mode_disable(uint32_t usart_periph);
Function descriptions	disable mute mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 receiver in mute mode */

usart_mute_mode_disable(USART0);

usart_mute_mode_wakeup_config

The description of usart_mute_mode_wakeup_config is shown as below:

Table 3-594. Function usart_mute_mode_wakeup_config

Function name	usart_mute_mode_wakeup_config
Function prototype	void usart_mute_mode_wakeup_config(uint32_t usart_periph, uint32_t wmethod);
Function descriptions	configure wakeup method in mute mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
wmethod	two methods be used to enter or exit the mute mode
USART_WM_IDLE	idle line
USART_WM_ADDR	address mask
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* configure USART0 wakeup method in mute mode */

usart_mute_mode_wakeup_config(USART0, USART_WM_IDLE);

usart_lin_mode_enable

The description of usart_lin_mode_enable is shown as below:

Table 3-595. Function usart_lin_mode_enable

Function name	usart_lin_mode_enable
Function prototype	void usart_lin_mode_enable(uint32_t usart_periph);
Function descriptions	enable LIN mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 LIN mode enable */

usart_lin_mode_enable(USART0);

usart_lin_mode_disable

The description of usart_lin_mode_disable is shown as below:

Table 3-596. Function usart_lin_mode_disable

Function name	usart_lin_mode_disable
Function prototype	void usart_lin_mode_disable(uint32_t usart_periph);
Function descriptions	disable LIN mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 LIN mode disable */

usart_lin_mode_disable(USART0);

usart_lin_break_dection_length_config

The description of usart_lin_break_dection_length_config is shown as below:

Table 3-597. Function usart_lin_break_dection_length_config

Function name	usart_lin_break_dection_length_config
Function prototype	void usart_lin_break_dection_length_config(uint32_t usart_periph, uint32_t lblen);
Function descriptions	configure lin break frame length
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Input parameter{in}	
lblen	LIN break detection length
USART_LBLEN_10B	10 bits break detection
USART_LBLEN_11B	11 bits break detection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure LIN break frame length */

usart_lin_break_dection_length_config(USART0, USART_LBLEN_10B);

usart_halfduplex_enable

The description of usart_halfduplex_enable is shown as below:

Table 3-598. Function usart_halfduplex_enable

Function name	usart_halfduplex_enable
Function prototype	void usart_halfduplex_enable(uint32_t usart_periph);
Function descriptions	enable half-duplex mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* enable USART0 half duplex mode*/

usart_halfduplex_enable(USART0);

usart_halfduplex_disable

The description of usart_halfduplex_disable is shown as below:

Table 3-599. Function usart_halfduplex_disable

Function name	usart_halfduplex_disable
Function prototype	void usart_halfduplex_disable(uint32_t usart_periph);
Function descriptions	disable half-duplex mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 half duplex mode*/

usart_halfduplex_disable(USART0);

usart_clock_enable

The description of usart_clock_enable is shown as below:

Table 3-600. Function usart_clock_enable

Function name	usart_clock_enable
Function prototype	void usart_clock_enable(uint32_t usart_periph);
Function descriptions	enable USART clock
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable USART0 CK pin */

usart_synchronous_clock_enable(USART0);

usart_clock_disable

The description of usart_clock_disable is shown as below:

Table 3-601. Function usart_clock_disable

Function name	usart_clock_disable
Function prototype	void usart_clock_disable(uint32_t usart_periph);
Function descriptions	disable USART clock
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 CK pin */

usart_synchronous_clock_disable(USART0);

usart_synchronous_clock_config

The description of usart_synchronous_clock_ config is shown as below:

Table 3-602. Function usart_synchronous_clock_ config

Function name	usart_synchronous_clock_ config
Function prototype	void usart_synchronous_clock_ config(uint32_t usart_periph, uint32_t clen, uint32_t cph, uint32_t cpl);
Function descriptions	configure USART synchronous mode parameters
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
clen	last bit clock pulse
USART_CLEN_NONE	clock pulse of the last data bit (MSB) is not output to the CK pin
USART_CLEN_EN	clock pulse of the last data bit (MSB) is output to the CK pin

Input parameter{in}	
cph	clock phase
USART_CPH_1CK	first clock transition is the first data capture edge
USART_CPH_2CK	second clock transition is the first data capture edge
Input parameter{in}	
cpl	clock polarity
USART_CPL_LOW	steady low value on CK pin
USART_CPL_HIGH	steady high value on CK pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 synchronous mode parameters */

usart_synchronous_clock_config(USART0,USART_CLEN_EN,USART_CPH_2CK, USART_CPL_HIGH);

usart_guard_time_config

The description of usart_guard_time_config is shown as below:

Table 3-603. Function usart_guard_time_config

Function name	usart_guard_time_config
Function prototype	void usart_guard_time_config(uint32_t usart_periph,uint32_t guat);
Function descriptions	configure guard time value in smartcard mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Input parameter{in}	
guat	guard time value
0-0x000000FF	guard time value
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 guard time value in smartcard mode */

usart_guard_time_config(USART0, 0x0000 0055);

usart_smartcard_mode_enable

The description of usart_smartcard_mode_enable is shown as below:

Table 3-604. Function usart_smartcard_mode_enable

Function name	usart_smartcard_mode_enable
Function prototype	void usart_smartcard_mode_enable(uint32_t usart_periph);
Function descriptions	enable smartcard mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 smartcard mode enable */

usart_smartcard_mode_enable(USART0);
```

usart_smartcard_mode_disable

The description of usart_smartcard_mode_disable is shown as below:

Table 3-605. Function usart_smartcard_mode_disable

Function name	usart_smartcard_mode_disable
Function prototype	void usart_smartcard_mode_disable(uint32_t usart_periph);
Function descriptions	disable smartcard mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 smartcard mode disable */

usart_smartcard_mode_disable(USART0);
```

usart_smartcard_mode_nack_enable

The description of usart_smartcard_mode_nack_enable is shown as below:

Table 3-606. Function usart_smartcard_mode_nack_enable

Function name	usart_smartcard_mode_nack_enable
Function prototype	void usart_smartcard_mode_nack_enable(uint32_t usart_periph);
Function descriptions	enable NACK in smartcard mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable USART0 NACK in smartcard mode */

usart_smartcard_mode_nack_enable(USART0);

usart_smartcard_mode_nack_disable

The description of usart_smartcard_mode_nack_disable is shown as below:

Table 3-607. Function usart_smartcard_mode_nack_disable

Function name	usart_smartcard_mode_nack_disable
Function prototype	void usart_smartcard_mode_nack_disable(uint32_t usart_periph);
Function descriptions	disable NACK in smartcard mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 NACK in smartcard mode */

usart_smartcard_mode_nack_disable(USART0);

usart_smartcard_autoretry_config

The description of usart_smartcard_autoretry_config is shown as below:

Table 3-608. Function usart_smartcard_autoretry_config

Function name	usart_smartcard_autoretry_config
Function prototype	void usart_smartcard_autoretry_config(uint32_t usart_periph, uint32_t scrtnum);
Function descriptions	configure smartcard auto-retry number
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Input parameter{in}	
scrtnum	smartcard auto-retry number
0x00000000-0x00000007	smartcard auto-retry number
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure smartcard auto-retry number */

usart_smartcard_autoretry_config(USART0, 0x00000007);

usart_block_length_config

The description of usart_block_length_config is shown as below:

Table 3-609. Function usart_block_length_config

Function name	usart_block_length_config
Function prototype	void usart_block_length_config(uint32_t usart_periph, uint32_t bl);
Function descriptions	configure block length
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Input parameter{in}	
bl	block length
0-0x000000FF	block length
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* configure block length in Smartcard T=1 reception */

usart_block_length_config(USART0, 0x000000FF);

usart_irda_mode_enable

The description of usart_irda_mode_enable is shown as below:

Table 3-610. Function usart_irda_mode_enable

Function name	usart_irda_mode_enable
Function prototype	void usart_irda_mode_enable(uint32_t usart_periph);
Function descriptions	enable IrDA mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable USART0 IrDA mode */

usart_irda_mode_enable(USART0);

usart_irda_mode_disable

The description of usart_irda_mode_disable is shown as below:

Table 3-611. Function usart_irda_mode_disable

Function name	usart_irda_mode_disable
Function prototype	void usart_irda_mode_disable(uint32_t usart_periph);
Function descriptions	disable IrDA mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 IrDA mode */

usart_irda_mode_disable(USART0);

usart_prescaler_config

The description of usart_prescaler_config is shown as below:

Table 3-612. Function usart_prescaler_config

Function name	usart_prescaler_config
Function prototype	void usart_prescaler_config(uint32_t usart_periph, uint8_t psc);
Function descriptions	configure the peripheral clock prescaler in USART IrDA low-power mode or SmartCard mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Input parameter{in}	
psc	clock prescaler
0x00-0xFF	clock prescaler
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the USART0 peripheral clock prescaler in USART IrDA low-power mode */

usart_prescaler_config(USART0, 0x00);

usart_irda_lowpower_config

The description of usart_irda_lowpower_config is shown as below:

Table 3-613. Function usart_irda_lowpower_config

Function name	usart_irda_lowpower_config
Function prototype	void usart_irda_lowpower_config(uint32_t usart_periph, uint32_t irlp);
Function descriptions	configure IrDA low-power
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Input parameter{in}	
irlp	IrDA low-power or normal

USART_IRLP_LOW	low-power
USART_IRLP_NORMAL	normal
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 IrDA low-power */

usart_irda_lowpower_config(USART0, USART_IRLP_LOW);

usart_hardware_flow_rts_config

The description of usart_hardware_flow_rts_config is shown as below:

Table 3-614. Function usart_hardware_flow_rts_config

Function name	usart_hardware_flow_rts_config
Function prototype	void usart_hardware_flow_rts_config(uint32_t usart_periph, uint32_t rtsconfig);
Function descriptions	configure hardware flow control RTS
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
rtsconfig	enable or disable RTS
USART_RTS_ENABLE	enable RTS
USART_RTS_DISABLE	disable RTS
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 hardware flow control RTS */

usart_hardware_flow_rts_config(USART0, USART_RTS_ENABLE);

usart_hardware_flow_cts_config

The description of usart_hardware_flow_cts_config is shown as below:

Table 3-615. Function usart_hardware_flow_cts_config

Function name	usart_hardware_flow_cts_config

Function prototype	void usart_hardware_flow_cts_config(uint32_t usart_periph, uint32_t ctsconfig);
Function descriptions	configure hardware flow control CTS
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
ctsconfig	enable or disable CTS
USART_CTS_ENABLE	enable CTS
USART_CTS_DISABLE	disable CTS
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 hardware flow control CTS */

usart_hardware_flow_cts_config(USART0, USART_CTS_ENABLE);

usart_rs485_driver_enable

The description of usart_rs485_driver_enable is shown as below:

Table 3-616. Function usart_rs485_driver_enable

Function name	usart_rs485_driver_enable
Function prototype	void usart_rs485_driver_enable(uint32_t usart_periph);
Function descriptions	enable USART RS485 driver
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable USART0 RS485 driver */

usart_rs485_driver_enable(USART0);

usart_rs485_driver_disable

The description of usart_rs485_driver_disable is shown as below:

Table 3-617. Function usart_rs485_driver_disable

Function name	usart_rs485_driver_disable
Function prototype	void usart_rs485_driver_disable(uint32_t usart_periph);
Function descriptions	disable USARTRS485 driver
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 RS485 driver */

usart_rs485_driver_disable (USART0);

usart_driver_assertime_config

The description of usart_driver_assertime_config is shown as below:

Table 3-618. Function usart_driver_assertime_config

Function name	usart_driver_assertime_config
Function prototype	void usart_driver_assertime_config(uint32_t usart_periph, uint32_t deatime);
Function descriptions	configure driver enable assertion time
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
deatime	driver enable assertion time
0-0x0000001F	driver enable assertion time
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set USART0 driver assertime */

usart_driver_assertime_config(USART0,0x0000001F);

usart_driver_deassertime_config

The description of usart_driver_deassertime_config is shown as below:

Table 3-619. Function usart_driver_deassertime_config

Function name	usart_driver_deassertime_config
Function prototype	void usart_driver_deassertime_config(uint32_t usart_periph, uint32_t dedtime);
Function descriptions	configure driver enable de-assertion time
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
deatime	driver enable de-assertion time
0x00000000-0x0000001F	driver enable de-assertion time
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set USART0 driver deassertime */

usart_driver_deassertime_config(USART0, 0x0000001F);
```

usart_depolarity_config

The description of usart_depolarity_config is shown as below:

Table 3-620. Function usart_depolarity_config

Function name	usart_depolarity_config
Function prototype	void usart_depolarity_config(uint32_t usart_periph, uint32_t dep);
Function descriptions	configure driver enable polarity mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
dep	DE signal
USART_DEP_HIGH	DE signal is active high
USART_DEP_LOW	DE signal is active low
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* configure driver enable polarity mode */

usart_driver_depolarity_config(USART0, USART_DEP_HIGH);

usart_dma_receive_config

The description of usart_dma_receive_config is shown as below:

Table 3-621. Function usart_dma_receive_config

Function name	usart_dma_receive_config
Function prototype	void usart_dma_receive_config(uint32_t usart_periph, uint8_t dmacmd);
Function descriptions	configure USART DMA reception
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
dmacmd	enable or disable DMA for reception
USART_RECEIVE_DMA_ENABLE	DMA enable for reception
USART_RECEIVE_DMA_DISABLE	DMA disable for reception
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 DMA enable for reception */

usart_dma_receive_config(USART0, USART_DENR_ENABLE);

usart_dma_transmit_config

The description of usart_dma_transmit_config is shown as below:

Table 3-622. Function usart_dma_transmit_config

Function name	usart_dma_transmit_config
Function prototype	void usart_dma_transmit_config(uint32_t usart_periph, uint8_t dmacmd);
Function descriptions	configure USART DMA transmission
Precondition	-
Input parameter{in}	

usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
dmacmd	enable or disable DMA for transmission
USART_TRANSMIT_DMA_ENABLE	DMA enable for transmission
USART_TRANSMIT_DMA_DISABLE	DMA disable for transmission
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 DMA enable for transmission */

usart_dma_transmit_config(USART0, USART_DENT_ENABLE);

usart_reception_error_dma_disable

The description of usart_reception_error_dma_disable is shown as below:

Table 3-623. Function usart_reception_error_dma_disable

Function name	usart_reception_error_dma_disable
Function prototype	void usart_reception_error_dma_disable(uint32_t usart_periph);
Function descriptions	disable DMA on reception error
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DMA on reception error */

usart_reception_error_dma_disable(USART0);

usart_reception_error_dma_enable

The description of usart_reception_error_dma_enable is shown as below:

Table 3-624. Function usart_reception_error_dma_enable

Function name	usart_reception_error_dma_enable

Function prototype	void usart_reception_error_dma_enable(uint32_t usart_periph);
Function descriptions	enable DMA on reception error
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DMA on reception error */

usart_reception_error_dma_ enable(USART0);

usart_wakeup_enable

The description of usart_wakeup_enable is shown as below:

Table 3-625. Function usart_wakeup_enable

Function name	usart_wakeup_enable
Function prototype	void usart_wakeup_enable(uint32_t usart_periph);
Function descriptions	enable USART to wakeup the mcu from deep-sleep mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 wake up enable */

usart_wakeup_enable(USART0);

usart_wakeup_disable

The description of usart_wakeup_disable is shown as below:

Table 3-626. Function usart_wakeup_disable

Function name	usart_wakeup_disable
Function prototype	void usart_wakeup_disable(uint32_t usart_periph);
Function descriptions	disable USART to wakeup the mcu from deep-sleep mode

Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 wake up disable */

usart_wakeup_disable(USART0);

usart_wakeup_mode_config

The description of usart_wakeup_mode_config is shown as below:

Table 3-627. Function usart_wakeup_mode_config

Function name	usart_wakeup_mode_config
Function prototype	void usart_wakeup_mode_config(uint32_t usart_periph, uint32_t wum);
Function descriptions	wakeup mode from deep-sleep mode
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0
Input parameter{in}	
wum	wakeup mode
USART_WUM_ADDR	WUF active on address match
USART_WUM_STARTB	WUF active on start bit
USART_WUM_RBNE	WUF active on RBNE
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 wake up mode */

usart_wakeup_mode_config(USART0, USART_WUM_ADDR);

usart_command_enable

The description of usart_command_enable is shown as below:

Table 3-628. Function usart_command_enable

Function name	usart_command_enable
Function prototype	void usart_command_enable(uint32_t usart_periph, uint32_t cmdtype);
Function descriptions	enable USART command
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
cmdtype	USART interrupt flag
USART_CMD_SBKCMD	send break command
USART_CMD_MMCMD	mute mode command
USART_CMD_RXFCMD	receive data flush command
USART_CMD_TXFCMD	transmit data flush request
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 command */

usart_command_enable(USART0, USART_CMD_SBKCMD);
```

usart_flag_get

The description of usart_flag_get is shown as below:

Table 3-629. Function usart_flag_get

Function name	usart_flag_get
Function prototype	FlagStatus usart_flag_get(uint32_t usart_periph, usart_flag_enum flag);
Function descriptions	get flag in STAT/RFCS register
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
flag	USART flags
USART_FLAG_PERR	parity error flag
USART_FLAG_FERR	frame error flag
USART_FLAG_NERR	noise error flag

USART_FLAG_ORERR	overrun error
USART_FLAG_IDLE	idle line detected flag
USART_FLAG_RBNE	read data buffer not empty
USART_FLAG_TC	transmission completed
USART_FLAG_TBE	transmit data register empty
USART_FLAG_LBD	LIN break detected flag
USART_FLAG_CTSF	CTS change flag
USART_FLAG_CTS	CTS level
USART_FLAG_RT	receiver timeout flag
USART_FLAG_EB	end of block flag
USART_FLAG_BSY	busy flag
USART_FLAG_AM	address match flag
USART_FLAG_SB	send break flag
USART_FLAG_RWU	receiver wakeup from mute mode
USART_FLAG_WU	wakeup from deep-sleep mode flag
USART_FLAG_TEA	transmit enable acknowledge flag
USART_FLAG_REA	receive enable acknowledge flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get flag USART0 state */

FlagStatus status;

status = usart_flag_get(USART0,USART_FLAG_TBE);

usart_flag_clear

The description of usart_flag_clear is shown as below:

Table 3-630. Function usart_flag_clear

Function name	usart_flag_clear
Function prototype	void usart_flag_clear(uint32_t usart_periph, usart_flag_enum flag);
Function descriptions	clear flag in STAT register
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
flag	USART flags
USART_FLAG_WU	wakeup from deep-sleep mode flag
USART_FLAG_AM	address match flag

USART_FLAG_EB	end of block flag
USART_FLAG_RT	receiver timeout flag
USART_FLAG_CTSF	CTS change flag
USART_FLAG_LBD	LIN break detected flag
USART_FLAG_TC	transmission complete flag
USART_FLAG_RBNE	ead data buffer not empty
USART_FLAG_IDLE	idle line detected flag
USART_FLAG_ORERR	overrun error flag
USART_FLAG_NERR	noise detected flag
USART_FLAG_FERR	frame error flag
USART_FLAG_PERR	parity error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear USART0 flag */

usart_flag_clear(USART0,USART_FLAG_TC);

usart_interrupt_enable

The description of usart_interrupt_enableis shown as below:

Table 3-631. Function usart_interrupt_enable

Function name	usart_interrupt_enable
Function prototype	void usart_interrupt_enable(uint32_t usart_periph, usart_interrupt_enum interrupt);
Function descriptions	enable USART interrupt
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
interrupt	interrupt type
USART_INT_IDLE	idle interrupt
USART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt enable interrupt
USART_INT_TC	transmission complete interrupt
USART_INT_TBE	transmit data register empty interrupt
USART_INT_PERR	parity error interrupt
USART_INT_AM	address match interrupt
USART_INT_RT	receiver timeout interrupt

USART_INT_EB	end of block interrupt
USART_INT_LBD	LIN break detection interrupt
USART_INT_ERR	error interrupt enable in multibuffer communication
USART_INT_CTS	CTS interrupt
USART_INT_WU	wakeup from deep-sleep mode interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable USART0 TBE interrupt */

usart_interrupt_enable(USART0, USART_INT_TBE);

usart_interrupt_disable

The description of usart_interrupt_disable is shown as below:

Table 3-632. Function usart_interrupt_disable

Function name	usart_interrupt_disable
Function prototype	void usart_interrupt_disable(uint32_t usart_periph, usart_interrupt_enum interrupt);
Function descriptions	disable USART interrupt
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
interrupt	USART interrupt flag
USART_INT_IDLE	idle interrupt
USART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt enable interrupt
USART_INT_TC	transmission complete interrupt
USART_INT_TBE	transmit data register empty interrupt
USART_INT_PERR	parity error interrupt
USART_INT_AM	address match interrupt
USART_INT_RT	receiver timeout interrupt
USART_INT_EB	end of block interrupt
USART_INT_LBD	LIN break detection interrupt
USART_INT_ERR	error interrupt enable in multibuffer communication
USART_INT_CTS	CTS interrupt
USART_INT_WU	wakeup from deep-sleep mode interrupt
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* disable USART0 TBE interrupt */

usart_interrupt_disable(USART0, USART_INT_TBE);

usart_interrupt_flag_get

The description of usart_interrupt_flag_get is shown as below:

Table 3-633. Function usart_interrupt_flag_get

Function name	usart_interrupt_flag_get
Function prototype	FlagStatus usart_interrupt_flag_get(uint32_t usart_periph, usart_interrupt_flag_enum int_flag);
Function descriptions	get USART interrupt and flag status
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
int_flag	USART interrupt flag
USART_INT_FLAG_EB	end of block interrupt and flag
USART_INT_FLAG_RT	receiver timeout interrupt and flag
USART_INT_FLAG_AM	address match interrupt and flag
USART_INT_FLAG_PERR	parity error interrupt and flag
USART_INT_FLAG_TBE	transmitter buffer empty interrupt and flag
USART_INT_FLAG_TC	transmission complete interrupt and flag
USART_INT_FLAG_RBNE	read data buffer not empty interrupt and flag
USART_INT_FLAG_RBNE_ORERR	read data buffer not empty interrupt and overrun error flag
USART_INT_FLAG_IDLE	IDLE line detected interrupt and flag
USART_INT_FLAG_LBD	LIN break detected interrupt and flag
USART_INT_FLAG_WU	wakeup from deep-sleep mode interrupt and flag
USART_INT_FLAG_CT	CTS interrupt and flag

S	
USART_INT_FLAG_ERR_NERR	error interrupt and noise error flag
USART_INT_FLAG_ERR_ORERR	error interrupt and overrun error
USART_INT_FLAG_ERR_FERR	error interrupt and frame error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the USART0 interrupt flag status */

FlagStatus status;

status = usart_interrupt_flag_get(USART0, USART_INT_FLAG_RBNE);

usart_interrupt_flag_clear

The description of usart_interrupt_flag_clear is shown as below:

Table 3-634. Function usart_interrupt_flag_clear

Function name	usart_interrupt_flag_clear
Function prototype	void usart_interrupt_flag_clear(uint32_t usart_periph, usart_interrupt_flag_enum int_flag);
Function descriptions	clear USART interrupt flag in STAT register
Precondition	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1
Input parameter{in}	
int_flag	USART interrupt flag
USART_INT_FLAG_PERR	parity error flag
USART_INT_FLAG_ERR_NERR	error interrupt and noise error flag
USART_INT_FLAG_RBNE_ORERR	read data buffer not empty interrupt and overrun error flag
USART_INT_FLAG_ERR_ORERR	error interrupt and overrun error
USART_INT_FLAG_ERR_FERR	error interrupt and frame error flag
USART_INT_FLAG_ID	idle line detected flag

LE	
USART_INT_FLAG_TC	transmission complete flag
USART_INT_FLAG_LBD	LIN break detected flag
USART_INT_FLAG_CTS	CTS change flag
USART_INT_FLAG_RT	receiver timeout flag
USART_INT_FLAG_EB	end of block flag
USART_INT_FLAG_AM	address match flag
USART_INT_FLAG_WU	wakeup from deep-sleep mode flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear the USART0 interrupt flag */

usart_interrupt_flag_clear(USART0, USART_INT_FLAG_TC);

3.24. WWDGT

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions.The WWDGT registers are listed in chapter ***3.24.1***, the FWDGT firmware functions are introduced in chapter ***3.24.2.***

3.24.1. Descriptions of Peripheral registers

WWDGT registers are listed in the table shown as below:

Table 3-635. WWDGT Registers

Registers	Descriptions
WWDGT_CTL	Control register
WWDGT_CFG	Configuration register
WWDGT_STAT	Status register

3.24.2. Descriptions of Peripheral functions

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

Table 3-636. WWDGT firmware function

Function name	Function description
wwdgt_deinit	reset the WWDGT counter configuration
wwdgt_enable	start the WWDGT counter
wwdgt_counter_update	configure the WWDGT counter value
wwdgt_config	configure counter value, window value, and prescaler divider value
wwdgt_interrupt_enable	enable early wakeup interrupt of WWDGT
wwdgt_flag_get	check early wakeup interrupt state of WWDGT
wwdgt_flag_clear	clear early wakeup interrupt state of WWDGT

wwdgt_deinit

The description of wwdgt_deinit is shown as below:

Table 3-637. Function wwdgt_deinit

Function name	wwdgt_deinit
Function prototype	void wwdgt_deinit(void);
Function descriptions	reset the WWDGT counter configuration
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the WWDGT configuration */

wwdgt_deinit ();
```

wwdgt_enable

The description of wwdgt_enable is shown as below:

Table 3-638. Function wwdgt_enable

Function name	wwdgt_enable
Function prototype	void wwdgt_enable (void);
Function descriptions	start the WWDGT counter
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* start the WWDGT counter */

wwdgt_enable ();

wwdgt_counter_update

The description of wwdgt_counter_update is shown as below:

Table 3-639. Function wwdgt_counter_update

Function name	wwdgt_counter_update
Function prototype	void wwdgt_counter_update(uint16_t counter_value);
Function descriptions	configure the WWDGT counter value
Precondition	-
Input parameter{in}	
counter_value	counter_value: 0x00000000 - 0x0000007F
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* update WWDGT counter to 0x7F */

wwdgt_counter_update(127);

wwdgt_config

The description of wwdgt_config is shown as below:

Table 3-640. Function wwdgt_config

Function name	wwdgt_config
Function prototype	void wwdgt_config(uint16_t counter, uint16_t window, uint32_t prescaler);
Function descriptions	configure counter value, window value, and prescaler divider value
Precondition	-
Input parameter{in}	
counter	counter: 0x00000000 - 0x0000007F
Input parameter{in}	
window	window: 0x00000000 - 0x0000007F
Input parameter{in}	
prescaler	wwdgt prescaler value
WWDGT_CFG_PSC_DIV1	the time base of WWDGT counter = (PCLK1/4096)/1

WWDGT_CFG_PSC_DIV2	the time base of WWDGT counter = (PCLK1/4096)/2
WWDGT_CFG_PSC_DIV4	the time base of WWDGT counter = (PCLK1/4096)/4
WWDGT_CFG_PSC_DIV8	the time base of WWDGT counter = (PCLK1/4096)/8
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* confiure WWDGT counter value to 0x7F, window value to 0x50, prescaler divider value to 8 */

wwdgt_config(127, 80, WWDGT_CFG_PSC_DIV8);

wwdgt_interrupt_enable

The description of wwdgt_interrupt_enable is shown as below:

Table 3-641. Function wwdgt_interrupt_enable

Function name	wwdgt_interrupt_enable
Function prototype	void wwdgt_interrupt_enable(void);
Function descriptions	enable early wakeup interrupt of WWDGT
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable early wakeup interrupt of WWDGT */

wwdgt_interrupt_enable ();

wwdgt_flag_get

The description of wwdgt_flag_get is shown as below:

Table 3-642. Function wwdgt_flag_get

Function name	wwdgt_flag_get
Function prototype	FlagStatus wwdgt_flag_get(void);
Function descriptions	check early wakeup interrupt state of WWDGT

Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* test if the counter value update has reached the 0x40 */
FlagStatus status;
status = wwdgt_flag_get ();
```

wwdgt_flag_clear

The description of wwdgt_flag_clear is shown as below:

Table 3-643. Function wwdgt_flag_clear

Function name	wwdgt_flag_clear
Function prototype	void wwdgt_flag_clear(void);
Function descriptions	clear early wakeup interrupt state of WWDGT
Precondition	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear early wakeup interrupt state of WWDGT */
wwdgt_flag_clear();
```

4. Revision history

Table 4-1. Revision history

Revison No.	Description	Date
1.0	Initial Release	Nov.20, 2019
1.1	1. Modified the description ***3.26. USART*** chapter. 2. Modified the description ***3.3. CAN*** chapter. 3. Modified the description ***3.14. I2C*** chapter.	Jun.30, 2022
1.2	1．DAC CMP consistency modification	Dec.31, 2023