GigaDevice Semiconductor Inc.

GD32F10x
ARM® Cortex™-M3 32-bit MCU

Firmware Library
User Guide

Revison 1.0

(Mar. 2018)

Table of Contents

List of Figures

List of Tables

1. Introduction

This manual introduces firmware library of GD32F10x 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 GD32F10x 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
BKP	Backup registers
CAN	Controller area network
CRC	CRC calculation unit
DAC	Digital-to-analog converter

Peripherals	Descriptions
DBG	Debug
DMA	Direct memory access controller
ENET	Ethernet
EXMC	External memory controller
EXTI	Interrupt/event controller
FMC	Flash memory controller
FWDGT	Free watchdog timer
GPIO/AFIO	General-purpose and alternate-function I/Os
I2C	Inter-integrated circuit interface
MISC	Nested Vectored Interrupt Controller
PMU	Power management unit
RCU	Reset and clock unit
RTC	Real-time Clock
SDIO	Secure digital input/output interface
SPI/I2S	Serial peripheral interface/Inter-IC sound
TIMER	TIMER
USART	Universal synchronous/asynchronous receiver /transmitter
WWDGT	Window watchdog timer
USBD	Universal Serial Bus full-speed device interface
USBFS	Universal serial bus full-speed interface

1.1.2. Naming rules

The firmware library naming rules are shown as below:

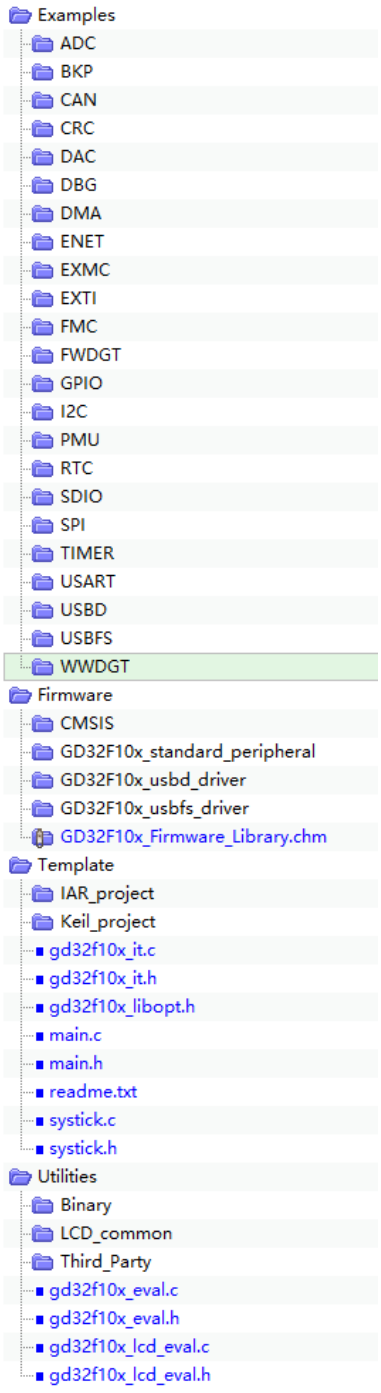
- The peripherals are shortened in XXX format, such as: ADC. More shorten information of peripherals refer to ***Peripherals***;
- The name of sourcefile and header file are started with “gd32f10x_”, such as: gd32f10x_adc.h;
- The constants used only in one file should be defined in the used file; the 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

GD32F10x_Firmware_Library, the file structure is shown as below:

Figure 2-1. File structure of firmware library of GD32F10x



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;
- gd32f10x_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);
- gd32f10x_it.c: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- gd32f10x.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 GD32F10x and system configuration file;
- GD32F10x_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;
- GD32F10x_usbd_driver subfolder includes all the related files about USBD peripheral:
  - Include subfolder includes the header files of USBD peripheral, users need not modify this folder;
  - Source subfolder includes the source files of USBD peripheral, users need not modify this folder;
- GD32F10x_usbfs_driver subfolder includes all the related files about USBFS peripheral:
  - Include subfolder includes the header files of USBFS peripheral, users need not modify this folder;
  - Source subfolder includes the source files of USBFS peripheral, 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.

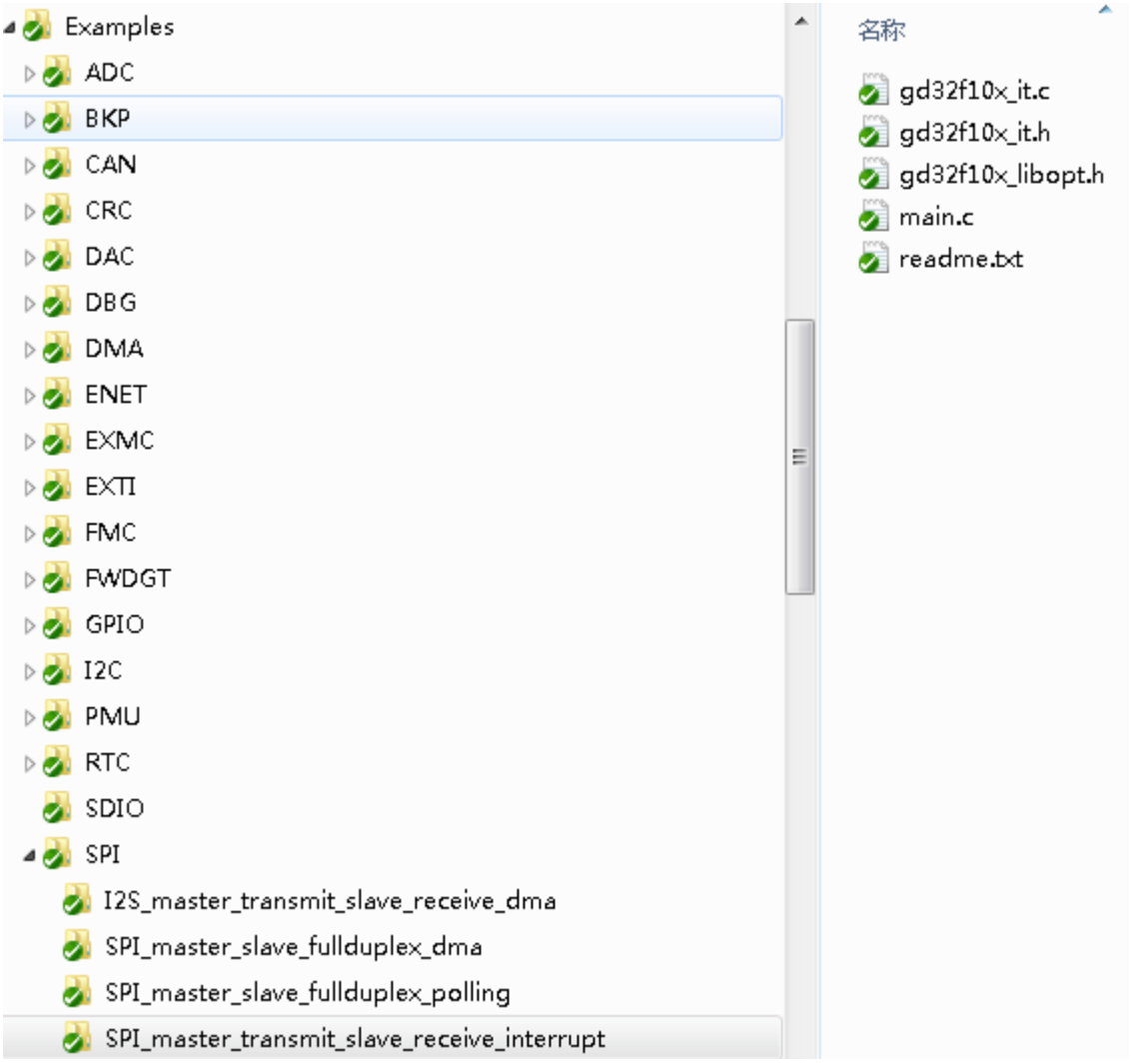
2.1.3. Template Folder

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

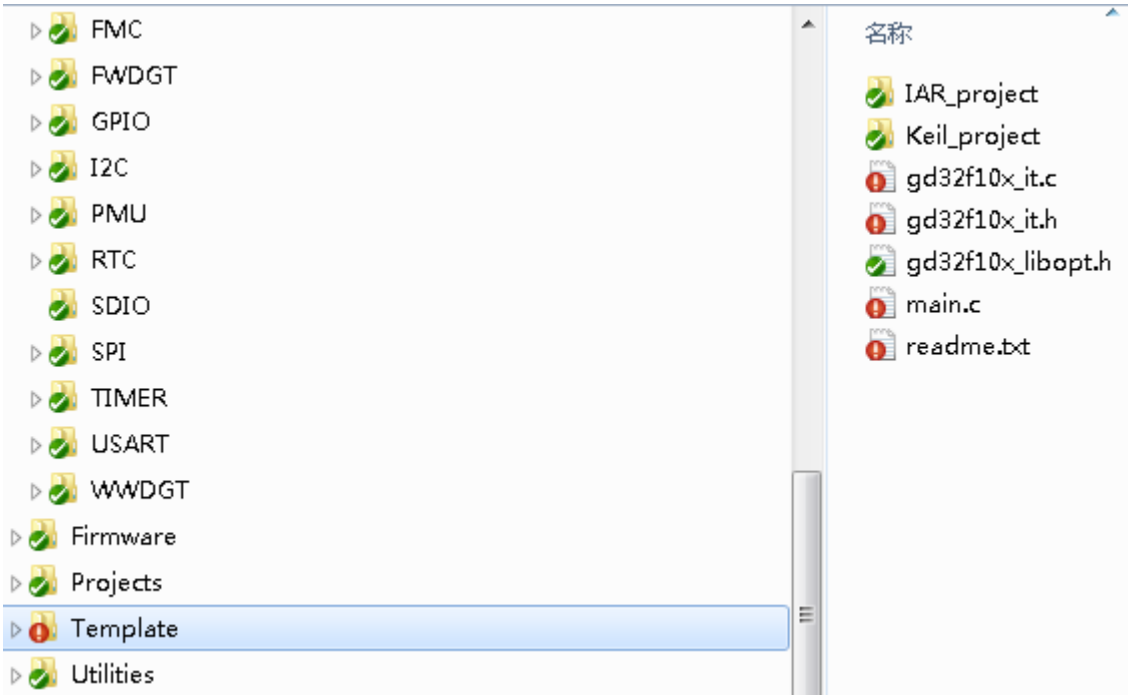


Copy files

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:

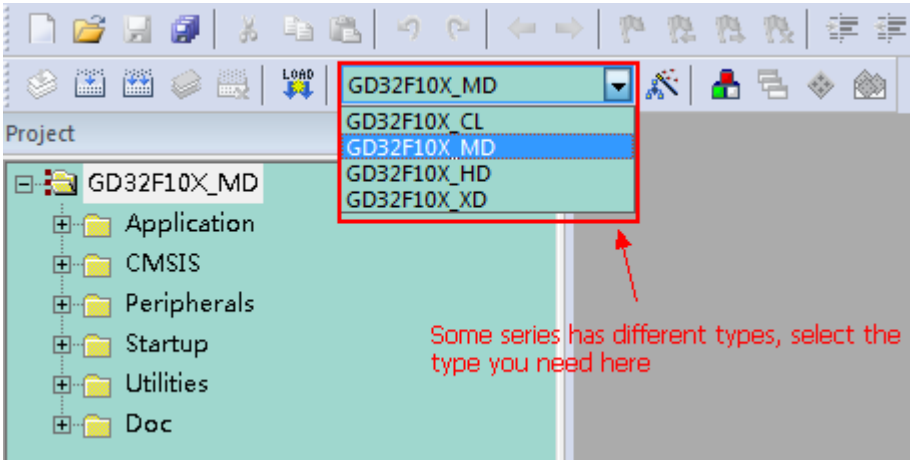
Figure 2-3. Copy the peripheral example files



Open a project

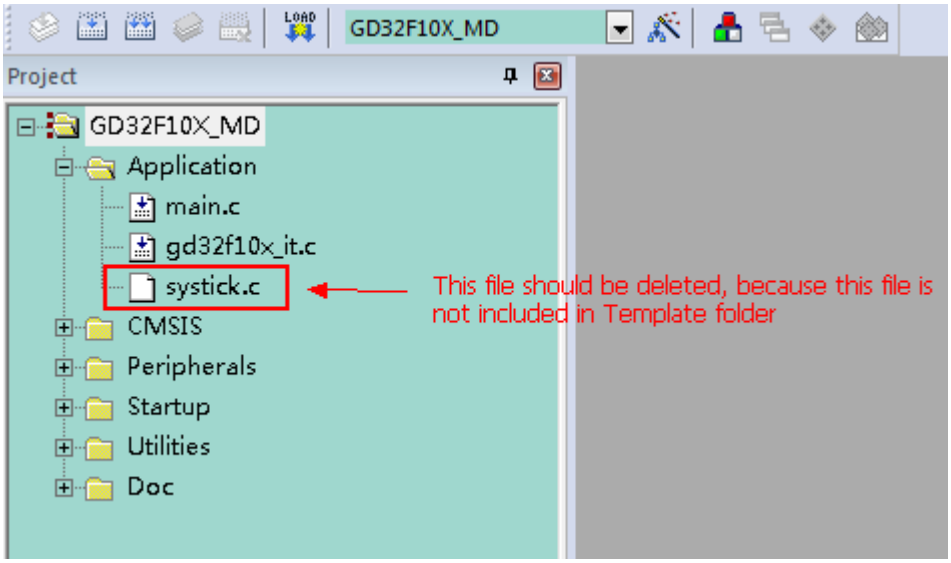
GD provides project in Keil and IAR, users can open project in different IDEs according to their need, such as ”Keil_project”, open \Template\Keil_project\Project.uvproj, shown as below:

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:

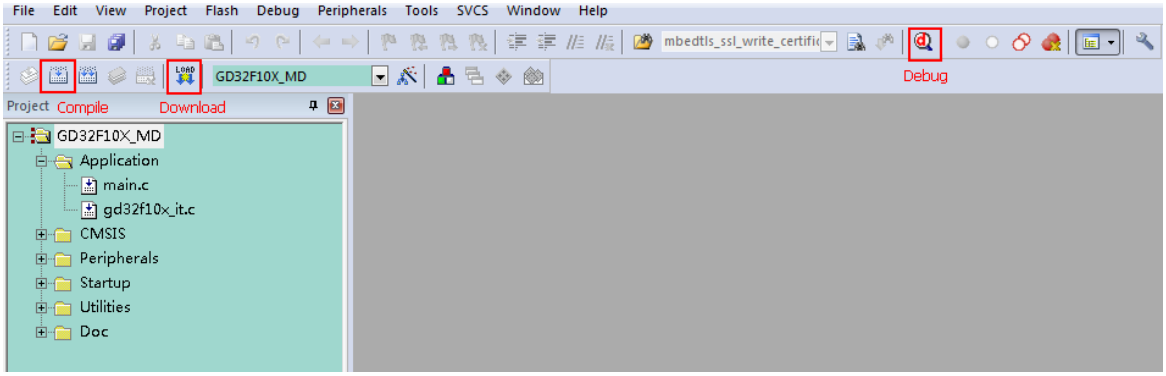
Figure 2-5. Configure project files



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、LCD_Commom and Third_Party subfolders include files for USB tests;
- gd32f10x_eval.h and gd32f10x_lcd_eval.h are related header files of the evaluation board about running the firmware examples;
- gd32f10x_eval.c and gd32f10x_lcd_eval.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
gd32f10x_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.
gd32f10x_it.h	Header file, including all the prototypes of interrupt service routines.
gd32f10x_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.
gd32f10x_xxx.h	The header file of peripheral PPP, including functions about peripheral PPP, and the variables used for functions.
gd32f10x_xxx.c	The C source file for driving peripheral PPP.
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
xxxx	Description of input parameters
Output parameter{out}	
Output parameter name	Description
xxxx	Description of output parameters
Return value	
Return value type	The range of return value

3.2. ADC

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

3.2.1. Descriptions of Peripheral registers

ADC registers are listed in the table shown as below:

Table 3-2. ADC Registers

Registers	Descriptions
ADC_STAT	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 (x=0..3)	Inserted channel data offset register x
ADC_WDHT	Watchdog high threshold register
ADC_WDLT	Watchdog low threshold register
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
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 ADCx peripheral
adc_mode_config	configure the ADC mode
adc_special_function_config	enable or disable ADC special function
adc_data_alignment_config	configure ADC data alignment
adc_enable	enable ADC interface
adc_disable	disable ADC interface
adc_calibration_enable	ADC calibration and reset calibration
adc_tempsensor_vrefint_enable	enable the temperature sensor and Vrefint channel

Function name	Function description
adc_tempsensor_vrefint_disable	disable the temperature sensor and Vrefint channel
adc_dma_mode_enable	enable DMA request
adc_dma_mode_disable	disable DMA request
adc_discontinuous_mode_config	configure ADC discontinuous mode
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_source_config	configure ADC external trigger source
adc_external_trigger_config	enable ADC external trigger
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_sync_mode_convert_value_read	read the last ADC0 and ADC1 conversion result data in sync mode
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_flag_get	get the ADC flag bits
adc_flag_clear	clear the ADC flag bits
adc_regular_software_startconv_flag_get	get the bit state of ADCx software start conversion
adc_inserted_software_startconv_flag_get	get the bit state of ADCx software inserted channel start conversion
adc_interrupt_flag_get	get the ADC interrupt bits

Function name	Function description
adc_interrupt_flag_clear	clear the ADC flag
adc_interrupt_enable	enable ADC interrupt
adc_interrupt_disable	disable ADC interrupt

adc_deinit

The description of adc_deinit is shown as below:

Table 3-4. Function adc_deinit

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

Example:

```
/* reset ADC0 */

adc_deinit(ADC0);
```

adc_mode_config

The description of adc_mode_config is shown as below:

Table 3-5. Function adc_mode_config

Function name	adc_mode_config
Function prototype	void adc_mode_config(uint32_t mode);
Function descriptions	Configure the ADCs sync mode

Precondition	-
The called functions	-
Input parameter{in}	
mode	ADC mode
ADC_MODE_FREE	all the ADC work independently
ADC_DAUL_REGULAL_PARALLEL_INSERTED_PARALLEL	ADC0 and ADC1 work in combined regular parallel + inserted parallel mode
ADC_DAUL_REGULAL_PARALLEL_INSERTED_ROTATION	ADC0 and ADC1 work in combined regular parallel + trigger rotation mode
ADC_DAUL_INSERTED_PARALLEL_REGULAL_FOLLOWUP_FAST	ADC0 and ADC1 work in combined inserted parallel + follow-up fast mode
ADC_DAUL_INSERTED_PARALLEL_REGULAL_FOLLOWUP_SLOW	ADC0 and ADC1 work in combined inserted parallel + follow-up slow mode
ADC_DAUL_INSERTED_PARALLEL	ADC0 and ADC1 work in inserted parallel mode only
ADC_DAUL_REGULAL_PARALLEL	ADC0 and ADC1 work in regular parallel mode only
ADC_DAUL_REGULAL_FOLLOWUP_FAST	ADC0 and ADC1 work in follow-up fast mode only
ADC_DAUL_REGULAL_FOLLOWUP_SLOW	ADC0 and ADC1 work in follow-up slow mode only
ADC_DAUL_INSERTED_TRRIGGER_ROTATION	ADC0 and ADC1 work in trigger rotation mode only
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the ADC sync mode */
adc_mode_config(ADC_MODE_FREE);
```

adc_special_function_config

The description of adc_special_function_config is shown as below:

Table 3-6. Function adc_special_function_config

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

Example:

/* enable ADC0 scan mode */

adc_special_function_config(ADC0, ADC_SCAN_MODE, ENABLE);

adc_data_alignment_config

The description of adc_data_alignment_config is shown as below:

Table 3-7. Function adc_data_alignment_config

Function name	adc_data_alignment_config
Function prototype	void adc_data_alignment_config(uint32_t adc_periph, uint32_t data_alignment);
Function descriptions	Configure ADCx data alignment
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
data_alignment	data alignment select
ADC_DATAALIGN_RIGHT	LSB alignment
ADC_DATAALIGN_LEFT	MSB alignment
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC0 data alignment */

adc_data_alignment_config(ADC0, ADC_DATAALIGN_RIGHT);

adc_enable

The description of adc_enable is shown as below:

Table 3-8. Function adc_enable

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

Example:

```
/* enable ADC0 */

adc_enable(ADC0);
```

adc_disable

The description of adc_disable is shown as below:

Table 3-9. Function adc_disable

Function name	adc_disable
Function prototype	void adc_disable(uint32_t adc_periph);
Function descriptions	Disable ADCx interface
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral

ADCx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

adc_calibration_enable

The description of adc_calibration_enable is shown as below:

Table 3-10. Function adc_calibration_enable

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

Example:

```
/* ADC0 calibration and reset calibration */
adc_calibration_enable(ADC0);
```

adc_tempsensor_vrefint_enable

The description of adc_tempsensor_vrefint_enable is shown as below:

Table 3-11. 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 3-12. 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_dma_mode_enable

The description of adc_dma_mode_enable is shown as below:

Table 3-13. Function adc_dma_mode_enable

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

Example:

/* enable ADC0 DMA request */

adc_dma_mode_enable(ADC0);

adc_dma_mode_disable

The description of adc_dma_mode_disable is shown as below:

Table 3-14. Function adc_dma_mode_disable

Function name	adc_dma_mode_disable
Function prototype	void adc_dma_mode_disable(uint32_t adc_periph);
Function descriptions	Disable ADCx DMA request

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

Example:

```
/* disable ADC0 DMA request */
adc_dma_mode_disable(ADC0);
```

adc_discontinuous_mode_config

The description of adc_discontinuous_mode_config is shown as below:

Table 3-15. Function adc_discontinuous_mode_config

Function name	adc_discontinuous_mode_config
Function prototype	void adc_discontinuous_mode_config(uint32_t adc_periph, uint8_t adc_channel_group, uint8_t length);
Function descriptions	Configure ADC discontinuous mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHA	inserted channel group

NNEL	
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 ADC0 discontinuous mode */
adc_discontinuous_mode_config(ADC0, ADC_REGULAR_CHANNEL, 6);
```

adc_channel_length_config

The description of adc_channel_length_config is shown as below:

Table 3-16. Function adc_channel_length_config

Function name	adc_channel_length_config
Function prototype	void adc_channel_length_config(uint32_t adc_periph, uint8_t adc_channel_group, uint32_t length);
Function descriptions	Configure the length of regular channel group or inserted channel group
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHA	inserted channel group

NNEL	
Input parameter{in}	
length	the length of the channel, regular channel 1-16, inserted channel 1-4
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the length of ADC0 regular channel */

adc_channel_length_config(ADC0, ADC_REGULAR_CHANNEL, 4);

adc_regular_channel_config

The description of adc_regular_channel_config is shown as below:

Table 3-17. Function adc_regular_channel_config

Function name	adc_regular_channel_config
Function prototype	void adc_regular_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
Function descriptions	Configure ADC regular channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
rank	the regular group sequence rank, this parameter must be between 0 to 15
Input parameter{in}	
adc_channel	the selected ADC channel
ADC_CHANNEL_x	ADC Channelx (x=0..17)(x=16 and x=17 are only for ADC0)
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 regular channel */

adc_regular_channel_config(ADC0, 1, ADC_CHANNEL_0, ADC_SAMPLETIME_7POINT5);

adc_inserted_channel_config

The description of adc_inserted_channel_config is shown as below:

Table 3-18. Function adc_inserted_channel_config

Function name	adc_inserted_channel_config
Function prototype	void adc_inserted_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
Function descriptions	Configure ADC inserted channel
Precondition	-

The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
rank	the inserted group sequencer rank, this parameter must be between 0 to 3
Input parameter{in}	
adc_channel	the selected ADC channel
ADC_CHANNEL_x	ADC Channelx (x=0..17)(x=16 and x=17 are only for ADC0)
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 3-19. Function adc_inserted_channel_offset_config

Function name	adc_inserted_channel_offset_config
Function prototype	void adc_inserted_channel_offset_config(uint32_t adc_periph, uint8_t inserted_channel, uint16_t offset);
Function descriptions	Configure ADC inserted channel offset
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
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 ADC0 inserted channel offset */

adc_inserted_channel_offset_config(ADC0, ADC_INSERTED_CHANNEL_0, 100);

adc_external_trigger_source_config

The description of adc_external_trigger_source_config is shown as below:

Table 3-20. Function adc_external_trigger_source_config

Function name	adc_external_trigger_ source_config
Function prototype	void adc_external_trigger_source_config(uint32_t adc_periph, uint8_t adc_channel_group, uint32_t external_trigger_source);
Function descriptions	Configure ADC external trigger source
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
external_trigger_source	regular or inserted group trigger source
ADC0_1_EXTTRIG_REGULAR_T0_CH0	TIMER0 CH0 event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T0_CH1	TIMER0 CH1 event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T0_CH2	TIMER0 CH2 event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T1_CH1	TIMER1 CH1 event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T2_TRGO	TIMER2 TRGO event select for regular channel
ADC0_1_EXTTRIG_RE	TIMER3 CH3 event select for regular channel

GULAR_T3_CH3	
ADC0_1_EXTTRIG_REGULAR_T7_TRGO	TIMER7 TRGO event select for regular channel
ADC0_1_EXTTRIG_REGULAR_EXTI_11	external interrupt line 11 for regular channel
ADC2_EXTTRIG_REGULAR_T2_CH0	TIMER2 CH0 event select for regular channel
ADC2_EXTTRIG_REGULAR_T1_CH2	TIMER1 CH2 event select for regular channel
ADC2_EXTTRIG_REGULAR_T0_CH2	TIMER0 CH2 event select for regular channel
ADC2_EXTTRIG_REGULAR_T7_CH0	TIMER7 CH0 event select for regular channel
ADC2_EXTTRIG_REGULAR_T7_TRGO	TIMER7 TRGO event select for regular channel
ADC2_EXTTRIG_REGULAR_T4_CH0	TIMER4 CH0 event select for regular channel
ADC2_EXTTRIG_REGULAR_T4_CH2	TIMER4 CH2 event select for regular channel
ADC0_1_2_EXTTRIG_REGULAR_NONE	software trigger for regular channel
ADC0_1_EXTTRIG_INSERTED_T0_TRGO	TIMER0 TRGO event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T0_CH3	TIMER0 CH3 event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T1_TRGO	TIMER1 TRGO event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T1_CH0	TIMER1 CH0 event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T2_CH3	TIMER2 CH3 event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T3_TRGO	TIMER3 TRGO event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_EXTI_15	external interrupt line 15 for inserted channel

ADC0_1_EXTTRIG_INSERTED_T7_CH3	TIMER7 CH3 event select for inserted channel
ADC2_EXTTRIG_INSERTED_T0_TRGO	TIMER0 TRGO event select for inserted channel
ADC2_EXTTRIG_INSERTED_T0_CH3	TIMER0 CH3 event select for inserted channel
ADC2_EXTTRIG_INSERTED_T3_CH2	TIMER3 CH2 event select for inserted channel
ADC2_EXTTRIG_INSERTED_T7_CH1	TIMER7 CH1 event select for inserted channel
ADC2_EXTTRIG_INSERTED_T7_CH3	TIMER7 CH3 event select for inserted channel
ADC2_EXTTRIG_INSERTED_T4_TRGO	TIMER4 TRGO event select for inserted channel
ADC2_EXTTRIG_INSERTED_T4_CH3	TIMER4 CH3 event select for inserted channel
ADC0_1_2_EXTTRIG_INSERTED_NONE	software trigger for inserted channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC0 regular channel external trigger source */

adc_external_trigger_source_config(ADC0,ADC_REGULAR_CHANNEL, ADC0_1_EXTTRIG_REGULAR_T0_CH0);

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(uint32_t adc_periph, uint8_t adc_channel_group, ControlStatus newvalue);

Function descriptions	Configure ADC external trigger
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable ADC0 inserted channel group external trigger */

adc_external_trigger_config(ADC0, ADC_INSERTED_CHANNEL_0, ENABLE);

adc_software_trigger_enable

The description of adc_software_trigger_enable is shown as below:

Table 3-22. Function adc_software_trigger_enable

Function name	adc_software_trigger_enable
Function prototype	void adc_software_trigger_enable(uint32_t adc_periph, uint8_t adc_channel_group);

Function descriptions	Enable ADC software trigger
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_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 ADC0 regular channel group software trigger */

adc_software_trigger_enable(ADC0, ADC_REGULAR_CHANNEL);

adc_regular_data_read

The description of adc_regular_data_read is shown as below:

Table 3-23. Function adc_regular_data_read

Function name	adc_regular_data_read
Function prototype	uint16_t adc_regular_data_read(uint32_t adc_periph);
Function descriptions	Read ADC regular group data register
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral

ADCx	x=0,1,2
Output parameter{out}	
-	-
Return value	
uint16_t	ADC conversion value (0-0xFFFF)

Example:

```
/* read ADC0 regular group data register */
uint16_t adc_value = 0;
adc_value = adc_regular_data_read(ADC0);
```

adc_inserted_data_read

The description of adc_inserted_data_read is shown as below:

Table 3-24. Function adc_inserted_data_read

Function name	adc_inserted_data_read
Function prototype	uint16_t adc_inserted_data_read(uint32_t adc_periph, uint8_t inserted_channel);
Function descriptions	Read ADC inserted group data register
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
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 ADC0 inserted group data register */

uint16_t adc_value = 0;

adc_value = adc_inserted_data_read(ADC0, ADC_INSERTED_CHANNEL_0);

adc_sync_mode_convert_value_read

The description of adc_sync_mode_convert_value_read is shown as below:

Table 3-25. Function adc_sync_mode_convert_value_read

Function name	adc_sync_mode_convert_value_read
Function prototype	uint32_t adc_sync_mode_convert_value_read(void);
Function descriptions	Read the last ADC0 and ADC1 conversion result data in sync mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	ADC conversion value (0-0xFFFFFFFF)

Example:

/* read the last ADC0 and ADC1 conversion result data in sync mode */

uint32_t adc_value = 0;

adc_value = adc_sync_mode_convert_value_read();

adc_watchdog_single_channel_enable

The description of adc_watchdog_single_channel_enable is shown as below:

Table 3-26. Function adc_watchdog_single_channel_enable

Function name	adc_watchdog_single_channel_enable
Function prototype	void adc_watchdog_single_channel_enable(uint32_t adc_periph, uint8_t adc_channel);

Function descriptions	Configure ADC analog watchdog single channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_channel	the selected ADC channel
ADC_CHANNEL_x	ADC Channelx(x=0..17) (x=16 and x=17 are only for ADC0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC0 analog watchdog single channel */

adc_watchdog_single_channel_enable(ADC0, ADC_CHANNEL_1);

adc_watchdog_group_channel_enable

The description of adc_watchdog_group_channel_enable is shown as below:

Table 3-27. Function adc_watchdog_group_channel_enable

Function name	adc_watchdog_group_channel_enable
Function prototype	void adc_watchdog_group_channel_enable(uint32_t adc_periph, uint8_t adc_channel_group);
Function descriptions	Configure ADC analog watchdog group channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2

Input parameter{in}	
adc_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 ADC0 analog watchdog group channel */

adc_watchdog_group_channel_enable(ADC0, ADC_REGULAR_CHANNEL);

adc_watchdog_disable

The description of adc_watchdog_disable is shown as below:

Table 3-28. Function adc_watchdog_disable

Function name	adc_watchdog_disable
Function prototype	void adc_watchdog_disable(uint32_t adc_periph);
Function descriptions	Disable ADC analog watchdog
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
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 3-29. Function adc_watchdog_threshold_config

Function name	adc_watchdog_threshold_config
Function prototype	void adc_watchdog_threshold_config(uint32_t adc_periph, uint16_t low_threshold, uint16_t high_threshold);
Function descriptions	Configure ADC analog watchdog threshold
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
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 ADC0 analog watchdog threshold */

adc_watchdog_threshold_config(ADC0, 0x0400, 0x0A00);

adc_flag_get

The description of adc_flag_get is shown as below:

Table 3-30. Function adc_flag_get

Function name	adc_flag_get
Function prototype	FlagStatus adc_flag_get(uint32_t adc_periph , uint32_t adc_flag);
Function descriptions	Get the ADC flag bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_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 ADC0 analog watchdog flag bits*/
FlagStatus flag_value;
flag_value = adc_flag_get(ADC0, ADC_FLAG_WDE);
```

adc_flag_clear

The description of adc_flag_clear is shown as below:

Table 3-31. Function adc_flag_clear

Function name	adc_flag_clear
Function prototype	void adc_flag_clear(uint32_t adc_periph, uint32_t adc_flag);
Function descriptions	Clear the ADC flag bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_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 ADC0 analog watchdog flag bits*/
adc_flag_clear(ADC0, ADC_FLAG_WDE);
```

adc_regular_software_startconv_flag_get

The description of adc_regular_software_startconv_flag_get is shown as below:

Table 3-32. Function adc_regular_software_startconv_flag_get

Function name	adc_regular_software_startconv_flag_get
Function prototype	FlagStatus adc_regular_software_startconv_flag_get(uint32_t adc_periph);

Function descriptions	Get the bit state of ADCx software regular channel start conversion
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the bit state of ADC0 software regular channel start conversion */
FlagStatus flag_value;
flag_value = adc_regular_software_startconv_flag_get(ADC0);
```

adc_inserted_software_startconv_flag_get

The description of adc_inserted_software_startconv_flag_get is shown as below:

Table 3-33. Function adc_inserted_software_startconv_flag_get

Function name	adc_inserted_software_startconv_flag_get
Function prototype	FlagStatus adc_inserted_software_startconv_flag_get(uint32_t adc_periph);
Function descriptions	Get the bit state of ADCx software inserted channel start conversion
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Output parameter{out}	
-	-
Return value	

FlagStatus	SET or RESET

Example:

/* get the bit state of ADC0 software inserted channel start conversion */

FlagStatus flag_value;

flag_value = adc_inserted_software_startconv_flag_get(ADC0);

adc_interrupt_flag_get

The description of adc_interrupt_flag_get is shown as below:

Table 3-34. Function adc_interrupt_flag_get

Function name	adc_interrupt_flag_get
Function prototype	FlagStatus adc_interrupt_flag_get(uint32_t adc_periph, uint32_t adc_interrupt);
Function descriptions	Get the ADC interrupt bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_interrupt	the adc interrupt bits
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	
FlagStatus	SET or RESET

Example:

/* get the ADC0 analog watchdog interrupt bits*/

FlagStatus flag_value;

flag_value = adc_interrupt_flag_get(ADC0, ADC_INT_WDE);

adc_interrupt_flag_clear

The description of adc_interrupt_flag_clear is shown as below:

Table 3-35. Function adc_interrupt_flag_clear

Function name	adc_interrupt_flag_clear
Function prototype	void adc_interrupt_flag_clear(uint32_t adc_periph, uint32_t adc_interrupt);
Function descriptions	Clear the ADC interrupt bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_interrupt	the adc interrupt bits
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:

/* clear the ADC0 analog watchdog interrupt bits*/

adc_interrupt_flag_clear(ADC0, ADC_INT_WDE);

adc_interrupt_enable

The description of adc_interrupt_enable is shown as below:

Table 3-36. Function adc_interrupt _enable

Function name	adc_interrupt_enable

Function prototype	void adc_interrupt_enable(uint32_t adc_periph, uint32_t adc_interrupt);
Function descriptions	Enable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_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 ADC0 analog watchdog interrupt */

adc_interrupt_enable(ADC0, ADC_INT_WDE);

adc_interrupt_disable

The description of adc_interrupt_disable is shown as below:

Table 3-37. Function adc_interrupt_disable

Function name	adc_interrupt_disable
Function prototype	void adc_interrupt_disable(uint32_t adc_periph , uint32_t adc_interrupt);
Function descriptions	Disable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	

adc_periph	ADC peripheral
ADCx	x=0,1,2
Input parameter{in}	
adc_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 ADC0 interrupt */
adc_interrupt_disable(ADC0, ADC_INT_WDE);
```

3.3. BKP

The Backup registers are located in the Backup domain that remains powered-on by V_{BAT} even if V_{DD} power is shut down, they are forty two 16-bit (84 bytes) registers for data protection of user application data, and the wake-up action from Standby mode or system reset do not affect these registers. The BKP registers are listed in chapter ***3.3.1***, the BKP firmware functions are introduced in chapter ***3.3.2***.

3.3.1. Descriptions of Peripheral registers

BKP registers are listed in the table shown as below:

Table 3-38. BKP Registers

Registers	Descriptions
BKP_DATAx (x= 0..41)	Backup data register
BKP_OCTL	RTC signal output control register
BKP_TPCTL	Tamper pin control register

Registers	Descriptions
BKP_TPCS	Tamper control and status register

3.3.2. Descriptions of Peripheral functions

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

Table 3-39. BKP firmware function

Function name	Function description
bkp_deinit	reset data registers
bkp_data_write	write data register
bkp_data_read	read data register
bkp_rtc_calibration_output_enable	enable RTC clock calibration output
bkp_rtc_calibration_output_disable	disable RTC clock calibration output
bkp_rtc_signal_output_enable	enable RTC alarm or second signal output
bkp_rtc_signal_output_disable	disable RTC alarm or second signal output
bkp_rtc_output_select	select RTC output, the RTC output can be select as alarm pulse or second pulse
bkp_rtc_calibration_value_set	set RTC clock calibration value
bkp_tamper_detection_enable	enable tamper detection
bkp_tamper_detection_disable	disable tamper detection
bkp_tamper_active_level_set	set tamper pin active level
bkp_interrupt_enable	enable tamper interrupt
bkp_interrupt_disable	disable tamper interrupt
bkp_flag_get	get bkp flag state
bkp_flag_clear	clear bkp flag state
bkp_interrupt_flag_get	get bkp interrupt flag state
bkp_interrupt_flag_clear	clear bkp interrupt flag state

bkp_deinit

The description of bkp_deinit is shown as below:

Table 3-40. Function bkp_deinit

Function name	bkp_deinit
Function prototype	void bkp_deinit(void);
Function descriptions	reset data registers
Precondition	-
The called functions	rcu_bkp_reset_enable / rcu_bkp_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset BKP registers */

bkp_deinit();
```

bkp_data_write

The description of bkp_data_write is shown as below:

Table 3-41. Function bkp_data_write

Function name	bkp_data_write
Function prototype	void bkp_data_write(bkp_data_register_enum register_number, uint16_t data);
Function descriptions	write data register
Precondition	-
The called functions	-
Input parameter{in}	
register_number	refer to bkp_data_register_enum
BKP_DATA_x(x = 0..41)	bkp data register number x
Input parameter{in}	

data	the data to be write in BKP data register
0-0xffff	data value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write BKP data register */
bkp_data_write(BKP_DATA_0, 0x1226);
```

bkp_data_read

The description of bkp_data_read is shown as below:

Table 3-42. Function bkp_read_data

Function name	bkp_data_read
Function prototype	uint16_t bkp_data_read(bkp_data_register_enum register_number);
Function descriptions	read data register
Precondition	-
The called functions	-
Input parameter{in}	
register_number	refer to bkp_data_register_enum
BKP_DATA_x(x = 0..41)	bkp data register number x
Output parameter{out}	
-	-
Return value	
uint16_t	0-0xffff

Example:

```
/* read BKP data register */
uint16_t data;
data = bkp_data_read(BKP_DATA_0);
```

bkp_rtc_calibration_output_enable

The description of bkp_rtc_calibration_output_enable is shown as below:

Table 3-43. Function bkp_rtc_calibration_output_enable

Function name	bkp_rtc_calibration_output_enable
Function prototype	void bkp_rtc_calibration_output_enable(void);
Function descriptions	enable RTC clock calibration output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RTC clock calibration output */

bkp_rtc_calibration_output_enable();
```

bkp_rtc_calibration_output_disable

The description of bkp_rtc_calibration_output_disable is shown as below:

Table 3-44. Function bkp_rtc_calibration_output_disable

Function name	bkp_rtc_calibration_output_disable
Function prototype	void bkp_rtc_calibration_output_disable(void);
Function descriptions	disable RTC clock calibration output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* disable RTC clock calibration output */

bkp_rtc_calibration_output_disable();

bkp_rtc_signal_output_enable

The description of bkp_rtc_signal_output_enable is shown as below:

Table 3-45. Function bkp_rtc_signal_output_enable

Function name	bkp_rtc_signal_output_enable
Function prototype	void bkp_rtc_signal_output_enable (void);
Function descriptions	enable RTC alarm or second signal output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable RTC alarm or second signal output */

bkp_rtc_signal_output_enable();

bkp_rtc_signal_output_disable

The description of bkp_rtc_signal_output_disable is shown as below:

Table 3-46. Function bkp_rtc_signal_output_disable

Function name	bkp_rtc_signal_output_disable
Function prototype	void bkp_rtc_signal_output_disable (void);

Function descriptions	disable RTC alarm or second signal output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC alarm or second signal output */
bkp_rtc_signal_output_disable();
```

bkp_rtc_output_select

The description of bkp_rtc_output_select is shown as below:

Table 3-47. Function bkp_rtc_output_select

Function name	bkp_rtc_output_select
Function prototype	void bkp_rtc_output_select (uint16_t outputsel);
Function descriptions	select RTC output, the RTC output can be select as alarm pulse or second pulse
Precondition	-
The called functions	-
Input parameter{in}	
outputsel	RTC output selection
RTC_OUTPUT_ALARM_PULSE	RTC alarm pulse is selected as the RTC output
RTC_OUTPUT_SECOND_PULSE	RTC second pulse is selected as the RTC output
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* select RTC output alarm signal output */

bkp_rtc_output_select(RTC_OUTPUT_ALARM_PULSE);

bkp_rtc_calibration_value_set

The description of bkp_rtc_calibration_value_set is shown as below:

Table 3-48. Function bkp_rtc_calibration_value_set

Function name	bkp_rtc_calibration_value_set
Function prototype	void bkp_rtc_calibration_value_set(uint8_t value);
Function descriptions	set RTC clock calibration value
Precondition	-
The called functions	-
Input parameter{in}	
value	RTC clock calibration value
0x00 - 0x7F	value
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set RTC clock calibration value */

bkp_rtc_calibration_value_set(0x7f);

bkp_tamper_detection_enable

The description of bkp_tamper_detection_enable is shown as below:

Table 3-49. Function bkp_tamper_detection_enable

Function name	bkp_tamper_detection_enable
Function prototype	void bkp_tamper_detection_enable (void);

Function descriptions	enable tamper detection
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable tamper pin detection */
bkp_tamper_detection_enable();
```

bkp_tamper_detection_disable

The description of bkp_tamper_detection_disable is shown as below:

Table 3-50. Function bkp_tamper_detection_disable

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

Example:

```
/* disable tamper pin detection */
```

bkp_tamper_detection_disable();

bkp_tamper_active_level_set

The description of bkp_tamper_active_level_set is shown as below:

Table 3-51. Function bkp_tamper_active_level_set

Function name	bkp_tamper_active_level_set
Function prototype	void bkp_tamper_active_level_set (uint16_t level);
Function descriptions	set tamper pin active level
Precondition	-
The called functions	-
Input parameter{in}	
level	tamper pin active level
TAMPER_PIN_ACTIVE_HIGH	the tamper pin is active high
TAMPER_PIN_ACTIVE_LOW	the tamper pin is active low
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set tamper pin active level high */

bkp_tamper_active_level_set(TAMPER_PIN_ACTIVE_HIGH);

bkp_interrupt_enable

The description of bkp_interrupt_enable is shown as below:

Table 3-52. Function bkp_interrupt_enable

Function name	bkp_interrupt_enable
Function prototype	void bkp_interrupt_enable (void);
Function descriptions	enable tamper interrupt
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable tamper pin interrupt */
bkp_interrupt_enable();
```

bkp_interrupt_disable

The description of bkp_interrupt_disable is shown as below:

Table 3-53. Function bkp_interrupt_disable

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

Example:

```
/* disable tamper pin interrupt */
bkp_interrupt_disable();
```

bkp_flag_get

The description of bkp_flag_get is shown as below:

Table 3-54. Function bkp_flag_get

Function name	bkp_flag_get
Function prototype	FlagStatus bkp_flag_get(uint16_t flag);
Function descriptions	get bkp flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	BKP flag
BKP_FLAG_TAMPER	tamper event flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get BKP flag state */
FlagStatus status;
status = bkp_flag_get(BKP_FLAG_TAMPER);
```

bkp_flag_clear

The description of bkp_flag_clear is shown as below:

Table 3-55. Function bkp_flag_clear

Function name	bkp_flag_clear
Function prototype	void bkp_flag_clear(uint16_t flag);
Function descriptions	clear bkp flag state
Precondition	-
The called functions	-
Input parameter{in}	

flag	BKP flag
BKP_FLAG_TAMPER	tamper event flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear BKP flag state */
bkp_flag_clear(BKP_FLAG_TAMPER);
```

bkp_interrupt_flag_get

The description of bkp_interrupt_flag_get is shown as below:

Table 3-56. Function bkp_interrupt_flag_get

Function name	bkp_interrupt_flag_get
Function prototype	FlagStatus bkp_interrupt_flag_get(uint16_t flag);
Function descriptions	get bkp interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	BKP interrupt flag
BKP_INT_FLAG_TAMPER	tamper interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get BKP interrupt flag state */
bkp_interrupt_flag_get(BKP_FLAG_TAMPER);
```

bkp_interrupt_flag_clear

The description of bkp_interrupt_flag_clear is shown as below:

Table 3-57. Function bkp_interrupt_flag_clear

Function name	bkp_interrupt_flag_clear
Function prototype	void bkp_interrupt_flag_ clear(uint16_t flag);
Function descriptions	clear bkp interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	BKP interrupt flag
BKP_INT_FLAG_TAMPER	tamper interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear BKP interrupt flag state */
bkp_interrupt_flag_clear(BKP_INT_FLAG_TAMPER);
```

3.4. CAN

CAN bus (for Controller Area Network) is a bus standard designed to allow microcontrollers and devices to communicate with each other without a host computer. The CAN registers are listed in chapter ***3.4.1***, the CAN firmware functions are introduced in chapter ***3.4.2***.

3.4.1. Descriptions of Peripheral registers

CAN registers are listed in the table shown as below:

Table 3-58. CAN Registers

Registers	Descriptions
CAN_CTL	Control register

Registers	Descriptions
CAN_STAT	Status register
CAN_TSTAT	Transmit status register
CAN_RFIFO0	Receive message FIFO0 register
CAN_RFIFO1	Receive message FIFO1 register
CAN_INTEN	Interrupt enable register
CAN_ERR	Error register
CAN_BT	Bit timing register
CAN_TMIx	Transmit mailbox identifier register
CAN_TMPx	Transmit mailbox property register
CAN_TMDATA0x	Transmit mailbox data0 register
CAN_TMDATA1x	Transmit mailbox data1 register
CAN_RFIFOMIx	Receive FIFO mailbox identifier register
CAN_RFIFOMPx	Receive FIFO mailbox property register
CAN_RFIFOMDATA0x	Receive FIFO mailbox data0 register
CAN_RFIFOMDATA1x	Receive FIFO mailbox data1 register
CAN_FCTL	Filter control register
CAN_FMCFG	Filter mode configuration register
CAN_FSCFG	Filter scale configuration register
CAN_FAFIFO	Filter associated FIFO registe
CAN_FW	Filter working register
CAN_FxDATAy	Filter x data y register

3.4.2. Descriptions of Peripheral functions

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

Table 3-59. CAN firmware function

Function name	Function description
can_deinit	deinitialize CAN
can_struct_para_init	initialize CAN struct
can_init	initialize CAN
can_filter_init	CAN filter init
can1_filter_start_bank	set can1 fliter start bank number
can_debug_freeze_enable	CAN debug freeze enable
can_debug_freeze_disable	CAN debug freeze disable
can_time_trigger_mode_enable	CAN time triggle mode enable
can_time_trigger_mode_disable	CAN time triggle mode disable
can_message_transmit	transmit CAN message
can_transmit_states	get CAN transmit state
can_transmission_stop	stop CAN transmission
can_message_receive	CAN receive message
can_fifo_release	CAN release fifo
can_receive_message_length_get	CAN receive message length
can_working_mode_set	CAN working mode
can_wakeup	CAN wakeup from sleep mode
can_error_get	CAN get error
can_receive_error_number_get	get CAN receive error number
can_transmit_error_number_get	get CAN transmit error number
can_interrupt_enable	CAN interrupt enable
can_interrupt_disable	CAN interrupt disable
can_flag_get	CAN get flag state
can_flag_clear	CAN clear flag state
can_interrupt_flag_get	CAN get interrupt flag state
can_interrupt_flag_clear	CAN clear interrupt flag state

Structure can_parameter_struct

Table 3-60. can_parameter_struct

Member name	Function description
working_mode	CAN working mode
resync_jump_width	CAN resynchronization jump width
time_segment_1	time segment 1
time_segment_2	time segment 2
time_triggered	time triggered communication mode
auto_bus_off_recovery	automatic bus-off recovery
auto_wake_up	automatic wake-up mode
auto_retrans	automatic retransmission mode
rec_fifo_overwrite	receive FIFO overwrite mode
trans_fifo_order	transmit FIFO order
prescaler	baudrate prescaler

Structure can_trasnmit_message_struct

Table 3-61. can_trasnmit_message_struct

Member name	Function description
tx_sfid	standard format frame identifier
tx_efid	extended format frame identifier
tx_ff	format of frame, standard or extended format
tx_ft	type of frame, data or remote
tx_dlen	data length
tx_data[8]	transmit data

Structure can_receive_message_struct

Table 3-62. can_receive_message_struct

Member name	Function description
rx_sfid	standard format frame identifier

rx_efid	extended format frame identifier
rx_ff	format of frame, standard or extended format
rx_ft	type of frame, data or remote
rx_dlen	data length
rx_data[8]	receive data
rx_fi	filtering index

Structure can_filter_parameter_struct

Table 3-63. can_filter_parameter_struct

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

can_deinit

The description of can_deinit is shown as below:

Table 3-64. Function can_deinit

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

CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

can_ struct_para_init

The description of can_struct_para_init is shown as below:

Table 3-65. Function can_struct_para_init

Function name	can_struct_para_init
Function prototype	void can_struct_para_init(can_struct_type_enum type, void* p_struct)
Function descriptions	initialize CAN parameter struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
type	the type of CAN parameter struct
CAN_INIT_STRUCT	CAN initiliaze parameters struct
CAN_FILTER_STRUCT	CAN filter parameters struct
CAN_TX_MESSAGE_STRUCT	CAN transmit message struct
CAN_RX_MESSAGE_STRUCT	CAN receive message struct
Input parameter{in}	
p_struct	the pointer of the specific struct
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* CAN parameter initialize*/

can_init(CAN_INIT_STRUCT, &can_parameter);

can_init

The description of can_init is shown as below:

Table 3-66. Function can_init

Function name	can_init
Function prototype	ErrStatus can_init(uint32_t can_periph, can_parameter_struct* can_parameter_init);
Function descriptions	initialize CAN
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
can_parameter_init	CAN parameter initialization stuct, the structure members can refer to members of the structure ***Table 3-60. can_parameter_struct***
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS / ERROR

Example:

/* CAN0 initialize*/

can_init(CAN0, &can_parameter);

can_filter_init

The description of can_filter_init is shown as below:

Table 3-67. Function can_filter_init

Function name	can_filter_init
Function prototype	void can_filter_init(can_filter_parameter_struct* can_filter_parameter_init);
Function descriptions	initialize CAN filter
Precondition	-
The called functions	-
Input parameter{in}	
can_filter_parameter_init	CAN filter initialization stuct, the structure members can refer to members of the structure ***Table 3-63. can_filter_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize CAN filter */
can_filter_init(&can_filter);
```

can1_filter_start_bank

The description of can1_filter_start_bank is shown as below:

Table 3-68. Function can1_filter_start_bank

Function name	can1_filter_start_bank
Function prototype	void can1_filter_start_bank(uint8_t start_bank);
Function descriptions	set CAN1 fliter start bank number
Precondition	-
The called functions	-
Input parameter{in}	
start_bank	CAN1 start bank number
1..27	start number
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* set CAN1 fliter start bank number 15*/

can1_filter_start_bank(15);

can_debug_freeze_enable

The description of can_debug_freeze_enable is shown as below:

Table 3-69. Function can_debug_freeze_enable

Function name	can_debug_freeze_enable
Function prototype	void can_debug_freeze_enable(uint32_t can_periph);
Function descriptions	enable CAN debug freeze
Precondition	-
The called functions	dbg_periph_enable
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable CAN0 debug freeze */

can_debug_freeze_enable(CAN0);

can_debug_freeze_disable

The description of can_debug_freeze_disable is shown as below:

Table 3-70. Function can_debug_freeze_disable

Function name	can_debug_freeze_disable
Function prototype	void can_debug_freeze_disable(uint32_t can_periph);

Function descriptions	disable CAN debug freeze
Precondition	-
The called functions	dbg_periph_disable
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CAN0 debug freeze */
can_debug_freeze_disable(CAN0);
```

can_time_trigger_mode_enable

The description of can_time_trigger_mode_enable is shown as below:

Table 3-71. Function can_time_trigger_mode_enable

Function name	can_time_trigger_mode_enable
Function prototype	void can_time_trigger_mode_enable(uint32_t can_periph);
Function descriptions	enable CAN time trigger mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable CAN0 time trigger mode */

can_time_trigger_mode_enable(CAN0);

can_time_trigger_mode_disable

The description of can_time_trigger_mode_disable is shown as below:

Table 3-72. Function can_time_trigger_mode_disable

Function name	can_time_trigger_mode_disable
Function prototype	void can_time_trigger_mode_disable(uint32_t can_periph);
Function descriptions	disable CAN time trigger mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable CAN0 time trigger mode */

can_time_trigger_mode_disable(CAN0);

can_message_transmit

The description of can_message_transmit is shown as below:

Table 3-73. Function can_message_transmit

Function name	can_message_transmit
Function prototype	uint8_t can_message_transmit(uint32_t can_periph, can_trasnmit_message_struct* transmit_message);
Function descriptions	transmit CAN message

Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
transmit_message	CAN transmit message stuct, the structure members can refer to members of the structure ***Table 3-61. can_trasnmit_message_struct***
Output parameter{out}	
-	-
Return value	
uint8_t	0x00-0x03

Example:

```
/* CAN0 transmit message and return the mailbox number*/
uint8_t transmit_mailbox = 0;
transmit_mailbox = can_message_transmit(CAN0, &transmit_message);
```

can_transmit_states

The description of can_transmit_states is shown as below:

Table 3-74. Function can_transmit_states

Function name	can_transmit_states
Function prototype	can_transmit_state_enum can_transmit_states(uint32_t can_periph, uint8_t mailbox_number);
Function descriptions	get CAN transmit state
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	

mailbox_number	Mailbox number
CAN_MAILBOXx	CAN_MAILBOXx(x=0,1,2)
Output parameter{out}	
-	-
Return value	
can_transmit_state_enum	0..4

Example:

```
/* CAN0 mailbox0 transmit state */
uint8_t transmit_state = 0;
transmit_state = can_transmit_states(CAN0, CAN_MAILBOX0);
```

can_transmission_stop

The description of can_transmission_stopt is shown as below:

Table 3-75. Function can_transmission_stop

Function name	can_transmission_stop
Function prototype	void can_transmission_stop(uint32_t can_periph, uint8_t mailbox_number);
Function descriptions	stop CAN transmission
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
mailbox_number	Mailbox number
CAN_MAILBOXx	CAN_MAILBOXx(x=0,1,2)
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* stop CAN0 mailbox0 transmission */

can_transmission_stop(CAN0, CAN_MAILBOX0);

can_message_receive

The description of can_message_receive is shown as below:

Table 3-76. Function can_message_receive

Function name	can_message_receive
Function prototype	void can_message_receive(uint32_t can_periph, uint8_t fifo_number, can_receive_message_struct* receive_message);
Function descriptions	CAN receive message
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
fifo_number	Fifo number
CAN_FIFOx	CAN_FIFOx(x=0,1)
Input parameter{in}	
receive_message	CAN message receive stuct, the structure members can refer to members of the structure ***Table 3-62. can_receive_message_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* CAN0 FIFO0 receive message */

can_message_receive(CAN0, CAN_FIFO0, &receive_message);

can_fifo_release

The description of can_fifo_release is shown as below:

Table 3-77. Function can_fifo_release

Function name	can_fifo_release
Function prototype	void can_fifo_release(uint32_t can_periph, uint8_t fifo_number);
Function descriptions	release FIFO0
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
fifo_number	Fifo number
CAN_FIFOx	CAN_FIFOx(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* CAN0 release FIFO0*/

can_fifo_release(CAN0, CAN_FIFO0);
```

can_receive_message_length_get

The description of can_receive_message_length_get is shown as below:

Table 3-78. Function can_receive_message_length_get

Function name	can_receive_message_length_get
Function prototype	uint8_t can_receive_message_length_get(uint32_t can_periph, uint8_t fifo_number);
Function descriptions	CAN receive message length
Precondition	-

The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
fifo_number	Fifo number
CAN_FIFOx	CAN_FIFOx(x=0,1)
Output parameter{out}	
-	-
Return value	
uint8_t	0..3

Example:

```
/* CAN0 FIFO0 receive message length */
uint8_t frame_number = 0;
frame_number = can_receive_message_length_get(CAN0, CAN_FIFO0);
```

can_working_mode_set

The description of can_working_mode_set is shown as below:

Table 3-79. Function can_working_mode_set

Function name	can_working_mode_set
Function prototype	ErrStatus can_working_mode_set(uint32_t can_periph, uint8_t working_mode);
Function descriptions	set CAN working mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	

can_working_mode	Mode select
CAN_MODE_INITIALIZE	Initialize mode
CAN_MODE_NORMAL	Normal mode
CAN_MODE_SLEEP	Sleep mode
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS / ERROR

Example:

/* set CAN0 working at initialize mode */

can_working_mode_set(CAN0, CAN_MODE_INITIALIZE);

can_wakeup

The description of can_wakeup is shown as below:

Table 3-80. Function can_wakeup

Function name	can_wakeup
Function prototype	ErrStatus can_wakeup(uint32_t can_periph);
Function descriptions	wake up CAN
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS / ERROR

Example:

/* wake up CAN0 */

can_wakeup (CAN0);

can_error_get

The description of can_error_get is shown as below:

Table 3-81. Function can_error_get

Function name	can_error_get
Function prototype	can_error_enum can_error_get(uint32_t can_periph);
Function descriptions	get CAN error type
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Output parameter{out}	
-	-
Return value	
can_error_enum	0..7

Example:

/* get CAN0 error type */

can_error_enum err_type;

err_type = can_error_get(CAN0);

can_receive_error_number_get

The description of can_receive_error_number_get is shown as below:

Table 3-82. Function can_receive_error_number_get

Function name	can_receive_error_number_get
Function prototype	uint8_t can_receive_error_number_get(uint32_t can_periph);
Function descriptions	get CAN receive error number
Precondition	-

The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Output parameter{out}	
-	-
Return value	
uint8_t	0..255

Example:

```
/* get CAN0 receive error number */
uint8_t error_num;
error_num = can_receive_error_number_get(CAN0);
```

can_transmit_error_number_get

The description of can_transmit_error_number_get is shown as below:

Table 3-83. Function can_transmit_error_number_get

Function name	can_transmit_error_number_get
Function prototype	uint8_t can_transmit_error_number_get(uint32_t can_periph);
Function descriptions	get CAN transmit error number
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Output parameter{out}	
-	-
Return value	
uint8_t	0..255

Example:

```
/* get CAN0 transmit error number */

uint8_t error_num;

error_num = can_transmit_error_number_get(CAN0);
```

can_interrupt_enable

The description of can_interrupt_enable is shown as below:

Table 3-84. Function can_interrupt_enable

Function name	can_interrupt_enable
Function prototype	void can_interrupt_enable(uint32_t can_periph, uint32_t interrupt);
Function descriptions	enable CAN interrupt
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
interrupt	Interrupt type
CAN_INT_TME	transmit mailbox empty interrupt enable
CAN_INT_RFNE0	receive FIFO0 not empty interrupt enable
CAN_INT_RFF0	receive FIFO0 full interrupt enable
CAN_INT_RFO0	receive FIFO0 overfull interrupt enable
CAN_INT_RFNE1	receive FIFO1 not empty interrupt enable
CAN_INT_RFF1	receive FIFO1 full interrupt enable
CAN_INT_RFO1	receive FIFO1 overfull interrupt enable
CAN_INT_WERR	warning error interrupt enable
CAN_INT_PERR	passive error interrupt enable
CAN_INT_BO	bus-off interrupt enable
CAN_INT_ERRN	error number interrupt enable
CAN_INT_ERR	error interrupt enable

CAN_INT_WU	wakeup interrupt enable
CAN_INT_SLPW	sleep working interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* CAN0 transmit mailbox empty interrupt enable */
can_interrupt_enable(CAN0, CAN_INT_TME);
```

can_interrupt_disable

The description of can_interrupt_disable is shown as below:

Table 3-85. Function can_interrupt_disable

Function name	can_interrupt_disable
Function prototype	void can_interrupt_disable(uint32_t can_periph, uint32_t interrupt);
Function descriptions	disable CAN interrupt
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
interrupt	Interrupt type
CAN_INT_TME	transmit mailbox empty interrupt enable
CAN_INT_RFNE0	receive FIFO0 not empty interrupt enable
CAN_INT_RFF0	receive FIFO0 full interrupt enable
CAN_INT_RFO0	receive FIFO0 overfull interrupt enable
CAN_INT_RFNE1	receive FIFO1 not empty interrupt enable
CAN_INT_RFF1	receive FIFO1 full interrupt enable

CAN_INT_RFO1	receive FIFO1 overfull interrupt enable
CAN_INT_WERR	warning error interrupt enable
CAN_INT_PERR	passive error interrupt enable
CAN_INT_BO	bus-off interrupt enable
CAN_INT_ERRN	error number interrupt enable
CAN_INT_ERR	error interrupt enable
CAN_INT_WU	wakeup interrupt enable
CAN_INT_SLPW	sleep working interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* CAN0 transmit mailbox empty interrupt disable */

can_interrupt_disable(CAN0, CAN_INT_TME);

can_flag_get

The description of can_flag_get is shown as below:

Table 3-86. Function can_flag_get

Function name	can_flag_get
Function prototype	FlagStatus can_flag_get(uint32_t can_periph, can_flag_enum flag);
Function descriptions	get CAN flag state
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
flag	CAN flags

CAN_FLAG_MTE2	mailbox 2 transmit error
CAN_FLAG_MTE1	mailbox 1 transmit error
CAN_FLAG_MTE0	mailbox 0 transmit error
CAN_FLAG_MTF2	mailbox 2 transmit finished
CAN_FLAG_MTF1	mailbox 1 transmit finished
CAN_FLAG_MTF0	mailbox 0 transmit finished
CAN_FLAG_RFO0	receive FIFO0 overfull
CAN_FLAG_RFF0	receive FIFO0 full
CAN_FLAG_RFO1	receive FIFO1 overfull
CAN_FLAG_RFF1	receive FIFO1 full
CAN_FLAG_BOERR	bus-off error
CAN_FLAG_PERR	passive error
CAN_FLAG_WERR	warning error
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

/* get CAN0 mailbox 0 transmit finished flag */

can_flag_get(CAN0, CAN_FLAG_MTF0);

can_flag_clear

The description of can_flag_clear is shown as below:

Table 3-87. Function can_flag_clear

Function name	can_flag_clear
Function prototype	void can_flag_clear(uint32_t can_periph, can_flag_enum flag);
Function descriptions	clear CAN flag state
Precondition	-
The called functions	-

Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
flag	CAN flags
CAN_FLAG_MTE2	mailbox 2 transmit error
CAN_FLAG_MTE1	mailbox 1 transmit error
CAN_FLAG_MTE0	mailbox 0 transmit error
CAN_FLAG_MTF2	mailbox 2 transmit finished
CAN_FLAG_MTF1	mailbox 1 transmit finished
CAN_FLAG_MTF0	mailbox 0 transmit finished
CAN_FLAG_RFO0	receive FIFO0 overfull
CAN_FLAG_RFF0	receive FIFO0 full
CAN_FLAG_RFO1	receive FIFO1 overfull
CAN_FLAG_RFF1	receive FIFO1 full
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear CAN0 mailbox 0 transmit error flag*/
can_flag_clear(CAN0, CAN_FLAG_MTE0);
```

can_interrupt_flag_get

The description of can_interrupt_flag_get is shown as below:

Table 3-88. Function can_interrupt_flag_get

Function name	can_interrupt_flag_get
Function prototype	FlagStatus can_interrupt_flag_get(uint32_t can_periph, can_interrupt_flag_enum flag);

Function descriptions	get CAN interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
flag	CAN interrupt flags
CAN_INT_FLAG_SLPIF	status change interrupt flag of sleep working mode entering
CAN_INT_FLAG_WUIF	status change interrupt flag of wakeup from sleep working mode
CAN_INT_FLAG_ERRIF	error interrupt flag
CAN_INT_FLAG_MTF2	mailbox 2 transmit finished interrupt flag
CAN_INT_FLAG_MTF1	mailbox 1 transmit finished interrupt flag
CAN_INT_FLAG_MTF0	mailbox 0 transmit finished interrupt flag
CAN_INT_FLAG_RFO0	receive FIFO0 overfull interrupt flag
CAN_INT_FLAG_RFF0	receive FIFO0 full interrupt flag
CAN_INT_FLAG_RFO1	receive FIFO1 overfull interrupt flag
CAN_INT_FLAG_RFF1	receive FIFO1 full interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get CAN0 mailbox 0 transmit finished interrupt flag */
FlagStatus Status;
Status = can_interrupt_flag_get(CAN0, CAN_INT_FLAG_MTF0);
```

can_interrupt_flag_clear

The description of can_interrupt_flag_clear is shown as below:

Table 3-89. Function can_interrupt_flag_clear

Function name	can_interrupt_flag_clear
Function prototype	void can_interrupt_flag_clear(uint32_t can_periph, can_interrupt_flag_enum flag);
Function descriptions	clear CAN interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection, the CAN1 only for GD32F10x_CL
Input parameter{in}	
flag	CAN interrupt flags
CAN_INT_FLAG_SLPIF	status change interrupt flag of sleep working mode entering
CAN_INT_FLAG_WUIF	status change interrupt flag of wakeup from sleep working mode
CAN_INT_FLAG_ERRIF	error interrupt flag
CAN_INT_FLAG_MTF2	mailbox 2 transmit finished interrupt flag
CAN_INT_FLAG_MTF1	mailbox 1 transmit finished interrupt flag
CAN_INT_FLAG_MTF0	mailbox 0 transmit finished interrupt flag
CAN_INT_FLAG_RFO0	receive FIFO0 overfull interrupt flag
CAN_INT_FLAG_RFF0	receive FIFO0 full interrupt flag
CAN_INT_FLAG_RFO1	receive FIFO1 overfull interrupt flag
CAN_INT_FLAG_RFF1	receive FIFO1 full interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear CAN0 mailbox 0 transmit finished interrupt flag */

can_interrupt_flag_clear (CAN0, CAN_INT_FLAG_MTF0);
```

3.5. CRC

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. The CRC registers are listed in chapter ***3.5.1***, the CRC firmware functions are introduced in chapter ***3.5.2***.

3.5.1. Descriptions of Peripheral registers

CRC registers are listed in the table shown as below:

Table 3-90. CRC Registers

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

3.5.2. Descriptions of Peripheral functions

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

Table 3-91. CRC firmware function

Function name	Function description
crc_deinit	deinit CRC calculation unit
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_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 3-92. 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_data_register_reset

The description of crc_data_register_reset is shown as below:

Table 3-93. 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 3-94. 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 3-95. 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 3-96. 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_single_data_calculate

The description of crc_single_data_calculate is shown as below:

Table 3-97. 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 3-98. 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	CRC calculate a 32-bit data array
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. Descriptions of Peripheral registers

DAC registers are listed in the table shown as below:

Table 3-99. DAC Registers

Registers	Descriptions
DAC_CTL	DAC control register
DAC_SWT	DAC software trigger register
DAC0_R12DH	DAC0 12-bit right-aligned data holding register

Registers	Descriptions
DAC0_L12DH	DAC0 12-bit left-aligned data holding register
DAC0_R8DH	DAC0 8-bit right-aligned data holding register
DAC1_R12DH	DAC1 12-bit right-aligned data holding register
DAC1_L12DH	DAC1 12-bit left-aligned data holding register
DAC1_R8DH	DAC1 8-bit right-aligned data holding register
DACC_R12DH	DAC concurrent mode 12-bit right-aligned data holding register
DACC_L12DH	DAC concurrent mode 12-bit left-aligned data holding register
DACC_R8DH	DAC concurrent mode 8-bit right-aligned data holding register
DAC0_DO	DAC0 data output register
DAC1_DO	DAC1 data output register

3.6.2. Descriptions of Peripheral functions

DAC registers are listed in the table shown as below:

Table 3-100. DAC firmware function

Function name	Function description
dac_deinit	deinitialize DAC
dac_enable	enable DAC
dac_disable	disable DAC
dac_dma_enable	dac_dma_enable
dac_dma_disable	dac_dma_disable
dac_output_buffer_enable	enable DAC output buffer
dac_output_buffer_disable	disable DAC output buffer
dac_output_value_get	get the last data 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

Function name	Function description
dac_software_trigger_disable	disable DAC software trigger
dac_wave_mode_config	configure DAC wave mode
dac_wave_bit_width_config	configure DAC wave bit width
dac_lfsr_noise_config	configure DAC LFSR noise mode
dac_triangle_noise_config	configure DAC triangle noise mode
dac_concurrent_enable	enable DAC concurrent mode
dac_concurrent_disable	disable DAC concurrent mode
dac_concurrent_software_trigger_enable	enable DAC concurrent software trigger
dac_concurrent_software_trigger_disable	disable DAC concurrent software trigger
dac_concurrent_output_buffer_enable	enable DAC concurrent buffer function
dac_concurrent_output_buffer_disable	disable DAC concurrent buffer function
dac_concurrent_data_set	set DAC concurrent mode data holding register value

dac_deinit

The description of dac_deinit is shown as below:

Table 3-101. Function dac_deinit

Function name	dac_deinit
Function prototype	void dac_deinit(void)
Function descriptions	Reset DAC peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* deinitialize DAC */

dac_deinit();

dac_enable

The description of dac_enable is shown as below:

Table 3-102. Function dac_enable

Function name	dac_enable
Function prototype	void dac_enable(uint32_t dac_periph)
Function descriptions	enable DAC
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DAC */

dac_enable(DAC0);

dac_disable

The description of dac_disable is shown as below:

Table 3-103. Function dac_disable

Function name	dac_disable
Function prototype	void dac_disable(uint32_t dac_periph)
Function descriptions	disable DAC

Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DAC */

dac_disable(DAC0);
```

dac_dma_enable

The description of dac_dma_enable is shown as below:

Table 3-104. Function dac_dma_enable

Function name	dac_dma_enable
Function prototype	void dac_dma_enable(uint32_t dac_periph)
Function descriptions	enable DAC DMA function
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DAC DMA function */

dac_dma_enable(DAC0);

dac_dma_disable

The description of dac_dma_disable is shown as below:

Table 3-105. Function dac_dma_disable

Function name	dac_dma_disable
Function prototype	void dac_dma_disable(uint32_t dac_periph)
Function descriptions	disable DAC DMA function
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DAC DMA function */

dac_dma_disable(DAC0);

dac_output_buffer_enable

The description of dac_output_buffer_enable is shown as below:

Table 3-106. Function dac_output_buffer_enable

Function name	dac_output_buffer_enable
Function prototype	void dac_output_buffer_enable(uint32_t dac_periph)
Function descriptions	enable DAC output buffer
Precondition	-
The called functions	-

Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DAC output buffer */

dac_output_buffer_enable(DAC0);

dac_output_buffer_disable

The description of dac_output_buffer_disable is shown as below:

Table 3-107. Function dac_output_buffer_disable

Function name	dac_output_buffer_disable
Function prototype	void dac_output_buffer_disable(uint32_t dac_periph)
Function descriptions	disable DAC output buffer
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DAC output buffer */

dac_output_buffer_disable(DAC0);

dac_output_value_get

The description of dac_output_value_get is shown as below:

Table 3-108. Function dac_output_value_get

Function name	dac_output_value_get
Function prototype	uint16_t dac_output_value_get(uint32_t dac_periph)
Function descriptions	get DAC output value
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	**-**
Return value	
uint16_t	DAC output data (0x0000 – 0x07FF)-

Example:

```
/* get DAC output value */
data = dac_output_value_get(DAC0);
```

dac_data_set

The description of dac_data_set is shown as below:

Table 3-109. Function dac_data_set

Function name	dac_data_set
Function prototype	void dac_data_set(uint32_t dac_periph, uint32_t dac_align, uint16_t data)
Function descriptions	set the DAC specified data holding register value
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral

DACx	peripheral selection(x =0,1)
Input parameter{in}	
dac_align	DAC align mode
DAC_ALIGN_8B_R	data right 8b alignment
DAC_ALIGN_12B_R	data right 12b alignment
DAC_ALIGN_12B_L	data left 12b alignment
Input parameter{in}	
data	The data sending to holding register
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set the DAC specified data holding register value */

dac_data_set(DAC0,DAC_ALIGN_8B_R,0xff);

dac_trigger_enable

The description of dac_trigger_enable is shown as below:

Table 3-110. Function dac_trigger_enable

Function name	dac_trigger_enable
Function prototype	void dac_trigger_enable(uint32_t dac_periph)
Function descriptions	enable DAC trigger
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* enable DAC trigger */

dac_trigger_enable(DAC0);

dac_trigger_disable

The description of dac_trigger_disable is shown as below:

Table 3-111. Function dac_trigger_disable

Function name	dac_trigger_disable
Function prototype	void dac_trigger_disable(uint32_t dac_periph)
Function descriptions	disable DAC trigger
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DAC trigger */

dac_trigger_disable(DAC0);

dac_trigger_source_config

The description of dac_trigger_source_config is shown as below:

Table 3-112. Function dac_trigger_source_config

Function name	dac_trigger_source_config
Function prototype	void dac_trigger_source_config(uint32_t dac_periph,uint32_t triggersource)

Function descriptions	set DAC trigger source
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Input parameter{in}	
triggersource	external triggers of DAC
DAC_TRIGGER_T1_TRGO	TIMER1 TRGO
DAC_TRIGGER_T2_TRGO	TIMER2 TRGO(for GD32F10X_CL)
DAC_TRIGGER_T3_TRGO	TIMER3 TRGO
DAC_TRIGGER_T4_TRGO	TIMER4 TRGO
DAC_TRIGGER_T5_TRGO	TIMER5 TRGO
DAC_TRIGGER_T6_TRGO	TIMER6 TRGO
DAC_TRIGGER_T7_TRGO	TIMER7 TRGO
DAC_TRIGGER_EXTI_9	EXTI interrupt line9 event
DAC_TRIGGER_SOFTWARE	software trigger
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set DAC trigger source*/

dac_trigger_source_config(DAC0,DAC_TRIGGER_T1_TRGO);

dac_software_trigger_enable

The description of dac_software_trigger_enable is shown as below:

Table 3-113. Function dac_software_trigger_enable

Function name	dac_software_trigger_enable
Function prototype	void dac_software_trigger_enable(uint32_t dac_periph)
Function descriptions	enable DAC software trigger
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DAC software trigger */

dac_software_trigger_enable(DAC0);

dac_software_trigger_disable

The description of dac_software_trigger_disable is shown as below:

Table 3-114. Function dac_software_trigger_disable

Function name	dac_software_trigger_disable
Function prototype	void dac_software_trigger_disable(uint32_t dac_periph)
Function descriptions	disable DAC software trigger
Precondition	-
The called functions	-
Input parameter{in}	

dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DAC software trigger */

dac_software_trigger_disable(DAC0);

dac_wave_mode_config

The description of dac_wave_mode_config is shown as below:

Table 3-115. Function dac_wave_mode_config

Function name	dac_wave_mode_config
Function prototype	void dac_wave_mode_config(uint32_t dac_periph, uint32_t wave_mode)
Function descriptions	configure DAC wave mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Input parameter{in}	
wave_mode	wave_mode
DAC_WAVE_DISABLE	wave disable
DAC_WAVE_MODE_LFSR	LFSR noise mode
DAC_WAVE_MODE_TRIANGLE	triangle noise mode
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* configure DAC wave mode */

dac_wave_mode_config(DAC0, DAC_WAVE_DISABLE);

dac_wave_bit_width_config

The description of dac_wave_bit_width_config is shown as below:

Table 3-116. Function dac_wave_bit_width_config

Function name	dac_wave_bit_width_config
Function prototype	void dac_wave_bit_width_config(uint32_t dac_periph, uint32_t bit_width)
Function descriptions	configure DAC wave bit width
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Input parameter{in}	
bit_width	noise wave bit width
DAC_WAVE_BIT_WIDTH_x	x = 1..12
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure DAC wave bit width */

dac_wave_bit_width_config(DAC0,DAC_WAVE_BIT_WIDTH_1);

dac_lfsr_noise_config

The description of dac_lfsr_noise_config is shown as below:

Table 3-117. Function dac_lfsr_noise_config

Function name	dac_lfsr_noise_config
Function prototype	void dac_lfsr_noise_config(uint32_t dac_periph, uint32_t unmask_bits)
Function descriptions	configure DAC LFSR noise mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Input parameter{in}	
unmask_bits	noise wave unmask bit width
DAC_LFSR_BIT0	unmask the LFSR bit0
DAC_LFSR_BITSx_0	unmask the LFSR bits[x:0]
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

dac_triangle_noise_config

The description of dac_triangle_noise_config is shown as below:

Table 3-118. Function dac_triangle_noise_config

Function name	dac_triangle_noise_config
Function prototype	void dac_triangle_noise_config(uint32_t dac_periph, uint32_t amplitude)
Function descriptions	configure DAC triangle noise mode

Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	peripheral
DACx	peripheral selection(x =0,1)
Input parameter{in}	
amplitude	the amplitude of triangle noise
DAC_TRIANGLE_AMPLITUDE_x	x = 2^n-1(n = 1..12)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure DAC triangle noise mode */

dac_triangle_noise_config(DAC0,DAC_TRIANGLE_AMPLITUDE_1);

dac_concurrent_enable

The description of dac_concurrent_enable is shown as below:

Table 3-119. Function dac_concurrent_enable

Function name	dac_concurrent_enable
Function prototype	void dac_concurrent_enable (void);
Function descriptions	enable DAC concurrent mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* enable DAC concurrent mode */

dac_concurrent_enable();

dac_concurrent_disable

The description of dac_concurrent_disable is shown as below:

Table 3-120. Function dac_concurrent_disable

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

Example:

/* disable DAC concurrent mode */

dac_concurrent_disable();

dac_concurrent_software_trigger_enable

The description of dac_concurrent_software_trigger_enable is shown as below:

Table 3-121. Function dac_concurrent_software_trigger_enable

Function name	dac_concurrent_software_trigger_enable
Function prototype	void dac_concurrent_software_trigger_enable (void);
Function descriptions	enable DAC concurrent software trigger function

Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC concurrent software trigger function */
dac_concurrent_software_trigger_enable();
```

dac_concurrent_software_trigger_disable

The description of dac_concurrent_software_trigger_disable is shown as below:

Table 3-122. Function dac_concurrent_software_trigger_disable

Function name	dac_concurrent_software_trigger_disable
Function prototype	void dac_concurrent_software_trigger_disable (void);
Function descriptions	disable DAC concurrent software trigger function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DAC concurrent software trigger function */
dac_concurrent_software_trigger_disable();
```

dac_concurrent_output_buffer_enable

The description of dac_concurrent_output_buffer_enable is shown as below:

Table 3-123. Function dac_concurrent_output_buffer_enable

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

Example:

```
/* enable DAC concurrent buffer function */
dac_concurrent_output_buffer_enable();
```

dac_concurrent_output_buffer_disable

The description of dac_concurrent_output_buffer_disable is shown as below:

Table 3-124. Function dac_concurrent_output_buffer_disable

Function name	dac_concurrent_output_buffer_disable
Function prototype	void dac_concurrent_output_buffer_disable(void);
Function descriptions	disable DAC concurrent buffer function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* disable DAC concurrent buffer function */

dac_concurrent_output_buffer_disable();

dac_concurrent_data_set

The description of dac_concurrent_data_set is shown as below:

Table 3-125. Function dac_concurrent_data_set

Function name	dac_concurrent_data_set
Function prototype	void dac_concurrent_data_set(uint32_t dac_align, uint16_t data0, uint16_t data1)
Function descriptions	set DAC concurrent mode data holding register value
Precondition	-
The called functions	-
Input parameter{in}	
dac_align	DAC align mode
DAC_ALIGN_8B_R	data right 8b alignment
DAC_ALIGN_12B_R	data right 12b alignment
DAC_ALIGN_12B_L	data left 12b alignment
Input parameter{in}	
data0	The data sending to holding register of DAC0
Input parameter{in}	
data1	The data sending to holding register of DAC1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC concurrent mode data holding register value */
dac_concurrent_data_set(DAC_ALIGN_8B_R,0xff,0xff);
```

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 3-126. DBG Registers

Registers	Descriptions
DBG_ID	DBG ID code register
DBG_CTL	DBG control register

3.7.2. Descriptions of Peripheral functions

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

Table 3-127. DBG firmware function

Function name	Function description
dbg_id_get	read DBG_ID code register
dbg_low_power_enable	enable low power behavior when the MCU is in debug mode
dbg_low_power_disable	disable low power behavior when the MCU is in debug mode
dbg_periph_enable	enable peripheral behavior when the MCU is in debug mode
dbg_periph_disable	disable peripheral behavior when the MCU is in debug mode
dbg_trace_pin_enable	enable trace pin assignment
dbg_trace_pin_disable	disable trace pin assignment
dbg_trace_pin_mode_set	set trace pin mode

Enum dbg_periph_enum

Table 3-128. 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_TIMER3_HOLD	hold TIMER3 counter when core is halted
DBG_CAN0_HOLD	debug CAN0 kept 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_TIMER4_HOLD	hold TIMER4 counter when core is halted
DBG_TIMER5_HOLD	hold TIMER5 counter when core is halted
DBG_TIMER6_HOLD	hold TIMER6 counter when core is halted
DBG_TIMER7_HOLD	hold TIMER7 counter when core is halted
DBG_CAN1_HOLD	debug CAN1 kept when core is halted (only for GD32F10x CL series)
DBG_TIMER11_HOLD	hold TIMER11 counter when core is halted (only for GD32F10x CL and XD series)
DBG_TIMER12_HOLD	hold TIMER12 counter when core is halted (only for GD32F10x CL and XD series)
DBG_TIMER13_HOLD	hold TIMER13 counter when core is halted (only for GD32F10x CL and XD series)
DBG_TIMER8_HOLD	hold TIMER8 counter when core is halted (only for GD32F10x CL and XD series)
DBG_TIMER9_HOLD	hold TIMER9 counter when core is halted (only for GD32F10x CL and XD series)
DBG_TIMER10_HOLD	hold TIMER10 counter when core is halted (only for GD32F10x CL and XD series)

dbg_id_get

The description of dbg_id_get is shown as below:

Table 3-129. 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 3-130. 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 3-131. 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_STANDBY	keep debugger connection during standby mode
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable low power behavior when the mcu is in debug mode */
dbg_low_power_disable(DBG_LOW_POWER_SLEEP);
```

dbg_periph_enable

The description of dbg_periph_enable is shown as below:

Table 3-132. 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 3-128. 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_CANx_HOLD	x=0,1,CAN1 is only available for GD32F10x CL series, hold CANx counter when core is halted
DBG_I2Cx_HOLD	x=0,1, hold I2Cx smbus when core is halted
DBG_TIMERx_HOLD	x=0,1,2,3,4,5,6,7,8,9,10,11,12,13 (TIMER8..13 are only available for GD32F10x CL and XD series), hold TIMERx 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 3-133. 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 3-128. 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_CANx_HOLD	x=0,1,CAN1 is only available for GD32F10x CL series, hold CANx counter when core is halted
DBG_I2Cx_HOLD	x=0,1, hold I2Cx smbus when core is halted
DBG_TIMERx_HOLD	x=0,1,2,3,4,5,6,7,8,9,10,11,12,13 (TIMER8..13 are only available for GD32F10x CL and XD series), hold TIMERx 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);

dbg_trace_pin_enable

The description of dbg_trace_pin_enable is shown as below:

Table 3-134. Function dbg_trace_pin_enable

Function name	dbg_trace_pin_enable

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

Example:

```
/* enable trace pin assignment */
dbg_trace_pin_enable();
```

dbg_trace_pin_disable

The description of dbg_trace_pin_disable is shown as below:

Table 3-135. Function dbg_trace_pin_disable

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

Example:

/* disable trace pin assignment */

dbg_trace_pin_disable();

dbg_trace_pin_mode_set

The description of dbg_trace_pin_mode_set is shown as below:

Table 3-136. Function dbg_trace_pin_mode_set

Function name	dbg_trace_pin_mode_set
Function prototype	void dbg_trace_pin_mode_set(uint32_t trace_mode);
Function descriptions	Trace pin mode selection
Precondition	-
The called functions	-
Input parameter{in}	
trace_mode	trace pin mode selection
TRACE_MODE_ASYNC	trace pin used for async mode
TRACE_MODE_SYNC_DATASIZE_1	trace pin used for sync mode and data size is 1
TRACE_MODE_SYNC_DATASIZE_2	trace pin used for sync mode and data size is 2
TRACE_MODE_SYNC_DATASIZE_4	trace pin used for sync mode and data size is 4
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* trace pin mode selection */

dbg_trace_pin_mode_set(TRACE_MODE_ASYNC);

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 3-137. 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 3-138. 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

Function name	Function description
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_flag_get	check DMA flag and interrupt enable bit is set or not
dma_interrupt_flag_clear	clear DMA a channel flag
dma_interrupt_enable	enable DMA interrupt
dma_interrupt_disable	disable DMA interrupt

Structure dma_parameter_struct

Table 3-139. Structure dma_parameter_struct

Member name	Function description
periph_addr	peripheral base address
periph_width	transfer data size of peripheral
memory_addr	memory base address
memory_width	transfer data size of memory

number	channel transfer number
priority	channel priority level
periph_inc	peripheral increasing mode
memory_inc	memory increasing mode
direction	channel data transfer direction

dma_deinit

The description of dma_deinit is shown as below:

Table 3-140. Function dma_deinit

Function name	dma_deinit
Function prototype	void dma_deinit(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	deinitialize DMA a channel registers
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA0 channel0 initialize */
dma_deinit(DMA0, DMA_CH0);
```

dma_struct_para_init

The description of dma_struct_para_init is shown as below:

Table 3-141. Function dma_struct_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	the initialization data needed to initialize DMA channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* Initialize the parameters of DMA struct with the default values */

dma_parameter_struct    init_struct
dma_struct_para_init (&init_struct);
```

dma_init

The description of dma_init is shown as below:

Table 3-142. Function dma_init

Function name	dma_init
Function prototype	void dma_init(uint32_t dma_periph, dma_channel_enum channelx, dma_parameter_struct init_struct);
Function descriptions	initialize DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral

DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to ***Table 3-139. Structure dma_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA0 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(DMA0, DMA_CH0, &dma_init_struct);
```

dma_circulation_enable

The description of dma_circulation_enable is shown as below:

Table 3-143. Function dma_circulation_enable

Function name	dma_circulation_enable
Function prototype	void dma_circulation_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable DMA circulation mode

Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA0 channel0 mode configuration */
dma_circulation_enable(DMA0, DMA_CH0);
```

dma_circulation_disable

The description of dma_circulation_disable is shown as below:

Table 3-144. Function dma_circulation_disable

Function name	dma_circulation_disable
Function prototype	void dma_circulation_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable DMA circulation mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	

channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA0 channel0 mode configuration */
dma_circulation_disable(DMA0, DMA_CH0);
```

dma_memory_to_memory_enable

The description of dma_memory_to_memory_enable is shown as below:

Table 3-145. Function dma_memory_to_memory_enable

Function name	dma_memory_to_memory_enable
Function prototype	void dma_memory_to_memory_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable memory to memory mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* DMA0 channel0 mode configuration */
dma_memory_to_memory_enable(DMA0, DMA_CH0);

dma_memory_to_memory_disable

The description of dma_memory_to_memory_disable is shown as below:

Table 3-146. Function dma_memory_to_memory_disable

Function name	dma_memory_to_memory_disable
Function prototype	void dma_memory_to_memory_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable memory to memory mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* DMA0 channel0 mode configuration */
dma_memory_to_memory_enable(DMA0, DMA_CH0);

dma_channel_enable

The description of dma_channel_enable is shown as below:

Table 3-147. Function dma_channel_enable

Function name	dma_channel_enable
Function prototype	void dma_channel_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA0 transfer */
dma_channel_enable(DMA0, DMA_CH0);
```

dma_channel_disable

The description of dma_channel_disable is shown as below:

Table 3-148. Function dma_channel_disable

Function name	dma_channel_disable
Function prototype	void dma_channel_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable DMA channel
Precondition	-
The called functions	-

Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA0 transfer */
dma_channel_disable(DMA0, DMA_CH0);
```

dma_periph_address_config

The description of dma_periph_address_config is shown as below:

Table 3-149. Function dma_periph_address_config

Function name	dma_periph_address_config
Function prototype	void dma_periph_address_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t address);
Function descriptions	set DMA peripheral base address
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..	DMA channel selection

6; DMA1: x=0..4)	
Input parameter{in}	
address	peripheral base address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
#define BANK0_WRITE_START_ADDR                ((uint32_t)0x08004000)

dma_periph_address_config(DMA0, DMA_CH0, BANK0_WRITE_START_ADDR);
```

dma_memory_address_config

The description of dma_memory_address_config is shown as below:

Table 3-150. Function dma_memory_address_config

Function name	dma_memory_address_config
Function prototype	void dma_memory_address_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t address);
Function descriptions	set DMA memory base address
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
address	memory base address
Output parameter{out}	

-	-
Return value	
-	-

Example:

uint8_t g_destbuf[TRANSFER_NUM];

dma_memory_address_config(DMA0, DMA_CH0, (uint32_t) g_destbuf);

dma_transfer_number_config

The description of dma_transfer_number_config is shown as below:

Table 3-151. Function dma_transfer_number_config

Function name	dma_transfer_number_config
Function prototype	void dma_transfer_number_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t number);
Function descriptions	set the number of remaining data to be transferred by the DMA
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
number	data transfer number
0-0xffff	number
Output parameter{out}	
-	-
Return value	
-	-

Example:

#define TRANSFER_NUM 0x400

dma_transfer_number_config(DMA0, DMA_CH0, TRANSFER_NUM);

dma_transfer_number_get

The description of dma_transfer_number_get is shown as below:

Table 3-152. Function dma_transfer_number_get

Function name	dma_transfer_number_get
Function prototype	uint32_t dma_transfer_number_get(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	get the number of remaining data to be transferred by the DMA
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
uint32_t	0-0xffff

Example:

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 3-153. Function dma_priority_config

Function name	dma_priority_config
Function prototype	void dma_priority_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t priority);
Function descriptions	configure priority level of DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

```
dma_priority_config(DMA0, DMA_CH0, DMA_PRIORITY_ULTRA_HIGH);
```

dma_memory_width_config

The description of dma_memory_width_config is shown as below:

Table 3-154. Function dma_memory_width_config

Function name	dma_memory_width_config
Function prototype	void dma_memory_width_config (uint32_t dma_periph, dma_channel_enum channelx, uint32_t mwidth);
Function descriptions	configure transfer data size of memory
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

```
dma_memory_width_config(DMA0, DMA_CH0, DMA_MEMORY_WIDTH_8BIT);
```

dma_periph_width_config

The description of dma_periph_width_config is shown as below:

Table 3-155. Function dma_periph_width_config

Function name	dma_periph_width_config
Function prototype	void dma_periph_width_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t pwidth);
Function descriptions	configure transfer data size of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

```
dma_periph_width_config(DMA0, DMA_CH0, DMA_PERIPHERAL_WIDTH_8BIT);
```

dma_memory_increase_enable

The description of dma_memory_increase_enable is shown as below:

Table 3-156. Function dma_memory_increase_enable

Function name	dma_memory_increase_enable
Function prototype	void dma_memory_increase_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable next address increasement algorithm of memory
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
dma_memory_increase_enable(DMA0, DMA_CH0);
```

dma_memory_increase_disable

The description of dma_memory_increase_disable is shown as below:

Table 3-157. Function dma_memory_increase_disable

Function name	dma_memory_increase_disable
Function prototype	void dma_memory_increase_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable next address increasement algorithm of memory
Precondition	-
The called functions	-

Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

dma_memory_increase_ disable(DMA0, DMA_CH0);

dma_periph_increase_enable

The description of dma_periph_increase_enable is shown as below:

Table 3-158. Function dma_periph_increase_enable

Function name	dma_periph_increase_enable
Function prototype	void dma_periph_increase_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable next address increasement algorithm of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection

Output parameter{out}	
-	-
Return value	
-	-

Example:

dma_periph_increase_enable(DMA0, DMA_CH0);

dma_periph_increase_disable

The description of dma_periph_increase_disable is shown as below:

Table 3-159. Function dma_periph_increase_disable

Function name	dma_periph_increase_disable
Function prototype	void dma_periph_increase_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable next address increasement algorithm of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

dma_periph_increase_disable(DMA0, DMA_CH0);

dma_transfer_direction_config

The description of dma_transfer_direction_config is shown as below:

Table 3-160. Function dma_transfer_direction_config

Function name	dma_transfer_direction_config
Function prototype	void dma_transfer_direction_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t direction);
Function descriptions	configure the direction of data transfer on the channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

```
dma_transfer_direction_config(DMA0, DMA_CH0, DMA_PERIPHERAL_TO_MEMORY);
```

dma_flag_get

The description of dma_flag_get is shown as below:

Table 3-161. Function dma_flag_get

Function name	dma_flag_get
Function prototype	FlagStatus dma_flag_get(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	check DMA flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

```
FlagStatus flag = RESET;

flag = dma_flag_get(DMA0, DMA_CH0, DMA_FLAG_FTF);
```

dma_flag_clear

The description of dma_flag_clear is shown as below:

Table 3-162. Function dma_flag_clear

Function name	dma_flag_clear
Function prototype	void dma_flag_clear(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear DMA a channel flag
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

```
dma_flag_clear(DMA0, DMA_CH0, DMA_FLAG_FTF);
```

dma_interrupt_flag_get

The description of dma_interrupt_flag_get is shown as below:

Table 3-163. Function dma_interrupt_flag_get

Function name	dma_interrupt_flag_get
Function prototype	FlagStatus dma_interrupt_flag_get(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	check DMA flag and interrupt enable bit is set or not
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

```
if(dma_interrupt_flag_get(DMA0, DMA_CH3, DMA_INT_FLAG_FTF)){
    dma_interrupt_flag_clear(DMA0, DMA_CH3, DMA_INT_FLAG_FTF);
}
```

dma_interrupt_flag_clear

The description of dma_interrupt_flag_clear is shown as below:

Table 3-164. Function dma_interrupt_flag_clear

Function name	dma_interrupt_flag_clear
Function prototype	void dma_interrupt_flag_clear(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear DMA a channel flag
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

```
if(dma_interrupt_flag_get(DMA0, DMA_CH3, DMA_INT_FLAG_FTF)){
    dma_interrupt_flag_clear(DMA0, DMA_CH3, DMA_INT_FLAG_G);
}
```

dma_interrupt_enable

The description of dma_interrupt_enable is shown as below:

Table 3-165. Function dma_interrupt_enable

Function name	dma_interrupt_enable
Function prototype	void dma_interrupt_enable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t source);
Function descriptions	enable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

```
/* DMA0 channel0 interrupt configuration */
```

dma_interrupt_enable(DMA0, DMA_CH0, DMA_INT_FTF);

dma_interrupt_disable

The description of dma_interrupt_disable is shown as below:

Table 3-166. Function dma_interrupt_disable

Function name	dma_interrupt_disable
Function prototype	void dma_interrupt_disable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t source);
Function descriptions	disable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	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:

/* DMA0 channel0 interrupt configuration */

dma_interrupt_ disable(DMA0, DMA_CH0, DMA_INT_FTF);

3.9. ENET

There is a media access controller (MAC) designed in Ethernet module to support 10/100Mbps interface speed. For more efficient data transfer between Ethernet and memory, a DMA controller is designed in this module. The support interface protocol for Ethernet is media independent interface (MII) and reduced media independent interface (RMII). The ENET registers are listed in chapter ***3.9.1***, the ENET firmware functions are introduced in chapter ***3.9.2***.

3.9.1. Descriptions of Peripheral registers

ENET registers are listed in the table shown as below:

Table 3-167. ENET Registers

Registers	Descriptions
ENET_MAC_CFG	MAC configuration register
ENET_MAC_FRMF	MAC frame filter register
ENET_MAC_HLH	MAC hash list high register
ENET_MAC_HLL	MAC hash list low register
ENET_MAC_PHY_CTL	MAC PHY control register
ENET_MAC_PHY_DATA	MAC MII data register
ENET_MAC_FCTL	MAC flow control register
ENET_MAC_FCTH	MAC flow control threshold register
ENET_MAC_VLT	MAC VLAN tag register
ENET_MAC_RWFF	MAC remote wakeup frame filter register
ENET_MAC_WUM	MAC wakeup management register
ENET_MAC_INTF	MAC interrupt flag register
ENET_MAC_INTMSK	MAC interrupt mask register
ENET_MAC_ADDR0H	MAC address 0 high register
ENET_MAC_ADDR0L	MAC address 0 low register

Registers	Descriptions
ENET_MAC_ADDR 1H	MAC address 1 high register
ENET_MAC_ADDR 1L	MAC address 1 low register
ENET_MAC_ADDT 2H	MAC address 2 high register
ENET_MAC_ADDR 2L	MAC address 2 low register
ENET_MAC_ADDR 3H	MAC address 3 high register
ENET_MAC_ADDR 3L	MAC address 3 low register
ENET_MSC_CTL	MSC control register
ENET_MSC_RINTF	MSC receive interrupt flag register
ENET_MSC_TINTF	MSC transmit interrupt flag register
ENET_MSC_RINT MSK	MSC receive interrupt mask register
ENET_MSC_TINTM SK	MSC transmit interrupt mask register
ENET_MSC_SCCN T	MSC transmitted good frames after a single collision counter register
ENET_MSC_MSCC NT	MSC transmitted good frames after more than a single collision counter register
ENET_MSC_TGFC NT	MSC transmitted good frames counter register
ENET_MSC_RFCE CNT	MSC received frames with CRC error counter register
ENET_MSC_RFAE CNT	MSC received frames with alignment error counter register
ENET_MSC_RGUF CNT	MSC received good unicast frames counter register
ENET_PTP_TSCTL	PTP time stamp control register

Registers	Descriptions
ENET_PTP_SSINC	PTP subsecond increment register
ENET_PTP_TSH	PTP time stamp high register
ENET_PTP_TSL	PTP time stamp low register
ENET_PTP_TSUH	PTP time stamp update high register
ENET_PTP_TSUL	PTP time stamp update low register
ENET_PTP_TSADDEND	PTP time stamp addend register
ENET_PTP_ETH	PTP expected time high register
ENET_PTP_ETL	PTP expected time low register
ENET_DMA_BCTL	DMA bus control register
ENET_DMA_TPEN	DMA transmit poll enable register
ENET_DMA_RPEN	DMA receive poll enable register
ENET_DMA_RDTADDR	DMA receive descriptor table address register
ENET_DMA_TDTADDR	DMA transmit descriptor table address register
ENET_DMA_STAT	DMA status register
ENET_DMA_CTL	DMA control register
ENET_DMA_INTEN	DMA interrupt enable register
ENET_DMA_MFBOCNT	DMA missed frame and buffer overflow counter register
ENET_DMA_CTDADDR	DMA current transmit descriptor address register
ENET_DMA_CRDADDR	DMA current receive descriptor address register
ENET_DMA_CTBADDR	DMA current transmit buffer address register
ENET_DMA_CRBADDR	DMA current receive buffer address register

3.9.2. Descriptions of Peripheral functions

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

Table 3-168. ENET firmware function

Function name	Function description
main function	
enet_deinit	deinitialize the ENET, and reset structure parameters for ENET initialization
enet_initpara_config	configure the parameters which are usually less cared for initialization
enet_init	initialize ENET peripheral with generally concerned parameters and the less cared parameters
enet_software_reset	reset all core internal registers located in CLK_TX and CLK_RX
enet_rxframe_size_get	check receive frame valid and return frame size
enet_descriptors_chain_init	initialize the dma tx/rx descriptors's parameters in chain mode
enet_descriptors_ring_init	initialize the dma tx/rx descriptors's parameters in ring mode
enet_frame_receive	handle current received frame data to application buffer
enet_frame_transmit	handle application buffer data to transmit it
enet_transmit_checksum_config	configure the transmit IP frame checksum offload calculation and insertion
enet_enable	ENET Tx and Rx function enable (include MAC and DMA module)
enet_disable	ENET Tx and Rx function disable (include MAC and DMA module)
enet_mac_address_set	configure MAC address
enet_mac_address_get	get MAC address
enet_flag_get	get the ENET MAC/MSC/PTP/DMA status flag
enet_flag_clear	clear the ENET DMA status flag
enet_interrupt_enable	enable ENET MAC/MSC/DMA interrupt
enet_interrupt_disable	disable ENET MAC/MSC/DMA interrupt
enet_interrupt_flag_get	get ENET MAC/MSC/DMA interrupt flag

Function name	Function description
enet_interrupt_flag_clear	clear ENET DMA interrupt flag
MAC function	
enet_tx_enable	ENET Tx function enable (include MAC and DMA module)
enet_tx_disable	ENET Tx function disable (include MAC and DMA module)
enet_rx_enable	ENET Rx function enable (include MAC and DMA module)
enet_rx_disable	ENET Rx function disable (include MAC and DMA module)
enet_registers_get	put registers value into the application buffer
enet_address_filter_enable	enable the MAC address filter
enet_address_filter_disable	disable the MAC address filter
enet_address_filter_config	configure the MAC address filter
enet_phy_config	PHY interface configuration (configure SMI clock and reset PHY chip)
enet_phy_write_read	write to/read from a PHY register
enet_phyloopback_enable	enable the loopback function of phy chip
enet_phyloopback_disable	disable the loopback function of phy chip
enet_forward_feature_enable	enable ENET forward feature
enet_forward_feature_disable	disable ENET forward feature
enet_fliter_feature_enable	enable ENET fliter feature
enet_fliter_feature_disable	disable ENET fliter feature
flow control function	
enet_pauseframe_generate	generate the pause frame, ENET will send pause frame after enable transmit flow control
enet_pauseframe_detect_config	configure the pause frame detect type
enet_pauseframe_config	configure the pause frame parameters
enet_flowcontrol_threshold_config	configure the threshold of the flow control(deactive and active threshold)
enet_flowcontrol_feature_enable	enable ENET flow control feature
enet_flowcontrol_feature_disable	disable ENET flow control feature

Function name	Function description
DMA function	
enet_dmaprocess_state_get	get the dma transmit/receive process state
enet_dmaprocess_resume	poll the dma transmission/reception enable
enet_rxprocess_check_recovery	check and recover the Rx process
enet_txfifo_flush	flush the ENET transmit fifo, and wait until the flush operation completes
enet_current_desc_address_get	get the transmit/receive address of current descriptor, or current buffer, or descriptor table
enet_desc_information_get	get the Tx or Rx descriptor information
enet_missed_frame_counter_get	get the number of missed frames during receiving
descriptor function	
enet_desc_flag_get	get the bit flag of ENET dma descriptor
enet_desc_flag_set	set the bit flag of ENET dma tx descriptor
enet_desc_flag_clear	clear the bit flag of ENET dma tx descriptor
enet_desc_receive_complete_bit_enable	when receiving the completed, set RS bit in ENET_DMA_STAT register will immediately set
enet_desc_receive_complete_bit_disable	when receiving the completed, set RS bit in ENET_DMA_STAT register will is set after a configurable delay time
enet_rxframe_drop	drop current receive frame
enet_dma_feature_enable	enable DMA feature
enet_dma_feature_disable	disable DMA feature
enet_ptp_normal_descriptors_chain_init	initialize the dma Tx/Rx descriptors's parameters in normal chain mode with ptp function
enet_ptp_normal_descriptors_ring_init	initialize the dma Tx/Rx descriptors's parameters in normal ring mode with ptp function
enet_ptpframe_receive_normal_mode	receive a packet data with timestamp values to application buffer, when the DMA is in normal mode
enet_ptpframe_transmit_normal_mode	send data with timestamp values in application buffer as a transmit packet, when the DMA is in normal mode

Function name	Function description
WUM function	
enet_wum_filter_register_pointer_reset	wakeup frame filter register pointer reset
enet_wum_filter_config	set the remote wakeup frame registers
enet_wum_feature_enable	enable wakeup management features
enet_wum_feature_disable	disable wakeup management features
MSC function	
enet_msc_counters_reset	reset the MAC statistics counters
enet_msc_feature_enable	enable the MAC statistics counter features
enet_msc_feature_disable	disable the MAC statistics counter features
enet_msc_counters_get	get MAC statistics counter
PTP function	
enet_ptp_subsecond_2_nanosecond	change subsecond to nanosecond
enet_ptp_nanosecond_2_subsecond	change nanosecond to subsecond
enet_ptp_feature_enable	enable the PTP features
enet_ptp_feature_disable	disable the PTP features
enet_ptp_timestamp_function_config	configure the PTP timestamp function
enet_ptp_subsecond_increment_config	configure the PTP system time subsecond increment value
enet_ptp_timestamp_addend_config	adjusting the PTP clock frequency only in fine update mode
enet_ptp_timestamp_update_config	initializing or adding/subtracting to second of the PTP system time
enet_ptp_expected_time_config	configure the PTP expected target time
enet_ptp_system_time_get	get the PTP current system time
enet_ptp_start	configure and start PTP timestamp counter
enet_ptp_finecorrection_adjfreq	adjust frequency in fine method by configure addend register
enet_ptp_coarsecorrection_systime_update	update system time in coarse method
enet_ptp_finecorrection_settime	set system time in fine method

Function name	Function description
enet_ptp_flag_get	get the ptp flag status
Other	
enet_initpara_reset	reset the ENET initpara struct, call it before using enet_initpara_config()

Structure enet_descriptors_struct

Table 3-169. Structure enet_descriptors_struct

member name	Function description
status	status
control_buffer_size	control and buffer1, buffer2 lengths
buffer1_addr	buffer1 address pointer/timestamp low
buffer2_next_desc_addr	buffer2 or next descriptor address pointer/timestamp high

Structure enet_ptp_systime_struct

Table 3-170. Structure enet_ptp_systime_struct

member name	Function description
second	second of system time
nanosecond	nanosecond of system time
sign	sign of system time

enet_deinit

The description of enet_deinit is shown as below:

Table 3-171. Function enet_deinit

Function name	enet_deinit
Function prototype	void enet_deinit(void);
Function descriptions	deinitialize the ENET, and reset structure parameters for ENET initialization
Precondition	-
The called functions	rcu_periph_reset_enable() /rcu_periph_reset_disable()/enet_initpara_reset()
Input parameter{in}	

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

Example:

enet_deinit();

enet_initpara_config

The description of enet_initpara_config is shown as below:

Table 3-172. Function enet_initpara_config

Function name	enet_initpara_config
Function prototype	void enet_initpara_config(enet_option_enum option, uint32_t para)
Function descriptions	configure the parameters which are usually less cared for initialization, this function must be called before enet_init(), otherwise configuration will be no effect
Precondition	enet_initpara_reset(void)
The called functions	-
Input parameter{in}	
option	different function option, which is related to several parameters only one parameter can be selected which is shown as below
FORWARD_OPTION	choose to configure the frame forward related parameters
DMABUS_OPTION	choose to configure the DMA bus mode related parameters
DMA_MAXBURST_OPTION	choose to configure the DMA max burst related parameters
DMA_ARBITRATION_OPTION	choose to configure the DMA arbitration related parameters
STORE_OPTION	choose to configure the store forward mode related parameters
DMA_OPTION	choose to configure the DMA related parameters
VLAN_OPTION	choose to configure vlan related parameters
FLOWCTL_OPTION	choose to configure flow control related parameters

HASHH_OPTION	choose to configure hash high
HASHL_OPTION	choose to configure hash low
FILTER_OPTION	choose to configure frame filter related parameters
HALFDUPLEX_OPTION	choose to configure halfduplex mode related parameters
TIMER_OPTION	choose to configure time counter related parameters
INTERFRAMEGAP_OPTION	choose to configure the inter frame gap related parameters
Input parameter{in}	
para (the value according to the parameter **option**)	all the related values should be configured which are shown as below example: **para** = (value1 | value2 | value3…)
When value of parameter **option** is *FORWARD_OPTION*	
value1	*ENET_AUTO_PADCRC_DROP_ENABLE* / *ENET_AUTO_PADCRC_DROP_DISABLE*
value2	*ENET_FORWARD_ERRFRAMES_ENABLE* / *ENET_FORWARD_ERRFRAMES_DISABLE*
value3	*ENET_FORWARD_UNDERSZ_GOODFRAMES_ENABLE* / *ENET_FORWARD_UNDERSZ_GOODFRAMES_DISABLE*
When value of parameter **option** is *DMABUS_OPTION*	
value1	*ENET_ADDRESS_ALIGN_ENABLE* / *ENET_ADDRESS_ALIGN_DISABLE*
value2	*ENET_FIXED_BURST_ENABLE* / *ENET_FIXED_BURST_DISABLE*
When value of parameter **option** is *DMA_MAXBURST_OPTION*	
value1	*ENET_RXDP_1BEAT* / *ENET_RXDP_2BEAT*/ *ENET_RXDP_4BEAT* / *ENET_RXDP_8BEAT* / *ENET_RXDP_16BEAT* / *ENET_RXDP_32BEAT* / *ENET_RXDP_4xPGBL_4BEAT* / *ENET_RXDP_4xPGBL_8BEAT* / *ENET_RXDP_4xPGBL_16BEAT* / *ENET_RXDP_4xPGBL_32BEAT* / *ENET_RXDP_4xPGBL_64BEAT* / *ENET_RXDP_4xPGBL_128BEAT*
value2	*ENET_PGBL_1BEAT* / *ENET_PGBL_2BEAT* / *ENET_PGBL_4BEAT* / *ENET_PGBL_8BEAT* / *ENET_PGBL_16BEAT* / *ENET_PGBL_32BEAT* / *ENET_PGBL_4xPGBL_4BEAT* / *ENET_PGBL_4xPGBL_8BEAT* / *ENET_PGBL_4xPGBL_16BEAT* / *ENET_PGBL_4xPGBL_32BEAT* / *ENET_PGBL_4xPGBL_64BEAT* / *ENET_PGBL_4xPGBL_128BEAT*
value3	*ENET_RXTX_DIFFERENT_PGBL* / *ENET_RXTX_SAME_PGBL*

When value of parameter **option** is *DMA_ARBITRATION_OPTION*	
value1	*ENET_ARBITRATION_RXPRIORTX* / *ENET_ARBITRATION_RXTX_1_1* / *ENET_ARBITRATION_RXTX_2_1* / *ENET_ARBITRATION_RXTX_3_1* / *ENET_ARBITRATION_RXTX_4_1*
When value of parameter **option** is *STORE_OPTION*	
value1	*ENET_RX_MODE_STOREFORWARD* / *ENET_RX_MODE_CUTTHROUGH*
value2	*ENET_TX_MODE_STOREFORWARD* / *ENET_TX_MODE_CUTTHROUGH*
value3	*ENET_RX_THRESHOLD_64BYTES* / *ENET_RX_THRESHOLD_32BYTES* / *ENET_RX_THRESHOLD_96BYTES* / *ENET_RX_THRESHOLD_128BYTES*
value4	*ENET_TX_THRESHOLD_64BYTES* / *ENET_TX_THRESHOLD_128BYTES* / *ENET_TX_THRESHOLD_192BYTES* / *ENET_TX_THRESHOLD_256BYTES* / *ENET_TX_THRESHOLD_40BYTES* / *ENET_TX_THRESHOLD_32BYTES* / *ENET_TX_THRESHOLD_24BYTES* / *ENET_TX_THRESHOLD_16BYTES*
When value of parameter **option** is *DMA_OPTION*	
value1	*ENET_FLUSH_RXFRAME_ENABLE* / *ENET_FLUSH_RXFRAME_DISABLE*
value2	*ENET_SECONDFRAME_OPT_ENABLE* / *ENET_SECONDFRAME_OPT_DISABLE*
When value of parameter **option** is *VLAN_OPTION*	
value1	*ENET_VLANTAGCOMPARISON_12BIT*/ *ENET_VLANTAGCOMPARISON_16BIT*
value2	*MAC_VLT_VLTI(regval)*
When value of parameter **option** is *FLOWCTL_OPTION*	
value1	*MAC_FCTL_PTM(regval)*
value2	*ENET_ZERO_QUANTA_PAUSE_ENABLE* / *ENET_ZERO_QUANTA_PAUSE_DISABLE*
value3	*ENET_PAUSETIME_MINUS4* / *ENET_PAUSETIME_MINUS28* / *ENET_PAUSETIME_MINUS144*/*ENET_PAUSETIME_MINUS256*
value4	*ENET_MAC0_AND_UNIQUE_ADDRESS_PAUSEDETECT* / *ENET_UNIQUE_PAUSEDETECT*
value5	*ENET_RX_FLOWCONTROL_ENABLE* /

	ENET_RX_FLOWCONTROL_DISABLE
value6	*ENET_TX_FLOWCONTROL_ENABLE* / *ENET_TX_FLOWCONTROL_DISABLE*
When value of parameter **option** is *HASHH_OPTION*	
value1	*0x0~0xFFFF FFFFU*
When value of parameter **option** is *HASHL_OPTION*	
value1	*0x0~0xFFFF FFFFU*
When value of parameter **option** is *FILTER_OPTION*	
value1	*ENET_SRC_FILTER_NORMAL_ENABLE* / *ENET_SRC_FILTER_INVERSE_ENABLE* / *ENET_SRC_FILTER_DISABLE*
value2	*ENET_DEST_FILTER_INVERSE_ENABLE* / *ENET_DEST_FILTER_INVERSE_DISABLE*
value3	*ENET_MULTICAST_FILTER_HASH_OR_PERFECT* / *ENET_MULTICAST_FILTER_HASH* / *ENET_MULTICAST_FILTER_PERFECT* / *ENET_MULTICAST_FILTER_NONE*
value4	*ENET_UNICAST_FILTER_EITHER* / *ENET_UNICAST_FILTER_HASH* / *ENET_UNICAST_FILTER_PERFECT*
value5	*ENET_PCFRM_PREVENT_ALL* / *ENET_PCFRM_PREVENT_PAUSEFRAME* / *ENET_PCFRM_FORWARD_ALL* / *ENET_PCFRM_FORWARD_FILTERED*
When value of parameter **option** is *HALFDUPLEX_OPTION*	
value1	*ENET_CARRIERSENSE_ENABLE* / *ENET_CARRIERSENSE_DISABLE*
value2	*ENET_RECEIVEOWN_ENABLE* / *ENET_RECEIVEOWN_DISABLE*
value3	*ENET_RETRYTRANSMISSION_ENABLE* / *ENET_RETRYTRANSMISSION_DISABLE*
value4	*ENET_BACKOFFLIMIT_10* / *ENET_BACKOFFLIMIT_8* / *ENET_BACKOFFLIMIT_4* / *ENET_BACKOFFLIMIT_1*
value5	*ENET_DEFERRALCHECK_ENABLE* / *ENET_DEFERRALCHECK_DISABLE*
When value of parameter **option** is *TIMER_OPTION*	
value1	*ENET_WATCHDOG_ENABLE* / *ENET_WATCHDOG_DISABLE*
value2	*ENET_JABBER_ENABLE* / *ENET_JABBER_DISABLE*

When value of parameter **option** is *INTERFRAMEGAP_OPTION*	
value1	*ENET_INTERFRAMEGAP_96BIT / ENET_INTERFRAMEGAP_88BIT / ENET_INTERFRAMEGAP_80BIT / ENET_INTERFRAMEGAP_72BIT / ENET_INTERFRAMEGAP_64BIT / ENET_INTERFRAMEGAP_56BIT / ENET_INTERFRAMEGAP_48BIT / ENET_INTERFRAMEGAP_40BIT*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_initpara_reset();
enet_initpara_config(DMA_OPTION, ENET_FLUSH_RXFRAME_ENABLE | ENET_SECONDFRAME_OPT_ENABLE | ENET_NORMAL_DESCRIPTOR);
```

enet_init

The description of enet_init is shown as below:

Table 3-173. Function enet_init

Function name	enet_init
Function prototype	ErrStatus enet_init(enet_mediamode_enum mediamode, enet_chksumconf_enum checksum, enet_frmrecept_enum recept);
Function descriptions	initialize ENET peripheral with generally concerned parameters and the less cared parameters
Precondition	enet_deinit ()
The called functions	enet_phy_config /enet_phy_write_read
Input parameter{in}	
mediamode	PHY mode and mac loopback configurations, only one parameter can be selected
ENET_AUTO_NEGOTIATION	PHY auto negotiation
ENET_100M_FULLDUPLEX	100Mbit/s, full-duplex
ENET_100M_HALFDU	100Mbit/s, half-duplex

PLEX	
ENET_10M_FULLDUPLEX	10Mbit/s, full-duplex
ENET_10M_HALFDUPLEX	10Mbit/s, half-duplex
ENET_LOOPBACKMODE	MAC in loopback mode at the MII
Input parameter{in}	
checksum	IP frame checksum offload function, only one parameter can be selected
ENET_NO_AUTOCHECKSUM	disable IP frame checksum function
ENET_AUTOCHECKSUM_DROP_FAILFRAMES	enable IP frame checksum function
ENET_AUTOCHECKSUM_ACCEPT_FAILFRAMES	enable IP frame checksum function, and the received frame with only payload error but no other errors will not be dropped
Input parameter{in}	
recept	frame filter function, only one parameter can be selected
ENET_PROMISCUOUS_MODE	promiscuous mode enabled
ENET_RECEIVEALL	all received frame are forwarded to application
ENET_BROADCAST_FRAMES_PASS	the address filters pass all received broadcast frames
ENET_BROADCAST_FRAMES_DROP	the address filters filter all incoming broadcast frames
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus  enet_init_status;

enet_init_status = enet_init(ENET_AUTO_NEGOTIATION, ENET_AUTOCHECKSUM_DR
```

OP_FAILFRAMES, ENET_BROADCAST_FRAMES_PASS);

enet_software_reset

The description of enet_software_reset is shown as below:

Table 3-174. Function enet_software_reset

Function name	enet_software_reset
Function prototype	ErrStatus enet_software_reset(void);
Function descriptions	reset all core internal registers located in CLK_TX and CLK_RX
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

ErrStatus reval_state = ERROR;

reval_state = enet_software_reset();

enet_rxframe_size_get

The description of enet_rxframe_size_get is shown as below:

Table 3-175. Function enet_rxframe_size_get

Function name	enet_rxframe_size_get
Function prototype	uint32_t enet_rxframe_size_get(void);
Function descriptions	check receive frame valid and return frame size
Precondition	-
The called functions	enet_rxframe_drop()
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
uint32_t	size of received frame 0x0 - 0x3FFF

Example:

```
uint32_t reval;

reval = enet_rxframe_size_get();
```

enet_descriptors_chain_init

The description of enet_descriptors_chain_init is shown as below:

Table 3-176. Function enet_descriptors_chain_init

Function name	enet_descriptors_chain_init
Function prototype	void enet_descriptors_chain_init(enet_dmadirection_enum direction);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in chain mode
Precondition	-
The called functions	-
Input parameter{in}	
direction	the descriptors which users want to init only one parameter can be selected which is shown as below
ENET_DMA_TX	DMA Tx descriptors
ENET_DMA_RX	DMA Rx descriptors
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_descriptors_chain_init(ENET_DMA_TX);
```

enet_descriptors_ring_init

The description of enet_descriptors_ring_init is shown as below:

Table 3-177. Function enet_descriptors_ring_init

Function name	enet_descriptors_ring_init
Function prototype	void enet_descriptors_ring_init(enet_dmadirection_enum direction);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in ring mode
Precondition	-
The called functions	-
Input parameter{in}	
direction	the descriptors which users want to init only one parameter can be selected which is shown as below
ENET_DMA_TX	DMA Tx descriptors
ENET_DMA_RX	DMA Rx descriptors
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_descriptors_ring_init(ENET_DMA_TX)

enet_frame_receive

The description of enet_frame_receive is shown as below:

Table 3-178. Function enet_frame_receive

Function name	enet_frame_receive
Function prototype	ErrStatus enet_frame_receive(uint8_t *buffer, uint32_t bufsize);
Function descriptions	handle current received frame data to application buffer
Precondition	-
The called functions	-
Input parameter{in}	
bufsize	the size of buffer which is the parameter in function, (0 -- 1524)
Output parameter{out}	

buffer	pointer to the received frame data if the input is NULL, user should copy data in application by himself
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
uint8_t data_buffer[1500];
uint32_t data_size;
enet_frame_receive(data_buffer, &data_size);
```

enet_frame_transmit

The description of enet_frame_transmit is shown as below:

Table 3-179. Function enet_frame_transmit

Function name	enet_frame_transmit
Function prototype	ErrStatus enet_frame_transmit(uint8_t *buffer, uint32_t length);
Function descriptions	handle application buffer data to transmit it
Precondition	-
The called functions	-
Input parameter{in}	
buffer	pointer to the frame data to be transmitted if the input is NULL, user should handle the data in application by himself
Input parameter{in}	
length	the length of frame data to be transmitted, (0 -- 1524)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
uint8_t data_buffer[1500];
uint32_t data_size = 800;
enet_frame_transmit (data_buffer, data_size);
```

enet_transmit_checksum_config

The description of enet_transmit_checksum_config is shown as below:

Table 3-180. Function enet_transmit_checksum_config

Function name	enet_transmit_checksum_config
Function prototype	void enet_transmit_checksum_config(enet_descriptors_struct *desc, uint32_t checksum);
Function descriptions	configure the transmit IP frame checksum offload calculation and insertion
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to configure, the structure members can refer to ***Table 3-169. Structure enet_descriptors_struct***
Input parameter{in}	
checksum	IP frame checksum configuration only one parameter can be selected which is shown as below
ENET_CHECKSUM_DISABLE	checksum insertion disabled
ENET_CHECKSUM_IPV4HEADER	only IP header checksum calculation and insertion are enabled
ENET_CHECKSUM_TCPUDPICMP_SEGMENT	TCP/UDP/ICMP checksum insertion calculated but pseudo-header
ENET_CHECKSUM_TCPUDPICMP_FULL	TCP/UDP/ICMP checksum insertion fully calculated
Output parameter{out}	
Return value	

Example:

```
enet_descriptors_struct rx_desc;
enet_transmit_checksum_config(rx_desc, ENET_CHECKSUM_TCPUDPICMP_FULL);
```

enet_enable

The description of enet_enable is shown as below:

Table 3-181. Function enet_enable

Function name	enet_enable
Function prototype	void enet_enable(void);
Function descriptions	ENET Tx and Rx function enable (include MAC and DMA module)
Precondition	-
The called functions	enet_tx_enable() /enet_rx_enable()
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_enable();

enet_disable

The description of enet_disable is shown as below:

Table 3-182. Function enet_disable

Function name	enet_disable
Function prototype	void enet_disable(void);
Function descriptions	ENET Tx and Rx function disable (include MAC and DMA module)
Precondition	-
The called functions	enet_tx_disable() /enet_rx_disable()
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

enet_disable();

enet_mac_address_set

The description of enet_mac_address_set is shown as below:

Table 3-183. Function enet_mac_address_set

Function name	enet_mac_address_set
Function prototype	void enet_mac_address_set(enet_macaddress_enum mac_addr, uint8_t paddr[]);
Function descriptions	configure MAC address
Precondition	-
The called functions	-
Input parameter{in}	
mac_addr	select which MAC address will be set only one parameter can be selected which is shown as below
ENET_MAC_ADDRESS0	set MAC address 0 filter
ENET_MAC_ADDRESS1	set MAC address 1 filter
ENET_MAC_ADDRESS2	set MAC address 2 filter
ENET_MAC_ADDRESS3	set MAC address 3 filter
Input parameter{in}	
paddr	the buffer pointer which stores the MAC address little-ending store, such as MAC address is aa:bb:cc:dd:ee:22, the buffer is {22, ee, dd, cc, bb, aa}
Output parameter{out}	
-	-
Return value	

-	-

Example:

```
netif->hwaddr[0] = 0x02;
netif->hwaddr[1] = 0xaa;
netif->hwaddr[2] = 0xbb;
netif->hwaddr[3] = 0xcc;
netif->hwaddr[4] = 0xdd;
netif->hwaddr[5] = 0xee;
enet_mac_address_set(ENET_MAC_ADDRESS0, netif->hwaddr);
```

enet_mac_address_get

The description of enet_mac_address_get is shown as below:

Table 3-184. Function enet_mac_address_get

Function name	enet_mac_address_get
Function prototype	void enet_mac_address_get(enet_macaddress_enum mac_addr, uint8_t paddr[]);
Function descriptions	get MAC address
Precondition	-
The called functions	-
Input parameter{in}	
mac_addr	select which MAC address will be set only one parameter can be selected which is shown as below
ENET_MAC_ADDRESS0	set MAC address 0 filter
ENET_MAC_ADDRESS1	set MAC address 1 filter
ENET_MAC_ADDRESS2	set MAC address 2 filter
ENET_MAC_ADDRESS3	set MAC address 3 filter
Output parameter{out}	

paddr	the buffer pointer which stores the MAC address little-ending store, such as MAC address is aa:bb:cc:dd:ee:22, the buffer is {22, ee, dd, cc, bb, aa}
Return value	
-	-

Example:

enet_mac_address_get (ENET_MAC_ADDRESS0, netif->hwaddr);

enet_flag_get

The description of enet_flag_get is shown as below:

Table 3-185. Function enet_flag_get

Function name	enet_flag_get
Function prototype	FlagStatus enet_flag_get(enet_flag_enum enet_flag);
Function descriptions	get the ENET MAC/MSC/PTP/DMA status flag
Precondition	-
The called functions	-
Input parameter{in}	
enet_flag	ENET status flag, refer to enet_flag_enum only one parameter can be selected which is shown as below
ENET_MAC_FLAG_MPKR	magic packet received flag
ENET_MAC_FLAG_WUFR	wakeup frame received flag
ENET_MAC_FLAG_FLOWCONTROL	flow control status flag
ENET_MAC_FLAG_WUM	WUM status flag
ENET_MAC_FLAG_MSC	MSC status flag
ENET_MAC_FLAG_MSCR	MSC receive status flag
ENET_MAC_FLAG_MS	MSC transmit status flag

CT	
ENET_MAC_FLAG_TMST	time stamp trigger status flag
ENET_PTP_FLAG_TSSCO	timestamp second counter overflow flag
ENET_PTP_FLAG_TTM	target time match flag
ENET_MSC_FLAG_RFCE	received frames CRC error flag
ENET_MSC_FLAG_RFAE	received frames alignment error flag
ENET_MSC_FLAG_RGUF	received good unicast frames flag
ENET_MSC_FLAG_TGFSC	transmitted good frames single collision flag
ENET_MSC_FLAG_TGFMSC	transmitted good frames more single collision flag
ENET_MSC_FLAG_TGF	transmitted good frames flag
ENET_DMA_FLAG_TS	transmit status flag
ENET_DMA_FLAG_TPS	transmit process stopped status flag
ENET_DMA_FLAG_TBU	transmit buffer unavailable status flag
ENET_DMA_FLAG_TJT	transmit jabber timeout status flag
ENET_DMA_FLAG_RO	receive overflow status flag
ENET_DMA_FLAG_TU	transmit underflow status flag
ENET_DMA_FLAG_RS	receive status flag
ENET_DMA_FLAG_RBU	receive buffer unavailable status flag
ENET_DMA_FLAG_RPS	receive process stopped status flag

ENET_DMA_FLAG_RWT	receive watchdog timeout status flag
ENET_DMA_FLAG_ET	early transmit status flag
ENET_DMA_FLAG_FBE	fatal bus error status flag
ENET_DMA_FLAG_ER	early receive status flag
ENET_DMA_FLAG_AI	abnormal interrupt summary flag
ENET_DMA_FLAG_NI	normal interrupt summary flag
ENET_DMA_FLAG_EB_DMA_ERROR	DMA error flag
ENET_DMA_FLAG_EB_TRANSFER_ERROR	transfer error flag
ENET_DMA_FLAG_EB_ACCESS_ERROR	access error flag
ENET_DMA_FLAG_MSC	MSC status flag
ENET_DMA_FLAG_WUM	WUM status flag
ENET_DMA_FLAG_TST	timestamp trigger status flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

enet_flag_get (ENET_DMA_FLAG_RS);

enet_flag_clear

The description of enet_flag_clear is shown as below:

Table 3-186. Function enet_flag_clear

Function name	enet_flag_clear
Function prototype	void enet_flag_clear(enet_flag_clear_enum enet_flag);

Function descriptions	clear the ENET DMA status flag
Precondition	-
The called functions	-
Input parameter{in}	
enet_flag	ENET DMA flag clear, refer to enet_flag_clear_enum only one parameter can be selected which is shown as below
ENET_DMA_FLAG_TS_CLR	transmit status flag clear
ENET_DMA_FLAG_TPS_CLR	transmit process stopped status flag clear
ENET_DMA_FLAG_TBU_CLR	transmit buffer unavailable status flag clear
ENET_DMA_FLAG_TJT_CLR	transmit jabber timeout status flag clear
ENET_DMA_FLAG_RO_CLR	receive overflow status flag clear
ENET_DMA_FLAG_TU_CLR	transmit underflow status flag clear
ENET_DMA_FLAG_RS_CLR	receive status flag clear
ENET_DMA_FLAG_RBU_CLR	receive buffer unavailable status flag clear
ENET_DMA_FLAG_RPS_CLR	receive process stopped status flag clear
ENET_DMA_FLAG_RWT_CLR	receive watchdog timeout status flag clear
ENET_DMA_FLAG_ET_CLR	early transmit status flag clear
ENET_DMA_FLAG_FBE_CLR	fatal bus error status flag clear
ENET_DMA_FLAG_ER_CLR	early receive status flag clear
ENET_DMA_FLAG_AI_CLR	abnormal interrupt summary flag clear

ENET_DMA_FLAG_NI_CLR	normal interrupt summary flag clear
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_flag_clear(ENET_DMA_FLAG_RS_CLR);

enet_interrupt_enable

The description of enet_interrupt_enable is shown as below:

Table 3-187. Function enet_interrupt_enable

Function name	enet_interrupt_enable
Function prototype	void enet_interrupt_enable(enet_int_enum enet_int);
Function descriptions	enable ENET MAC/MSC/DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
enet_int	ENET interrupt only one parameter can be selected which is shown as below
ENET_MAC_INT_WUMIM	WUM interrupt mask
ENET_MAC_INT_TMSTIM	timestamp trigger interrupt mask
ENET_MSC_INT_RFCEIM	received frame CRC error interrupt mask
ENET_MSC_INT_RFAEIM	received frames alignment error interrupt mask
ENET_MSC_INT_RGUFIM	received good unicast frames interrupt mask
ENET_MSC_INT_TGFSCIM	transmitted good frames single collision interrupt mask

ENET_MSC_INT_TGFMSCIM	transmitted good frames more single collision interrupt mask
ENET_MSC_INT_TGFIM	transmitted good frames interrupt mask
ENET_DMA_INT_TIE	transmit interrupt enable
ENET_DMA_INT_TPSIE	transmit process stopped interrupt enable
ENET_DMA_INT_TBUIE	transmit buffer unavailable interrupt enable
ENET_DMA_INT_TJTIE	transmit jabber timeout interrupt enable
ENET_DMA_INT_ROIE	receive overflow interrupt enable
ENET_DMA_INT_TUIE	transmit underflow interrupt enable
ENET_DMA_INT_RIE	receive interrupt enable
ENET_DMA_INT_RBUIE	receive buffer unavailable interrupt enable
ENET_DMA_INT_RPSIE	receive process stopped interrupt enable
ENET_DMA_INT_RWTIE	receive watchdog timeout interrupt enable
ENET_DMA_INT_ETIE	early transmit interrupt enable
ENET_DMA_INT_FBEIE	fatal bus error interrupt enable
ENET_DMA_INT_ERIE	early receive interrupt enable
ENET_DMA_INT_AIE	abnormal interrupt summary enable
ENET_DMA_INT_NIE	normal interrupt summary enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_interrupt_enable(ENET_DMA_INT_NIE);
```

enet_interrupt_disable

The description of enet_interrupt_disable is shown as below:

Table 3-188. Function enet_interrupt_disable

Function name	enet_interrupt_disable
Function prototype	void enet_interrupt_disable(enet_int_enum enet_int);
Function descriptions	disable ENET MAC/MSC/DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
enet_int	ENET interrupt only one parameter can be selected which is shown as below
ENET_MAC_INT_WUMIM	WUM interrupt mask
ENET_MAC_INT_TMSTIM	timestamp trigger interrupt mask
ENET_MSC_INT_RFCEIM	received frame CRC error interrupt mask
ENET_MSC_INT_RFAEIM	received frames alignment error interrupt mask
ENET_MSC_INT_RGUFIM	received good unicast frames interrupt mask
ENET_MSC_INT_TGFSCIM	transmitted good frames single collision interrupt mask
ENET_MSC_INT_TGFMSCIM	transmitted good frames more single collision interrupt mask
ENET_MSC_INT_TGFIM	transmitted good frames interrupt mask
ENET_DMA_INT_TIE	transmit interrupt enable
ENET_DMA_INT_TPSIE	transmit process stopped interrupt enable
ENET_DMA_INT_TBUIE	transmit buffer unavailable interrupt enable

ENET_DMA_INT_TJTIE	transmit jabber timeout interrupt enable
ENET_DMA_INT_ROIE	receive overflow interrupt enable
ENET_DMA_INT_TUIE	transmit underflow interrupt enable
ENET_DMA_INT_RIE	receive interrupt enable
ENET_DMA_INT_RBUIE	receive buffer unavailable interrupt enable
ENET_DMA_INT_RPSIE	receive process stopped interrupt enable
ENET_DMA_INT_RWTIE	receive watchdog timeout interrupt enable
ENET_DMA_INT_ETIE	early transmit interrupt enable
ENET_DMA_INT_FBEIE	fatal bus error interrupt enable
ENET_DMA_INT_ERIE	early receive interrupt enable
ENET_DMA_INT_AIE	abnormal interrupt summary enable
ENET_DMA_INT_NIE	normal interrupt summary enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_interrupt_disable(ENET_DMA_INT_NIE);
```

enet_interrupt_flag_get

The description of enet_interrupt_flag_get is shown as below:

Table 3-189. Function enet_interrupt_flag_get

Function name	enet_interrupt_flag_get
Function prototype	FlagStatus enet_interrupt_flag_get(enet_int_flag_enum int_flag);
Function descriptions	get ENET MAC/MSC/DMA interrupt flag
Precondition	-

The called functions	-
Input parameter{in}	
int_flag	ENET interrupt flag only one parameter can be selected which is shown as below
ENET_MAC_INT_FLAG_WUM	WUM status flag
ENET_MAC_INT_FLAG_MSC	MSC status flag
ENET_MAC_INT_FLAG_MSCR	MSC receive status flag
ENET_MAC_INT_FLAG_MSCT	MSC transmit status flag
ENET_MAC_INT_FLAG_TMST	time stamp trigger status flag
ENET_MSC_INT_FLAG_RFCE	received frames CRC error flag
ENET_MSC_INT_FLAG_RFAE	received frames alignment error flag
ENET_MSC_INT_FLAG_RGUF	received good unicast frames flag
ENET_MSC_INT_FLAG_TGFSC	transmitted good frames single collision flag
ENET_MSC_INT_FLAG_TGFMSC	transmitted good frames more single collision flag
ENET_MSC_INT_FLAG_TGF	transmitted good frames flag
ENET_DMA_INT_FLAG_TS	transmit status flag
ENET_DMA_INT_FLAG_TPS	transmit process stopped status flag
ENET_DMA_INT_FLAG_TBU	transmit buffer unavailable status flag
ENET_DMA_INT_FLAG_TJT	transmit jabber timeout status flag

ENET_DMA_INT_FLAG_RO	receive overflow status flag
ENET_DMA_INT_FLAG_TU	transmit underflow status flag
ENET_DMA_INT_FLAG_RS	receive status flag
ENET_DMA_INT_FLAG_RBU	receive buffer unavailable status flag
ENET_DMA_INT_FLAG_RPS	receive process stopped status flag
ENET_DMA_INT_FLAG_RWT	receive watchdog timeout status flag
ENET_DMA_INT_FLAG_ET	early transmit status flag
ENET_DMA_INT_FLAG_FBE	fatal bus error status flag
ENET_DMA_INT_FLAG_ER	early receive status flag
ENET_DMA_INT_FLAG_AI	abnormal interrupt summary flag
ENET_DMA_INT_FLAG_NI	normal interrupt summary flag
ENET_DMA_INT_FLAG_MSC	MSC status flag
ENET_DMA_INT_FLAG_WUM	WUM status flag
ENET_DMA_INT_FLAG_TST	timestamp trigger status flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

enet_interrupt_flag_get(ENET_DMA_INT_FLAG_RS);

enet_interrupt_flag_clear

The description of enet_interrupt_flag_clear is shown as below:

Table 3-190. Function enet_interrupt_flag_clear

Function name	enet_interrupt_flag_clear
Function prototype	void enet_interrupt_flag_clear(enet_int_flag_clear_enum int_flag_clear);
Function descriptions	clear ENET DMA interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag_clear	clear ENET interrupt flag only one parameter can be selected which is shown as below
ENET_DMA_INT_FLAG_TS_CLR	transmit status flag
ENET_DMA_INT_FLAG_TPS_CLR	transmit process stopped status flag
ENET_DMA_INT_FLAG_TBU_CLR	transmit buffer unavailable status flag
ENET_DMA_INT_FLAG_TJT_CLR	transmit jabber timeout status flag
ENET_DMA_INT_FLAG_RO_CLR	receive overflow status flag
ENET_DMA_INT_FLAG_TU_CLR	transmit underflow status flag
ENET_DMA_INT_FLAG_RS_CLR	receive status flag
ENET_DMA_INT_FLAG_RBU_CLR	receive buffer unavailable status flag
ENET_DMA_INT_FLAG_RPS_CLR	receive process stopped status flag
ENET_DMA_INT_FLAG_RWT_CLR	receive watchdog timeout status flag

ENET_DMA_INT_FLAG_ET_CLR	early transmit status flag
ENET_DMA_INT_FLAG_FBE_CLR	fatal bus error status flag
ENET_DMA_INT_FLAG_ER_CLR	early receive status flag
ENET_DMA_INT_FLAG_AI_CLR	abnormal interrupt summary flag
ENET_DMA_INT_FLAG_NI_CLR	normal interrupt summary flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_interrupt_flag_clear(ENET_DMA_INT_FLAG_RS);

enet_tx_enable

The description of enet_tx_enable is shown as below:

Table 3-191. Function enet_tx_enable

Function name	enet_tx_enable
Function prototype	void enet_tx_enable(void);
Function descriptions	ENET Tx function enable (include MAC and DMA module)
Precondition	-
The called functions	enet_txfifo_flush()
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_tx_enable();

enet_tx_disable

The description of enet_tx_disable is shown as below:

Table 3-192. Function enet_tx_disable

Function name	enet_tx_disable
Function prototype	void enet_tx_disable(void);
Function descriptions	ENET Tx function disable (include MAC and DMA module)
Precondition	-
The called functions	enet_txfifo_flush()
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_tx_disable();

enet_rx_enable

The description of enet_rx_enable is shown as below:

Table 3-193. Function enet_rx_enable

Function name	enet_rx_enable
Function prototype	void enet_rx_enable(void);
Function descriptions	ENET Rx function enable (include MAC and DMA module)
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_rx_enable();

enet_rx_disable

The description of enet_rx_disable is shown as below:

Table 3-194. Function enet_rx_disable

Function name	enet_rx_disable
Function prototype	void enet_rx_disable(void);
Function descriptions	ENET Rx function disable (include MAC and DMA module)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_rx_disable();

enet_registers_get

The description of enet_registers_get is shown as below:

Table 3-195. Function enet_registers_get

Function name	enet_registers_get
Function prototype	void enet_registers_get(enet_registers_type_enum type, uint32_t *preg, uint32_t num);

Function descriptions	put registers value into the application buffer
Precondition	-
The called functions	-
Input parameter{in}	
type	register type which will be get only one parameter can be selected which is shown as below
ALL_MAC_REG	get the registers within the offset scope range ENET_MAC_CFG to ENET_MAC_FCTH
ALL_MSC_REG	get the registers within the offset scope range ENET_MSC_CTL to ENET_MSC_RGUFCNT
ALL_PTP_REG	get the registers within the offset scope range ENET_PTP_TSCTL to ENET_PTP_PPSCTL
ALL_DMA_REG	get the registers within the offset scope range ENET_DMA_BCTL to ENET_DMA_CRBADDR
Input parameter{in}	
num	the number of registers that the user want to get, (0 -- 54)
Output parameter{out}	
preg	the application buffer pointer for storing the register value
Return value	
-	-

Example:

```
uint32_t register_buffer[5];

enet_registers_get(ALL_MAC_REG, 5, register_buffer);
```

enet_address_filter_enable

The description of enet_address_filter_enable is shown as below:

Table 3-196. Function enet_address_filter_enable

Function name	enet_address_filter_enable
Function prototype	void enet_address_filter_enable(enet_macaddress_enum mac_addr);
Function descriptions	enable the MAC address filter
Precondition	-

The called functions	--
Input parameter{in}	
mac_addr	select which MAC address will be enable only one parameter can be selected which is shown as below
ENET_MAC_ADDRESS1	enable MAC address 1 filter
ENET_MAC_ADDRESS2	enable MAC address 2 filter
ENET_MAC_ADDRESS3	enable MAC address 3 filter
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_address_filter_enable(ENET_MAC_ADDRESS1);

enet_address_filter_disable

The description of enet_address_filter_disable is shown as below:

Table 3-197. Function enet_address_filter_disable

Function name	enet_address_filter_disable
Function prototype	void enet_address_filter_disable(enet_macaddress_enum mac_addr);
Function descriptions	disable the MAC address filter
Precondition	-
The called functions	-
Input parameter{in}	
mac_addr	select which MAC address will be enable only one parameter can be selected which is shown as below
ENET_MAC_ADDRESS1	enable MAC address 1 filter
ENET_MAC_ADDRESS2	enable MAC address 2 filter

ENET_MAC_ADDRESS3	enable MAC address 3 filter
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_address_filter_disable(ENET_MAC_ADDRESS1);

enet_address_filter_config

The description of enet_address_filter_config is shown as below:

Table 3-198. Function enet_address_filter_config

Function name	enet_address_filter_config
Function prototype	void enet_address_filter_config(enet_macaddress_enum mac_addr, uint32_t addr_mask, uint32_t filter_type);
Function descriptions	configure the MAC address filter
Precondition	-
The called functions	-
Input parameter{in}	
mac_addr	select which MAC address will be enable only one parameter can be selected which is shown as below
ENET_MAC_ADDRESS1	enable MAC address 1 filter
ENET_MAC_ADDRESS2	enable MAC address 2 filter
ENET_MAC_ADDRESS3	enable MAC address 3 filter
Input parameter{in}	
addr_mask	select which MAC address bytes will be mask one or more parameters can be selected which are shown as below
ENET_ADDRESS_MASK_BYTE0	mask ENET_MAC_ADDR1L[7:0] bits

ENET_ADDRESS_MASK_BYTE1	mask ENET_MAC_ADDR1L[15:8] bits
ENET_ADDRESS_MASK_BYTE2	mask ENET_MAC_ADDR1L[23:16] bits
ENET_ADDRESS_MASK_BYTE3	mask ENET_MAC_ADDR1L [31:24] bits
ENET_ADDRESS_MASK_BYTE4	mask ENET_MAC_ADDR1H [7:0] bits
ENET_ADDRESS_MASK_BYTE5	mask ENET_MAC_ADDR1H [15:8] bits
Input parameter{in}	
filter_type	select which MAC address filter type will be selected only one parameter can be selected which is shown as below
ENET_ADDRESS_FILTER_SA	The MAC address is used to compared with the SA field of the received frame
ENET_ADDRESS_FILTER_DA	The MAC address is used to compared with the DA field of the received frame
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_address_filter_config(ENET_MAC_ADDRESS1, ENET_ADDRESS_MASK_BYTE0 | ENET_ADDRESS_MASK_BYTE1 | ENET_ADDRESS_MASK_BYTE2, ENET_ADDRESS_FILTER_DA);
```

enet_phy_config

The description of enet_phy_config is shown as below:

Table 3-199. Function enet_phy_config

Function name	enet_phy_config
Function prototype	ErrStatus enet_phy_config(void);
Function descriptions	PHY interface configuration (configure SMI clock and reset PHY chip)

Precondition	-
The called functions	rcu_clock_freq_get()/enet_phy_write_read()
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

enet_phy_config();

enet_phy_write_read

The description of enet_phy_write_read is shown as below:

Table 3-200. Function enet_phy_write_read

Function name	enet_phy_write_read
Function prototype	ErrStatus enet_phy_write_read(enet_phydirection_enum direction, uint16_t phy_address, uint16_t phy_reg, uint16_t *pvalue);
Function descriptions	write to / read from a PHY register
Precondition	-
The called functions	-
Input parameter{in}	
direction	only one parameter can be selected which is shown as below
ENET_PHY_WRITE	write data to phy register
ENET_PHY_READ	read data from phy register
Input parameter{in}	
phy_address	0x0 - 0x1F
Input parameter{in}	
phy_reg	0x0 - 0x1F
Input parameter{in}	

pvalue	the value will be written to the PHY register in ENET_PHY_WRITE direction
Output parameter{out}	
pvalue	the value will be read from the PHY register in ENET_PHY_READ direction
Return value	
ErrStatus	ERROR or SUCCESS

Example:

uint16_t temp_phy = 0U;

phy_state = enet_phy_write_read(ENET_PHY_WRITE, PHY_ADDRESS, PHY_REG_BCR, &temp_phy);

enet_phyloopback_enable

The description of enet_phyloopback_enable is shown as below:

Table 3-201. Function enet_phyloopback_enable

Function name	enet_phyloopback_enable
Function prototype	ErrStatus enet_phyloopback_enable(void);
Function descriptions	enable the loopback function of PHY chip
Precondition	-
The called functions	enet_phy_write_read()
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

phy_state = enet_phyloopback_enable();

enet_phyloopback_disable

The description of enet_phyloopback_disable is shown as below:

Table 3-202. Function enet_phyloopback_disable

Function name	enet_phyloopback_disable
Function prototype	ErrStatus enet_phyloopback_disable(void);
Function descriptions	disable the loopback function of PHY chip
Precondition	-
The called functions	enet_phy_write_read
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

phy_state = enet_phyloopback_disable();

enet_forward_feature_enable

The description of enet_forward_feature_enable is shown as below:

Table 3-203. Function enet_forward_feature_enable

Function name	enet_forward_feature_enable
Function prototype	void enet_forward_feature_enable(uint32_t feature);
Function descriptions	enable ENET forward feature
Precondition	-
The called functions	-
Input parameter{in}	
feature	the feature of ENET forward mode one or more parameters can be selected which are shown as below
ENET_AUTO_PADCRC_DROP	the function of the MAC strips the Pad/FCS field on received frames
ENET_FORWARD_ERRFRAMES	the function that all frame received with error except runt error are forwarded to memory

ENET_FORWARD_UNDERSZ_GOODFRAMES	the function that forwarding undersized good frames
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_forward_feature_enable(ENET_AUTO_PADCRC_DROP);

enet_forward_feature_disable

The description of enet_forward_feature_disable is shown as below:

Table 3-204. Function enet_forward_feature_disable

Function name	enet_forward_feature_disable
Function prototype	void enet_forward_feature_disable(uint32_t feature);
Function descriptions	disable ENET forward feature
Precondition	-
The called functions	-
Input parameter{in}	
feature	the feature of ENET forward mode one or more parameters can be selected which are shown as below
ENET_AUTO_PADCRC_DROP	the function of the MAC strips the Pad/FCS field on received frames
ENET_FORWARD_ERRFRAMES	the function that all frame received with error except runt error are forwarded to memory
ENET_FORWARD_UNDERSZ_GOODFRAMES	the function that forwarding undersized good frames
Output parameter{out}	
-	-
Return value	

-	-

Example:

enet_forward_feature_disable (ENET_AUTO_PADCRC_DROP);

enet_fliter_feature_enable

The description of enet_fliter_feature_enable is shown as below:

Table 3-205. Function enet_fliter_feature_enable

Function name	enet_fliter_feature_enable
Function prototype	void enet_fliter_feature_enable(uint32_t feature);
Function descriptions	enable ENET fliter feature
Precondition	-
The called functions	-
Input parameter{in}	
feature	the feature of ENET fliter mode one or more parameters can be selected which are shown as below
ENET_SRC_FILTER	filter source address function
ENET_SRC_FILTER_INVERSE	inverse source address filtering result function
ENET_DEST_FILTER_INVERSE	inverse DA filtering result function
ENET_MULTICAST_FILTER_PASS	pass all multicast frames function
ENET_MULTICAST_FILTER_HASH_MODE	HASH multicast filter function
ENET_UNICAST_FILTER_HASH_MODE	HASH unicast filter function
ENET_FILTER_MODE_EITHER	HASH or perfect filter function
Output parameter{out}	
-	-
Return value	

-	-

Example:

enet_fliter_feature_enable(ENET_SRC_FILTER);

enet_fliter_feature_disable

The description of enet_fliter_feature_disable is shown as below:

Table 3-206. Function enet_fliter_feature_disable

Function name	enet_fliter_feature_disable
Function prototype	void enet_fliter_feature_disable(uint32_t feature);
Function descriptions	disable ENET fliter feature
Precondition	-
The called functions	-
Input parameter{in}	
feature	the feature of ENET fliter mode one or more parameters can be selected which are shown as below
ENET_SRC_FILTER	filter source address function
ENET_SRC_FILTER_INVERSE	inverse source address filtering result function
ENET_DEST_FILTER_INVERSE	inverse DA filtering result function
ENET_MULTICAST_FILTER_PASS	pass all multicast frames function
ENET_MULTICAST_FILTER_HASH_MODE	HASH multicast filter function
ENET_UNICAST_FILTER_HASH_MODE	HASH unicast filter function
ENET_FILTER_MODE_EITHER	HASH or perfect filter function
Output parameter{out}	
-	-
Return value	

-	-

Example:

enet_fliter_feature_disable(ENET_SRC_FILTER_INVERSE);

enet_pauseframe_generate

The description of enet_pauseframe_generate is shown as below:

Table 3-207. Function enet_pauseframe_generate

Function name	enet_pauseframe_generate
Function prototype	ErrStatus enet_pauseframe_generate(void);
Function descriptions	generate the pause frame, ENET will send pause frame after enable transmit flow control this function only use in full-dulex mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

ErrStatus reval;

reval = enet_pauseframe_generate();

enet_pauseframe_detect_config

The description of enet_pauseframe_detect_config is shown as below:

Table 3-208. Function enet_pauseframe_detect_config

Function name	enet_pauseframe_detect_config
Function prototype	void enet_pauseframe_detect_config(uint32_t detect);
Function descriptions	configure the pause frame detect type
Precondition	-

The called functions	-
Input parameter{in}	
detect	pause frame detect type only one parameter can be selected which is shown as below
ENET_MAC0_AND_UNIQUE_ADDRESS_PAUSEDETECT	besides the unique multicast address, MAC can also use the MAC0 address to detecting pause frame
ENET_UNIQUE_PAUSEDETECT	only the unique multicast address for pause frame which is specified in IEEE802.3 can be detected
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_pauseframe_detect_config(ENET_UNIQUE_PAUSEDETECT);

enet_pauseframe_config

The description of enet_pauseframe_config is shown as below:

Table 3-209. Function enet_pauseframe_config

Function name	enet_pauseframe_config
Function prototype	void enet_pauseframe_config(uint32_t pausetime, uint32_t pause_threshold);
Function descriptions	configure the pause frame parameters
Precondition	-
The called functions	-
Input parameter{in}	
pausetime	pause time in transmit pause control frame, (0 – 0xFFFF)
Input parameter{in}	
pause_threshold	the threshold of the pause timer for retransmitting frames automatically, this value must make sure to be less than configured pause time, only one parameter can be selected which is shown as below

ENET_PAUSETIME_MINUS4	pause time minus 4 slot times
ENET_PAUSETIME_MINUS28	pause time minus 28 slot times
ENET_PAUSETIME_MINUS144	pause time minus 144 slot times
ENET_PAUSETIME_MINUS256	pause time minus 256 slot times
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_pauseframe_config(30, ENET_PAUSETIME_MINUS4);

enet_flowcontrol_threshold_config

The description of enet_flowcontrol_threshold_config is shown as below:

Table 3-210. Function enet_flowcontrol_threshold_config

Function name	enet_flowcontrol_threshold_config
Function prototype	void enet_flowcontrol_threshold_config(uint32_t deactive, uint32_t active);
Function descriptions	configure the threshold of the flow control(deactive and active threshold)
Precondition	-
The called functions	-
Input parameter{in}	
deactive	the threshold of the deactive flow control, this value should always be less than active flow control value, only one parameter can be selected which is shown as below
ENET_DEACTIVE_THRESHOLD_256BYTES	threshold level is 256 bytes
ENET_DEACTIVE_THRESHOLD_512BYTES	threshold level is 512 bytes
ENET_DEACTIVE_TH	threshold level is 768 bytes

RESHOLD_768BYTES	
ENET_DEACTIVE_THRESHOLD_1024BYTES	threshold level is 1024 bytes
ENET_DEACTIVE_THRESHOLD_1280BYTES	threshold level is 1280 bytes
ENET_DEACTIVE_THRESHOLD_1536BYTES	threshold level is 1536 bytes
ENET_DEACTIVE_THRESHOLD_1792BYTES	threshold level is 1792 bytes
Input parameter{in}	
active	the threshold of the active flow control, only one parameter can be selected which is shown as below
ENET_ACTIVE_THRESHOLD_256BYTES	threshold level is 256 bytes
ENET_ACTIVE_THRESHOLD_512BYTES	threshold level is 512 bytes
ENET_ACTIVE_THRESHOLD_768BYTES	threshold level is 768 bytes
ENET_ACTIVE_THRESHOLD_1024BYTES	threshold level is 1024 bytes
ENET_ACTIVE_THRESHOLD_1280BYTES	threshold level is 1280 bytes
ENET_ACTIVE_THRESHOLD_1536BYTES	threshold level is 1536 bytes
ENET_ACTIVE_THRESHOLD_1792BYTES	threshold level is 1792 bytes
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_flowcontrol_threshold_config(ENET_DEACTIVE_THRESHOLD_256BYTES, ENET_ACTIVE_THRESHOLD_256BYTES);

enet_flowcontrol_feature_enable

The description of enet_flowcontrol_feature_enable is shown as below:

Table 3-211. Function enet_flowcontrol_feature_enable

Function name	enet_flowcontrol_feature_enable
Function prototype	void enet_flowcontrol_feature_enable(uint32_t feature);
Function descriptions	enable ENET flow control feature
Precondition	-
The called functions	-
Input parameter{in}	
feature	the feature of ENET flow control mode one or more parameters can be selected which are shown as below
ENET_ZERO_QUANTA_PAUSE	the automatic zero-quanta generation function
ENET_TX_FLOWCONTROL	the flow control operation in the MAC
ENET_RX_FLOWCONTROL	decoding function for the received pause frame and process it
ENET_BACK_PRESSURE	back pressure operation in the MAC(only use in half-dulex mode)
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_flowcontrol_feature_enable(ENET_ZERO_QUANTA_PAUSE);

enet_flowcontrol_feature_disable

The description of enet_flowcontrol_feature_disable is shown as below:

Table 3-212. Function enet_flowcontrol_feature_disable

Function name	enet_flowcontrol_feature_disable
Function prototype	void enet_flowcontrol_feature_disable(uint32_t feature);
Function descriptions	disable ENET flow control feature
Precondition	-
The called functions	-
Input parameter{in}	
feature	the feature of ENET flow control mode one or more parameters can be selected which are shown as below
ENET_ZERO_QUANTA_PAUSE	the automatic zero-quanta generation function
ENET_TX_FLOWCONTROL	the flow control operation in the MAC
ENET_RX_FLOWCONTROL	decoding function for the received pause frame and process it
ENET_BACK_PRESSURE	back pressure operation in the MAC(only use in half-dulex mode)
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_flowcontrol_feature_disable(ENET_ZERO_QUANTA_PAUSE);

enet_dmaprocess_state_get

The description of enet_dmaprocess_state_get is shown as below:

Table 3-213. Function enet_dmaprocess_state_get

Function name	enet_dmaprocess_state_get
Function prototype	uint32_t enet_dmaprocess_state_get(enet_dmadirection_enum direction);
Function descriptions	get the dma transmit/receive process state
Precondition	-

The called functions	-
Input parameter{in}	
direction	choose the direction of DMA process which users want to resume only one parameter can be selected which is shown as below
ENET_DMA_TX	DMA transmit process
ENET_DMA_RX	DMA receive process
Output parameter{out}	
-	-
Return value	
uint32_t	state of dma process, the value range shows below: ENET_RX_STATE_STOPPED / ENET_RX_STATE_FETCHING / ENET_RX_STATE_WAITING / ENET_RX_STATE_SUSPENDED / ENET_RX_STATE_CLOSING / ENET_RX_STATE_QUEUING / ENET_TX_STATE_STOPPED / ENET_TX_STATE_FETCHING / ENET_TX_STATE_WAITING / ENET_TX_STATE_READING / ENET_TX_STATE_SUSPENDED / ENET_TX_STATE_CLOSING

Example:

```
uint32_t reval;
reval = enet_dmaprocess_state_get(ENET_DMA_RX);
if(ENET_RX_STATE_SUSPENDED == reval){
    do…
}
```

enet_dmaprocess_resume

The description of enet_dmaprocess_resume is shown as below:

Table 3-214. Function enet_dmaprocess_resume

Function name	enet_dmaprocess_resume
Function prototype	void enet_dmaprocess_resume(enet_dmadirection_enum direction);
Function descriptions	poll the DMA transmission/reception enable by writing any value to the ENET_DMA_TPEN/ENET_DMA_RPEN register, this will make the DMA to resume transmission/reception
Precondition	-

The called functions	-
Input parameter{in}	
direction	choose the direction of DMA process which users want to resume only one parameter can be selected which is shown as below
ENET_DMA_TX	DMA transmit process
ENET_DMA_RX	DMA receive process
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_dmaprocess_resume(ENET_DMA_RX);
```

enet_rxprocess_check_recovery

The description of enet_rxprocess_check_recovery is shown as below:

Table 3-215. Function enet_rxprocess_check_recovery

Function name	enet_rxprocess_check_recovery
Function prototype	void enet_rxprocess_check_recovery(void);
Function descriptions	check and recover the Rx process
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_rxprocess_check_recovery();
```

enet_txfifo_flush

The description of enet_txfifo_flush is shown as below:

Table 3-216. Function enet_txfifo_flush

Function name	enet_txfifo_flush
Function prototype	ErrStatus enet_txfifo_flush(void);
Function descriptions	flush the ENET transmit FIFO, and wait until the flush operation completes
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus reval;

reval = enet_txfifo_flush();
```

enet_current_desc_address_get

The description of enet_current_desc_address_get is shown as below:

Table 3-217. Function enet_current_desc_address_get

Function name	enet_current_desc_address_get
Function prototype	uint32_t enet_current_desc_address_get(enet_desc_reg_enum addr_get);
Function descriptions	get the transmit/receive address of current descriptor, or current buffer, or descriptor table
Precondition	-
The called functions	-
Input parameter{in}	
addr_get	choose the address which users want to get only one parameter can be selected which is shown as below

ENET_RX_DESC_TABLE	the start address of the receive descriptor table
ENET_RX_CURRENT_DESC	the start descriptor address of the current receive descriptor read by the RxDMA controller
ENET_RX_CURRENT_BUFFER	the current receive buffer address being read by the RxDMA controller
ENET_TX_DESC_TABLE	the start address of the transmit descriptor table
ENET_TX_CURRENT_DESC	the start descriptor address of the current transmit descriptor read by the TxDMA controller
ENET_TX_CURRENT_BUFFER	the current transmit buffer address being read by the TxDMA controller
Output parameter{out}	
-	-
Return value	
uint32_t	0- 0xFFFFFFFF

Example:

```
uint32_t reval;
reval = enet_current_desc_address_get(ENET_RX_CURRENT_DESC);
```

enet_desc_information_get

The description of enet_desc_information_get is shown as below:

Table 3-218. Function enet_desc_information_get

Function name	enet_desc_information_get
Function prototype	uint32_t enet_desc_information_get(enet_descriptors_struct *desc, enet_descstate_enum info_get);
Function descriptions	get the Tx or Rx descriptor information
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to get information, the structure members can refer to ***Table 3-169. Structure enet_descriptors***

	struct
Input parameter{in}	
info_get	the descriptor information type which is selected only one parameter can be selected which is shown as below
RXDESC_BUFFER_1_SIZE	receive buffer 1 size
RXDESC_BUFFER_2_SIZE	receive buffer 2 size
RXDESC_FRAME_LENGTH	the byte length of the received frame that was transferred to the buffer
TXDESC_COLLISION_COUNT	the number of collisions occurred before the frame was transmitted
RXDESC_BUFFER_1_ADDR	the buffer1 address of the Rx frame
TXDESC_BUFFER_1_ADDR	the buffer1 address of the Tx frame
Output parameter{out}	
-	-
Return value	
uint32_t	descriptor information if value is 0xFFFFFFFFU, means the false input parameter

Example:

```
uint32_t reval;
reval = enet_desc_information_get(rx_desc, RXDESC_FRAME_LENGTH);
```

enet_missed_frame_counter_get

The description of enet_missed_frame_counter_get is shown as below:

Table 3-219. Function enet_missed_frame_counter_get

Function name	enet_missed_frame_counter_get
Function prototype	void enet_missed_frame_counter_get(uint32_t *rxfifo_drop, uint32_t *rxdma_drop);
Function descriptions	get the number of missed frames during receiving

Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
rxfifo_drop	pointer to the number of frames dropped by RxFIFO
Output parameter{out}	
rxdma_drop	pointer to the number of frames missed by the RxDMA controller
Return value	
-	-

Example:

```
uint32_t rxcnt, txcnt;

enet_missed_frame_counter_get(&rxcnt, &txcnt);
```

enet_desc_flag_get

The description of enet_desc_flag_get is shown as below:

Table 3-220. Function enet_desc_flag_get

Function name	enet_desc_flag_get
Function prototype	FlagStatus enet_desc_flag_get(enet_descriptors_struct *desc, uint32_t desc_flag);
Function descriptions	get the bit flag of ENET DMA descriptor
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to get flag, the structure members can refer to ***Table 3-169. Structure enet_descriptors_struct***
Input parameter{in}	
desc_flag (the value according to the parameter **desc**)	the bit flag of ENET DMA descriptor only one parameter can be selected which is shown as below

When type of parameter **desc** is TX	
ENET_TDES0_DB	deferred
ENET_TDES0_UFE	underflow error
ENET_TDES0_EXD	excessive deferral
ENET_TDES0_VFRM	VLAN frame
ENET_TDES0_ECO	excessive collision
ENET_TDES0_LCO	late collision
ENET_TDES0_NCA	no carrier
ENET_TDES0_LCA	loss of carrier
ENET_TDES0_IPPE	IP payload error
ENET_TDES0_FRMF	frame flushed
ENET_TDES0_JT	jabber timeout
ENET_TDES0_ES	error summary
ENET_TDES0_IPHE	IP header error
ENET_TDES0_TTMSS	transmit timestamp status
ENET_TDES0_TCHM	the second address chained mode
ENET_TDES0_TERM	transmit end of ring mode
ENET_TDES0_TTSEN	transmit timestamp function enable
ENET_TDES0_DPAD	disable adding pad
ENET_TDES0_DCRC	disable CRC
ENET_TDES0_FSG	first segment
ENET_TDES0_LSG	last segment
ENET_TDES0_INTC	interrupt on completion
ENET_TDES0_DAV	DAV bit
When type of parameter **desc** is RX	
ENET_RDES0_PCERR	payload checksum error
ENET_RDES0_CERR	CRC error
ENET_RDES0_DBERR	dribble bit error

ENET_RDES0_RERR	receive error
ENET_RDES0_RWDT	receive watchdog timeout
ENET_RDES0_FRMT	frame type
ENET_RDES0_LCO	late collision
ENET_RDES0_IPHERR	IP frame header error
ENET_RDES0_LDES	last descriptor
ENET_RDES0_FDES	first descriptor
ENET_RDES0_VTAG	VLAN tag
ENET_RDES0_OERR	overflow error
ENET_RDES0_LERR	length error
ENET_RDES0_SAFF	SA filter fail
ENET_RDES0_DERR	descriptor error
ENET_RDES0_ERRS	error summary
ENET_RDES0_DAFF	destination address filter fail
ENET_RDES0_DAV	descriptor available
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

FlagStatus reval;

reval = enet_desc_flag_get(p_txdesc, ENET_TDES0_TCHM);

enet_desc_flag_set

The description of enet_desc_flag_set is shown as below:

Table 3-221. Function enet_desc_flag_set

Function name	enet_desc_flag_set
Function prototype	void enet_desc_flag_set(enet_descriptors_struct *desc, uint32_t desc_flag);

Function descriptions	set the bit flag of ENET DMA descriptor
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to set flag, the structure members can refer to ***Table 3-169. Structure enet_descriptors_struct***
Input parameter{in}	
desc_flag (the value according to the parameter **desc**)	the bit flag of ENET DMA descriptor only one parameter can be selected which is shown as below
When type of parameter **desc** is TX	
ENET_TDES0_VFRM	VLAN frame
ENET_TDES0_FRMF	frame flushed
ENET_TDES0_TCHM	the second address chained mode
ENET_TDES0_TERM	transmit end of ring mode
ENET_TDES0_TTSEN	transmit timestamp function enable
ENET_TDES0_DPAD	disable adding pad
ENET_TDES0_DCRC	disable CRC
ENET_TDES0_FSG	first segment
ENET_TDES0_LSG	last segment
ENET_TDES0_INTC	interrupt on completion
ENET_TDES0_DAV	DAV bit
When type of parameter **desc** is RX	
ENET_RDES0_DAV	descriptor available
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_desc_flag_set(p_txdesc, ENET_TDES0_TCHM);

enet_desc_flag_clear

The description of enet_desc_flag_clear is shown as below:

Table 3-222. Function enet_desc_flag_clear

Function name	enet_desc_flag_clear
Function prototype	void enet_desc_flag_clear(enet_descriptors_struct *desc, uint32_t desc_flag);
Function descriptions	clear the bit flag of ENET DMA descriptor
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to set flag, the structure members can refer to ***Table 3-169. Structure enet_descriptors_struct***
Input parameter{in}	
desc_flag (the value according to the parameter **desc**)	the bit flag of ENET DMA descriptor only one parameter can be selected which is shown as below
When type of parameter **desc** is TX	
ENET_TDES0_VFRM	VLAN frame
ENET_TDES0_FRMF	frame flushed
ENET_TDES0_TCHM	the second address chained mode
ENET_TDES0_TERM	transmit end of ring mode
ENET_TDES0_TTSEN	transmit timestamp function enable
ENET_TDES0_DPAD	disable adding pad
ENET_TDES0_DCRC	disable CRC
ENET_TDES0_FSG	first segment
ENET_TDES0_LSG	last segment
ENET_TDES0_INTC	interrupt on completion
ENET_TDES0_DAV	DAV bit

When type of parameter **desc** is RX	
ENET_RDES0_DAV	descriptor available
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_desc_flag_clear(p_txdesc, ENET_TDES0_TCHM);

enet_desc_receive_complete_bit_enable

The description of enet_desc_receive_complete_bit_enable is shown as below:

Table 3-223. Function enet_desc_receive_complete_bit_enable

Function name	enet_desc_receive_complete_bit_enable
Function prototype	void enet_desc_receive_complete_bit_enable(enet_descriptors_struct *desc);
Function descriptions	when receiving the completed, set RS bit in ENET_DMA_STAT register will set
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to get flag, the structure members can refer to ***Table 3-169. Structure enet_descriptors_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_desc_receive_complete_bit_enable(p_rxdesc);

enet_desc_receive_complete_bit_disable

The description of enet_desc_receive_complete_bit_disable is shown as below:

Table 3-224. Function enet_desc_receive_complete_bit_disable

Function name	enet_desc_receive_complete_bit_disable
Function prototype	void enet_desc_receive_complete_bit_disable(enet_descriptors_struct *desc);
Function descriptions	when receiving the completed, set RS bit in ENET_DMA_STAT register will not set
Precondition	-
The called functions	-
Input parameter{in}	
desc	the descriptor pointer which users want to get flag, the structure members can refer to ***Table 3-169. Structure enet_descriptors_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_desc_receive_complete_bit_disable(p_rxdesc);

enet_rxframe_drop

The description of enet_rxframe_drop is shown as below:

Table 3-225. Function enet_rxframe_drop

Function name	enet_rxframe_drop
Function prototype	void enet_rxframe_drop(void);
Function descriptions	drop current receive frame
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

enet_rxframe_drop();

enet_dma_feature_enable

The description of enet_dma_feature_enable is shown as below:

Table 3-226. Function enet_dma_feature_enable

Function name	enet_dma_feature_enable
Function prototype	void enet_dma_feature_enable(uint32_t feature);
Function descriptions	enable DMA feature
Precondition	-
The called functions	-
Input parameter{in}	
feature	the feature of DMA mode one or more parameters can be selected which are shown as below
ENET_NO_FLUSH_RXFRAME	RxDMA does not flushes frames function
ENET_SECONDFRAME_OPT	TxDMA controller operate on second frame function
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_dma_feature_enable(ENET_NO_FLUSH_RXFRAME);

enet_dma_feature_disable

The description of enet_dma_feature_disable is shown as below:

Table 3-227. Function enet_dma_feature_disable

Function name	enet_dma_feature_disable

Function prototype	void enet_dma_feature_disable(uint32_t feature);
Function descriptions	disable DMA feature
Precondition	-
The called functions	-
Input parameter{in}	
feature	the feature of DMA mode one or more parameters can be selected which are shown as below
ENET_NO_FLUSH_RXFRAME	RxDMA does not flushes frames function
ENET_SECONDFRAME_OPT	TxDMA controller operate on second frame function
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_dma_feature_disable(ENET_NO_FLUSH_RXFRAME);

enet_ptp_normal_descriptors_chain_init

The description of enet_ptp_normal_descriptors_chain_init is shown as below:

Table 3-228. Function enet_ptp_normal_descriptors_chain_init

Function name	enet_ptp_normal_descriptors_chain_init
Function prototype	void enet_ptp_normal_descriptors_chain_init(enet_dmadirection_enum direction, enet_descriptors_struct *desc_ptptab);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in normal chain mode with PTP function
Precondition	-
The called functions	-
Input parameter{in}	
direction	the descriptors which users want to init only one parameter can be selected which is shown as below

ENET_DMA_TX	DMA Tx descriptors
ENET_DMA_RX	DMA Rx descriptors
Input parameter{in}	
desc_ptptab	pointer to the first descriptor address of PTP Rx descriptor table, the structure members can refer to ***Table 3-169. Structure enet_descriptors_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_descriptors_struct ptp_rxdesc_tab[ENET_RXBUF_NUM];

enet_ptp_normal_descriptors_chain_init(ENET_DMA_RX, ptp_rxdesc_tab);

enet_ptp_normal_descriptors_ring_init

The description of enet_ptp_normal_descriptors_ring_init is shown as below:

Table 3-229. Function enet_ptp_normal_descriptors_ring_init

Function name	enet_ptp_normal_descriptors_ring_init
Function prototype	void enet_ptp_normal_descriptors_ring_init(enet_dmadirection_enum direction, enet_descriptors_struct *desc_ptptab);
Function descriptions	initialize the DMA Tx/Rx descriptors's parameters in normal ring mode with PTP function
Precondition	-
The called functions	-
Input parameter{in}	
direction	the descriptors which users want to init only one parameter can be selected which is shown as below
ENET_DMA_TX	DMA Tx descriptors
ENET_DMA_RX	DMA Rx descriptors
Input parameter{in}	

desc_ptptab	pointer to the first descriptor address of PTP Rx descriptor table, the structure members can refer to ***Table 3-169. Structure enet_descriptors_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_descriptors_struct ptp_rxdesc_tab[ENET_RXBUF_NUM];
enet_ptp_normal_descriptors_ring_init(ENET_DMA_RX, ptp_rxdesc_tab);
```

enet_ptpframe_receive_normal_mode

The description of enet_ptpframe_receive_normal_mode is shown as below:

Table 3-230. Function enet_ptpframe_receive_normal_mode

Function name	enet_ptpframe_receive_normal_mode
Function prototype	ErrStatus enet_ptpframe_receive_normal_mode(uint8_t *buffer, uint32_t bufsize, uint32_t timestamp[]);
Function descriptions	receive a packet data with timestamp values to application buffer, when the DMA is in normal mode
Precondition	-
The called functions	-
Input parameter{in}	
bufsize	the size of buffer which is the parameter in function
Output parameter{out}	
timestamp	pointer to the table which stores the timestamp high and low
Output parameter{out}	
buffer	pointer to the application buffer if the input is NULL, user should copy data in application by himself
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
uint32_t rx_buffer[500];

uint32_t time_stamp[2];

ErrStatus status;

status = enet_ptpframe_receive_normal_mode(rx_buffer, 500, time_stamp);
```

enet_ptpframe_transmit_normal_mode

The description of enet_ptpframe_transmit_normal_mode is shown as below:

Table 3-231. Function enet_ptpframe_transmit_normal_mode

Function name	enet_ptpframe_transmit_normal_mode
Function prototype	ErrStatus enet_ptpframe_transmit_normal_mode(uint8_t *buffer, uint32_t length, uint32_t timestamp[]);
Function descriptions	send data with timestamp values in application buffer as a transmit packet, when the DMA is in normal mode
Precondition	-
The called functions	--
Input parameter{in}	
buffer	pointer on the application buffer if the input is NULL, user should copy data in application by himself
Input parameter{in}	
length	the length of frame data to be transmitted
Output parameter{out}	
timestamp	pointer to the table which stores the timestamp high and low if the input is NULL, timestamp is ignored
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
uint32_t tx_buffer[500];

uint32_t time_stamp[2];

ErrStatus status;

status = enet_ptpframe_transmit_normal_mode(tx_buffer, 500, time_stamp);
```

enet_wum_filter_register_pointer_reset

The description of enet_wum_filter_register_pointer_reset is shown as below:

Table 3-232. Function enet_wum_filter_register_pointer_reset

Function name	enet_wum_filter_register_pointer_reset
Function prototype	void enet_wum_filter_register_pointer_reset(void);
Function descriptions	wakeup frame filter register pointer reset
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_wum_filter_register_pointer_reset();

enet_wum_filter_config

The description of enet_wum_filter_config is shown as below:

Table 3-233. Function enet_wum_filter_config

Function name	enet_wum_filter_config
Function prototype	void enet_wum_filter_config(uint32_t pdata[]);
Function descriptions	set the remote wakeup frame registers
Precondition	-
The called functions	-
Input parameter{in}	
pdata	pointer to buffer data which is written to remote wakeup frame registers (8 words total)
Output parameter{out}	
-	-

Return value	
-	-

Example:

uint32_t wum_data[8];

enet_wum_filter_config (wum_data);

enet_wum_feature_enable

The description of enet_wum_feature_enable is shown as below:

Table 3-234. Function enet_wum_feature_enable

Function name	enet_wum_feature_enable
Function prototype	void enet_wum_feature_enable(uint32_t feature);
Function descriptions	enable wakeup management features
Precondition	-
The called functions	-
Input parameter{in}	
feature	one or more parameters can be selected which are shown as below
ENET_WUM_POWER_DOWN	power down mode
ENET_WUM_MAGIC_PACKET_FRAME	enable a wakeup event due to magic packet reception
ENET_WUM_WAKE_UP_FRAME	enable a wakeup event due to wakeup frame reception
ENET_WUM_GLOBAL_UNICAST	any received unicast frame passed filter is considered to be a wakeup frame
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_wum_feature_enable(ENET_WUM_POWER_DOWN);

enet_wum_feature_disable

The description of enet_wum_feature_disable is shown as below:

Table 3-235. Function enet_wum_feature_disable

Function name	enet_wum_feature_disable
Function prototype	void enet_wum_feature_disable(uint32_t feature)
Function descriptions	disable wakeup management features
Precondition	-
The called functions	-
Input parameter{in}	
feature	one or more parameters can be selected which are shown as below
ENET_WUM_MAGIC_PACKET_FRAME	enable a wakeup event due to magic packet reception
ENET_WUM_WAKE_UP_FRAME	enable a wakeup event due to wakeup frame reception
ENET_WUM_GLOBAL_UNICAST	any received unicast frame passed filter is considered to be a wakeup frame
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_wum_feature_disable(ENET_WUM_POWER_DOWN);

enet_msc_counters_reset

The description of enet_msc_counters_reset is shown as below:

Table 3-236. Function enet_msc_counters_reset

Function name	enet_msc_counters_reset
Function prototype	void enet_msc_counters_reset(void);
Function descriptions	reset the MAC statistics counters
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_msc_counters_reset();
```

enet_msc_feature_enable

The description of enet_msc_feature_enable is shown as below:

Table 3-237. Function enet_msc_feature_enable

Function name	enet_msc_feature_enable
Function prototype	void enet_msc_feature_enable(uint32_t feature);
Function descriptions	enable the MAC statistics counter features
Precondition	-
The called functions	-
Input parameter{in}	
feature	one or more parameters can be selected which are shown as below
ENET_MSC_COUNTER_STOP_ROLLOVER	counter stop rollover
ENET_MSC_RESET_ON_READ	reset on read
ENET_MSC_COUNTERS_FREEZE	MSC counter freeze
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_msc_feature_enable(ENET_MSC_COUNTER_STOP_ROLLOVER);

enet_msc_feature_disable

The description of enet_msc_feature_disable is shown as below:

Table 3-238. Function enet_msc_feature_disable

Function name	enet_msc_feature_disable
Function prototype	void enet_msc_feature_disable(uint32_t feature);
Function descriptions	disable the MAC statistics counter features
Precondition	-
The called functions	-
Input parameter{in}	
feature	one or more parameters can be selected which are shown as below
ENET_MSC_COUNTER_STOP_ROLLOVER	counter stop rollover
ENET_MSC_RESET_ON_READ	reset on read
ENET_MSC_COUNTERS_FREEZE	MSC counter freeze
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_msc_feature_disable(ENET_MSC_COUNTER_STOP_ROLLOVER);

enet_msc_counters_get

The description of enet_msc_counters_get is shown as below:

Table 3-239. Function enet_msc_counters_get

Function name	enet_msc_counters_get
Function prototype	uint32_t enet_msc_counters_get(enet_msc_counter_enum counter);

Function descriptions	get MAC statistics counter
Precondition	-
The called functions	-
Input parameter{in}	
counter	MSC counters which is selected only one parameter can be selected which is shown as below
ENET_MSC_TX_SCCNT	MSC transmitted good frames after a single collision counter
ENET_MSC_TX_MSCCNT	MSC transmitted good frames after more than a single collision counter
ENET_MSC_TX_TGFCNT	MSC transmitted good frames counter
ENET_MSC_RX_RFCECNT	MSC received frames with CRC error counter
ENET_MSC_RX_RFAECNT	MSC received frames with alignment error counter
ENET_MSC_RX_RGUFCNT	MSC received good unicast frames counter
Output parameter{out}	
-	-
Return value	
uint32_t	the MSC counter value

Example:

```
uin32_t reval;

reval = enet_msc_counters_get(ENET_MSC_TX_SCCNT);
```

enet_ptp_subsecond_2_nanosecond

The description of enet_ptp_subsecond_2_nanosecond is shown as below:

Table 3-240. Function enet_ptp_subsecond_2_nanosecond

Function name	enet_ptp_subsecond_2_nanosecond
Function prototype	uint32_t enet_ptp_subsecond_2_nanosecond(uint32_t subsecond);

Function descriptions	change subsecond to nanosecond
Precondition	-
The called functions	-
Input parameter{in}	
subsecond	subsecond value (0 – 0x7FFF FFFF)
Output parameter{out}	
-	-
Return value	
uint32_t	the nanosecond value (0 -- 499999999)

Example:

uin32_t reval;

reval = enet_ptp_subsecond_2_nanosecond(50);

enet_ptp_nanosecond_2_subsecond

The description of enet_ptp_nanosecond_2_subsecond is shown as below:

Table 3-241. Function enet_ptp_nanosecond_2_subsecond

Function name	enet_ptp_nanosecond_2_subsecond
Function prototype	uint32_t enet_ptp_nanosecond_2_subsecond(uint32_t nanosecond);
Function descriptions	change nanosecond to subsecond
Precondition	-
The called functions	-
Input parameter{in}	
nanosecond	nanosecond value (0 -- 499999999)
Output parameter{out}	
-	-
Return value	
uint32_t	the subsecond value (0 – 0x7FFF FFFF)

Example:

uin32_t reval;

reval = enet_ptp_nanosecond_2_subsecond(50);

enet_ptp_feature_enable

The description of enet_ptp_feature_enable is shown as below:

Table 3-242. Function enet_ptp_feature_enable

Function name	enet_ptp_feature_enable
Function prototype	void enet_ptp_feature_enable(uint32_t feature);
Function descriptions	enable the PTP features
Precondition	-
The called functions	-
Input parameter{in}	
feature	the feature of ENET PTP mode one or more parameters can be selected which are shown as below
ENET_RXTX_TIMESTAMP	timestamp function for transmit and receive frames
ENET_PTP_TIMESTAMP_INT	timestamp interrupt trigger
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_ptp_feature_enable(ENET_RXTX_TIMESTAMP);

enet_ptp_feature_disable

The description of enet_ptp_feature_disable is shown as below:

Table 3-243. Function enet_ptp_feature_disable

Function name	enet_ptp_feature_disable
Function prototype	void enet_ptp_feature_disable(uint32_t feature);
Function descriptions	disable the PTP features
Precondition	-

The called functions	-
Input parameter{in}	
feature	the feature of ENET PTP mode one or more parameters can be selected which are shown as below
ENET_RXTX_TIMESTAMP	timestamp function for transmit and receive frames
ENET_PTP_TIMESTAMP_INT	timestamp interrupt trigger
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_ptp_feature_disable(ENET_RXTX_TIMESTAMP);

enet_ptp_timestamp_function_config

The description of enet_ptp_timestamp_function_config is shown as below:

Table 3-244. Function enet_ptp_timestamp_function_config

Function name	enet_ptp_timestamp_function_config
Function prototype	ErrStatus enet_ptp_timestamp_function_config(enet_ptp_function_enum func);
Function descriptions	configure the PTP timestamp function
Precondition	-
The called functions	-
Input parameter{in}	
func	only one parameter can be selected which is shown as below
ENET_PTP_ADDEND_UPDATE	addend register update
ENET_PTP_SYSTIME_UPDATE	timestamp update
*ENET_PTP_SYSTIME_*	timestamp initialize

INIT	
ENET_PTP_FINEMODE	the system timestamp uses the fine method for updating
ENET_PTP_COARSEMODE	the system timestamp uses the coarse method for updating
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

ErrStatus reval;

reval = enet_ptp_timestamp_function_config(ENET_PTP_ADDEND_UPDATE);

enet_ptp_subsecond_increment_config

The description of enet_ptp_subsecond_increment_config is shown as below:

Table 3-245. Function enet_ptp_subsecond_increment_config

Function name	enet_ptp_subsecond_increment_config
Function prototype	void enet_ptp_subsecond_increment_config(uint32_t subsecond);
Function descriptions	configure system time subsecond increment value
Precondition	-
The called functions	-
Input parameter{in}	
subsecond	the value will be added to the subsecond value of system time, this value must be between 0 and 0xFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_ptp_subsecond_increment_config(0x1F);

enet_ptp_timestamp_addend_config

The description of enet_ptp_timestamp_addend_config is shown as below:

Table 3-246. Function enet_ptp_timestamp_addend_config

Function name	enet_ptp_timestamp_addend_config
Function prototype	void enet_ptp_timestamp_addend_config(uint32_t add);
Function descriptions	adjusting the clock frequency only in fine update mode
Precondition	-
The called functions	-
Input parameter{in}	
add	the value will be added to the accumulator register to achieve time synchronization (0 – 0xFFFF FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_ptp_timestamp_addend_config(0x1FFF);

enet_ptp_timestamp_update_config

The description of enet_ptp_timestamp_update_config is shown as below:

Table 3-247. Function enet_ptp_timestamp_update_config

Function name	enet_ptp_timestamp_update_config
Function prototype	void enet_ptp_timestamp_update_config(uint32_t sign, uint32_t second, uint32_t subsecond);
Function descriptions	initialize or add/subtract to second of the system time
Precondition	-
The called functions	-
Input parameter{in}	
sign	timestamp update positive or negative sign, only one parameter can be selected which is shown as below

ENET_PTP_ADD_TO_TIME	update value is added to system time
ENET_PTP_SUBSTRACT_FROM_TIME	timestamp update value is subtracted from system time
Input parameter{in}	
second	initializing or adding/subtracting to second of the system time (0 – 0xFFFF FFFF)
Input parameter{in}	
subsecond	the current subsecond of the system time with 0.46 ns accuracy (0 – 0x7FFF FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_ptp_timestamp_update_config(ENET_PTP_ADD_TO_TIME, 0, 0);

enet_ptp_expected_time_config

The description of enet_ptp_expected_time_config is shown as below:

Table 3-248. Function enet_ptp_expected_time_config

Function name	enet_ptp_expected_time_config
Function prototype	void enet_ptp_expected_time_config(uint32_t second, uint32_t nanosecond);
Function descriptions	configure the expected target time
Precondition	-
The called functions	-
Input parameter{in}	
second	the expected target second time (0 – 0xFFFF FFFF)
Input parameter{in}	
nanosecond	the expected target nanosecond time (signed) (0 – 0xFFFF FFFF)
Output parameter{out}	

-	-
Return value	
-	-

Example:

enet_ptp_expected_time_config(2000, 0);

enet_ptp_system_time_get

The description of enet_ptp_system_time_get is shown as below:

Table 3-249. Function enet_ptp_system_time_get

Function name	enet_ptp_system_time_get
Function prototype	void enet_ptp_system_time_get(enet_ptp_systime_struct *systime_struct);
Function descriptions	get the current system time
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
systime_struct	pointer to a enet_ptp_systime_struct structure which contains parameters of PTP system time, the structure members can refer to ***Table 3-170. Structure enet_ptp_systime_struct***
Return value	
-	-

Example:

enet_ptp_systime_struct systime；

enet_ptp_system_time_get(&systime);

enet_ptp_start

The description of enet_ptp_start is shown as below:

Table 3-250. Function enet_ptp_start

Function name	enet_ptp_start

Function prototype	void enet_ptp_start(int32_t updatemethod, uint32_t init_sec, uint32_t init_subsec, uint32_t carry_cfg, uint32_t accuracy_cfg);
Function descriptions	configure and start PTP timestamp counter
Precondition	-
The called functions	enet_interrupt_disable/ enet_ptp_feature_enable/ enet_ptp_subsecond_increment_config/ enet_ptp_timestamp_addend_config/ enet_ptp_timestamp_function_config/ enet_ptp_flag_get/ enet_ptp_timestamp_update_config
Input parameter{in}	
updatemethod	method for updating
ENET_PTP_FINEMODE	fine correction method
ENET_PTP_COARSEMODE	coarse correction method
Input parameter{in}	
init_sec	second value for initializing system time (0 – 0xFFFF FFFF)
Input parameter{in}	
init_subsec	subsecond value for initializing system time (0 – 0x7FFF FFFF)
Input parameter{in}	
carry_cfg	the value to be added to the accumulator register (in fine method is used) (0 – 0xFFFF FFFF)
Input parameter{in}	
accuracy_cfg	the value to be added to the subsecond value of system time (0 – 0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

enet_ptp_start(ENET_PTP_FINEMODE, 0, 0, 50, 0);

enet_ptp_finecorrection_adjfreq

The description of enet_ptp_finecorrection_adjfreq is shown as below:

Table 3-251. Function enet_ptp_finecorrection_adjfreq

Function name	enet_ptp_finecorrection_adjfreq
Function prototype	void enet_ptp_finecorrection_adjfreq(int32_t carry_cfg);
Function descriptions	adjust frequency in fine method by configure addend register
Precondition	-
The called functions	enet_ptp_timestamp_addend_config/ enet_ptp_timestamp_function_config
Input parameter{in}	
carry_cfg	the value to be added to the accumulator register (0 – 0xFFFF FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_ptp_finecorrection_adjfreq(50);
```

enet_ptp_coarsecorrection_systime_update

The description of enet_ptp_coarsecorrection_systime_update is shown as below:

Table 3-252. Function enet_ptp_coarsecorrection_systime_update

Function name	enet_ptp_coarsecorrection_systime_update
Function prototype	void enet_ptp_coarsecorrection_systime_update(enet_ptp_systime_struct *systime_struct);
Function descriptions	update system time in coarse method
Precondition	-
The called functions	enet_ptp_nanosecond_2_subsecond/ enet_ptp_timestamp_update_config/ enet_ptp_timestamp_function_config/ enet_ptp_flag_get/ enet_ptp_timestamp_addend_config
Input parameter{in}	
systime_struct	pointer to a enet_ptp_systime_struct structure which contains parameters of PTP system time, the structure members can refer to ***Table 3-170. Structure enet_ptp_systime_struct***
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
enet_ptp_systime_struct systime_struct;
systime_struct.second = 0x0000FFFF;
systime_struct.nanosecond = 0;
systime_struct.sign = ENET_PTP_TIME_POSITIVE;
enet_ptp_coarsecorrection_systime_update (&systime_struct);
```

enet_ptp_finecorrection_settime

The description of enet_ptp_finecorrection_settime is shown as below:

Table 3-253. Function enet_ptp_finecorrection_settime

Function name	enet_ptp_finecorrection_settime
Function prototype	void enet_ptp_finecorrection_settime(enet_ptp_systime_struct * systime_struct);
Function descriptions	set system time in fine method
Precondition	-
The called functions	enet_ptp_nanosecond_2_subsecond/ enet_ptp_timestamp_update_config/ enet_ptp_timestamp_function_config/ enet_ptp_flag_get
Input parameter{in}	
systime_struct	pointer to a enet_ptp_systime_struct structure which contains parameters of PTP system time, the structure members can refer to ***Table 3-170. Structure enet_ptp_systime_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_ptp_systime_struct systime_struct;
systime_struct.second = 0x0000FFFF;
```

systime_struct.nanosecond = 0;

systime_struct.sign = ENET_PTP_TIME_POSITIVE;

enet_ptp_finecorrection_settime(&systime_struct);

enet_ptp_flag_get

The description of enet_ptp_flag_get is shown as below:

Table 3-254. Function enet_ptp_flag_get

Function name	enet_ptp_flag_get
Function prototype	FlagStatus enet_ptp_flag_get(uint32_t flag);
Function descriptions	get the ptp flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	ptp flag status to be checked
ENET_PTP_ADDEND_UPDATE	addend register update
ENET_PTP_SYSTIME_UPDATE	timestamp update
ENET_PTP_SYSTIME_INIT	timestamp initialize
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

FlagStatus reval;

reval = enet_ptp_flag_get(ENET_PTP_ADDEND_UPDATE);

enet_initpara_reset

The description of enet_initpara_reset is shown as below:

Table 3-255. Function enet_initpara_reset

Function name	enet_initpara_reset
Function prototype	void enet_initpara_reset(void);
Function descriptions	reset the ENET initpara struct, call it before using enet_initpara_config()
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
enet_initpara_reset();
```

3.10. EXMC

The external memory controller EXMC, is used as a translator for MCU to access a variety of external memory. The EXMC registers are listed in chapter ***3.10.1***, the EXMC firmware functions are introduced in chapter ***3.10.2***.

3.10.1. Descriptions of Peripheral registers

EXMC registers are listed in the table shown as below:

Table 3-256. EXMC Registers

Registers	Descriptions
EXMC_SNCTLx (x=0, 1, 2, 3)	SRAM/NOR Flash control registers
EXMC_SNTCFGx (x=0, 1, 2, 3)	SRAM/NOR Flash timing configuration registers
EXMC_SNWTCFGx (x=0, 1, 2, 3)	SRAM/NOR Flash write timing configuration registers
EXMC_NPCTLx	NAND Flash/PC Card control registers

Registers	Descriptions
(x=1, 2, 3)	
EXMC_NPINTENx (x=1, 2, 3)	NAND Flash/PC Card interrupt enable registers
EXMC_NPCTCFGx (x=1, 2, 3)	NAND Flash/PC Card common space timing configuration registers
EXMC_NPATCFGx (x=1, 2, 3)	NAND Flash/PC Card attribute space timing configuration registers
EXMC_PIOTCFG3	PC Card I/O space timing configuration register
EXMC_NECCx (x=1, 2)	NAND Flash ECC registers

3.10.2. Descriptions of Peripheral functions

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

Table 3-257. EXMC firmware function

Function name	Function description
exmc_norsram_deinit	deinitialize EXMC NOR/SRAM region
exmc_norsram_struct_para_init	initialize the struct exmc_norsram_parameter_struct
exmc_norsram_init	initialize EXMC NOR/SRAM region
exmc_norsram_enable	enable EXMC NOR/PSRAM bank region
exmc_norsram_disable	disable EXMC NOR/PSRAM bank region
exmc_nand_deinit	deinitialize EXMC NAND bank
exmc_nand_init	initialize EXMC NAND bank
exmc_nand_struct_para_init	initialize the struct exmc_nand_parameter_struct
exmc_nand_enable	enable NAND bank
exmc_nand_disable	disable NAND bank
exmc_nand_ecc_config	enable or disable the EXMC NAND ECC function
exmc_ecc_get	get the EXMC ECC value
exmc_pccard_deinit	deinitialize EXMC PC card bank
exmc_pccard_init	initialize EXMC PC card bank

Function name	Function description
exmc_pccard_struct_para_init	initialize the struct exmc_pccard_parameter_struct
exmc_pccard_enable	enable PC Card Bank
exmc_pccard_disable	disable PC Card Bank
exmc_interrupt_enable	enable EXMC interrupt
exmc_interrupt_disable	disable EXMC interrupt
exmc_interrupt_flag_get	check EXMC interrupt flag is set or not
exmc_interrupt_flag_clear	clear EXMC interrupt flag
exmc_flag_get	check EXMC flag is set or not
exmc_flag_clear	clear EXMC flag

Structure exmc_norsram_timing_parameter_struct

Table 3-258. Structure exmc_norsram_timing_parameter_struct

Member name	Function description
asyn_access_mode	asynchronous access mode
syn_data_latency	configure the data latency
syn_clk_division	configure the clock divide ratio
bus_latency	configure the bus latency
asyn_data_setuptime	configure the data setup time,asynchronous access mode valid
asyn_address_holdtime	configure the address hold time,asynchronous access mode valid
asyn_address_setuptime	configure the data setup time,asynchronous access mode valid

Structure exmc_norsram_parameter_struct

Table 3-259. Structure exmc_norsram_parameter_struct

Member name	Function description
norsram_region	select the region of EXMC NOR/SRAM bank
write_mode	the write mode, synchronous mode or asynchronous mode
extended_mode	enable or disable the extended mode

asyn_wait	enable or disable the asynchronous wait function
nwait_signal	enable or disable the NWAIT signal while in synchronous bust mode
memory_write	enable or disable the write operation
nwait_config	NWAIT signal configuration
wrap_burst_mode	enable or disable the wrap burst mode
nwait_polarity	specifies the polarity of NWAIT signal from memory
burst_mode	enable or disable the burst mode
databus_width	specifies the databus width of external memory
memory_type	specifies the type of external memory
address_data_mux	specifies whether the data bus and address bus are multiplexed
read_write_timing	timing parameters for read and write if the extended mode is not used or the timing parameters for read if the extended mode is used
write_timing	timing parameters for write when the extended mode is used

Structure exmc_nand_pccard_timing_parameter_struct

Table 3-260. Structure exmc_nand_pccard_timing_parameter_struct

Member name	Function description
databus_hiztime	configure the dadtabus HiZ time for write operation
holdtime	configure the address hold time(or the data hold time for write operation)
waittime	configure the minimum wait time
setuptime	configure the address setup time

Structure exmc_nand_parameter_struct

Table 3-261. Structure exmc_nand_parameter_struct

Member name	Function description
nand_bank	select the bank of NAND
ecc_size	the page size for the ECC calculation
atr_latency	configure the latency of ALE low to RB low
ctr_latency	configure the latency of CLE low to RB low
ecc_logic	enable or disable the ECC calculation logic

databus_width	the NAND flash databus width
wait_feature	enables or disables the wait feature
common_space_timing	the timing parameters for NAND flash common space
attribute_space_timing	the timing parameters for NAND flash attribute space

Structure exmc_pccard_parameter_struct

Table 3-262. Structure exmc_pccard_parameter_struct

Member name	Function description
atr_latency	configure the latency of ALE low to RB low
ctr_latency	configure the latency of CLE low to RB low
wait_feature	enables or disables the Wait feature
common_space_timing	the timing parameters for NAND flash common space
attribute_space_timing	the timing parameters for NAND flash attribute space
io_space_timing	the timing parameters for NAND flash IO space

exmc_norsram_deinit

The description of exmc_norsram_deinit is shown as below:

Table 3-263. Function exmc_norsram_deinit

Function name	exmc_norsram_deinit
Function prototype	void exmc_norsram_deinit(uint32_t norsram_region);
Function descriptions	deinitialize EXMC NOR/SRAM region
Precondition	-
The called functions	-
Input parameter{in}	
norsram_region	select the region of bank0
EXMC_BANK0_NORSRAM_REGIONx(x=0..3)	region x of bank0

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize EXMC NOR/SRAM region 0 */
exmc_norsram_deinit(EXMC_BANK0_NORSRAM_REGION0);
```

exmc_norsram_struct_para_init

The description of exmc_norsram_struct_para_init is shown as below:

Table 3-264. Function exmc_norsram_struct_para_init

Function name	exmc_norsram_struct_para_init
Function prototype	void exmc_norsram_struct_para_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);
Function descriptions	initialize the struct exmc_norsram_parameter_struct
Precondition	-
The called functions	-
Input parameter{in}	
exmc_norsram_init_struct	Structure for initialization, the structure members can refer to ***Table 3-259. Structure exmc_norsram_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the struct nor_init_struct */
exmc_norsram_parameter_struct nor_init_struct;
exmc_norsram_struct_para_init(&nor_init_struct);
```

exmc_norsram_init

The description of exmc_norsram_init is shown as below:

Table 3-265. Function exmc_norsram_init

Function name	exmc_norsram_init
Function prototype	void exmc_norsram_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);
Function descriptions	initialize EXMC NOR/SRAM region
Precondition	-
The called functions	-
Input parameter{in}	
exmc_norsram_init_struct	Structure for initialization, the structure members can refer to ***Table 3-259. Structure exmc_norsram_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize EXMC NOR/SRAM region */
exmc_norsram_parameter_struct nor_init_struct;
exmc_norsram_timing_parameter_struct nor_timing_init_struct;
/* configure timing parameter */
nor_timing_init_struct.asyn_access_mode = EXMC_ACCESS_MODE_A;
nor_timing_init_struct.syn_data_latency = EXMC_DATALAT_2_CLK;
nor_timing_init_struct.syn_clk_division = EXMC_SYN_CLOCK_RATIO_DISABLE;
nor_timing_init_struct.bus_latency = 1;
nor_timing_init_struct.asyn_data_setuptime = 7;
nor_timing_init_struct.asyn_address_holdtime = 2;
nor_timing_init_struct.asyn_address_setuptime = 5;
/* configure EXMC bus parameters */
nor_init_struct.norsram_region = EXMC_BANK0_NORSRAM_REGION0;
nor_init_struct.write_mode = EXMC_ASYN_WRITE;
nor_init_struct.extended_mode = DISABLE;
```

```
nor_init_struct.asyn_wait = DISABLE;
nor_init_struct.nwait_signal = DISABLE;
nor_init_struct.memory_write = ENABLE;
nor_init_struct.nwait_config = EXMC_NWAIT_CONFIG_BEFORE;
nor_init_struct.wrap_burst_mode = DISABLE;
nor_init_struct.nwait_polarity = EXMC_NWAIT_POLARITY_LOW;
nor_init_struct.burst_mode = DISABLE;
nor_init_struct.databus_width = EXMC_NOR_DATABUS_WIDTH_16B;
nor_init_struct.memory_type = EXMC_MEMORY_TYPE_NOR;
nor_init_struct.address_data_mux = ENABLE;
nor_init_struct.read_write_timing = &nor_timing_init_struct;
nor_init_struct.write_timing = &nor_timing_init_struct;
exmc_norsram_init(&nor_init_struct);
```

exmc_norsram_enable

The description of exmc_norsram_enable is shown as below:

Table 3-266. Function exmc_norsram_enable

Function name	exmc_norsram_enable
Function prototype	void exmc_norsram_enable(uint32_t norsram_region);
Function descriptions	enable EXMC NOR/PSRAM bank region
Precondition	-
The called functions	-
Input parameter{in}	
norsram_region	specifie the region of NOR/PSRAM bank
EXMC_BANK0_NORSRAM_REGIONx(x=0..3)	region x of bank0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable region 0 of bank0 */

exmc_norsram_enable(EXMC_BANK0_NORSRAM_REGION0);

exmc_norsram_disable

The description of exmc_norsram_disable is shown as below:

Table 3-267. Function exmc_norsram_disable

Function name	exmc_norsram_disable
Function prototype	void exmc_norsram_disable(uint32_t norsram_region);
Function descriptions	disable EXMC NOR/PSRAM bank region
Precondition	-
The called functions	-
Input parameter{in}	
norsram_region	specifie the region of NOR/PSRAM bank
EXMC_BANK0_NORSRAM_REGIONx(x=0..3)	region x of bank0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable region 0 of bank0 */

exmc_norsram_disable(EXMC_BANK0_NORSRAM_REGION0);

exmc_nand_deinit

The description of exmc_nand_deinit is shown as below:

Table 3-268. Function exmc_nand_deinit

Function name	exmc_nand_deinit
Function prototype	void exmc_nand_deinit(uint32_t nand_bank);
Function descriptions	deinitialize EXMC NAND bank

Precondition	-
The called functions	-
Input parameter{in}	
nand_bank	select the bank of NAND
EXMC_BANKx_NAND(x=1,2)	bank of NAND
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize bank1 */
exmc_nand_deinit(EXMC_BANK1_NAND);
```

exmc_nand_init

The description of exmc_nand_init is shown as below:

Table 3-269. Function exmc_nand_init

Function name	exmc_nand_init
Function prototype	void exmc_nand_init(exmc_nand_parameter_struct* exmc_nand_init_struct);
Function descriptions	initialize EXMC NAND bank
Precondition	-
The called functions	-
Input parameter{in}	
exmc_nand_init_struct	Structure for initialization, the structure members can refer to ***Table 3-261. Structure exmc_nand_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize EXMC NAND bank */
exmc_nand_parameter_struct nand_init_struct;
exmc_nand_pccard_timing_parameter_struct nand_timing_init_struct;
nand_timing_init_struct.setuptime = 1;
nand_timing_init_struct.waittime = 3;
nand_timing_init_struct.holdtime = 2;
nand_timing_init_struct.databus_hiztime = 2;
nand_init_struct.nand_bank = EXMC_BANK1_NAND;
nand_init_struct.ecc_size = EXMC_ECC_SIZE_2048BYTES;
nand_init_struct.atr_latency = EXMC_ALE_RE_DELAY_1_HCLK;
nand_init_struct.ctr_latency = EXMC_CLE_RE_DELAY_1_HCLK;
nand_init_struct.ecc_logic = ENABLE;
nand_init_struct.databus_width = EXMC_NAND_DATABUS_WIDTH_8B;
nand_init_struct.wait_feature = ENABLE;
nand_init_struct.common_space_timing = &nand_timing_init_struct;
nand_init_struct.attribute_space_timing = &nand_timing_init_struct;
exmc_nand_init(&nand_init_struct);
```

exmc_nand_struct_para_init

The description of exmc_nand_struct_para_init is shown as below:

Table 3-270. Function exmc_nand_struct_para_init

Function name	exmc_nand_struct_para_init
Function prototype	void exmc_nand_struct_para_init(exmc_nand_parameter_struct* exmc_nand_init_struct);
Function descriptions	initialize the struct exmc_nand_parameter_struct
Precondition	-
The called functions	-
Input parameter{in}	
exmc_nand_init_struc	Structure for initialization, the structure members can refer to ***Table 3-261.***

t	*Structure exmc_nand_parameter_struct*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the struct nand_init_struct */
exmc_nand_parameter_struct nand_init_struct;
exmc_norsram_struct_para_init(&nor_init_struct);
```

exmc_nand_enable

The description of exmc_nand_enable is shown as below:

Table 3-271. Function exmc_nand_enable

Function name	exmc_nand_enable
Function prototype	void exmc_nand_enable(uint32_t nand_bank);
Function descriptions	enable NAND bank
Precondition	-
The called functions	-
Input parameter{in}	
nand_bank	specify the NAND bank
EXMC_BANKx_NAND(x=1,2)	bank of NAND
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable NAND bank */
exmc_nand_enable(EXMC_BANK1_NAND);
```

exmc_nand_disable

The description of exmc_nand_disable is shown as below:

Table 3-272. Function exmc_nand_disable

Function name	exmc_nand_disable
Function prototype	void exmc_nand_disable(uint32_t nand_bank);
Function descriptions	disable NAND bank
Precondition	-
The called functions	-
Input parameter{in}	
nand_bank	specifie the NAND bank
EXMC_BANKx_NAND(x=1,2)	bank of NAND
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable NAND bank */
exmc_nand_disable(EXMC_BANK1_NAND);
```

exmc_nand_ecc_config

The description of exmc_nand_ecc_config is shown as below:

Table 3-273. Function exmc_nand_ecc_config

Function name	exmc_nand_ecc_config
Function prototype	void exmc_nand_ecc_config(uint32_t nand_bank, ControlStatus newvalue);
Function descriptions	enable or disable the EXMC NAND ECC function
Precondition	-
The called functions	-
Input parameter{in}	
nand_bank	specify the NAND bank

EXMC_BANKx_NAND(x=1,2)	bank of NAND
Input parameter{in}	
newvalue	enable or disable
ENABLE	enable
DISABLE	disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the EXMC NAND ECC function */
exmc_nand_ecc_config(EXMC_BANK1_NAND, ENABLE);
```

exmc_ecc_get

The description of exmc_ecc_get is shown as below:

Table 3-274. Function exmc_ecc_get

Function name	exmc_ecc_get
Function prototype	uint32_t exmc_ecc_get(uint32_t nand_bank);
Function descriptions	get the EXMC ECC value
Precondition	-
The called functions	-
Input parameter{in}	
nand_bank	specifie the NAND bank
EXMC_BANKx_NAND(x=1,2)	bank of NAND
Output parameter{out}	
-	-
Return value	
uint32_t	0-0xFFFFFFFF

Example:

```
/* get the EXMC ECC value */

uint32_t value;

value = exmc_ecc_get(EXMC_BANK1_NAND);
```

exmc_pccard_deinit

The description of exmc_pccard_deinit is shown as below:

Table 3-275. Function exmc_pccard_deinit

Function name	exmc_pccard_deinit
Function prototype	void exmc_pccard_deinit(void);
Function descriptions	deinitialize EXMC PC card bank
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize EXMC PC card bank */

exmc_pccard_deinit();
```

exmc_pccard_init

The description of exmc_pccard_init is shown as below:

Table 3-276. Function exmc_pccard_init

Function name	exmc_pccard_init
Function prototype	void exmc_pccard_init(exmc_pccard_parameter_struct* exmc_pccard_init_struct);
Function descriptions	initialize EXMC PC card bank

Precondition	-
The called functions	-
Input parameter{in}	
exmc_pccard_init_struct	Structure for initialization, the structure members can refer to ***Table 3-262. Structure exmc_pccard_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize EXMC PC card bank */
exmc_pccard_parameter_struct pccard_init_struct;
exmc_nand_pccard_timing_parameter_struct pccard_timing_init_struct;
pccard_timing_init_struct.databus_hiztime = 2;
pccard_timing_init_struct.holdtime = 2;
pccard_timing_init_struct.waittime = 3;
pccard_timing_init_struct.setuptime = 1;
pccard_init_struct.atr_latency = EXMC_ALE_RE_DELAY_1_HCLK;
pccard_init_struct.ctr_latency = EXMC_CLE_RE_DELAY_1_HCLK;
pccard_init_struct.wait_feature = DISABLE;
pccard_init_struct.common_space_timing = &pccard _timing_init_struct;
pccard_init_struct.attribute_space_timing = &pccard _timing_init_struct;
pccard_init_struct.io_space_timing = &pccard _timing_init_struct;
exmc_pccard_init(&pccard_init_struct);
```

exmc_pccard_struct_para_init

The description of exmc_pccard_struct_para_init is shown as below:

Table 3-277. Function exmc_pccard_struct_para_init

Function name	exmc_pccard_ struct_para_init
Function prototype	void exmc_pccard_struct_para_init(exmc_pccard_parameter_struct*

	exmc_pccard_init_struct);
Function descriptions	initialize the struct exmc_pccard_parameter_struct
Precondition	-
The called functions	-
Input parameter{in}	
Output parameter{out}	
exmc_pccard_init_struct	Structure for initialization, the structure members can refer to ***Table 3-262. Structure exmc_pccard_parameter_struct***
Return value	
-	-

Example:

```
/* initialize the struct exmc_pccard_parameter_struct */
exmc_pccard_parameter_struct pccard_init_struct;
exmc_pccard_struct_para_init(&pccard_init_struct);
```

exmc_pccard_enable

The description of exmc_pccard_enable is shown as below:

Table 3-278. Function exmc_pccard_enable

Function name	exmc_pccard_enable
Function prototype	void exmc_pccard_enable(void);
Function descriptions	enable PC Card Bank
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable PC Card bank */

exmc_pccard_enable();

exmc_pccard_disable

The description of exmc_pccard_disable is shown as below:

Table 3-279. Function exmc_pccard_disable

Function name	exmc_pccard_disable
Function prototype	void exmc_pccard_disable(void);
Function descriptions	disable PC Card Bank
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable PC Card bank */

exmc_pccard_disable();

exmc_interrupt_enable

The description of exmc_interrupt_enable is shown as below:

Table 3-280. Function exmc_interrupt_enable

Function name	exmc_interrupt_enable
Function prototype	void exmc_interrupt_enable(uint32_t bank, uint32_t interrupt_source);
Function descriptions	enable EXMC interrupt
Precondition	-
The called functions	-

Input parameter{in}	
bank	specifies the NAND bank,PC card bank
EXMC_BANK1_NAND	the NAND bank1
EXMC_BANK2_NAND	the NAND bank2
EXMC_BANK3_PCCARD	the PC card bank
Input parameter{in}	
interrupt_source	specify get which interrupt flag
EXMC_NAND_PCCARD_INT_RISE	interrupt source of rising edge
EXMC_NAND_PCCARD_INT_LEVEL	interrupt source of high-level
EXMC_NAND_PCCARD_INT_FALL	interrupt source of falling edge
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable EXMC interrupt */
exmc_interrupt_enable(EXMC_BANK1_NAND, EXMC_NAND_PCCARD_INT_RISE);
```

exmc_interrupt_disable

The description of exmc_interrupt_disable is shown as below:

Table 3-281. Function exmc_interrupt_disable

Function name	exmc_interrupt_disable
Function prototype	void exmc_interrupt_disable(uint32_t bank, uint32_t interrupt_source);
Function descriptions	disable EXMC interrupt
Precondition	-
The called functions	-

Input parameter{in}	
bank	specifies the NAND bank,PC card bank
EXMC_BANK1_NAND	the NAND bank1
EXMC_BANK2_NAND	the NAND bank2
EXMC_BANK3_PCCARD	the PC card bank
Input parameter{in}	
interrupt_source	specify get which interrupt flag
EXMC_NAND_PCCARD_INT_RISE	interrupt source of rising edge
EXMC_NAND_PCCARD_INT_LEVEL	interrupt source of high-level
EXMC_NAND_PCCARD_INT_FALL	interrupt source of falling edge
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable EXMC interrupt */
exmc_interrupt_disable(EXMC_BANK1_NAND, EXMC_NAND_PCCARD_INT_RISE);
```

exmc_interrupt_flag_get

The description of exmc_interrupt_flag_get is shown as below:

Table 3-282. Function exmc_interrupt_flag_get

Function name	exmc_interrupt_flag_get
Function prototype	FlagStatus exmc_interrupt_flag_get(uint32_t bank, uint32_t interrupt_source);
Function descriptions	check EXMC interrupt flag is set or not
Precondition	-
The called functions	-

Input parameter{in}	
bank	specifies the NAND bank , PC card bank
EXMC_BANK1_NAND	the NAND bank1
EXMC_BANK2_NAND	the NAND bank2
EXMC_BANK3_PCCARD	the PC Card bank
Input parameter{in}	
interrupt_source	Interrupt flag
EXMC_NAND_PCCARD_INT_RISE	interrupt source of rising edge
EXMC_NAND_PCCARD_INT_LEVEL	interrupt source of high-level
EXMC_NAND_PCCARD_INT_FALL	interrupt source of falling edge
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check EXMC interrupt flag is set or not */

FlagStatus status;

status = exmc_interrupt_flag_get(EXMC_BANK1_NAND,
EXMC_NAND_PCCARD_INT_RISE);
```

exmc_interrupt_flag_clear

The description of exmc_interrupt_flag_clear is shown as below:

Table 3-283. Function exmc_interrupt_flag_clear

Function name	exmc_interrupt_flag_clear
Function prototype	void exmc_interrupt_flag_clear(uint32_t bank, uint32_t interrupt_source);
Function descriptions	clear EXMC interrupt flag
Precondition	-

The called functions	-
Input parameter{in}	
bank	specifies the NAND bank , PC card bank
EXMC_BANK1_NAND	the NAND bank1
EXMC_BANK2_NAND	the NAND bank2
EXMC_BANK3_PCCARD	the PC Card bank
Input parameter{in}	
interrupt_source	Interrupt flag
EXMC_NAND_PCCARD_INT_RISE	interrupt source of rising edge
EXMC_NAND_PCCARD_INT_LEVEL	interrupt source of high-level
EXMC_NAND_PCCARD_INT_FALL	interrupt source of falling edge
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear EXMC interrupt flag */

exmc_interrupt_flag_clear(EXMC_BANK1_NAND, EXMC_NAND_PCCARD_INT_RISE);

exmc_flag_get

The description of exmc_flag_get is shown as below:

Table 3-284. Function exmc_flag_get

Function name	exmc_flag_get
Function prototype	FlagStatus exmc_flag_get(uint32_t bank, uint32_t flag);
Function descriptions	check EXMC flag is set or not
Precondition	-

The called functions	-
Input parameter{in}	
bank	specifies the NAND bank , PC card bank
EXMC_BANK1_NAND	the NAND bank1
EXMC_BANK2_NAND	the NAND bank2
EXMC_BANK3_PCCARD	the PC Card bank
Input parameter{in}	
flag	flag
EXMC_NAND_PCCARD_FLAG_RISE	interrupt rising edge status
EXMC_NAND_PCCARD_FLAG_LEVEL	interrupt high-level status
EXMC_NAND_PCCARD_FLAG_FALL	interrupt falling edge status
EXMC_NAND_PCCARD_FLAG_FIFOE	FIFO empty flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check EXMC flag is set or not */
FlagStatus status;
status = exmc_flag_get(EXMC_BANK1_NAND, EXMC_NAND_PCCARD_FLAG_RISE);
```

exmc_flag_clear

The description of exmc_flag_clear is shown as below:

Table 3-285. Function exmc_flag_clear

Function name	exmc_flag_clear
Function prototype	void exmc_flag_clear(uint32_t bank, uint32_t flag);

Function descriptions	clear EXMC flag
Precondition	-
The called functions	-
Input parameter{in}	
bank	specifies the NAND bank , PC card bank
EXMC_BANK1_NAND	the NAND bank1
EXMC_BANK2_NAND	the NAND bank2
EXMC_BANK3_PCCARD	the PC Card bank
Input parameter{in}	
flag	flag
EXMC_NAND_PCCARD_FLAG_RISE	interrupt rising edge status
EXMC_NAND_PCCARD_FLAG_LEVEL	interrupt high-level status
EXMC_NAND_PCCARD_FLAG_FALL	interrupt falling edge status
EXMC_NAND_PCCARD_FLAG_FIFOE	FIFO empty flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear EXMC flag */
exmc_flag_clear(EXMC_BANK1_NAND, EXMC_NAND_PCCARD_FLAG_RISE);
```

3.11. EXTI

EXTI is the interrupt/event controller in the MCU. It contains up to 20 independent edge detectors and generates interrupt requests or events to the processer. The EXTI registers are listed in chapter ***3.11.1***, the EXTI firmware functions are introduced in chapter ***3.11.2***.

3.11.1. Descriptions of Peripheral registers

EXTI registers are listed in the table shown as below:

Table 3-286. 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

3.11.2. Descriptions of Peripheral functions

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

Table 3-287. EXTI firmware function

Function name	Function description
exti_deinit	reset EXTI
exti_init	initialize EXTI line x
exti_interrupt_enable	enable EXTI line x interrupt
exti_event_enable	enable EXTI line x event
exti_interrupt_disable	disable EXTI line x interrupt
exti_event_disable	disable EXTI line x event
exti_flag_get	get EXTI line x flag
exti_flag_clear	clear EXTI line x flag
exti_interrupt_flag_get	get EXTI line x interrupt flag
exti_interrupt_flag_clear	clear EXTI line x interrupt flag
exti_software_interrupt_enable	enable EXTI line x software interrupt
exti_software_interrupt_disable	disable EXTI line x software interrupt

exti_deinit

The description of exti_deinit is shown as below:

Table 3-288. Function exti_deinit

Function name	exti_deinit
Function prototype	void exti_deinit(void);
Function descriptions	reset EXTI. reset the value of all EXTI registers into initial values
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 3-289. 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 EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19

Input parameter{in}	
mode	EXTI mode
EXTI_INTERRUPT	interrupt mode
EXTI_EVENT	event mode
Input parameter{in}	
trig_type	trigger type
EXTI_TRIG_RISING	rising edge trigger
EXTI_TRIG_FALLING	falling edge trigger
EXTI_TRIG_BOTH	rising edge and falling edge trigger
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 3-290. Function exti_interrupt_enable

Function name	exti_interrupt_enable
Function prototype	void exti_interrupt_enable(exti_line_enum linex);
Function descriptions	enable EXTI line x interrupt
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* enable the interrupts from EXTI line 0 */

exti_interrupt_enable(EXTI_0);

exti_event_enable

The description of exti_event_enable is shown as below:

Table 3-291. Function exti_event_enable

Function name	exti_event_enable
Function prototype	void exti_event_enable(exti_line_enum linex);
Function descriptions	enable EXTI line x event
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the events from EXTI line 0 */

exti_event_enable(EXTI_0);

exti_interrupt_disable

The description of exti_interrupt_disable is shown as below:

Table 3-292. Function exti_interrupt_disable

Function name	exti_interrupt_disable

Function prototype	void exti_interrupt_disable(exti_line_enum linex);
Function descriptions	disable EXTI line x interrupt
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the interrupts from EXTI line 0 */
exti_interrupt_disable(EXTI_0);
```

exti_event_disable

The description of exti_event_disable is shown as below:

Table 3-293. Function exti_event_disable

Function name	exti_event_disable
Function prototype	void exti_event_disable(exti_line_enum linex);
Function descriptions	disable EXTI line x event
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* disable the events from EXTI line 0 */

exti_event_disable(EXTI_0);

exti_flag_get

The description of exti_flag_get is shown as below:

Table 3-294. 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 flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
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 3-295. 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 flag

Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
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 3-296. 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 flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
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 3-297. 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 flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear EXTI line 0 interrupt flag status */

exti_interrupt_flag_clear(EXTI_0);

exti_software_interrupt_enable

The description of exti_software_interrupt_enable is shown as below:

Table 3-298. 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 EXTI line x software interrupt
Precondition	-
The called functions	-

Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
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 3-299. 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 EXTI line x software interrupt
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x
EXTI_x	x=0,1,2..19
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable EXTI line 0 software interrupt */
exti_software_interrupt_disable(EXTI_0);
```

3.12. FMC

There is flash controller and option byte for GD32F10x series. The FMC registers are listed in chapter ***3.12.1*** the FMC firmware functions are introduced in chapter ***3.12.2***.

3.12.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

Table 3-300. FMC Registers

Registers	Descriptions
FMC_WS	Wait state register
FMC _KEY0	Unlock key register 0
FMC _OBKEY	Option byte unlock key register
FMC _STAT0	Status register 0
FMC _CTL0	Control register 0
FMC _ADDR0	Address register 0
FMC _OBSTAT	Option byte status register
FMC _WP	Erase/Program Protection register
FMC _KEY1	Unlock key register 1
FMC _STAT1	Status register 1
FMC _CTL1	Control register 1
FMC _ADDR1	Address register 1
FMC _WSEN	Wait state enable register
FMC _PID	Product ID register

3.12.2. Descriptions of Peripheral functions

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

Table 3-301. FMC firmware function

Function name	Function description
fmc_wscnt_set	set the FMC wait state counter
fmc_unlock	unlock the main FMC operation

Function name	Function description
fmc_bank0_unlock	unlock the FMC bank0 operation
fmc_bank1_unlock	unlock the FMC bank1 operation
fmc_lock	lock the main FMC operation
fmc_bank0_lock	lock the bank0 FMC operation
fmc_bank1_lock	lock the bank1 FMC operation
fmc_page_erase	FMC erase page
fmc_mass_erase	FMC erase whole chip
fmc_bank0_erase	FMC erase whole bank0
fmc_bank1_erase	FMC erase whole bank1
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_erase	erase the option byte
ob_write_protection_enable	enable write protection
ob_security_protection_config	configure the option byte security protection
ob_user_write	write the FMC user option byte
ob_data_program	program option bytes data
ob_user_get	get the FMC user option byte
ob_data_get	get OB_DATA in register FMC_OBSTAT
ob_write_protection_get	get the FMC option byte write protection
ob_spc_get	get option byte security protection code value
fmc_interrupt_enable	enable FMC interrupt
fmc_interrupt_disable	disable FMC interrupt
fmc_flag_get	check flag is set or not
fmc_flag_clear	clear the FMC flag
fmc_interrupt_flag_get	get FMC interrupt flag state

Function name	Function description
fmc_interrupt_flag_clear	clear FMC interrupt flag state
fmc_bank0_state_get	return the FMC bank0 state
fmc_bank1_state_get	return the FMC bank1 state
fmc_bank0_ready_wait	check FMC bank0 ready or not
fmc_bank1_ready_wait	check FMC bank1 ready or not

fmc_state_enum

Table 3-302. fmc_state_enum

enum name	enum description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_PGERR	program error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error

fmc_int_enum

Table 3-303. fmc_int_enum

enum name	enum description
FMC_INT_BANK0_END	enable FMC bank0 end of program interrupt
FMC_INT_BANK0_ERR	enable FMC bank0 error interrupt
FMC_INT_BANK1_END	enable FMC bank1 end of program interrupt
FMC_INT_BANK1_ERR	enable FMC bank1 error interrupt

fmc_flag_enum

Table 3-304. fmc_flag_enum

enum name	enum description
FMC_FLAG_BANK0	FMC bank0 busy flag

enum name	enum description
_BUSY	
FMC_FLAG_BANK0_PGERR	FMC bank0 operation error flag
FMC_FLAG_BANK0_WPERR	FMC bank0 erase/program protection error flag
FMC_FLAG_BANK0_END	FMC bank0 end of operation flag
FMC_FLAG_OBERR	FMC option bytes read error flag
FMC_FLAG_BANK1_BUSY	FMC bank1 busy flag
FMC_FLAG_BANK1_PGERR	FMC bank1 operation error flag
FMC_FLAG_BANK1_WPERR	FMC bank1 erase/program protection error flag
FMC_FLAG_BANK1_END	FMC bank1 end of operation flag

fmc_interrupt_flag_enum

Table 3-305. fmc_interrupt_flag_enum

enum name	enum description
FMC_INT_FLAG_BANK0_PGERR	FMC bank0 operation error interrupt flag
FMC_INT_FLAG_BANK0_WPERR	FMC bank0 erase/program protection error interrupt flag
FMC_INT_FLAG_BANK0_END	FMC bank0 end of operation interrupt flag
FMC_INT_FLAG_BANK1_PGERR	FMC bank1 operation error interrupt flag
FMC_INT_FLAG_BANK1_WPERR	FMC bank1 erase/program protection error interrupt flag
FMC_INT_FLAG_BANK1_END	FMC bank1 end of operation interrupt flag

fmc_wscnt_set

The description of fmc_wscnt_set is shown as below:

Table 3-306. 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	rcu_periph_reset_enable / rcu_periph_reset_disable
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_unlock

The description of fmc_unlock is shown as below:

Table 3-307. 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_bank0_unlock

The description of fmc_bank0_unlock is shown as below:

Table 3-308. Function fmc_bank0_unlock

Function name	fmc_bank0_unlock
Function prototype	void fmc_bank0_unlock(void);
Function descriptions	unlock the main FMC bank0 operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock the main FMC bank0 operation */
fmc_bank0_unlock( );
```

fmc_bank1_unlock

The description of fmc_bank1_unlock is shown as below:

Table 3-309. Function fmc_bank1_unlock

Function name	fmc_bank1_unlock
Function prototype	void fmc_bank1_unlock(void);
Function descriptions	unlock the main FMC bank1 operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock the main FMC bank1 operation */
fmc_bank1_unlock( );
```

fmc_lock

The description of fmc_lock is shown as below:

Table 3-310. 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_bank0_lock

The description of fmc_bank0_lock is shown as below:

Table 3-311. Function fmc_bank0_lock

Function name	fmc_bank0_lock
Function prototype	void fmc_bank0_lock(void);
Function descriptions	lock the main FMC bank0 operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* lock the main FMC bank0 operation */

fmc_bank0_lock( );

fmc_bank1_lock

The description of fmc_bank1_lock is shown as below:

Table 3-312. Function fmc_bank1_lock

Function name	fmc_bank1_lock
Function prototype	void fmc_bank1_lock(void);
Function descriptions	lock the main FMC bank1 operation
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the main FMC bank1 operation */

fmc_bank1_lock( );
```

fmc_page_erase

The description of fmc_page_erase is shown as below:

Table 3-313. 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_bank0_ready_wait / fmc_bank1_ready_wait
Input parameter{in}	
page_address	the page address to be erased
Output parameter{out}	
-	-
Return value	
fmc_state_enum	*state of FMC*

Example:

```
/* erase page */

fmc_state_enum state = fmc_page_erase( 0x08004000);
```

fmc_mass_erase

The description of fmc_mass_erase is shown as below:

Table 3-314. 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_bank0_ready_wait/ fmc_bank1_ready_wait
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* erase whole chip */

fmc_state_enum state = fmc_mass_erase( );

fmc_bank0_erase

The description of fmc_bank0_erase is shown as below:

Table 3-315. Function fmc_bank0_erase

Function name	fmc_bank0_erase
Function prototype	fmc_state_enum fmc_ bank0_erase(void);
Function descriptions	erase bank0 whole chip
Precondition	fmc_unlock
The called functions	fmc_bank0_ready_wait
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* erase bank0 whole chip */

fmc_state_enum state = fmc_bank0_erase( );

fmc_bank1_erase

The description of fmc_bank1_erase is shown as below:

Table 3-316. Function fmc_bank1_erase

Function name	fmc_bank1_erase
Function prototype	fmc_state_enum fmc_bank1_erase( );
Function descriptions	erase bank1 whole chip
Precondition	fmc_unlock
The called functions	fmc_bank1_ready_wait
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* erase bank1 whole chip */

fmc_state_enum state = fmc_bank1_erase( );

fmc_word_program

The description of fmc_word_program is shown as below:

Table 3-317. 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_bank0_ready_wait / fmc_bank1_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*

Example:

/* program a word at the corresponding address */

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

fmc_halfword_program

The description of fmc_halfword_program is shown as below:

Table 3-318. 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 halfword at the corresponding address
Precondition	fmc_unlock
The called functions	fmc_bank0_ready_wait/ fmc_bank1_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*

Example:

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

fmc_state_enum state = fmc_halfword_program(0x08004000,0xaabb);

ob_unlock

The description of ob_unlock is shown as below:

Table 3-319. 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 3-320. Function ob_lock

Function name	ob_lock
Function prototype	void ob_lock(void);
Function descriptions	lock the option byte operation
Precondition	fmc_lock
The called functions	-

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

Example:

/* lock the option byte operation */

ob_lock( );

ob_erase

The description of ob_erase is shown as below:

Table 3-321. Function ob_erase

Function name	ob_erase
Function prototype	void ob_erase(void);
Function descriptions	erase the FMC option byte
Precondition	ob_unlock
The called functions	fmc_bank0_ready_wait
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* erase the FMC option byte */

ob_ erase( );

ob_write_protection_enable

The description of ob_write_protection_enable is shown as below:

Table 3-322. Function ob_write_protection_enable

Function name	ob_write_protection_enable
Function prototype	fmc_state_enum ob_write_protection_enable(uint32_t ob_wp);
Function descriptions	enable write protection
Precondition	ob_unlock
The called functions	fmc_bank0_ready_wait
Input parameter{in}	
ob_wp	enable write protection
OB_WPx	write protect specify sector x(x=0..31)
OB_WP_ALL	write protect all sector
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* enable write protection */

fmc_state_enum state = ob_write_protection_enable(OB_WP7);

ob_security_protection_config

The description of ob_security_protection_config is shown as below:

Table 3-323. 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_bank0_ready_wait
Input parameter{in}	
ob_spc	specify security protection
FMC_NSPC	no security protection

FMC_USPC	under security protection
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* enable security protection */

fmc_state_enum state = ob_security_protection_config(FMC_USPC);

ob_user_write

The description of ob_user_write is shown as below:

Table 3-324. Function ob_user_write

Function name	ob_user_write
Function prototype	fmc_state_enum ob_user_write(uint8_t ob_fwdgt, uint8_t ob_deepsleep, uint8_t ob_stdby, uint8_t ob_boot);
Function descriptions	program the FMC user option byte
Precondition	ob_unlock
The called functions	fmc_bank0_ready_wait
Input parameter{in}	
ob_fwdgt	option byte watchdog value
OB_FWDGT_SW	software free watchdog
OB_FWDGT_HW	hardware free watchdog
Input parameter{in}	
ob_deepsleep	option byte deepsleep reset value
OB_DEEPSLEEP_NRST	no reset when entering deepsleep mode
OB_DEEPSLEEP_RST	generate a reset instead of entering deepsleep mode
Input parameter{in}	
ob_stdby	option byte standby reset value
OB_STDBY_NRST	no reset when entering standby mode

OB_STDBY_RST	generate a reset instead of entering standby mode
Input parameter{in}	
ob_boot	specifies the option byte boot bank value
OB_BOOT_B0	boot from bank0
OB_BOOT_B1	boot from bank1 or bank0 if bank1 is void
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* configure user option byte */

fmc_state_enum state = ob_user_write(OB_FWDGT_HW, OB_DEEPSLEEP_RST, OB_STDBY_RST, OB_BOOT_B1);

ob_data_program

The description of ob_data_program is shown as below:

Table 3-325. Function ob_data_program

Function name	ob_data_program
Function prototype	fmc_state_enum ob_data_program(uint32_t address, uint8_t data);
Function descriptions	program option bytes data
Precondition	ob_unlock
The called functions	fmc_bank0_ready_wait
Input parameter{in}	
address	0x1ffff804 / 0x1ffff806
data	the byte to be programmed
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* program option bytes data */

fmc_state_enum state = ob_data_program(0x1ffff804, 0x56);

ob_user_get

The description of ob_user_get is shown as below:

Table 3-326. Function ob_user_get

Function name	ob_user_get
Function prototype	uint8_t ob_user_get(void);
Function descriptions	get the FMC user option byte
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	the FMC user option byte values(0xF0 – 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 3-327. Function ob_data_get

Function name	ob_data_get
Function prototype	uint8_t ob_data_get(void);
Function descriptions	get the FMC data option byte
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	the FMC data option byte values(0x0 – 0xFF)

Example:

/* get the FMC data option byte */

uint8_t data = ob_data_get( );

ob_write_protection_get

The description of ob_ write_protection_get is shown as below:

Table 3-328. Function ob_write_protection_get

Function name	ob_ write_protection_get
Function prototype	uint32_t ob_write_protection_get(void);
Function descriptions	get the FMC option byte write protection
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	the FMC write protection option byte value(0x0 – 0XFFFFFFFF)

Example:

/* get the FMC option byte write protection */

uint32_t wp = ob_write_protection_get( );

ob_spc_get

The description of ob_spc_get is shown as below:

Table 3-329. Function ob_spc_get

Function name	ob_spc_get
Function prototype	FlagStatus ob_spc_get(void);
Function descriptions	get the FMC option byte security protection
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the FMC option byte security protection */

FlagStatus spc = ob_spc_get( );

fmc_interrupt_enable

The description of fmc_interrupt_enable is shown as below:

Table 3-330. Function fmc_interrupt_enable

Function name	fmc_interrupt_enable
Function prototype	void fmc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable FMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the FMC interrupt source
FMC_INT_BANK0_END	FMC bank0 end of program interrupt
FMC_INT_BANK0_ERR	FMC bank0 error interrupt

FMC_INT_BANK1_END	FMC bank1 end of program interrupt
FMC_INT_BANK1_ERR	FMC bank1 error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable FMC interrupt */

fmc_interrupt_enable(FMC_INT_BANK0_END);
```

fmc_interrupt_disable

The description of fmc_interrupt_disable is shown as below:

Table 3-331. Function fmc_interrupt_disable

Function name	fmc_interrupt_disable
Function prototype	void fmc_interrupt_disable(uint32_t interrupt);
Function descriptions	disable FMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the FMC interrupt source
FMC_INT_BANK0_END	FMC bank0 end of program interrupt
FMC_INT_BANK0_ERR	FMC bank0 error interrupt
FMC_INT_BANK1_END	FMC bank1 end of program interrupt
FMC_INT_BANK1_ERR	FMC bank1 error interrupt
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable FMC interrupt */
fmc_interrupt_disable(FMC_INT_BANK0_END);
```

fmc_flag_get

The description of fmc_flag_get is shown as below:

Table 3-332. Function fmc_flag_get

Function name	fmc_flag_get
Function prototype	FlagStatus fmc_flag_get(uint32_t flag);
Function descriptions	check flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
flag	check FMC flag
FMC_FLAG_BANK0_BUSY	FMC bank0 busy flag bit
FMC_FLAG_BANK0_PGERR	FMC bank0 operation error flag bit
FMC_FLAG_BANK0_WPERR	FMC bank0 erase/program protection error flag bit
FMC_FLAG_BANK0_END	FMC bank0 end of operation flag bit
FMC_FLAG_BANK1_BUSY	FMC bank1 busy flag bit
FMC_FLAG_BANK1_PGERR	FMC bank1 operation error flag bit
FMC_FLAG_BANK1_WPERR	FMC bank1 erase/program protection error flag bit

FMC_FLAG_BANK1_END	FMC bank1 end of operation flag bit
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get FMC flag */

FlagStatus flag = fmc_flag_get(FMC_FLAG_BANK0_END);

fmc_flag_clear

The description of fmc_flag_clear is shown as below:

Table 3-333. Function fmc_flag_clear

Function name	fmc_flag_clear
Function prototype	void fmc_flag_clear(uint32_t flag);
Function descriptions	clear the FMC flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	clear FMC flag
FMC_FLAG_BANK0_PGERR	FMC bank0 operation error flag bit
FMC_FLAG_BANK0_WPERR	FMC bank0 erase/program protection error flag bit
FMC_FLAG_BANK0_END	FMC bank0 end of operation flag bit
FMC_FLAG_BANK1_PGERR	FMC bank1 operation error flag bit
FMC_FLAG_BANK1_WPERR	FMC bank1 erase/program protection error flag bit
FMC_FLAG_BANK1_E	FMC bank1 end of operation flag bit

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

Example:

/* get FMC flag */

fmc_flag_clear(FMC_FLAG_BANK0_END);

fmc_interrupt_flag_get

The description of fmc_interrupt_flag_get is shown as below:

Table 3-334. Function fmc_interrupt_flag_get

Function name	fmc_interrupt_flag_get
Function prototype	FlagStatus fmc_interrupt_flag_get(fmc_interrupt_flag_enum flag);
Function descriptions	get FMC interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	FMC flag
FMC_INT_FLAG_BANK0_PGERR	FMC bank0 operation error flag bit
FMC_INT_FLAG_BANK0_WPERR	FMC bank0 erase/program protection error flag bit
FMC_INT_FLAG_BANK0_END	FMC bank0 end of operation flag bit
FMC_INT_FLAG_BANK1_PGERR	FMC bank1 operation error flag bit
FMC_INT_FLAG_BANK1_WPERR	FMC bank1 erase/program protection error flag bit

FMC_INT_FLAG_BANK1_END	FMC bank1 end of operation flag bit
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get FMC flag */

FlagStatus flag = fmc_interrupt_flag_get(FMC_INT_FLAG_BANK0_PGERR);

fmc_interrupt_flag_clear

The description of fmc_interrupt_flag_get is shown as below:

Table 3-335. Function fmc_interrupt_flag_clear

Function name	fmc_interrupt_flag_ clear
Function prototype	FlagStatus fmc_interrupt_flag_ clear (fmc_interrupt_flag_enum flag);
Function descriptions	clear FMC interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	clear FMC flag
FMC_INT_FLAG_BANK0_PGERR	FMC bank0 operation error flag bit
FMC_INT_FLAG_BANK0_WPERR	FMC bank0 erase/program protection error flag bit
FMC_INT_FLAG_BANK0_END	FMC bank0 end of operation flag bit
FMC_INT_FLAG_BANK1_PGERR	FMC bank1 operation error flag bit
*FMC_*	FMC bank1 erase/program protection error flag bit

INT_FLAG_BANK1_WPERR	
FMC_INT_FLAG_BANK1_END	FMC bank1 end of operation flag bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear FMC flag */

fmc_interrupt_flag_get(FMC_INT_FLAG_BANK0_PGERR);

fmc_bank0_state_get

The description of fmc_bank0_state_get is shown as below:

Table 3-336. Function fmc_bank0_state_get

Function name	fmc_bank0_state_get
Function prototype	fmc_state_enum fmc_bank0_state_get(void);
Function descriptions	get the FMC bank0 state
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* get the FMC bank0 state */

fmc_state_enum state = fmc_bank0_state_get( );

fmc_bank1_state_get

The description of fmc_bank1_state_get is shown as below:

Table 3-337. Function fmc_bank1_state_get

Function name	fmc_bank1_state_get
Function prototype	fmc_state_enum fmc_bank1_state_get(void);
Function descriptions	get the FMC bank1 state
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

```
/* get the FMC bank1 state */
fmc_state_enum state = fmc_bank1_state_get( );
```

fmc_bank0_ready_wait

The description of fmc_bank0_ready_waitis shown as below:

Table 3-338. Function fmc_bank0_ready_wait

Function name	fmc_bank0_ready_wait
Function prototype	fmc_state_enum fmc_bank0_ready_wait(uint32_t timeout);
Function descriptions	check whether FMC bank0 is ready or not
Precondition	-
The called functions	fmc_bank0_state_get()
Input parameter{in}	
timeout	count of loop
Output parameter{out}	

-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* check whether FMC bank0 is ready or not */

fmc_state_enum state = fmc_bank0_ready_wait(0x00001000);

fmc_bank1_ready_wait

The description of fmc_bank1_ready_waitis shown as below:

Table 3-339. Function fmc_bank0_ready_wait

Function name	fmc_bank1_ready_wait
Function prototype	fmc_state_enum fmc_bank1_ready_wait(uint32_t timeout);
Function descriptions	check whether FMC bank1 is ready or not
Precondition	-
The called functions	fmc_bank1_state_get
Input parameter{in}	
timeout	count of loop
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

/* check whether FMC bank1 is ready or not */

fmc_state_enum state = fmc_bank1_ready_wait(0x00001000);

3.13. 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 ***3.13.2***.

3.13.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

Table 3-340. FWDGT Registers

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

3.13.2. Descriptions of Peripheral functions

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

Table 3-341. 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 free watchdog timer counter
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 3-342. Function fwdgt_write_enable

Function name	fwdgt_write_enable
Function prototype	void fwdgt_write_enable(void);
Function descriptions	enable write access to FWDGT_PSC and FWDGT_RLD
Precondition	-
The called functions	-
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 3-343. 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	-
The called functions	-
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 3-344. Function fwdgt_enable

Function name	fwdgt_enable
Function prototype	void fwdgt_enable(void);
Function descriptions	start the free watchdog timer counter
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* start the free watchdog timer counter */

fwdgt_enable( );

fwdgt_counter_reload

The description of fwdgt_counter_reload is shown as below:

Table 3-345. Function fwdgt_counter_reload

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

-	-

Example:

/* reload FWDGT counter */

fwdgt_counter_reload( );

fwdgt_config

The description of fwdgt_config is shown as below:

Table 3-346. Function fwdgt_config

Function name	fwdgt_config
Function prototype	ErrStatus fwdgt_config(uint16_t reload_value, uint8_t prescaler_div);
Function descriptions	configure counter reload value, and prescaler divider value
Precondition	-
The called functions	-
Input parameter{in}	
reload_value	specify reload value(0x0000 - 0x0FFF)-
Input parameter{in}	
prescaler_div	FWDGT prescaler value-
FWDGT_PSC_DIV4	FWDGT prescaler set to 4
FWDGT_PSC_DIV8	FWDGT prescaler set to 8
FWDGT_PSC_DIV16	FWDGT prescaler set to 16
FWDGT_PSC_DIV32	FWDGT prescaler set to 32
FWDGT_PSC_DIV64	FWDGT prescaler set to 64
FWDGT_PSC_DIV128	FWDGT prescaler set to 128
FWDGT_PSC_DIV256	FWDGT prescaler set to 256
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

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

ErrStatus status;

status = fwdgt_config(2*500, FWDGT_PSC_DIV64);
```

fwdgt_flag_get

The description of fwdgt_flag_get is shown as below:

Table 3-347. Function fwdgt_flag_get fwdgt_write_disable

Function name	fwdgt_flag_get
Function prototype	FlagStatus fwdgt_flag_get(uint16_t flag);
Function descriptions	get flag state of FWDGT
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag to get
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
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);

if(status == RESET)

{

...

}else

{

...
```

}

3.14. GPIO

GPIO is used to implement logic input/output functions for the devices. The ADC registers are listed in chapter ***3.14.1***, the ADC firmware functions are introduced in chapter ***3.14.2***.

3.14.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

Table 3-348. GPIO Registers

Registers	Descriptions
GPIOx_CTL0	Port control register 0
GPIOx_CTL1	Port control register 1
GPIOx_ISTAT	Port input status register
GPIOx_OCTL	Port output control register
GPIOx_BOP	Port bit operate register
GPIOx_BC	Port bit clear register
GPIOx_LOCK	Port configuration lock register
AFIO_EC	Event control register
AFIO_PCF0	AFIO port configuration register 0
AFIO_EXTISS0	EXTI sources selection register 0
AFIO_EXTISS1	EXTI sources selection register 1
AFIO_EXTISS2	EXTI sources selection register 2
AFIO_EXTISS3	EXTI sources selection register 3
AFIO_PCF1	AFIO port configuration register 1

3.14.2. Descriptions of Peripheral functions

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

Table 3-349. GPIO firmware function

Function name	Function description
gpio_deinit	reset GPIO port

Function name	Function description
gpio_afio_deinit	reset alternate function I/O(AFIO)
gpio_init	GPIO parameter initialization
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_pin_remap_config	configure GPIO pin remap
gpio_exti_source_select	select GPIO pin exti sources
gpio_event_output_config	configure GPIO pin event output
gpio_event_output_enable	enable GPIO pin event output
gpio_event_output_disable	disable GPIO pin event output
gpio_pin_lock	lock GPIO pin bit
gpio_ethernet_phy_select	select ethernet MII or RMII PHY

gpio_deinit

The description of gpio_deinit is shown as below:

Table 3-350. Function gpio_deinit

Function name	gpio_deinit
Function prototype	void gpio_deinit(uint32_t gpio_periph);
Function descriptions	reset GPIO port
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
gpio_periph	GPIO port

GPIOx	GPIOx(x = A,B,C,D,E,F,G)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset GPIOA */
gpio_deinit(GPIOA);
```

gpio_afio_deinit

The description of gpio_afio_deinit is shown as below:

Table 3-351. Function gpio_afio_deinit

Function name	gpio_afio_deinit
Function prototype	void gpio_afio_deinit(void);
Function descriptions	reset alternate function I/O(AFIO)
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset alternate function */
gpio_afio_deinit();
```

gpio_init

The description of gpio_init is shown as below:

Table 3-352. Function gpio_init

Function name	gpio_init
Function prototype	void gpio_init(uint32_t gpio_periph,uint32_t mode,uint32_t speed,uint32_t pin);
Function descriptions	GPIO parameter initialization
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G)
Input parameter{in}	
mode	gpio pin mode
GPIO_MODE_AIN	analog input mode
GPIO_MODE_IN_FLOATING	floating input mode
GPIO_MODE_IPD	pull-down input mode
GPIO_MODE_IPU	pull-up input mode
GPIO_MODE_OUT_OD	GPIO output with open-drain
GPIO_MODE_OUT_PP	GPIO output with push-pull
GPIO_MODE_AF_OD	AFIO output with open-drain
GPIO_MODE_AF_PP	AFIO output with push-pull
Input parameter{in}	
speed	gpio output max speed value
GPIO_OSPEED_10MHZ	output max speed 10MHz
GPIO_OSPEED_2MHZ	output max speed 2MHz
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 analog input mode*/

gpio_init(GPIOA, GPIO_MODE_AIN, GPIO_OSPEED_50MHZ, GPIO_PIN_0);

gpio_bit_set

The description of gpio_bit_set is shown as below:

Table 3-353. 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
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G)
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 3-354. 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 bit
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G)
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 3-355. 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,E,F,G)
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 3-356. 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,E,F,G)
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 to Port A */

gpio_port_write (GPIOA, 0xA5);

gpio_input_bit_get

The description of gpio_input_bit_get is shown as below:

Table 3-357. 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,E,F,G)
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;

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 3-358. 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 port input status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G)
Output parameter{out}	
-	-
Return value	
uint16_t	0x00-0xFF

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 3-359. Function gpio_output_bit_get

Function name	gpio_output_bit_get
Function prototype	FlagStatus gpio_output_bit_get(uint32_t gpio_periph,uint32_t pin);
Function descriptions	get GPIO pin output status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G)
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 3-360. 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 port output status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G)
Output parameter{out}	
-	-
Return value	
uint16_t	0x00-0xFF

Example:

```
/* get output value of Port A */
uint16_t port_state;
port_state = gpio_output_port_get (GPIOA);
```

gpio_pin_remap_config

The description of gpio_pin_remap_config is shown as below:

Table 3-361. Function gpio_pin_remap_config

Function name	gpio_pin_remap_config
Function prototype	void gpio_pin_remap_config(uint32_t gpio_remap, ControlStatus newvalue);
Function descriptions	configure GPIO pin remap
Precondition	-
The called functions	-
Input parameter{in}	
gpio_remap	select the pin to remap
GPIO_SPI0_REMAP	SPI0 remapping

GPIO_I2C0_REMAP	I2C0 remapping
GPIO_USART0_REMAP	USART0 remapping
GPIO_USART1_REMAP	USART1 remapping
GPIO_USART2_PARTIAL_REMAP	USART2 partial remapping
GPIO_USART2_FULL_REMAP	USART2 full remapping
GPIO_TIMER0_PARTIAL_REMAP	TIMER0 partial remapping
GPIO_TIMER0_FULL_REMAP	TIMER0 full remapping
GPIO_TIMER1_PARTIAL_REMAP1	TIMER1 partial remapping
GPIO_TIMER1_PARTIAL_REMAP2	TIMER1 partial remapping
GPIO_TIMER1_FULL_REMAP	TIMER1 full remapping
GPIO_TIMER2_PARTIAL_REMAP	TIMER2 partial remapping
GPIO_TIMER2_FULL_REMAP	TIMER2 full remapping
GPIO_TIMER3_REMAP	TIMER3 remapping
GPIO_CAN_PARTIAL_REMAP	CAN partial remapping(only for GD32F10X_MD, GD32F10X_HD devices and GD32F10X_XD devices)
GPIO_CAN_FULL_REMAP	CAN full remapping(only for GD32F10X_MD, GD32F10X_HD devices and GD32F10X_XD devices)
GPIO_CAN0_PARTIAL_REMAP	CAN0 partial remapping(only for GD32F10X_CL devices)
GPIO_CAN0_FULL_REMAP	CAN0 full remapping(only for GD32F10X_CL devices)
GPIO_PD01_REMAP	PD01 remapping

GPIO_TIMER4CH3_IREMAP	TIMER4 channel3 internal remapping(only for GD32F10X_CL devices and GD32F10X_HD devices)
GPIO_ADC0_ETRGINS_REMAP	ADC0 external trigger inserted conversion remapping(only for GD32F10X_MD, GD32F10X_HD devices and GD32F10X_XD devices)
GPIO_ADC0_ETRGREG_REMAP	ADC0 external trigger regular conversion remapping(only for GD32F10X_MD, GD32F10X_HD devices and GD32F10X_XD devices)
GPIO_ADC1_ETRGINS_REMAP	ADC1 external trigger inserted conversion remapping(only for GD32F10X_MD, GD32F10X_HD devices and GD32F10X_XD devices)
GPIO_ADC1_ETRGREG_REMAP	ADC1 external trigger regular conversion remapping(only for GD32F10X_MD, GD32F10X_HD devices and GD32F10X_XD devices)
GPIO_ENET_REMAP	ENET remapping(only for GD32F10X_CL devices)
GPIO_CAN1_REMAP	CAN1 remapping(only for GD32F10X_CL devices)
GPIO_SWJ_NONJTRST_REMAP	full SWJ(JTAG-DP + SW-DP),but without NJTRST
GPIO_SWJ_SWDPENABLE_REMAP	JTAG-DP disabled and SW-DP enabled
GPIO_SWJ_DISABLE_REMAP	JTAG-DP disabled and SW-DP disabled
GPIO_SPI2_REMAP	SPI2 remapping(only for GD32F10X_CL devices)
GPIO_TIMER1ITI1_REMAP	TIMER1 internal trigger 1 remapping(only for GD32F10X_CL devices)
GPIO_PTP_PPS_REMAP	ethernet PTP PPS remapping(only for GD32F10X_CL devices)
GPIO_TIMER8_REMAP	TIMER8 remapping
GPIO_TIMER9_REMAP	TIMER9 remapping
GPIO_TIMER10_REMAP	TIMER10 remapping
GPIO_TIMER12_REMAP	TIMER12 remapping
GPIO_TIMER13_REMAP	TIMER13 remapping
GPIO_EXMC_NADV_R	EXMC_NADV connect/disconnect

EMAP	
Input parameter{in}	
newvalue	ENABLE or DISABLE
ENABLE	*enable*
DISABLE	*disable*
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*enable SPI0 remapping*/

gpio_pin_remap_config (GPIO_SPI0_REMAP, ENABLE);

gpio_exti_source_select

The description of gpio_exti_source_select is shown as below:

Table 3-362. Function gpio_exti_source_select

Function name	gpio_exti_source_select
Function prototype	void gpio_exti_source_select(uint8_t gpio_outputport,uint8_t gpio_outputpin);
Function descriptions	select GPIO pin exti sources
Precondition	-
The called functions	-
Input parameter{in}	
gpio_outputport	gpio event output port
GPIO_PORT_SOURCE_GPIOx	output port source (x=A..G)
Input parameter{in}	
gpio_outputpin	gpio event output pin
GPIO_PIN_SOURCE_x	Pin number(x=0..15)
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* config PA0 as EXTI source*/
gpio_exti_source_select(GPIO_PORT_SOURCE_GPIOA, GPIO_PIN_SOURCE_0);
```

gpio_event_output_config

The description of gpio_event_output_config is shown as below:

Table 3-363. Function gpio_event_output_config

Function name	gpio_event_output_config
Function prototype	void gpio_event_output_config(uint8_t gpio_outputport,uint8_t gpio_outputpin);
Function descriptions	configure GPIO pin event output
Precondition	-
The called functions	-
Input parameter{in}	
gpio_outputport	gpio event output port
GPIO_EVENT_PORT_GPIOx	event output port x (x=A..G)
Input parameter{in}	
gpio_outputpin	gpio event output pin
GPIO_EVENT_PIN_x	Pin number (x=0..15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* Config PA0 as the output of event */
gpio_event_output_config(GPIO_EVENT_PORT_GPIOA, GPIO_EVENT_PIN_0);
```

gpio_event_output_enable

The description of gpio_event_output_enable is shown as below:

Table 3-364. Function gpio_event_output_enable

Function name	gpio_event_output_enable
Function prototype	void gpio_event_output_enable(void);
Function descriptions	enable GPIO pin event output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable GPIO pin event output */

gpio_event_output_enable();
```

gpio_event_output_disable

The description of gpio_event_output_disable is shown as below:

Table 3-365. Function gpio_event_output_disable

Function name	gpio_event_output_disable
Function prototype	void gpio_event_output_disable(void);
Function descriptions	disable GPIO pin event output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable GPIO pin event output */
gpio_event_output_disable();
```

gpio_pin_lock

The description of gpio_pin_lock is shown as below:

Table 3-366. 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
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	*GPIOx(x = A,B,C,D,E,F,G)*
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);
```

gpio_ethernet_phy_select

The description of gpio_ethernet_phy_select is shown as below:

Table 3-367. Function gpio_ethernet_phy_select

Function name	gpio_ethernet_phy_select
Function prototype	void gpio_ethernet_phy_select(uint32_t gpio_enetsel);
Function descriptions	select ethernet MII or RMII PHY
Precondition	-
The called functions	-
Input parameter{in}	
gpio_enetsel	ethernet MII or RMII PHY selection
GPIO_ENET_PHY_MII	configure ethernet MAC for connection with an MII PHY
GPIO_ENET_PHY_RMII	configure ethernet MAC for connection with an RMII PHY
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ethernet MAC for connection with an RMII PHY */
gpio_ethernet_phy_select(GPIO_ENET_PHY_RMII);
```

3.15. 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 ***3.15.1***, the I2C firmware functions are introduced in chapter ***3.15.2***.

3.15.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

Table 3-368. 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

3.15.2. Descriptions of Peripheral functions

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

Table 3-369. 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	SMBus type selection
i2c_ack_config	whether or not to send an ACK
i2c_ackpos_config	configure I2C POAP position
i2c_master_addressing	master send slave address
i2c_dualaddr_enable	dual-address mode switch
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

Function name	Function description
i2c_data_receive	I2C receive data function
i2c_dma_enable	I2C DMA mode enable
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	software reset I2C
i2c_pec_enable	I2C PEC calculation on or off
i2c_pec_transfer_enable	I2C whether to transfer PEC value
i2c_pec_value_get	packet error checking value
i2c_smbus_issue_alert	I2C issue alert through SMBA pin
i2c_smbus_arp_enable	I2C ARP protocol in SMBus switch
i2c_flag_get	check I2C flag is set or not
i2c_flag_clear	clear I2C flag
i2c_interrupt_enable	enable I2C interrupt
i2c_interrupt_disable	disable I2C interrupt
i2c_interrupt_flag_get	check I2C interrupt flag
i2c_interrupt_flag_clear	clear I2C interrupt flag

i2c_deinit

The description of i2c_deinit is shown as below:

Table 3-370. 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)
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 3-371. 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	I2C clock configure
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
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 3-372. 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)
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	7bits
I2C_ADDFORMAT_10BITS	10bits

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 3-373. 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	SMBus type selection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
type	Device or host
I2C_SMBUS_DEVICE	device
I2C_SMBUS_HOST	host
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 3-374. 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)
Input parameter{in}	
ack	I2C peripheral
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 3-375. Function i2c_ackpos_config

Function name	i2c_ackpos_config
Function prototype	void i2c_ackpos_config(uint32_t i2c_periph, uint32_t pos);
Function descriptions	I2C POAP position configure
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
pos	ACK position
I2C_ACKPOS_CURRENT	whether to send ACK or not for the current
I2C_ACKPOS_NEXT	whether to send ACK or not 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 3-376. 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)
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 3-377. Function i2c_dualaddr_enable

Function name	i2c_dualaddr_enable
Function prototype	void i2c_dualaddr_enable(uint32_t i2c_periph, uint32_t dualaddr);
Function descriptions	dual-address mode switch
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)

Input parameter{in}	
dualaddr	Enable or disable
I2C_DUADEN_DISABLE	disable dual-address mode
I2C_DUADEN_ENABLE	enable dual-address mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C0 dual-address*/
i2c_dualaddr_enable(I2C0, I2C_DUADEN_ENABLE);
```

i2c_enable

The description of i2c_enable is shown as below:

Table 3-378. 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)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable I2C0 */

i2c_enable(I2C0);

i2c_disable

The description of i2c_disable is shown as below:

Table 3-379. 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)
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 3-380. 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)
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 3-381. 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)
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 3-382. 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)
Input parameter{in}	
data	transmit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 transmit data */

i2c_data_transmit(I2C0);
```

i2c_data_receive

The description of i2c_data_receive is shown as below:

Table 3-383. 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)
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_enable

The description of i2c_dma_enable is shown as below:

Table 3-384. Function i2c_dma_enable

Function name	i2c_dma_enable
Function prototype	void i2c_dma_enable(uint32_t i2c_periph, uint32_t dmastate);
Function descriptions	enable I2C DMA mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
dmastate	on or off
I2C_DMA_ON	DMA mode enable
I2C_DMA_OFF	DMA mode disable
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* I2C0 DMA mode enable */

i2c_dma_enable(I2C0, I2C_DMA_ON);

i2c_dma_last_transfer_config

The description of i2c_dma_last_transfer_config is shown as below:

Table 3-385. 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)
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 3-386. 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)
Input parameter{in}	
stretchpara	SCL stretching enable or disable
I2C_SCLSTRETCH_ENABLE	SCL stretching is enabled
I2C_SCLSTRETCH_DISABLE	SCL stretching is disabled
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 3-387. 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)
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 3-388. 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	software reset I2C
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral

I2Cx	(x=0,1)
Input parameter{in}	
sreset	under 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_enable

The description of i2c_pec_enable is shown as below:

Table 3-389. Function i2c_pec_enable

Function name	i2c_pec_enable
Function prototype	void i2c_pec_enable(uint32_t i2c_periph, uint32_t pecstate);
Function descriptions	I2C PEC calculation on or off
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
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_enable(I2C0, I2C_PEC_ENABLE);

i2c_pec_transfer_enable

The description of i2c_pec_transfer_enable is shown as below:

Table 3-390. Function i2c_pec_transfer_enable

Function name	i2c_pec_transfer_enable
Function prototype	void i2c_pec_transfer_enable(uint32_t i2c_periph, uint32_t pecpara);
Function descriptions	I2C whether to transfer PEC value
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
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_enable(I2C0, I2C_PECTRANS_ENABLE);

i2c_pec_value_get

The description of i2c_pec_value_get is shown as below:

Table 3-391. 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)
Output parameter{out}	
-	-
Return value	
uint8_t	0..255

Example:

/* I2C0 get packet error checking value */

uint8_t pec_value;

pec_value = i2c_pec_value_get(I2C0);

i2c_smbus_issue_alert

The description of i2c_smbus_issue_alert is shown as below:

Table 3-392. Function i2c_smbus_issue_alert

Function name	i2c_smbus_issue_alert
Function prototype	void i2c_smbus_issue_alert(uint32_t i2c_periph, uint32_t smbuspara);
Function descriptions	I2C issue alert through SMBA pin
Precondition	-
The called functions	-

Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
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_issue_alert(I2C0, I2C_SALTSEND_ENABLE);

i2c_smbus_arp_enable

The description of i2c_smbus_arp_enable is shown as below:

Table 3-393. Function i2c_smbus_arp_enable

Function name	i2c_smbus_arp_enable
Function prototype	void i2c_smbus_arp_enable(uint32_t i2c_periph, uint32_t arpstate);
Function descriptions	enable or disable I2C ARP protocol in SMBus switch
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
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_enable(I2C0, I2C_ARP_ENABLE);
```

i2c_flag_get

The description of i2c_flag_get is shown as below:

Table 3-394. Function i2c_flag_get

Function name	i2c_flag_get
Function prototype	FlagStatus i2c_flag_get(uint32_t i2c_periph,uint32_t 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)
Input parameter{in}	
flag	specify get which flag
I2C_FLAG_SBSEND	start condition send out
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	overrun or underrun 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_TRS	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 3-395. Function i2c_flag_clear

Function name	i2c_flag_clear
Function prototype	void i2c_flag_clear(uint32_t i2c_periph, uint32_t flag);
Function descriptions	clear I2C flag
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
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
I2C_FLAG_ADDSEND	cleared by reading I2C_STAT0 and reading I2C_STAT1
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 3-396. 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)
Input parameter{in}	
interrupt	interrupt type
I2C_INT_ERR	error interrupt enable
I2C_INT_EV	event interrupt enable
I2C_INT_BUF	*buffer interrupt enable*
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable I2C0 error interrupt */

i2c_interrupt_enable(I2C0, I2C_INT_EV);

i2c_interrupt_disable

The description of i2c_interrupt_disable is shown as below:

Table 3-397. 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)
Input parameter{in}	
interrupt	interrupt type
I2C_INT_ERR	error interrupt disable
I2C_INT_EV	event interrupt disable
I2C_INT_BUF	buffer interrupt disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable I2C0 error interrupt */

i2c_interrupt_disable(I2C0, I2C_INT_EV);

i2c_interrupt_flag_get

The description of i2c_interrupt_flag_get is shown as below:

Table 3-398. 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	check I2C interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)

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 3-399. 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
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
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);
```

3.16. MISC

MISC is a software package that provide the interfaces for NVIC and SysTick.The NVIC and SysTick registers are listed in chapter ***3.16.1***, the MISC firmware functions are introduced in chapter ***3.16.2***.

3.16.1. Descriptions of Peripheral registers

Table 3-400. NVIC Registers

Registers	Descriptions
ISER(1)	Interrupt Set Enable Register
ICER(1)	Interrupt Clear Enable Register
ISPR(1)	Interrupt Set Pending Register
ICPR(1)	Interrupt Clear Pending Register
IABR(1)	Interrupt Active bit Register
IP(1)	Interrupt Priority Register
STIR(1)	Software Trigger Interrupt Register
CPUID(2)	CPUID Base Register
ICSR(2)	Interrupt Control and State Register
VTOR(2)	Vector Table Offset Register
AIRCR(2)	Application Interrupt and Reset Control Register
SCR(2)	System Control Register
CCR(2)	Configuration Control Register

Registers	Descriptions
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 3-401. 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

3.16.2. Descriptions of Peripheral functions

Enum IRQn_Type

Table3-402. IRQn_Type

Member name	Function description
WWDGT_IRQn	window watchDog timer interrupt

Member name	Function description
LVD_IRQn	LVD through EXTI line detect interrupt
TAMPER_IRQn	tamper through EXTI line detect
RTC_IRQn	RTC through EXTI line interrupt
FMC_IRQn	FMC interrupt
RCU_CTC_IRQn	RCU and CTC interrupt
EXTI0_IRQn	EXTI line 0 interrupts
EXTI1_IRQn	EXTI line 1 interrupts
EXTI2_IRQn	EXTI line 2 interrupts
EXTI3_IRQn	EXTI line 3 interrupts
EXTI4_IRQn	EXTI line 4 interrupts
DMA0_Channel0_IRQn	DMA0 channel0 interrupt
DMA0_Channel1_IRQn	DMA0 channel1 interrupt
DMA0_Channel2_IRQn	DMA0 channel2 interrupt
DMA0_Channel3_IRQn	DMA0 channel3 interrupt
DMA0_Channel4_IRQn	DMA0 channel4 interrupt
DMA0_Channel5_IRQn	DMA0 channel5 interrupt
DMA0_Channe6_IRQn	DMA0 channel6 interrupt
ADC0_1_IRQn	ADC0 and ADC1 interrupt
USBD_HP_CAN0_TX_IRQn	USBD high priority interrupts or CAN0 transmit interrupts
USBD_LP_CAN0_RX0_IRQn	USBD low priority interrupts or CAN0 receive0 interrupts
CAN0_RX1_IRQn	CAN0 receive1 interrupts
CAN0_EWMC_IRQn	CAN0 EWMC interrupts
EXTI5_9_IRQn	EXTI[9:5] interrupts
TIMER0_BRK_IRQn	TIMER0 break interrupts
TIMER0_UP_IRQn	TIMER0 update interrupts
TIMER0_TRG_CMT_IRQn	TIMER0 trigger and commutation interrupts
TIMER0_Channel_IRQn	TIMER0 channel capture compare interrupts
TIMER0_BRK_TIMER8_IRQn	TIMER0 break and TIMER8 interrupts
TIMER0_UP_TIMER9_IRQn	TIMER0 update and TIMER9 interrupts
TIMER0_TRG_CMT_TIMER10_IRQn	TIMER0 trigger and commutation and TIMER10 interrupts
TIMER1_IRQn	TIMER1 interrupt
TIMER2_IRQn	TIMER2 interrupt
TIMER3_IRQn	TIMER3 interrupt
I2C0_EV_IRQn	I2C0 event interrupt
I2C0_ER_IRQn	I2C0 error interrupt
I2C1_EV_IRQn	I2C1 event interrupt
I2C1_ER_IRQn	I2C1 error interrupt
SPI0_IRQn	SPI0 interrupt
SPI1_IRQn	SPI1 interrupt
USART0_IRQn	USART0 interrupt
USART1_IRQn	USART1 interrupt

Member name	Function description
USART2_IRQn	USART2 interrupt
EXTI10_15_IRQn	EXTI[15:10] interrupts
RTC_Alarm_IRQn	RTC alarm interrupt
USBD_WKUP_IRQn	USBD wakeup interrupt
USBFS_WKUP_IRQn	USBFS wakeup interrupt
TIMER7_BRK_IRQn	TIMER7 break interrupts
TIMER7_UP_IRQn	TIMER7 update interrupts
TIMER7_TRG_CMT_IRQn	TIMER7 trigger and commutation interrupts
TIMER7_BRK_TIMER11_IRQn	TIMER7 break and TIMER11 interrupts
TIMER7_UP_TIMER12_IRQn	TIMER7 update and TIMER12 interrupts
TIMER7_TRG_CMT_TIMER13_IRQn	TIMER7 trigger and commutation and TIMER13 interrupts
TIMER7_Channel_IRQn	TIMER7 channel capture compare interrupts
ADC2_IRQn	ADC2 global interrupt
EXMC_IRQn	EXMC global interrupt
SDIO_IRQn	SDIO global interrupt
TIMER4_IRQn	TIMER4 global interrupt
SPI2_IRQn	SPI2 global interrupt
UART3_IRQn	UART3 global interrupt
UART4_IRQn	UART4 global interrupt
TIMER5_IRQn	TIMER5 global interrupt
TIMER6_IRQn	TIMER6 global interrupt
DMA1_Channel0_IRQn	DMA1 channel0 global interrupt
DMA1_Channel1_IRQn	DMA1 channel1 global interrupt
DMA1_Channel2_IRQn	DMA1 channel2 global interrupt
DMA1_Channel3_IRQn	DMA1 channel3 global interrupt
DMA1_Channel4_IRQn	DMA1 channel4 global interrupt
DMA1_Channel3_Channel4_IRQn	DMA1 channel3 and channel4 global interrupt
ENET_IRQn	ENET global interrupt
ENET_WKUP_IRQn	ENET Wakeup interrupt
CAN1_TX_IRQn	CAN1 transmit interrupt
CAN1_RX0_IRQn	CAN1 receive0 interrupt
CAN1_RX1_IRQn	CAN1 receive1 interrupt
CAN1_EWMC_IRQn	CAN1 EWMC interrupt
USBFS_IRQn	USBFS global interrupt

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

Table 3-403. MISC firmware function

Function name	Function description
nvic_priority_group_set	set the priority group

Function name	Function description
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

nvic_priority_group_set

The description of nvic_priority_group_set is shown as below:

Table 3-404. Function nvic_priority_group_set

Function name	nvic_priority_group_set
Function prototype	void nvic_priority_group_set(uint32_t nvic_prigroup);
Function descriptions	configure bits length of 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 3-405. 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, configure the priority of interrupt
Precondition	-
The called functions	nvic_priority_group_set
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to enum ***Table3-402. IRQn_Type*****Type** **Member name** | **Function descrip** WWDGT_IRQn | window watchDog time
Input parameter{in}	
nvic_irq_pre_priority	the pre-emption priority needed to set (0~4)
Input parameter{in}	
nvic_irq_sub_priority	the subpriority needed to set (0~4)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable window watchDog timer interrupt , pre-emption priority is 1, subpriority is 1 */
nvic_irq_enable(WWDGT_IRQn, 1, 1);

nvic_irq_disable

The description of nvic_irq_disable is shown as below:

Table 3-406. 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***Table3-402. IRQn_Type***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable window watchDog timer interrupt */
nvic_irq_disable(WWDGT_IRQn);
```

nvic_vector_table_set

The description of nvic_vector_table_set is shown as below:

Table 3-407. 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_FLAS	Flash base address

H	
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 3-408. Function system_lowpower_set

Function name	system_lowpower_set
Function prototype	void system_lowpower_set(uint8_t lowpower_mode);
Function descriptions	the state of the low power mode management
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 3-409. Function system_lowpower_reset

Function name	system_lowpower_reset
Function prototype	void system_lowpower_reset(uint8_t lowpower_mode);
Function descriptions	the state of the low power mode management
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 3-410. 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);

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

3.17.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

Table 3-411. PMU Registers

Registers	Descriptions
PMU_CTL	Control register
PMU_CS	Control and status register

3.17.2. Descriptions of Peripheral functions

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

Table 3-412. PMU firmware function

Function name	Function description
pmu_deinit	deinitialize the PMU
pmu_lvd_select	select low voltage detector threshold
pmu_lvd_disable	disable PMU lvd
pmu_to_sleepmode	PMU work at sleep mode
pmu_to_deepsleepmode	PMU work at deepsleep mode
pmu_to_standbymode	pmu work at standby mode
pmu_wakeup_pin_enable	enable wakeup pin
pmu_wakeup_pin_disable	disable wakeup pin
pmu_backup_write_enable	enable backup domain write
pmu_backup_write_disable	disable backup domain write
pmu_flag_get	get flag state
pmu_flag_clear	clear flag bit

pmu_deinit

The description of pmu_deinit is shown as below:

Table 3-413. Function pmu_deinit

Function name	pmu_deinit
Function prototype	void pmu_deinit(void);
Function descriptions	deinitialize the PMU
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 3-414. 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 3-415. 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 3-416. 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 3-417. 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 normal work 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 3-418. 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_standbymode(WFI_CMD);
```

pmu_wakeup_pin_enable

The description of pmu_wakeup_pin_enable is shown as below:

Table 3-419. Function pmu_wakeup_pin_enable

Function name	pmu_wakeup_pin_enable
Function prototype	void pmu_wakeup_pin_enable(void);
Function descriptions	enable wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable wakeup pin */
pmu_wakeup_pin_enable();
```

pmu_wakeup_pin_disable

The description of pmu_wakeup_pin_disable is shown as below:

Table 3-420. Function pmu_wakeup_pin_disable

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

-	-

Example:

/* disable wakeup pin */

pmu_wakeup_pin_disable();

pmu_backup_write_enable

The description of pmu_backup_write_enable is shown as below:

Table 3-421. Function pmu_backup_write_enable

Function name	pmu_backup_write_enable
Function prototype	void pmu_backup_write_enable (void);
Function descriptions	enable write access to the registers in backup domain
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 3-422. Function pmu_backup_write_disable

Function name	pmu_backup_write_disable
Function prototype	void pmu_backup_write_disable (void);
Function descriptions	disable write access to the registers in backup domain
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable backup domain write */
pmu_backup_write_disable();
```

pmu_flag_get

The description of pmu_flag_get is shown as below:

Table 3-423. 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	low voltage detector status flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get flag state */

FlagStatus status;

status = pmu_flag_get(PMU_FLAG_WAKEUP);
```

pmu_flag_clear

The description of pmu_flag_clear is shown as below:

Table 3-424. Function pmu_flag_clear

Function name	pmu_flag_clear
Function prototype	void pmu_flag_clear(uint32_t flag_reset);
Function descriptions	clear flag bit
Precondition	-
The called functions	-
Input parameter{in}	
flag_reset	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);
```

3.18. RCU

RCU is the reset and clock unit. Reset Control includes the control of three kinds of reset: power reset, system reset and backup domain reset. The Clock Control unit provides a range of frequencies and clock functions. The RCU registers are listed in chapter ***3.18.1***, the RCU firmware functions are introduced in chapter ***3.18.2***.

3.18.1. Descriptions of Peripheral registers

RCU registers (MD, HD, XD series) are listed in the table shown as below:

Table 3-425. RCU Registers (MD, HD, XD series)

Registers	Descriptions
RCU_CTL	Control register
RCU_CFG0	Clock configuration register 0
RCU_INT	Clock interrupt register
RCU_APB2RST	APB2 reset register
RCU_APB1RST	APB1 reset register
RCU_AHBEN	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_DSV	Deep-sleep mode voltage register

RCU registers (CL series) are listed in the table shown as below:

Table 3-426. RCU Registers (CL series)

Registers	Descriptions
RCU_CTL	Control register
RCU_CFG0	Clock configuration register 0
RCU_INT	Clock interrupt register
RCU_APB2RST	APB2 reset register
RCU_APB1RST	APB1 reset register
RCU_AHBEN	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

Registers	Descriptions
RCU_AHBRST	AHB reset register
RCU_CFG1	Clock configuration register 1
RCU_DSV	Deep-sleep mode voltage register

3.18.2. Descriptions of Peripheral functions

Table 3-427. 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_ckout0_config	configure the CK_OUT0 clock source
rcu_pll_config	configure the main PLL clock
rcu_predv0_config	configure the PREDV0 division factor
rcu_predv1_config	configure the PREDV1 division factor
rcu_pll1_config	configure the PLL1 clock
rcu_pll2_config	configure the PLL2 clock
rcu_adc_clock_config	configure the ADC prescaler factor

Function name	Function description
rcu_usb_clock_config	configure the USB prescaler factor
rcu_rtc_clock_config	configure the RTC clock source selection
rcu_i2s1_clock_config	configure the I2S1 clock source selection
rcu_i2s2_clock_config	configure the I2S2 clock source selection
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_deepsleep_voltage_set	set the deep-sleep mode voltage value
rcu_clock_freq_get	get the system clock, bus clock frequency

rcu_deinit

The description of rcu_deinit is shown as below:

Table 3-428. 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	rcu_osci_stab_wait
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize RCU */

rcu_deinit();
```

rcu_periph_clock_enable

The description of rcu_periph_clock_enable is shown as below:

Table 3-429. 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 rcu_periph_enum
RCU_GPIOx	GPIO ports clock (x=A,B,C,D,E,F,G)
RCU_AF	alternate function clock
RCU_CRC	CRC clock
RCU_DMAx	DMAx clock (x=0,1)
RCU_ENET	ENET clock (CL series available)
RCU_ENETTX	ENETTX clock (CL series available)
RCU_ENETRX	ENETRX clock (CL series available)

RCU_USBD	USBD clock (HD,XD series available)
RCU_USBFS	USBFS clock (CL series available)
RCU_EXMC	EXMC clock
RCU_TIMERx	TIMERx clock (x=0,1,2,3,4,5,6,7,8,9,10,11,12,13), TIMER8..13 are only available for XD series
RCU_WWDGT	WWDGT clock
RCU_SPIx	SPIx clock (x=0,1,2)
RCU_USARTx	USARTx clock (x=0,1,2)
RCU_UARTx	UARTx clock (x=3,4)
RCU_I2Cx	I2Cx clock (x=0,1)
RCU_CANx	CANx clock (x=0,1), CAN1 is only available for CL series
RCU_PMU	PMU clock
RCU_DAC	DAC clock
RCU_RTC	RTC clock
RCU_ADCx	ADCx clock (x=0,1,2), ADC2 is not available for CL series
RCU_SDIO	SDIO clock (not available for CL series)
RCU_BKPI	BKP interface 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 3-430. 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 rcu_periph_enum
RCU_GPIOx	GPIO ports clock (x=A,B,C,D,E,F,G)
RCU_AF	alternate function clock
RCU_CRC	CRC clock
RCU_DMAx	DMAx clock (x=0,1)
RCU_ENET	ENET clock (CL series available)
RCU_ENETTX	ENETTX clock (CL series available)
RCU_ENETRX	ENETRX clock (CL series available)
RCU_USBD	USBD clock (HD,XD series available)
RCU_USBFS	USBFS clock (CL series available)
RCU_EXMC	EXMC clock
RCU_TIMERx	TIMERx clock (x=0,1,2,3,4,5,6,7,8,9,10,11,12,13), TIMER8..13 are only available for XD series
RCU_WWDGT	WWDGT clock
RCU_SPIx	SPIx clock (x=0,1,2)
RCU_USARTx	USARTx clock (x=0,1,2)
RCU_UARTx	UARTx clock (x=3,4)
RCU_I2Cx	I2Cx clock (x=0,1)
RCU_CANx	CANx clock (x=0,1), CAN1 is only available for CL series
RCU_PMU	PMU clock
RCU_DAC	DAC clock
RCU_RTC	RTC clock
RCU_ADCx	ADCx clock (x=0,1,2), ADC2 is not available for CL series
RCU_SDIO	SDIO clock (not available for CL series)

RCU_BKPI	BKP interface 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 3-431. 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 rcu_periph_sleep_enum
RCU_FMC_SLP	FMC clock
RCU_SRAM_SLP	SRAM clock
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 3-432. 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 rcu_periph_sleep_enum
RCU_FMC_SLP	FMC clock
RCU_SRAM_SLP	SRAM clock
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 3-433. 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 rcu_periph_reset_enum
RCU_GPIOxRST	reset GPIO ports clock (x=A,B,C,D,E,F,G)
RCU_AFRST	reset alternate function clock
RCU_ENETRST	reset ENET clock (CL series available)
RCU_USBDRST	reset USBD clock (HD,XD series available)
RCU_USBFSRST	reset USBFS clock (CL series available)
RCU_TIMERxRST	reset TIMERx clock (x=0,1,2,3,4,5,6,7,8,9,10,11,12,13), TIMER8..13 are only available for XD series
RCU_WWDGTRST	reset WWDGT clock
RCU_SPIxRST	reset SPIx clock (x=0,1,2)
RCU_USARTxRST	reset USARTx clock (x=0,1,2)
RCU_UARTxRST	reset UARTx clock (x=3,4)
RCU_I2CxRST	reset I2Cx clock (x=0,1)
RCU_CANxRST	reset CANx clock (x=0,1), CAN1 is only available for CL series)
RCU_PMURST	reset PMU clock
RCU_DACRST	reset DAC clock
RCU_ADCxRST	reset ADCx clock (x=0,1,2), ADC2 is not available for CL series)
RCU_BKPIRST	reset BKPI clock
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 3-434. 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 rcu_periph_reset_enum
RCU_GPIOxRST	reset GPIO ports clock (x=A,B,C,D,E,F,G)
RCU_AFRST	reset alternate function clock
RCU_ENETRST	reset ENET clock (CL series available)
RCU_USBDRST	reset USBD clock (HD,XD series available)
RCU_USBFSRST	reset USBFS clock (CL series available)
RCU_TIMERxRST	reset TIMERx clock (x=0,1,2,3,4,5,6,7,8,9,10,11,12,13), TIMER8..13 are only available for XD series
RCU_WWDGTRST	reset WWDGT clock
RCU_SPIxRST	reset SPIx clock (x=0,1,2)
RCU_USARTxRST	reset USARTx clock (x=0,1,2)
RCU_UARTxRST	reset UARTx clock (x=3,4)
RCU_I2CxRST	reset I2Cx clock (x=0,1)
RCU_CANxRST	reset CANx clock (x=0,1), CAN1 is only available for CL series)
RCU_PMURST	reset PMU clock
RCU_DACRST	reset DAC clock
RCU_ADCxRST	reset ADCx clock (x=0,1,2), ADC2 is not available for CL series)
RCU_BKPIRST	reset BKPI clock
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 3-435. 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 3-436. 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 3-437. 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 3-438. 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	which clock is selected as CK_SYS source
RCU_SCSS_IRC8M	CK_IRC8M is selected as the CK_SYS source
RCU_SCSS_HXTAL	CK_HXTAL is selected as the CK_SYS source
RCU_SCSS_PLL	CK_PLL is selected as the CK_SYS source

Example:

/* get the CK_SYS source */

uint32_t temp_cksys_status;

temp_cksys_status = rcu_system_clock_source_get();

rcu_ahb_clock_config

The description of rcu_ahb_clock_config is shown as below:

Table 3-439. 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 3-440. 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_DIV1	select CK_AHB as CK_APB1
RCU_APB1_CKAHB_DIV2	select CK_AHB/2 as CK_APB1
RCU_APB1_CKAHB_D	select CK_AHB/4 as CK_APB1

IV4	
RCU_APB1_CKAHB_DIV8	select CK_AHB/8 as CK_APB1
RCU_APB1_CKAHB_DIV16	select CK_AHB/16 as CK_APB1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure CK_AHB/16 as CK_APB1 */

rcu_apb1_clock_config(RCU_APB1_CKAHB_DIV16);

rcu_apb2_clock_config

The description of rcu_apb2_clock_config is shown as below:

Table 3-441. 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_DIV1	select CK_AHB as CK_APB2
RCU_APB2_CKAHB_DIV2	select CK_AHB/2 as CK_APB2
RCU_APB2_CKAHB_DIV4	select CK_AHB/4 as CK_APB2
RCU_APB2_CKAHB_DIV8	select CK_AHB/8 as CK_APB2

RCU_APB2_CKAHB_DIV16	select CK_AHB/16 as CK_APB2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB2 */
rcu_apb2_clock_config(RCU_APB2_CKAHB_DIV8);
```

rcu_ckout0_config

The description of rcu_ckout0_config is shown as below:

Table 3-442. Function rcu_ckout0_config

Function name	rcu_ckout0_config
Function prototype	void rcu_ckout0_config(uint32_t ckout0_src);
Function descriptions	configure the CK_OUT0 clock source
Precondition	-
The called functions	-
Input parameter{in}	
ckout0_src	CK_OUT0 clock source selection
RCU_CKOUT0SRC_NONE	no clock selected
RCU_CKOUT0SRC_CKSYS	select system clock CK_SYS
RCU_CKOUT0SRC_IRC8M	select high speed 8M internal oscillator clock
RCU_CKOUT0SRC_HXTAL	select HXTAL
RCU_CKOUT0SRC_CKPLL_DIV2	select (CK_PLL / 2) clock
RCU_CKOUT0SRC_C	select CK_PLL1 clock

KPLL1	
RCU_CKOUT0SRC_CKPLL2_DIV2	select (CK_PLL2 / 2) clock
RCU_CKOUT0SRC_EXT1	select EXT1 clock
RCU_CKOUT0SRC_CKPLL2	select CK_PLL2 clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the HXTAL as CK_OUT0 clock source */

rcu_ckout0_config(RCU_CKOUT0SRC_HXTAL);

rcu_pll_config

The description of rcu_pll_config is shown as below:

Table 3-443. 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 main PLL clock
Precondition	-
The called functions	-
Input parameter{in}	
pll_src	PLL clock source selection
RCU_PLLSRC_IRC8M_DIV2	IRC8M/2 clock is selected as source clock of PLL
RCU_PLLSRC_HXTAL	HXTAL is selected as source clock of PLL
Input parameter{in}	
pll_mul	PLL clock multiplication factor
RCU_PLL_MULx	PLL clock * x (x = 2..32 in XD series, x = 2..14, 6.5, 16..32 in CL series)

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the PLL */

rcu_pll_config(RCU_PLLSRC_HXTAL, RCU_PLL_MUL10);

rcu_predv0_config (MD、HD、XD seires)

The description of rcu_predv0_config is shown as below:

Table 3-444. Function rcu_predv0_config

Function name	rcu_predv0_config
Function prototype	void rcu_predv0_config(uint32_t predv0_div);
Function descriptions	configure the PREDV0 division factor
Precondition	-
The called functions	-
Input parameter{in}	
predv0_div	PREDV0 division factor
RCU_PREDV0_DIVx	PREDV0 input source clock is divided by x, (x = 1,2)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the PREDV0 division factor */

rcu_predv0_config(RCU_PREDV0_DIV2);

rcu_predv0_config (CL seires)

The description of rcu_predv0_config is shown as below:

Table 3-445. Function rcu_predv0_config

Function name	rcu_predv0_config
Function prototype	void rcu_predv0_config(uint32_t predv0_source, uint32_t predv0_div);
Function descriptions	configure the PREDV0 division factor
Precondition	-
The called functions	-
Input parameter{in}	
predv0_source	PREDV0 input clock source selection
RCU_PREDV0SRC_HXTAL	select HXTAL as PREDV0 input source clock
RCU_PREDV0SRC_CKPLL1	select CK_PLL1 as PREDV0 input source clock
Input parameter{in}	
predv0_div	PREDV0 division factor
RCU_PREDV0_DIVx	PREDV0 input source clock is divided x (x=1..16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the PREDV0 division factor */

rcu_predv0_config(RCU_PREDV0SRC_HXTAL, RCU_PREDV0_DIV4);

rcu_predv1_config (CL seires)

The description of rcu_predv1_config is shown as below:

Table 3-446. Function rcu_predv1_config

Function name	rcu_predv1_config
Function prototype	void rcu_predv1_config(uint32_t predv1_div);
Function descriptions	configure the PREDV1 division factor
Precondition	-

The called functions	-
Input parameter{in}	
predv1_div	PREDV1 division factor
RCU_PREDV1_DIVx	PREDV1 input source clock is divided x (x=1..16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PREDV1 division factor */
rcu_predv1_config(RCU_PREDV1_DIV8);
```

rcu_pll1_config (CL seires)

The description of rcu_pll1_config is shown as below:

Table 3-447. Function rcu_pll1_config

Function name	rcu_pll1_config
Function prototype	void rcu_pll1_config(uint32_t pll_mul);
Function descriptions	configure the PLL1 clock
Precondition	-
The called functions	-
Input parameter{in}	
pll_mul	PLL clock multiplication factor
RCU_PLL1_MULx	PLL1 clock * x，(x = 8..16, 20)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PLL1 clock */
```

rcu_pll1_config(RCU_PLL1_MUL8);

rcu_pll2_config (CL seires)

The description of rcu_pll2_config is shown as below:

Table 3-448. Function rcu_pll2_config

Function name	rcu_pll2_config
Function prototype	void rcu_pll2_config(uint32_t pll_mul)
Function descriptions	configure the PLL2 clock
Precondition	-
The called functions	-
Input parameter{in}	
pll_mul	PLL clock multiplication factor
RCU_PLL2_MULx	PLL2 clock * x, (x = 8..16, 20)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the PLL2 clock */

rcu_pll2_config(RCU_PLL2_MUL8);

rcu_adc_clock_config

The description of rcu_adc_clock_config is shown as below:

Table 3-449. Function rcu_adc_clock_config

Function name	rcu_adc_clock_config
Function prototype	void rcu_adc_clock_config(uint32_t adc_psc);
Function descriptions	configure the ADC prescaler factor
Precondition	-
The called functions	-
Input parameter{in}	

adc_psc	ADC prescaler factor
RCU_CKADC_CKAPB2_DIV2	CK_ADC = CK_APB2 / 2
RCU_CKADC_CKAPB2_DIV4	CK_ADC = CK_APB2 / 4
RCU_CKADC_CKAPB2_DIV6	CK_ADC = CK_APB2 / 6
RCU_CKADC_CKAPB2_DIV8	CK_ADC = CK_APB2 / 8
RCU_CKADC_CKAPB2_DIV12	CK_ADC = CK_APB2 / 12
RCU_CKADC_CKAPB2_DIV16	CK_ADC = CK_APB2 / 16
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the ADC prescaler factor */

rcu_adc_clock_config(RCU_CKADC_CKAPB2_DIV8);

rcu_usb_clock_config

The description of rcu_usb_clock_config is shown as below:

Table 3-450. Function rcu_usb_clock_config

Function name	rcu_usb_clock_config
Function prototype	void rcu_usb_clock_config(uint32_t usb_psc);
Function descriptions	configure the USB prescaler factor
Precondition	-
The called functions	-
Input parameter{in}	
usb_psc	USB prescaler factor

RCU_CKUSB_CKPLL_DIV1_5	CK_USBFS = CK_PLL / 1.5
RCU_CKUSB_CKPLL_DIV1	CK_USBFS = CK_PLL / 1
RCU_CKUSB_CKPLL_DIV2_5	CK_USBFS = CK_PLL / 2.5
RCU_CKUSB_CKPLL_DIV2	CK_USBFS = CK_PLL / 2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the USB prescaler factor */

rcu_usb_clock_config(RCU_CKUSB_CKPLL_DIV2_5);

rcu_rtc_clock_config

The description of rcu_rtc_clock_config is shown as below:

Table 3-451. 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	select CK_HXTAL/128 as RTC source clock

_DIV_128	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the RTC clock source selection */
rcu_rtc_clock_config(RCU_RTCSRC_IRC40K);
```

rcu_i2s1_clock_config (CL seires)

The description of rcu_i2s1_clock_config is shown as below:

Table 3-452. Function rcu_i2s1_clock_config

Function name	rcu_i2s1_clock_config
Function prototype	void rcu_i2s1_clock_config(uint32_t i2s_clock_source);
Function descriptions	configure the I2S1 clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
i2s_clock_source	I2S clock source selection
RCU_I2S1SRC_CKSYS	select system clock as I2S1 source clock
RCU_I2S1SRC_CKPLL2_MUL2	select CK_PLL2 * 2 as I2S1 source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the I2S1 clock source selection */
rcu_i2s1_clock_config(RCU_I2S1SRC_CKPLL2_MUL2);
```

rcu_i2s2_clock_config (CL seires)

The description of rcu_i2s2_clock_config is shown as below:

Table 3-453. Function rcu_i2s2_clock_config

Function name	rcu_i2s2_clock_config
Function prototype	void rcu_i2s2_clock_config(uint32_t i2s_clock_source);
Function descriptions	configure the I2S2 clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
i2s_clock_source	I2S clock source selection
RCU_I2S2SRC_CKSYS	select system clock as I2S2 source clock
RCU_I2S2SRC_CKPLL2_MUL2	select CK_PLL2 * 2 as I2S2 source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the I2S2 clock source selection */
rcu_i2s2_clock_config(RCU_I2S2SRC_CKPLL2_MUL2);
```

rcu_flag_get

The description of rcu_flag_get is shown as below:

Table 3-454. 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 rcu_flag_enum
RCU_FLAG_IRC8MSTB	IRC8M stabilization flag
RCU_FLAG_HXTALSTB	HXTAL stabilization flag
RCU_FLAG_PLLSTB	PLL stabilization flag
RCU_FLAG_PLL1STB	PLL1 stabilization flag(CL series only)
RCU_FLAG_PLL2STB	PLL2 stabilization flag(CL series only)
RCU_FLAG_LXTALSTB	LXTAL stabilization flag
RCU_FLAG_IRC40KSTB	IRC40K stabilization 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 3-455. 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 3-456. 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 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_PLL1STB	PLL1 stabilization interrupt flag(CL series only)
RCU_INT_FLAG_PLL2STB	PLL2 stabilization interrupt flag(CL series only)
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 3-457. 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 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_PLL1STB_CLR	PLL1 stabilization interrupt flag clear(CL series only)
RCU_INT_FLAG_PLL2STB_CLR	PLL2 stabilization interrupt flag clear(CL series only)
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 3-458. 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 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_PLL1STB	PLL1 stabilization interrupt enable(CL series only)
RCU_INT_PLL2STB	PLL2 stabilization interrupt enable(CL series only)
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 3-459. 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 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_PLL1STB	PLL1 stabilization interrupt enable(CL series only)
RCU_INT_PLL2STB	PLL2 stabilization interrupt enable(CL series only)
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 3-460. 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	rcu_flag_get
Input parameter{in}	
osci	oscillator types, refer to 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_IRC40K	internal 40K RC oscillator(IRC40K)

RCU_PLL_CK	phase locked loop(PLL)
RCU_PLL1_CK	phase locked loop 1(CL series only)
RCU_PLL2_CK	phase locked loop 2(CL series only)
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 3-461. Function rcu_osci_on

Function name	rcu_osci_on
Function prototype	void rcu_osci_on(rcu_osci_type_enum osci);
Function descriptions	turn on the oscillator
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to rcu_osci_type_enum
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_LXTAL	low speed crystal oscillator(LXTAL)
RCU_IRC8M	internal 8M RC oscillators(IRC8M)
RCU_IRC40K	internal 40K RC oscillator(IRC40K)
RCU_PLL_CK	phase locked loop(PLL)
RCU_PLL1_CK	phase locked loop 1(CL series only)
RCU_PLL2_CK	phase locked loop 2(CL series only)

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 3-462. Function rcu_osci_off

Function name	rcu_osci_off
Function prototype	void rcu_osci_off(rcu_osci_type_enum osci);
Function descriptions	turn off the oscillator
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to rcu_osci_type_enum
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_LXTAL	low speed crystal oscillator(LXTAL)
RCU_IRC8M	internal 8M RC oscillators(IRC8M)
RCU_IRC40K	internal 40K RC oscillator(IRC40K)
RCU_PLL_CK	phase locked loop(PLL)
RCU_PLL1_CK	phase locked loop 1(CL series only)
RCU_PLL2_CK	phase locked loop 2(CL series only)
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 3-463. 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 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 3-464. 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 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 3-465. 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 3-466. 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 3-467. Function rcu_irc8m_adjust_value_set

Function name	rcu_irc8m_adjust_value_set
Function prototype	void rcu_irc8m_adjust_value_set(uint8_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_deepsleep_voltage_set

The description of rcu_deepsleep_voltage_set is shown as below:

Table 3-468. Function rcu_deepsleep_voltage_set

Function name	rcu_deepsleep_voltage_set
Function prototype	void rcu_deepsleep_voltage_set(uint32_t dsvol);
Function descriptions	set the deep-sleep mode voltage value
Precondition	-
The called functions	-
Input parameter{in}	
dsvol	deep sleep mode voltage
RCU_DEEPSLEEP_V_1_2	the core voltage is 1.2V in deep-sleep mode
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_1);

rcu_clock_freq_get

The description of rcu_clock_freq_get is shown as below:

Table 3-469. 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 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
Output parameter{out}	
-	-
Return value	
ck_freq	clock frequency of system, AHB, APB1, APB2

Example:

uint32_t temp_freq;

/* get the system clock frequency */

temp_freq = rcu_clock_freq_get(CK_SYS);

3.19. RTC

The Real-time Clock (RTC) is usually used as a clock-calendar. The ones in the Backup Domain consist of a 32-bit up-counter, an alarm, a prescaler, a divider and the RTC clock configuration register. The RTC registers are listed in chapter ***3.19.1***, the FWDGT firmware functions are introduced in chapter ***3.19.2***.

3.19.1. Descriptions of Peripheral registers

RTC registers are listed in the table shown as below:

Table 3-470. RTC Registers

Registers	Descriptions
RTC_INTEN	Interrupt enable register
RTC_CTL	Control register
RTC_PSCH	Prescaler high register
RTC_PSCL	Prescaler low register
RTC_DIVH	Divider high register
RTC_DIVL	Divider low register
RTC_CNTH	counter high register
RTC_CNTL	counter low register
RTC_ALRMH	Alarm high register
RTC_ALRML	Alarm low register

3.19.2. Descriptions of Peripheral functions

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

Table 3-471.RTC firmware function

Function name	Function description
rtc_configuration_mode_enter	enter RTC configuration mode
rtc_configuration_mode_exit	exit RTC configuration mode
rtc_counter_set	set RTC counter value
rtc_prescaler_set	set RTC prescaler value
rtc_lwoff_wait	wait RTC last write operation finished flag set

Function name	Function description
rtc_register_sync_wait	wait RTC registers synchronized flag set
rtc_alarm_config	set RTC alarm value
rtc_counter_get	get RTC counter value
rtc_divider_get	get RTC divider value
rtc_flag_get	get RTC flag status
rtc_flag_clear	clear RTC flag status
rtc_interrupt_flag_get	get RTC interrupt flag status
rtc_interrupt_flag_clear	clear RTC interrupt flag status
rtc_interrupt_enable	enable RTC interrupt
rtc_interrupt_disable	disable RTC interrupt

rtc_configuration_mode_enter

The description of rtc_configuration_mode_enter is shown as below:

Table 3-472. Function rtc_configuration_mode_enter

Function name	rtc_configuration_mode_enter
Function prototype	void rtc_configuration_mode_enter(void);
Function descriptions	enter RTC configuration mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enter RTC configuration mode */
rtc_configuration_mode_enter( );
```

rtc_configuration_mode_exit

The description of rtc_configuration_mode_exit is shown as below:

Table 3-473. Function rtc_configuration_mode_exit

Function name	rtc_configuration_mode_exit
Function prototype	void rtc_configuration_mode_exit(void);
Function descriptions	exit RTC configuration mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* exit RTC configuration mode */
rtc_configuration_mode_ exit( );
```

rtc_counter_set

The description of rtc_counter_set is shown as below:

Table 3-474. Function rtc_counter_set

Function name	rtc_counter_set
Function prototype	void rtc_counter_set(uint32_t cnt);
Function descriptions	set RTC counter value
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set).
The called functions	rtc_configuration_mode_enter / rtc_configuration_mode_exit
Input parameter{in}	
cnt	RTC counter value（0-0xFFFF FFFF）
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* set counter value to 0xFFFF */
rtc_counter_set (0xFFFF);
```

rtc_prescaler_set

The description of rtc_prescaler_set is shown as below:

Table 3-475. Function rtc_prescaler_set

Function name	rtc_prescaler_set
Function prototype	void rtc_prescaler_set(uint32_t psc);
Function descriptions	set RTC prescaler value
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set).
The called functions	rtc_configuration_mode_enter / rtc_configuration_mode_exit
Input parameter{in}	
psc	RTC prescaler value（0-0x000F FFFF）
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* set RTC prescaler value to 0x7FFFF */
rtc_prescaler_set (0x7FFFF);
```

rtc_lwoff_wait

The description of rtc_lwoff_wait is shown as below:

Table 3-476. Function rtc_lwoff_wait

Function name	rtc_lwoff_wait
Function prototype	void rtc_lwoff_wait(void);
Function descriptions	wait RTC last write operation finished flag set
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */

rtc_lwoff_wait( );
```

rtc_register_sync_wait

The description of rtc_register_sync_wait is shown as below:

Table 3-477. Function rtc_register_sync_wait

Function name	rtc_register_sync_wait
Function prototype	void rtc_register_sync_wait(void);
Function descriptions	wait RTC registers synchronized flag set
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* wait for RTC registers synchronization */

rtc_register_sync_wait( );

rtc_alarm_config

The description of rtc_alarm_config is shown as below:

Table 3-478. Function rtc_alarm_config

Function name	rtc_alarm_config
Function prototype	void rtc_alarm_config(uint32_t alarm);
Function descriptions	set RTC alarm value
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set). -
The called functions	rtc_configuration_mode_enter / rtc_configuration_mode_exit
Input parameter{in}	
alarm	RTC alarm value（0-0xFFFF FFFF）
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* wait until last write operation on RTC registers has finished */

rtc_lwoff_wait( );

/* set alarm value to 0xFFFF */

rtc_alarm_config (0xFFFF);

rtc_counter_get

The description of rtc_counter_get is shown as below:

Table 3-479. Function rtc_counter_get

Function name	rtc_counter_get
Function prototype	uint32_t rtc_counter_get(void);
Function descriptions	get RTC counter value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	the value of RTC counter

Example:

```
/* get the counter value */
uint32_t rtc_counter_value;
rtc_counter_value = rtc_counter_get( );
```

rtc_divider_get

The description of rtc_divider_get is shown as below:

Table 3-480. Function rtc_divider_get

Function name	rtc_divider_get
Function prototype	uint32_t rtc_divider_get(void);
Function descriptions	get RTC divider value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
uint32_t	the value of RTC divider

Example:

/* get the current RTC divider value */

uint32_t rtc_divider_value;

rtc_divider_value = rtc_divider_get( );

rtc_flag_get

The description of rtc_flag_getrtc_interrupt_enable is shown as below:

Table 3-481. Function rtc_flag_get

Function name	rtc_flag_get
Function prototype	FlagStatus rtc_flag_get(uint32_t flag);
Function descriptions	get RTC flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which RTC flag status to get
RTC_FLAG_SECOND	second interrupt flag
RTC_FLAG_ALARM	alarm interrupt flag
RTC_FLAG_OVERFLOW	overflow interrupt flag
RTC_FLAG_RSYN	registers synchronized flag
RTC_FLAG_LWOF	last write operation finished flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the RTC alarm flag status */

FlagStatus alarm_status;

alarm_status = rtc_flag_get(RTC_FLAG_ALARM);

rtc_flag_clear

The description of rtc_flag_clear is shown as below:

Table 3-482. Function rtc_flag_clear

Function name	rtc_flag_clear
Function prototype	void rtc_flag_clear(uint32_t flag);
Function descriptions	clear RTC flag status
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set).
The called functions	-
Input parameter{in}	
flag	specify which RTC flag status to clear
RTC_FLAG_SECOND	second interrupt flag
RTC_FLAG_ALARM	alarm interrupt flag
RTC_FLAG_OVERFLOW	overflow interrupt flag
RTC_FLAG_RSYN	registers synchronized flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* clear the RTC alarm flag */
rtc_flag_clear(RTC_FLAG_ALARM);
```

rtc_interrupt_flag_get

The description of rtc_interrupt_flag_get is shown as below:

Table 3-483. Function rtc_interrupt_flag_get

Function name	rtc_interrupt_flag_get
Function prototype	FlagStatus rtc_interrupt_flag_get(uint32_t flag);
Function descriptions	get RTC interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag status to get
RTC_INT_FLAG_SECOND	second interrupt flag
RTC_INT_FLAG_ALARM	alarm interrupt flag
RTC_INT_FLAG_OVERFLOW	overflow interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the RTC overflow interrupt flag status */

FlagStatus overflow_status;

overflow_status = rtc_interrupt_flag_get(RTC_INT_FLAG_OVERFLOW);

rtc_interrupt_flag_clear

The description of rtc_interrupt_flag_clear is shown as below:

Table 3-484. Function rtc_interrupt_flag_clear

Function name	rtc_interrupt_flag_clear
Function prototype	void rtc_interrupt_flag_clear(uint32_t flag);
Function descriptions	clear RTC interrupt flag status
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set).

The called functions	-
Input parameter{in}	
flag	specify which flag status to clear
RTC_INT_FLAG_SECOND	second interrupt flag
RTC_INT_FLAG_ALARM	alarm interrupt flag
RTC_INT_FLAG_OVERFLOW	overflow interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* clear the RTC alarm interrupt flag */
rtc_interrupt_flag_clear(RTC_FLAG_ALARM);
```

rtc_interrupt_enable

The description of rtc_interrupt_enable is shown as below:

Table 3-485. Function rtc_interrupt_enable

Function name	rtc_interrupt_enable
Function prototype	void rtc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable RTC interrupt
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set).
The called functions	-
Input parameter{in}	
interrupt	specify which RTC interrupt to enable

RTC_INT_SECOND	second interrupt
RTC_INT_ALARM	alarm interrupt
RTC_INT_OVERFLOW	overflow interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* enable the RTC second interrupt */
rtc_interrupt_enable(RTC_INT_SECOND);
```

rtc_interrupt_disable

The description of rtc_interrupt_disable is shown as below:

Table 3-486. Function rtc_interrupt_disable

Function name	rtc_interrupt_disable
Function prototype	void rtc_interrupt_disable(uint32_t interrupt);
Function descriptions	disable RTC interrupt
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set).
The called functions	-
Input parameter{in}	
interrupt	specify which RTC interrupt to disable
RTC_INT_SECOND	second interrupt
RTC_INT_ALARM	alarm interrupt
RTC_INT_OVERFLOW	overflow interrupt
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* disable the RTC second interrupt */
rtc_interrupt_disable(RTC_INT_SECOND);
```

3.20. SDIO

The secure digital input/output interface (SDIO) defines the SD, SD I/O, MMC and CE-ATA card host interface, which provides command/data transfer between the AHB system bus and SD memory cards, SD I/O cards, Multimedia Card (MMC) and CE-ATA devices. The SDIO registers are listed in chapter ***3.20.1***, the SDIO firmware functions are introduced in chapter ***3.20.2***.

3.20.1. Descriptions of Peripheral registers

SDIO registers are listed in the table shown as below:

Table 3-487. SDIO Registers

Registers	Descriptions
SDIO_PWRCTL	Power control register
SDIO_CLKCTL	Clock control register
SDIO_CMDAGMT	Command argument register
SDIO_CMDCTL	Command control register
SDIO_RSPCMDIDX	Command index response register
SDIO_RESPx x=0..3	Response register
SDIO_DATATO	Data timeout register
SDIO_DATALEN	Data length register
SDIO_DATACTL	Data control register
SDIO_DATACNT	Data counter register

Registers	Descriptions
SDIO_STAT	Status register
SDIO_INTC	Interrupt clear register
SDIO_INTEN	Interrupt enable register
SDIO_FIFOCNT	FIFO counter register
SDIO_FIFO	FIFO data register

3.20.2. Descriptions of Peripheral functions

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

Table 3-488. SDIO firmware function

Function name	Function description
sdio_deinit	deinitialize the SDIO
sdio_clock_config	configure the SDIO clock
sdio_hardware_clock_enable	enable hardware clock control
sdio_hardware_clock_disable	disable hardware clock control
sdio_bus_mode_set	set different SDIO card bus mode
sdio_power_state_set	set the SDIO power state
sdio_power_state_get	get the SDIO power state
sdio_clock_enable	enable SDIO_CLK clock output
sdio_clock_disable	disable SDIO_CLK clock output
sdio_command_response_config	configure the command and response
sdio_wait_type_set	set the command state machine wait type
sdio_csm_enable	enable the CSM(command state machine)
sdio_csm_disable	disable the CSM(command state machine)
sdio_command_index_get	get the last response command index
sdio_response_get	get the response for the last received command
sdio_data_config	configure the data timeout, data length and data block size
sdio_data_transfer_config	configure the data transfer mode and direction
sdio_dsm_enable	enable the DSM(data state machine) for data transfer

Function name	Function description
sdio_dsm_disable	disable the DSM(data state machine)
sdio_data_write	write data(one word) to the transmit FIFO
sdio_data_read	read data(one word) from the receive FIFO
sdio_data_counter_get	get the number of remaining data bytes to be transferred to card
sdio_fifo_counter_get	get the number of words remaining to be written or read from FIFO
sdio_dma_enable	enable the DMA request for SDIO
sdio_dma_disable	disable the DMA request for SDIO
sdio_flag_get	get the flags state of SDIO
sdio_flag_clear	clear the pending flags of SDIO
sdio_interrupt_enable	enable the SDIO interrupt
sdio_interrupt_disable	disable the SDIO interrupt
sdio_interrupt_flag_get	get the interrupt flags state of SDIO
sdio_interrupt_flag_clear	clear the interrupt pending flags of SDIO
sdio_readwait_enable	enable the read wait mode(SD I/O only)
sdio_readwait_disable	disable the read wait mode(SD I/O only)
sdio_stop_readwait_enable	enable the function that stop the read wait process(SD I/O only)
sdio_stop_readwait_disable	disable the function that stop the read wait process(SD I/O only)
sdio_readwait_type_set	set the read wait type(SD I/O only)
sdio_operation_enable	enable the SD I/O mode specific operation(SD I/O only)
sdio_operation_disable	disable the SD I/O mode specific operation(SD I/O only)
sdio_suspend_enable	enable the SD I/O suspend operation(SD I/O only)
sdio_suspend_disable	disable the SD I/O suspend operation(SD I/O only)
sdio_ceata_command_enable	enable the CE-ATA command(CE-ATA only)
sdio_ceata_command_disable	disable the CE-ATA command(CE-ATA only)
sdio_ceata_interrupt_enable	enable the CE-ATA interrupt(CE-ATA only)

Function name	Function description
sdio_ceata_interrupt_disable	disable the CE-ATA interrupt(CE-ATA only)
sdio_ceata_command_completion_enable	enable the CE-ATA command completion signal(CE-ATA only)
sdio_ceata_command_completion_disable	disable the CE-ATA command completion signal(CE-ATA only)

sdio_deinit

The description of sdio_deinit is shown as below:

Table 3-489. Function sdio_deinit

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

Example:

```
/* deinitialize the SDIO */
sdio_deinit();
```

sdio_clock_config

The description of sdio_clock_config is shown as below:

Table 3-490. Function sdio_clock_config

Function name	sdio_clock_config
Function prototype	void sdio_clock_config(uint32_t clock_edge, uint32_t clock_bypass, uint32_t clock_powersave, uint16_t clock_division);

Function descriptions	configure the SDIO clock
Precondition	-
The called functions	-
Input parameter{in}	
clock_edge	SDIO_CLK clock edge
SDIO_SDIOCLKEDGE_RISING	select the rising edge of the SDIOCLK to generate SDIO_CLK
SDIO_SDIOCLKEDGE_FALLING	select the falling edge of the SDIOCLK to generate SDIO_CLK
Input parameter{in}	
clock_bypass	clock bypass
SDIO_CLOCKBYPASS_ENABLE	clock bypass
SDIO_CLOCKBYPASS_DISABLE	no bypass
Input parameter{in}	
clock_powersave	SDIO_CLK clock dynamic switch on/off for power saving
SDIO_CLOCKPWRSAVE_ENABLE	SDIO_CLK closed when bus is idle
SDIO_CLOCKPWRSAVE_DISABLE	SDIO_CLK clock is always on
Input parameter{in}	
clock_division	clock division, less than 256
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the SDIO clock */

sdio_clock_config(SDIO_SDIOCLKEDGE_RISING, SDIO_CLOCKBYPASS_DISABLE,
SDIO_CLOCKPWRSAVE_DISABLE, SD_CLK_DIV_TRANS);
```

sdio_hardware_clock_enable

The description of sdio_hardware_clock_enable is shown as below:

Table 3-491. Function sdio_hardware_clock_enable

Function name	sdio_hardware_clock_enable
Function prototype	void sdio_hardware_clock_enable(void);
Function descriptions	enable hardware clock control
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable hardware clock control */
sdio_hardware_clock_enable();
```

sdio_hardware_clock_disable

The description of sdio_hardware_clock_disable is shown as below:

Table 3-492. Function sdio_hardware_clock_disable

Function name	sdio_hardware_clock_disable
Function prototype	void sdio_hardware_clock_disable(void);
Function descriptions	disable hardware clock control
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable hardware clock control */
sdio_hardware_clock_disable();
```

sdio_bus_mode_set

The description of sdio_bus_mode_set is shown as below:

Table 3-493. Function sdio_bus_mode_set

Function name	sdio_bus_mode_set
Function prototype	void sdio_bus_mode_set(uint32_t bus_mode);
Function descriptions	set different SDIO card bus mode
Precondition	-
The called functions	-
Input parameter{in}	
bus_mode	SDIO card bus mode
SDIO_BUSMODE_1BIT	1-bit SDIO card bus mode
SDIO_BUSMODE_4BIT	4-bit SDIO card bus mode
SDIO_BUSMODE_8BIT	8-bit SDIO card bus mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO bus mode */
sdio_bus_mode_set(SDIO_BUSMODE_1BIT);
```

sdio_power_state_set

The description of sdio_power_state_set is shown as below:

Table 3-494. Function sdio_power_state_set

Function name	sdio_power_state_set
Function prototype	void sdio_power_state_set(uint32_t power_state);
Function descriptions	set the SDIO power state
Precondition	-
The called functions	-
Input parameter{in}	
power_state	SDIO power state
SDIO_POWER_ON	SDIO power on
SDIO_POWER_OFF	SDIO power off
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set SDIO power state */

sdio_power_state_set(SDIO_POWER_ON);

sdio_power_state_get

The description of sdio_power_state_get is shown as below:

Table 3-495. Function sdio_power_state_get

Function name	sdio_power_state_get
Function prototype	uint32_t sdio_power_state_get(void);
Function descriptions	get the SDIO power state
Precondition	-
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
uint32_t	SDIO_POWER_ON / SDIO_POWER_OFF

Example:

```
/* get the SDIO power state */
uint32_t sdio_power_value;
sdio_power_value = sdio_power_state_get();
```

sdio_clock_enable

The description of sdio_clock_enable is shown as below:

Table 3-496. Function sdio_clock_enable

Function name	sdio_clock_enable
Function prototype	void sdio_clock_enable(void);
Function descriptions	enable SDIO_CLK clock output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SDIO_CLK clock output */
sdio_clock_enable();
```

sdio_clock_disable

The description of sdio_clock_disable is shown as below:

Table 3-497. Function sdio_clock_disable

Function name	sdio_clock_disable
Function prototype	void sdio_clock_disable(void);
Function descriptions	disable SDIO_CLK clock output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SDIO_CLK clock output */
sdio_clock_disable();
```

sdio_command_response_config

The description of sdio_command_response_config is shown as below:

Table 3-498. Function sdio_command_response_config

Function name	sdio_command_response_config
Function prototype	void sdio_command_response_config(uint32_t cmd_index, uint32_t cmd_argument, uint32_t response_type);
Function descriptions	configure the command and response
Precondition	-
The called functions	-
Input parameter{in}	
cmd_index	command index, refer to the related specifications
Input parameter{in}	
cmd_argument	command argument, refer to the related specifications
Input parameter{in}	

response_type	response type
SDIO_RESPONSETYPE_NO	no response
SDIO_RESPONSETYPE_SHORT	short response
SDIO_RESPONSETYPE_LONG	long response
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* CMD2(SD_CMD_ALL_SEND_CID) command response config*/
sdio_command_response_config(SD_CMD_ALL_SEND_CID, (uint32_t)0x0, SDIO_RESPONSETYPE_LONG);
```

sdio_wait_type_set

The description of sdio_wait_type_set is shown as below:

Table 3-499. Function sdio_wait_type_set

Function name	sdio_wait_type_set
Function prototype	void sdio_wait_type_set(uint32_t wait_type);
Function descriptions	set the command state machine wait type
Precondition	-
The called functions	-
Input parameter{in}	
wait_type	wait type
SDIO_WAITTYPE_NO	not wait interrupt
SDIO_WAITTYPE_INTERRUPT	wait interrupt
SDIO_WAITTYPE_DATAEND	wait the end of data transfer

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set the command state machine wait type */

sdio_wait_type_set(SDIO_WAITTYPE_NO);

sdio_csm_enable

The description of sdio_csm_enable is shown as below:

Table 3-500. Function sdio_csm_enable

Function name	sdio_csm_enable
Function prototype	void sdio_csm_enable(void);
Function descriptions	enable the CSM(command state machine)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the CSM(command state machine) */

sdio_csm_enable();

sdio_csm_disable

The description of sdio_csm_disable is shown as below:

Table 3-501. Function sdio_csm_disable

Function name	sdio_csm_disable

Function prototype	void sdio_csm_disable(void);
Function descriptions	disable the CSM(command state machine)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CSM(command state machine) */
sdio_csm_disable();
```

sdio_command_index_get

The description of sdio_command_index_get is shown as below:

Table 3-502. Function sdio_command_index_get

Function name	sdio_command_index_get
Function prototype	uint8_t sdio_command_index_get(void);
Function descriptions	get the last response command index
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	last response command index

Example:

```
/* get SDIO command index */

uint8_t sdio_commond_value;

sdio_commond_value = sdio_command_index_get();
```

sdio_response_get

The description of sdio_response_get is shown as below:

Table 3-503. Function sdio_response_get

Function name	sdio_response_get
Function prototype	uint32_t sdio_response_get(uint32_t responsex);
Function descriptions	get the response for the last received command
Precondition	-
The called functions	-
Input parameter{in}	
responsex	SDIO response
SDIO_RESPONSE0	card response[31:0]/card response[127:96]
SDIO_RESPONSE1	card response[95:64]
SDIO_RESPONSE2	card response[63:32]
SDIO_RESPONSE3	card response[31:1], plus bit 0
Output parameter{out}	
-	-
Return value	
uint32_t	response for the last received command

Example:

```
/* store the CID0 numbers */

uint32_t sdio_cid[0];

sdio_cid[0] = sdio_response_get(SDIO_RESPONSE0);
```

sdio_data_config

The description of sdio_data_config is shown as below:

Table 3-504. Function sdio_data_config

Function name	sdio_data_config
Function prototype	void sdio_data_config(uint32_t data_timeout, uint32_t data_length, uint32_t data_blocksize);
Function descriptions	configure the data timeout, data length and data block size
Precondition	-
The called functions	-
Input parameter{in}	
data_timeout	data timeout period in card bus clock periods
Input parameter{in}	
data_length	number of data bytes to be transferred
Input parameter{in}	
data_blocksize	size of data block for block transfer
SDIO_DATABLOCKSIZE_1BYTE	block size = 1 byte
SDIO_DATABLOCKSIZE_2BYTES	block size = 2 bytes
SDIO_DATABLOCKSIZE_4BYTES	block size = 4 bytes
SDIO_DATABLOCKSIZE_8BYTES	block size = 8 bytes
SDIO_DATABLOCKSIZE_16BYTES	block size = 16 bytes
SDIO_DATABLOCKSIZE_32BYTES	block size = 32 bytes
SDIO_DATABLOCKSIZE_64BYTES	block size = 64 bytes
SDIO_DATABLOCKSIZE_128BYTES	block size = 128 bytes
SDIO_DATABLOCKSIZE_256BYTES	block size = 256 bytes
SDIO_DATABLOCKSIZ	block size = 512 bytes

E_512BYTES	
SDIO_DATABLOCKSIZE_1024BYTES	block size = 1024 bytes
SDIO_DATABLOCKSIZE_2048BYTES	block size = 2048 bytes
SDIO_DATABLOCKSIZE_4096BYTES	block size = 4096 bytes
SDIO_DATABLOCKSIZE_8192BYTES	block size = 8192 bytes
SDIO_DATABLOCKSIZE_16384BYTES	block size = 16384 bytes
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure SDIO data */

sdio_data_config(0, 0, SDIO_DATABLOCKSIZE_1BYTE);

sdio_data_transfer_config

The description of sdio_data_transfer_config is shown as below:

Table 3-505. Function sdio_data_transfer_config

Function name	sdio_data_transfer_config
Function prototype	void sdio_data_transfer_config(uint32_t transfer_mode, uint32_t transfer_direction);
Function descriptions	configure the data transfer mode and direction
Precondition	-
The called functions	-
Input parameter{in}	
transfer_mode	mode of data transfer
SDIO_TRANSMODE_BLOCK	block transfer

SDIO_TRANSMODE_STREAM	stream transfer or SDIO multibyte transfer
Input parameter{in}	
transfer_direction	data transfer direction, read or write
SDIO_TRANSDIRECTION_TOCARD	write data to card
SDIO_TRANSDIRECTION_TOSDIO	read data from card
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure SDIO data transmisson */

sdio_data_transfer_config(SDIO_TRANSDIRECTION_TOSDIO, SDIO_TRANSMODE_BLOCK);

sdio_dsm_enable

The description of sdio_dsm_enable is shown as below:

Table 3-506. Function sdio_dsm_enable

Function name	sdio_dsm_enable
Function prototype	void sdio_dsm_enable(void);
Function descriptions	enable the DSM(data state machine) for data transfer
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable the DSM(data state machine) */

sdio_dsm_enable();

sdio_dsm_disable

The description of sdio_dsm_disable is shown as below:

Table 3-507. Function sdio_dsm_disable

Function name	sdio_dsm_disable
Function prototype	void sdio_dsm_disable(void);
Function descriptions	disable the DSM(data state machine)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the DSM(data state machine) */

sdio_dsm_disable();

sdio_data_write

The description of sdio_data_write is shown as below:

Table 3-508. Function sdio_data_write

Function name	sdio_data_write
Function prototype	void sdio_data_write(uint32_t data);
Function descriptions	write data(one word) to the transmit FIFO
Precondition	-

The called functions	-
Input parameter{in}	
data	32-bit data write to card
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* write data(one word) to the transmit FIFO */

sdio_data_write(0x0000 0001);

sdio_data_read

The description of sdio_data_read is shown as below:

Table 3-509. Function sdio_data_read

Function name	sdio_data_read
Function prototype	uint32_t sdio_data_read(void);
Function descriptions	read data(one word) from the receive FIFO
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	received data

Example:

/* read data(one word) from the receive FIFO */

sdio_data_read();

sdio_data_counter_get

The description of sdio_data_counter_get is shown as below:

Table 3-510. Function sdio_data_counter_get

Function name	sdio_data_counter_get
Function prototype	uint32_t sdio_data_counter_get(void);
Function descriptions	get the number of remaining data bytes to be transferred to card
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	number of remaining data bytes to be transferred

Example:

```
/* get the number of remaining data bytes to be transferred to card */
uint32_t sdio_data_value;
sdio_data_value = sdio_data_counter_get();
```

sdio_fifo_counter_get

The description of sdio_fifo_counter_get is shown as below:

Table 3-511. Function sdio_data_counter_get

Function name	sdio_fifo_counter_get
Function prototype	uint32_t sdio_fifo_counter_get(void);
Function descriptions	get the number of words remaining to be written or read from FIFO
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
uint32_t	remaining number of words

Example:

```
/* get the number of words remaining to be written or read from FIFO */
uint32_t sdio_fifo_value;
sdio_fifo_value = sdio_fifo_counter_get();
```

sdio_dma_enable

The description of sdio_dma_enable is shown as below:

Table 3-512. Function sdio_dma_enable

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

Example:

```
/* enable the SDIO DMA */
sdio_dma_enable();
```

sdio_dma_disable

The description of sdio_dma_disable is shown as below:

Table 3-513. Function sdio_dma_disable

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

Example:

```
/* disable the SDIO DMA */

sdio_dma_disable();
```

sdio_flag_get

The description of sdio_flag_get is shown as below:

Table 3-514. Function sdio_flag_get

Function name	sdio_flag_get
Function prototype	FlagStatus sdio_flag_get(uint32_t flag);
Function descriptions	get the flags state of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
flag	flags state of SDIO
SDIO_FLAG_CCRCERR	command response received (CRC check failed) flag
SDIO_FLAG_DTCRCERR	data block sent/received (CRC check failed) flag

SDIO_FLAG_CMDTMOUT	command response timeout flag
SDIO_FLAG_DTTMOUT	data timeout flag
SDIO_FLAG_TXURE	transmit FIFO underrun error occurs flag
SDIO_FLAG_RXORE	received FIFO overrun error occurs flag
SDIO_FLAG_CMDRECV	command response received (CRC check passed) flag
SDIO_FLAG_CMDSEND	command sent (no response required) flag
SDIO_FLAG_DTEND	data end (data counter, SDIO_DATACNT, is zero) flag
SDIO_FLAG_STBITE	start bit error in the bus flag
SDIO_FLAG_DTBLKEND	data block sent/received (CRC check passed) flag
SDIO_FLAG_CMDRUN	command transmission in progress flag
SDIO_FLAG_TXRUN	data transmission in progress flag
SDIO_FLAG_RXRUN	data reception in progress flag
SDIO_FLAG_TFH	transmit FIFO is half empty flag: at least 8 words can be written into the FIFO
SDIO_FLAG_RFH	receive FIFO is half full flag: at least 8 words can be read in the FIFO
SDIO_FLAG_TFF	transmit FIFO is full flag
SDIO_FLAG_RFF	receive FIFO is full flag
SDIO_FLAG_TFE	transmit FIFO is empty flag
SDIO_FLAG_RFE	receive FIFO is empty flag
SDIO_FLAG_TXDTVAL	data is valid in transmit FIFO flag
SDIO_FLAG_RXDTVAL	data is valid in receive FIFO flag
SDIO_FLAG_SDIOINT	SD I/O interrupt received flag
SDIO_FLAG_ATAEND	CE-ATA command completion signal received (only for CMD61) flag
Output parameter{out}	
-	-

Return value	
FlagStatus	SET or RESET

Example:

/* get the flags state of SDIO */

FlagStatus flag_value;

flag_value = sdio_flag_get(SDIO_FLAG_RFH);

sdio_flag_clear

The description of sdio_flag_clear is shown as below:

Table 3-515. Function sdio_flag_clear

Function name	sdio_flag_clear
Function prototype	void sdio_flag_clear(uint32_t flag);
Function descriptions	clear the pending flags of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
flag	flags state of SDIO
SDIO_FLAG_CCRCERR	command response received (CRC check failed) flag
SDIO_FLAG_DTCRCERR	data block sent/received (CRC check failed) flag
SDIO_FLAG_CMDTMOUT	command response timeout flag
SDIO_FLAG_DTTMOUT	data timeout flag
SDIO_FLAG_TXURE	transmit FIFO underrun error occurs flag
SDIO_FLAG_RXORE	received FIFO overrun error occurs flag
SDIO_FLAG_CMDRECV	command response received (CRC check passed) flag
SDIO_FLAG_CMDSEND	command sent (no response required) flag

SDIO_FLAG_DTEND	data end (data counter, SDIO_DATACNT, is zero) flag
SDIO_FLAG_STBITE	start bit error in the bus flag
SDIO_FLAG_DTBLKEND	data block sent/received (CRC check passed) flag
SDIO_FLAG_SDIOINT	SD I/O interrupt received flag
SDIO_FLAG_ATAEND	CE-ATA command completion signal received (only for CMD61) flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear the pending flags of SDIO */

sdio_flag_clear(SDIO_FLAG_DTCRCERR);

sdio_interrupt_enable

The description of sdio_interrupt_enable is shown as below:

Table 3-516. Function sdio_interrupt_enable

Function name	sdio_interrupt_enable
Function prototype	void sdio_interrupt_enable(uint32_t int_flag);
Function descriptions	enable the SDIO interrupt
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt flags state of SDIO
SDIO_INT_CCRCERR	SDIO CCRCERR interrupt
SDIO_INT_DTCRCERR	SDIO DTCRCERR interrupt
SDIO_INT_CMDTMOUT	SDIO CMDTMOUT interrupt
SDIO_INT_DTTMOUT	SDIO DTTMOUT interrupt

SDIO_INT_TXURE	SDIO TXURE interrupt
SDIO_INT_RXORE	SDIO_INT_RXORE
SDIO_INT_CMDRECV	SDIO CMDRECV interrupt
SDIO_INT_CMDSEND	SDIO CMDSEND interrupt
SDIO_INT_DTEND	SDIO DTEND interrupt
SDIO_INT_STBITE	SDIO STBITE interrupt
SDIO_INT_DTBLKEND	SDIO DTBLKEND interrupt
SDIO_INT_CMDRUN	SDIO CMDRUN interrupt
SDIO_INT_TXRUN	SDIO TXRUN interrupt
SDIO_INT_RXRUN	SDIO RXRUN interrupt
SDIO_INT_TFH	SDIO TFH interrupt
SDIO_INT_RFH	SDIO RFH interrupt
SDIO_INT_TFF	SDIO TFF interrupt
SDIO_INT_RFF	SDIO RFF interrupt
SDIO_INT_TFE	SDIO TFE interrupt
SDIO_INT_RFE	SDIO RFE interrupt
SDIO_INT_TXDTVAL	SDIO TXDTVAL interrupt
SDIO_INT_RXDTVAL	SDIO RXDTVAL interrupt
SDIO_INT_SDIOINT	SDIO SDIOINT interrupt
SDIO_INT_ATAEND	SDIO ATAEND interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the SDIO corresponding interrupts */

sdio_interrupt_enable(SDIO_INT_CCRCERR | SDIO_INT_DTTMOUT | SDIO_INT_RXORE | SDIO_INT_DTEND | SDIO_INT_STBITE);

sdio_interrupt_disable

The description of sdio_interrupt_disable is shown as below:

Table 3-517. Function sdio_interrupt_disable

Function name	sdio_interrupt_disable
Function prototype	void sdio_interrupt_disable(uint32_t int_flag);
Function descriptions	disable the SDIO interrupt
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt flags state of SDIO
SDIO_INT_CCRCERR	SDIO CCRCERR interrupt
SDIO_INT_DTCRCERR	SDIO DTCRCERR interrupt
SDIO_INT_CMDTMOUT	SDIO CMDTMOUT interrupt
SDIO_INT_DTTMOUT	SDIO DTTMOUT interrupt
SDIO_INT_TXURE	SDIO TXURE interrupt
SDIO_INT_RXORE	SDIO_INT_RXORE
SDIO_INT_CMDRECV	SDIO CMDRECV interrupt
SDIO_INT_CMDSEND	SDIO CMDSEND interrupt
SDIO_INT_DTEND	SDIO DTEND interrupt
SDIO_INT_STBITE	SDIO STBITE interrupt
SDIO_INT_DTBLKEND	SDIO DTBLKEND interrupt
SDIO_INT_CMDRUN	SDIO CMDRUN interrupt
SDIO_INT_TXRUN	SDIO TXRUN interrupt
SDIO_INT_RXRUN	SDIO RXRUN interrupt
SDIO_INT_TFH	SDIO TFH interrupt
SDIO_INT_RFH	SDIO RFH interrupt
SDIO_INT_TFF	SDIO TFF interrupt

SDIO_INT_RFF	SDIO RFF interrupt
SDIO_INT_TFE	SDIO TFE interrupt
SDIO_INT_RFE	SDIO RFE interrupt
SDIO_INT_TXDTVAL	SDIO TXDTVAL interrupt
SDIO_INT_RXDTVAL	SDIO RXDTVAL interrupt
SDIO_INT_SDIOINT	SDIO SDIOINT interrupt
SDIO_INT_ATAEND	SDIO ATAEND interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the SDIO interrupt */

sdio_interrupt_disable(SDIO_INT_DTCRCERR);

sdio_interrupt_flag_get

The description of sdio_interrupt_flag_get is shown as below:

Table 3-518. Function sdio_interrupt_flag_get

Function name	sdio_interrupt_flag_get
Function prototype	FlagStatus sdio_interrupt_flag_get(uint32_t int_flag);
Function descriptions	get the interrupt flags state of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt flags state of SDIO
SDIO_INT_CCRCERR	SDIO CCRCERR interrupt
SDIO_INT_DTCRCERR	SDIO DTCRCERR interrupt
SDIO_INT_CMDTMOUT	SDIO CMDTMOUT interrupt

SDIO_INT_DTTMOUT	SDIO DTTMOUT interrupt
SDIO_INT_TXURE	SDIO TXURE interrupt
SDIO_INT_RXORE	SDIO_INT_RXORE
SDIO_INT_CMDRECV	SDIO CMDRECV interrupt
SDIO_INT_CMDSEND	SDIO CMDSEND interrupt
SDIO_INT_DTEND	SDIO DTEND interrupt
SDIO_INT_STBITE	SDIO STBITE interrupt
SDIO_INT_DTBLKEND	SDIO DTBLKEND interrupt
SDIO_INT_CMDRUN	SDIO CMDRUN interrupt
SDIO_INT_TXRUN	SDIO TXRUN interrupt
SDIO_INT_RXRUN	SDIO RXRUN interrupt
SDIO_INT_TFH	SDIO TFH interrupt
SDIO_INT_RFH	SDIO RFH interrupt
SDIO_INT_TFF	SDIO TFF interrupt
SDIO_INT_RFF	SDIO RFF interrupt
SDIO_INT_TFE	SDIO TFE interrupt
SDIO_INT_RFE	SDIO RFE interrupt
SDIO_INT_TXDTVAL	SDIO TXDTVAL interrupt
SDIO_INT_RXDTVAL	SDIO RXDTVAL interrupt
SDIO_INT_SDIOINT	SDIO SDIOINT interrupt
SDIO_INT_ATAEND	SDIO ATAEND interrupt
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the interrupt flags state of SDIO */
FlagStatus flag_value;
```

```
flag_value = sdio_interrupt_flag_get(SDIO_INT_FLAG_DTEND);
```

sdio_interrupt_flag_clear

The description of sdio_interrupt_flag_clear is shown as below:

Table 3-519. Function sdio_interrupt_flag_clear

Function name	sdio_interrupt_flag_clear
Function prototype	void sdio_interrupt_flag_clear(uint32_t int_flag);
Function descriptions	clear the interrupt pending flags of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt flags state of SDIO
SDIO_INT_CCRCERR	SDIO CCRCERR interrupt
SDIO_INT_DTCRCERR	SDIO DTCRCERR interrupt
SDIO_INT_CMDTMOUT	SDIO CMDTMOUT interrupt
SDIO_INT_DTTMOUT	SDIO DTTMOUT interrupt
SDIO_INT_TXURE	SDIO TXURE interrupt
SDIO_INT_RXORE	SDIO_INT_RXORE
SDIO_INT_CMDRECV	SDIO CMDRECV interrupt
SDIO_INT_CMDSEND	SDIO CMDSEND interrupt
SDIO_INT_DTEND	SDIO DTEND interrupt
SDIO_INT_STBITE	SDIO STBITE interrupt
SDIO_INT_DTBLKEND	SDIO DTBLKEND interrupt
SDIO_INT_SDIOINT	SDIO SDIOINT interrupt
SDIO_INT_ATAEND	SDIO ATAEND interrupt
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* clear the interrupt pending flags of SDIO */

sdio_interrupt_flag_clear(SDIO_INT_DTEND);

sdio_readwait_enable

The description of sdio_readwait_enable is shown as below:

Table 3-520. Function sdio_readwait_enable

Function name	sdio_readwait_enable
Function prototype	void sdio_readwait_enable(void);
Function descriptions	enable the read wait mode(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the read wait mode(SD I/O only) */

sdio_readwait_enable();

sdio_readwait_disable

The description of sdio_readwait_disable is shown as below:

Table 3-521. Function sdio_readwait_disable

Function name	sdio_readwait_disable
Function prototype	void sdio_readwait_disable(void);
Function descriptions	disable the read wait mode(SD I/O only)
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the read wait mode(SD I/O only) */

sdio_readwait_disable();

sdio_stop_readwait_enable

The description of sdio_stop_readwait_enable is shown as below:

Table 3-522. Function sdio_stop_readwait_enable

Function name	sdio_stop_readwait_enable
Function prototype	void sdio_stop_readwait_enable(void);
Function descriptions	enable the function that stop the read wait process(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the function that stop the read wait process(SD I/O only) */

sdio_stop_readwait_enable();

sdio_stop_readwait_disable

The description of sdio_stop_readwait_disable is shown as below:

Table 3-523. Function sdio_stop_readwait_disable

Function name	sdio_stop_readwait_disable
Function prototype	void sdio_stop_readwait_disable(void);
Function descriptions	disable the function that stop the read wait process(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the function that stop the read wait process(SD I/O only) */
sdio_stop_readwait_disable();
```

sdio_readwait_type_set

The description of sdio_readwait_type_set is shown as below:

Table 3-524. Function sdio_readwait_type_set

Function name	sdio_readwait_type_set
Function prototype	void sdio_readwait_type_set(uint32_t readwait_type);
Function descriptions	set the read wait type(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
readwait_type	SD I/O read wait type
SDIO_READWAITTYPE_CLK	read wait control by stopping SDIO_CLK

SDIO_READWAITTYPE_DAT2	read wait control using SDIO_DAT[2]
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set the read wait type(SD I/O only) */

sdio_readwait_type_set(SDIO_READWAITTYPE_CLK);

sdio_operation_enable

The description of sdio_operation_enable is shown as below:

Table 3-525. Function sdio_operation_enable

Function name	sdio_operation_enable
Function prototype	void sdio_operation_enable(void);
Function descriptions	enable the SD I/O mode specific operation(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the SD I/O mode specific operation(SD I/O only) */

sdio_operation_enable();

sdio_operation_disable

The description of sdio_operation_disable is shown as below:

Table 3-526. Function sdio_operation_disable

Function name	sdio_operation_disable
Function prototype	void sdio_operation_disable(void);
Function descriptions	disable the SD I/O mode specific operation(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the SD I/O mode specific operation(SD I/O only) */

void sdio_operation_disable();

sdio_suspend_enable

The description of sdio_suspend_enable is shown as below:

Table 3-527. Function sdio_suspend_enable

Function name	sdio_suspend_enable
Function prototype	void sdio_suspend_enable(void);
Function descriptions	enable the SD I/O suspend operation(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable the SD I/O suspend operation(SD I/O only) */

sdio_suspend_enable();

sdio_suspend_disable

The description of sdio_suspend_disable is shown as below:

Table 3-528. Function sdio_suspend_disable

Function name	sdio_suspend_disable
Function prototype	void sdio_suspend_disable(void);
Function descriptions	disable the SD I/O suspend operation(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the SD I/O suspend operation(SD I/O only) */

sdio_suspend_disable();

sdio_ceata_command_enable

The description of sdio_ceata_command_enable is shown as below:

Table 3-529. Function sdio_ceata_command_enable

Function name	sdio_ceata_command_enable
Function prototype	void sdio_ceata_command_enable(void);
Function descriptions	enable the CE-ATA command(CE-ATA only)
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the CE-ATA command(CE-ATA only) */

sdio_ceata_command_enable();

sdio_ceata_command_disable

The description of sdio_ceata_command_disable is shown as below:

Table 3-530. Function sdio_ceata_command_disable

Function name	sdio_ceata_command_disable
Function prototype	void sdio_ceata_command_disable(void);
Function descriptions	disable the CE-ATA command(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the CE-ATA command(CE-ATA only) */

sdio_ceata_command_disable();

sdio_ceata_interrupt_enable

The description of sdio_ceata_interrupt_enable is shown as below:

Table 3-531. Function sdio_ceata_interrupt_enable

Function name	sdio_ceata_interrupt_enable
Function prototype	void sdio_ceata_interrupt_enable(void);
Function descriptions	enable the CE-ATA interrupt(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the CE-ATA interrupt(CE-ATA only) */

sdio_ceata_interrupt_enable();
```

sdio_ceata_interrupt_disable

The description of sdio_ceata_interrupt_disable is shown as below:

Table 3-532. Function sdio_ceata_interrupt_disable

Function name	sdio_ceata_interrupt_disable
Function prototype	void sdio_ceata_interrupt_disable(void);
Function descriptions	disable the CE-ATA interrupt(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* disable the CE-ATA interrupt(CE-ATA only) */

sdio_ceata_interrupt_disable();

sdio_ceata_command_completion_enable

The description of sdio_ceata_command_completion_enable is shown as below:

Table 3-533. Function sdio_ceata_command_completion_enable

Function name	sdio_ceata_command_completion_enable
Function prototype	void sdio_ceata_command_completion_enable(void);
Function descriptions	enable the CE-ATA command completion signal(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the CE-ATA command completion signal(CE-ATA only) */

sdio_ceata_command_completion_enable();

sdio_ceata_command_completion_disable

The description of sdio_ceata_command_completion_disable is shown as below:

Table 3-534. Function sdio_ceata_command_completion_disable

Function name	sdio_ceata_command_completion_disable
Function prototype	void sdio_ceata_command_completion_disable(void);

Function descriptions	disable the CE-ATA command completion signal(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CE-ATA command completion signal(CE-ATA only) */
sdio_ceata_command_completion_disable();
```

3.21. SPI

The SPI/I2S module can communicate with external devices using the SPI protocol or the I2S audio protocol. The SPI/I2S registers are listed in chapter ***3.21.1*** , the SPI/I2S firmware functions are introduced in chapter ***3.21.2***.

3.21.1. Descriptions of Peripheral registers

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

Table 3-535. SPI/I2S Registers

Registers	Descriptions
SPI_CTL0	SPI control register 0
SPI_CTL1	SPI control register 1
SPI_STAT	SPI status register
SPI_DATA	SPI data register
SPI_CRCPOLY	SPI CRC polynomial register
SPI_RCRC	SPI receive CRC register
SPI_TCRC	SPI transmit CRC register

Registers	Descriptions
SPI_I2SCTL	SPI/I2S control register
SPI_I2SPSC	SPI/I2S clock prescaler register

3.21.2. Descriptions of Peripheral functions

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

Table 3-536. 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
spi_init	Initialize SPI parameter
spi_enable	Enable SPI
spi_disable	Disable SPI
i2s_init	Initialize I2S parameter
i2s_psc_config	Configure I2S 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	SPI NSS pin high level in software mode
spi_nss_internal_low	SPI NSS pin low level in software mode
spi_dma_enable	Enable SPI DMA function
spi_dma_disable	Disable SPI DMA function
spi_i2s_data_frame_format_config	Configure SPI/I2S data frame format
spi_i2s_data_transmit	SPI transmit data
spi_i2s_data_receive	SPI receive data
spi_bidirectional_transfer_config	Configure SPI bidirectional transfer direction
spi_crc_polynomial_set	Set SPI CRC polynomial
spi_crc_polynomial_get	Get SPI CRC polynomial

Function name	Function description
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_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
spi_i2s_flag_get	Get SPI and I2S flag status
spi_crc_error_clear	Clear SPI CRC error flag status

Structure spi_parameter_struct

Table 3-537. 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_16BIT, SPI_FRAMESIZE_8BIT)
nss	SPI NSS control by handware 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_1EDGESPI_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 3-538. Function spi_i2s_deinit

Function name	spi_i2s_deinit
Function prototype	void spi_i2s_deinit(uint32_t spi_periph);
Function descriptions	Reset SPI and I2S peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
spi_periph	SPI/I2S peripheral
SPIx	x=0,1,2
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 3-539. 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 parameter stuct, the structure members can refer to members of the structure***Table 3-537. spi_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of SPI struct */
spi_parameter_struct spi_struct;
spi_struct->device_mode = SPI_SLAVE;
spi_struct->trans_mode = SPI_TRANSMODE_FULLDUPLEX;
spi_struct->frame_size = SPI_FRAMESIZE_8BIT;
spi_struct->nss = SPI_NSS_HARD;
spi_struct->clock_polarity_phase = SPI_CK_PL_LOW_PH_1EDGE;
spi_struct->prescale = SPI_PSC_2;
spi_struct_para_init(&spi_struct);
```

spi_init

The description of spi_init is shown as below:

Table 3-540. Function spi_init

Function name	spi_init
Function prototype	void spi_init(uint32_t spi_periph, spi_parameter_struct* spi_struct);
Function descriptions	Initialize SPI peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
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 3-537. 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 3-541. Function spi_enable

Function name	spi_enable
Function prototype	void spi_enable(uint32_t spi_periph);
Function descriptions	Enable SPI
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
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 3-542. Function spi_disable

Function name	spi_disable
Function prototype	void spi_disable(uint32_t spi_periph);
Function descriptions	Disable SPI
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 3-543. 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 I2S peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=1,2
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 I2S1 */

i2s_init(SPI1, I2S_MODE_MASTERTX, I2S_STD_PHILLIPS, I2S_CKPL_LOW);

i2s_psc_config

The description of i2s_psc_config is shown as below:

Table 3-544. 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 I2S prescaler
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=1,2
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_16K	audio sample rate is 16KHz
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 I2S1 prescaler */

i2s_psc_config(SPI1, I2S_AUDIOSAMPLE_44K, I2S_FRAMEFORMAT_DT16B_CH16B, I2S_MCKOUT_DISABLE);

i2s_enable

The description of i2s_enable is shown as below:

Table 3-545. Function i2s_enable

Function name	i2s_enable
Function prototype	void i2s_enable(uint32_t spi_periph);
Function descriptions	Enable I2S
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2S1*/

i2s_enable(SPI1);
```

i2s_disable

The description of i2s_disable is shown as below:

Table 3-546. Function i2s_disable

Function name	i2s_disable
Function prototype	void i2s_disable(uint32_t spi_periph);
Function descriptions	Disable I2S
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=1,2
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* disable I2S1*/

i2s_disable(SPI1);

spi_nss_output_enable

The description of spi_nss_output_enable is shown as below:

Table 3-547. Function spi_nss_output_enable

Function name	spi_nss_output_enable
Function prototype	void spi_nss_output_enable(uint32_t spi_periph);
Function descriptions	Enable SPI NSS output function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI 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 3-548. Function spi_nss_output_disable

Function name	spi_nss_output_disable

Function prototype	void spi_nss_output_disable(uint32_t spi_periph);
Function descriptions	Disable SPI NSS output function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI 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 3-549. Function spi_nss_internal_high

Function name	spi_nss_internal_high
Function prototype	void spi_nss_internal_high(uint32_t spi_periph);
Function descriptions	SPI NSS pin high level in software mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
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 3-550. Function spi_nss_internal_low

Function name	spi_nss_internal_low
Function prototype	void spi_nss_internal_low(uint32_t spi_periph);
Function descriptions	SPI NSS pin low level in software mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
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 3-551. 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 SPI 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 3-552. 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 SPI 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 3-553. 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 SPI/I2S 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_16BIT	SPI frame size is 16 bits
SPI_FRAMESIZE_8BIT	SPI frame size is 8 bits
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* configure SPI1/I2S1 data frame format size is 16 bits */

spi_i2s_data_frame_format_config(SPI1, SPI_FRAMESIZE_16BIT);

spi_i2s_data_transmit

The description of spi_i2s_data_transmit is shown as below:

Table 3-554. 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:

uint16_t spi0_send_data = 0xAA:;

/* SPI0 transmit data */

spi_i2s_data_transmit(SPI0, spi0_send_data);

spi_i2s_data_receive

The description of spi_i2s_data_receive is shown as below:

Table 3-555. 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:

```
uint16_t spi0_receive_data;
/* SPI0 receive data */
spi0_receive_data = spi_i2s_data_receive(SPI0);
```

spi_bidirectional_transfer_config

The description of spi_bidirectional_transfer_config is shown as below:

Table 3-556. Function spi_bidirectional_transfer_config

Function name	spi_bidirectional_transfer_config
Function prototype	void spi_bidirectional_transfer_config(uint32_t spi_periph, uint32_t transfer_direction);
Function descriptions	Configure SPI bidirectional transfer direction
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
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_crc_polynomial_set

The description of spi_crc_polynomial_set is shown as below:

Table 3-557. 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 3-558. 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 (0-0xFFFF)

Example:

uint16_t crc_data;

/* get SPI0 CRC polynomial */

crc_data = spi_crc_polynomial_get(SPI0);

spi_crc_on

The description of spi_crc_on is shown as below:

Table 3-559. Function spi_crc_on

Function name	spi_crc_on
Function prototype	void spi_crc_on(uint32_t spi_periph);

Function descriptions	Turn on SPI 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 3-560. Function spi_crc_off

Function name	spi_crc_off
Function prototype	void spi_crc_off(uint32_t spi_periph);
Function descriptions	Turn off SPI 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 3-561. 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 3-562. 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 (0-0xFFFF)

Example:

```
uint16_t crc_value;

/* get SPI0 CRC send value */

crc_value = spi_crc_get(SPI0, SPI_CRC_TX);
```

spi_i2s_interrupt_enable

The description of spi_i2s_interrupt_enable is shown as below:

Table 3-563. Function spi_i2s_interrupt_enable

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

interrupt	SPI/I2S interrupt
SPI_I2S_INT_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, transmission underrun error and format 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 3-564. Function spi_i2s_interrupt_disable

Function name	spi_i2s_interrupt_disable
Function prototype	void spi_i2s_interrupt_disable(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	Disable SPI and I2S interrupt
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
interrupt	SPI/I2S interrupt
SPI_I2S_INT_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, transmission underrun error and format 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 3-565. Function spi_i2s_interrupt_flag_get

Function name	spi_i2s_interrupt_flag_get
Function prototype	FlagStatus spi_i2s_interrupt_flag_get(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	Get SPI and I2S interrupt status
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
interrupt	SPI/I2S interrupt flag status
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 */

FlagStatus Flag_ interrupt = RESET;

Flag_ interrupt = spi_i2s_interrupt_flag_get(SPI0, SPI_I2S_INT_FLAG_TBE);

spi_i2s_flag_get

The description of spi_i2s_flag_get is shown as below:

Table 3-566. Function spi_i2s_flag_get

Function name	spi_i2s_flag_get
Function prototype	FlagStatus spi_i2s_flag_get(uint32_t spi_periph, uint32_t flag);
Function descriptions	Get SPI and I2S flag status
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
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_I2S_INT_FLAG_RXORERR	receive overrun error flag
SPI_FLAG_CONFERR	mode config error flag

SPI_FLAG_CRCERR	CRC error 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 */
FlagStatus Flag = RESET;
Flag = spi_i2s_flag_get(SPI0, SPI_FLAG_TBE);
```

spi_crc_error_clear

The description of spi_crc_error_clear is shown as below:

Table 3-567. 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);
```

3.22. TIMER

The timers have a 16-bit counter that can be used as an unsigned counter and supports both input capture and output compare. Timers (TIMERx) are divided into five sorts: advanced timer (TIMERx, x=0, 7), general level0 timer (TIMERx, x=1, 2, 3, 4), general level1 timer (TIMERx, x=8, 11), general level2 timer (TIMERx, x=9, 10, 12, 13), Basic timer (TIMERx, x=5, 6). The specific functions of different types of timer are different. The TIMER registers are listed in chapter ***3.22.1***, the TIMER firmware functions are introduced in chapter ***3.22.2***.

3.22.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

Table 3-568. TIMERx Registers

Registers	Descriptions
TIMER_CTL0	Control register 0
TIMER_CTL1	Control register 1
TIMER_SMCFG	Slave mode configuration register
TIMER_DMAINTEN	DMA and interrupt enable register
TIMER_INTF	Interrupt flag register
TIMER_SWEVG	Software event generation register
TIMER_CHCTL0	Channel control register 0
TIMER_CHCTL1	Channel control register 1
TIMER_CHCTL2	Channel control register 2
TIMER_CNT	Counter register
TIMER_PSC	Prescaler register
TIMER_CAR	Counter auto reload register
TIMER_CREP	Counter repetition register
TIMER_CH0CV	Channel 0 capture/compare value register
TIMER_CH1CV	Channel 1 capture/compare value register
TIMER_CH2CV	Channel 2 capture/compare value register

Registers	Descriptions
TIMER_CH3CV	Channel 3 capture/compare value register
TIMER_CCHP	Channel complementary protection register
TIMER_DMACFG	DMA configuration register
TIMER_DMATB	DMA transfer buffer register

3.22.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

Table 3-569.TIMERx firmware function

Function name	Function description
timer_deinit	deinit a TIMER
timer_struct_para_init	initialize TIMER init parameter struct
timer_init	initialize TIMER counterx
timer_enable	enable a TIMER
timer_disable	disable a TIMER
timer_auto_reload_shadow_enable	enable the auto reload shadow function
timer_auto_reload_shadow_disable	disable the auto reload shadow function
timer_update_event_enable	enable the update event
timer_update_event_disable	disable the update event
timer_counter_alignment	set TIMER counter alignment mode
timer_counter_up_direction	set TIMER counter up direction
timer_counter_down_direction	set TIMER counter down direction
timer_prescaler_config	configure TIMER prescaler
timer_repetition_value_config	configure TIMER repetition register value
timer_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

Function name	Function description
timer_update_source_config	configure TIMER update source
timer_dma_enable	enable the TIMER DMA
timer_dma_disable	disable the TIMER DMA
timer_channel_dma_request_source_select	channel DMA request source selection
timer_dma_transfer_config	configure the TIMER DMA transfer
timer_event_software_generate	software generate events
timer_break_struct_para_init	initialize TIMER break parameter struct
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	enable or disable TIMER primary output function
timer_channel_control_shadow_config	enable or disable channel capture/compare control shadow register
timer_channel_control_shadow_update_config	configure TIMER channel control shadow register update control
timer_channel_output_struct_para_init	initialize TIMER channel output parameter struct
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

Function name	Function description
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 TIMER channel input parameter struct
timer_input_capture_config	configure TIMER input capture parameter
timer_channel_input_capture_prescaler_config	configure TIMER channel input capture prescaler value
timer_channel_capture_value_register_read	read TIMER channel capture compare register value
timer_input_pwm_capture_config	configure TIMER input pwm capture function
timer_hall_mode_config	configure TIMER hall sensor mode
timer_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_clock_config	configure TIMER the internal trigger as external clock input
timer_external_trigger_as_external_clock_config	configure TIMER the external trigger as external clock input
timer_external_clock_mode0_config	configure TIMER the external clock mode 0
timer_external_clock_mode1_config	configure TIMER the external clock mode 1
timer_external_clock_mode1_disable	disable TIMER the external clock mode 1

Function name	Function description
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

Structure timer_parameter_struct

Table 3-570. 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 3-571. 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）

Member name	Function description
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 3-572. Structure timer_oc_parameter_struct

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

Structure timer_ic_parameter_struct

Table 3-573. 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 3-574. 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..13)	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 3-575. 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 TIMER init parameter struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
initpara	init parameter struct, the structure members can refer to ***Table 3-570. Structure timer_parameter_struct***

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the TIMER structure */
timer_parameter_struct initpara;
initpara->prescaler = 0U;
initpara->alignedmode = TIMER_COUNTER_EDGE;
initpara->counterdirection = TIMER_COUNTER_UP;
initpara->period = 65535U;
initpara->clockdivision = TIMER_CKDIV_DIV1;
initpara->repetitioncounter = 0U;
timer_struct_para_init(&initpara);
```

timer_init

The description of timer_init is shown as below:

Table 3-576. 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..13)	TIMER peripheral selection
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to ***Table 3-570. Structure timer_parameter_structcture*** **timer_parameter_struct**

		Member name	**Function description**
		prescaler	prescaler value（0~65535）
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 3-577. 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..13)	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 3-578. 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..13)	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 3-579. 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..13)	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 3-580. 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..13)	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 3-581. 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..13)	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 3-582. 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..13)	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 3-583. Function timer_counter_alignment

Function name	timer_counter_alignment
Function prototype	void timer_counter_alignment(uint32_t timer_periph, uint16_t aligned);
Function descriptions	set TIMER counter alignment mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7..13)	TIMER peripheral selection
Input parameter{in}	
aligned	alignment mode
TIMER_COUNTER_ED	No center-aligned mode (edge-aligned mode). The direction of the counter

GE	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 3-584. 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..4,7..13)	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 3-585. 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..4,7..13)	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 3-586. 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..13)	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_UPDATE	the prescaler is loaded at the next update event
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 prescaler */

timer_prescaler_config(TIMER0, 3000, TIMER_PSC_RELOAD_NOW);

timer_repetition_value_config

The description of timer_repetition_value_config is shown as below:

Table 3-587. Function timer_repetition_value_config

Function name	timer_repetition_value_config

Function prototype	void timer_repetition_value_config(uint32_t timer_periph, uint8_t repetition);
Function descriptions	configure TIMER repetition register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	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 3-588. Function timer_autoreload_value_config

Function name	timer_autoreload_value_config
Function prototype	void timer_autoreload_value_config(uint32_t timer_periph, uint32_t autoreload);
Function descriptions	configure TIMER autoreload register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection

Input parameter{in}	
autoreload	the counter auto-reload value
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 3-589. Function timer_counter_value_config

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

Example:

/* configure TIMER0 counter register value */

timer_counter_value_config(TIMER0);

timer_counter_read

The description of timer_counter_read is shown as below:

Table 3-590. 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..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint16_t	counter value(0x0000~0xFFFF)

Example:

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

timer_prescaler_read

The description of timer_prescaler_read is shown as below:

Table 3-591. 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..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint16_t	prescaler register value(0x0000~0xFFFF)

Example:

```
/* read TIMER0 prescaler value */
uint16_t i = 0;
i = timer_prescaler_read(TIMER0);
```

timer_single_pulse_mode_config

The description of timer_single_pulse_mode_config is shown as below:

Table 3-592. 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..8,11)	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 3-593. Function timer_update_source_config

Function name	timer_update_source_config
Function prototype	void timer_update_source_config(uint32_t timer_periph, uint8_t update);
Function descriptions	configure TIMER update source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	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_dma_enable

The description of timer_dma_enable is shown as below:

Table 3-594. 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..7)
TIMER_DMA_CH0D	channel 0 DMA enable,TIMERx(x=0..4,7)
TIMER_DMA_CH1D	channel 1 DMA enable,TIMERx(x=0..4,7)
TIMER_DMA_CH2D	channel 2 DMA enable,TIMERx(x=0..4,7)
TIMER_DMA_CH3D	channel 3 DMA enable,TIMERx(x=0..4,7)
TIMER_DMA_CMTD	commutation DMA request enable,TIMERx(x=0,7)
TIMER_DMA_TRGD	trigger DMA enable,TIMERx(x=0..4,7)
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable the TIMER0 update DMA */

timer_dma_enable(TIMER0, TIMER_DMA_UPD);

timer_dma_disable

The description of timer_dma_disable is shown as below:

Table 3-595. 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 disable,TIMERx(x=0..7)
TIMER_DMA_CH0D	channel 0 DMA disable,TIMERx(x=0..4,7)
TIMER_DMA_CH1D	channel 1 DMA disable,TIMERx(x=0..4,7)
TIMER_DMA_CH2D	channel 2 DMA disable,TIMERx(x=0..4,7)
TIMER_DMA_CH3D	channel 3 DMA disable,TIMERx(x=0..4,7)
TIMER_DMA_CMTD	commutation DMA request disable,TIMERx(x=0,7)
TIMER_DMA_TRGD	trigger DMA disable,TIMERx(x=0..4,7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the TIMER0 update DMA */

timer_dma_disable(TIMER0, TIMER_DMA_UPD);

timer_channel_dma_request_source_select

The description of timer_channel_dma_request_source_select is shown as below:

Table 3-596. 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, uint8_t dma_request);
Function descriptions	channel DMA request source selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7)	TIMER peripheral selection
Input parameter{in}	
dma_request	channel DMA request source selection
TIMER_DMAREQUEST_CHANNELEVENT	DMA request of channel y is sent when channel y event occurs
TIMER_DMAREQUEST_UPDATEEVENT	DMA request of channel y is sent when update event occurs
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* TIMER0 channel DMA request of channel y 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 3-597. 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	please refer to the following parameters
Input parameter{in}	
dma_baseaddr	DMA transfer access start address
TIMER_DMACFG_DMATA_CTL0	DMA transfer address is TIMER_CTL0,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CTL1	DMA transfer address is TIMER_CTL1,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_SMCFG	DMA transfer address is TIMER_SMCFG,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_DMAINTEN	DMA transfer address is TIMER_DMAINTEN,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_INTF	DMA transfer address is TIMER_INTF,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_SWEVG	DMA transfer address is TIMER_SWEVG,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CHCTL0	DMA transfer address is TIMER_CHCTL0,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CHCTL1	DMA transfer address is TIMER_CHCTL1,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CHCTL2	DMA transfer address is TIMER_CHCTL2,TIMERx(x=0..4,7)

TIMER_DMACFG_DMATA_CNT	DMA transfer address is TIMER_CNT,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_PSC	DMA transfer address is TIMER_PSC,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CAR	MA transfer address is TIMER_CAR,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CREP	DMA transfer address is TIMER_CREP,TIMERx(x=0,7)
TIMER_DMACFG_DMATA_CH0CV	DMA transfer address is TIMER_CH0CV,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CH1CV	DMA transfer address is TIMER_CH1CV,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CH2CV	DMA transfer address is TIMER_CH2CV,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CH3CV	DMA transfer address is TIMER_CH3CV,TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CCHP	DMA transfer address is TIMER_CCHP,TIMERx(x=0,7)
TIMER_DMACFG_DMATA_DMACFG	DMA transfer address is TIMER_DMACFG,TIMERx(x=0..4,7)
Input parameter{in}	
dma_lenth	DMA transfer count, TIMER_DMACFG_DMATC_xTRANSFER(x=1..18)
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 3-598. 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..13)
TIMER_EVENT_SRC_CH0G	channel 0 capture or compare event generation,TIMERx(x=0..4,7..13)
TIMER_EVENT_SRC_CH1G	channel 1 capture or compare event generation,TIMERx(x=0..4,7,8,11)
TIMER_EVENT_SRC_CH2G	channel 2 capture or compare event generation,TIMERx(x=0..4,7)
TIMER_EVENT_SRC_CH3G	channel 3 capture or compare event generation,TIMERx(x=0..4,7)
TIMER_EVENT_SRC_CMTG	channel commutation event generation,TIMERx(x=0,7)
TIMER_EVENT_SRC_TRGG	trigger event generation,TIMERx(x=0..4,7,8,11)
TIMER_EVENT_SRC_BRKG	break event generation,TIMERx(x=0,7)
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 3-599. 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 TIMER break parameter struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
breakpara	TIMER break parameter struct, refer to ***Table 3-571. Structure timer_break_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize the TIMER break parameter structure */

timer_break_parameter_struct breakpara;

breakpara->runoffstate = TIMER_ROS_STATE_DISABLE;

breakpara->ideloffstate = TIMER_IOS_STATE_DISABLE;

breakpara->deadtime = 0U;

breakpara->breakpolarity = TIMER_BREAK_POLARITY_LOW;

breakpara->outputautostate = TIMER_OUTAUTO_DISABLE;

breakpara->protectmode = TIMER_CCHP_PROT_OFF;

breakpara->breakstate = TIMER_BREAK_DISABLE;

timer_break_struct_para_init(&breakpara);

timer_break_config

The description of timer_break_config is shown as below:

Table 3-600. Function timer_break_config

Function name	timer_break_config
Function prototype	void timer_break_config(uint32_t timer_periph, timer_break_parameter_struct* breakpara);
Function descriptions	configure TIMER break function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
breakpara	TIMER break parameter struct, refer to ***Table 3-571. 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 3-601. 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=0,7)	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 3-602. 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=0,7)	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 t is shown as below:

Table 3-603. 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=0,7)	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 t is shown as below:

Table 3-604. 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=0,7)	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 3-605. 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	enable or disable TIMER primary output function
Precondition	-
The called functions	-
Input parameter{in}	

timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* 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 3-606. 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	enable or disable channel capture/compare control shadow register
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
ENABLE	enable function

DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* channel 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 3-607. Function timer_channel_control_shadow_update_config

Function name	timer_channel_control_shadow_update_config
Function prototype	void timer_channel_control_shadow_update_config(uint32_t timer_periph, uint32_t ccuctl);
Function descriptions	configure TIMER channel control shadow register update control
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	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 3-608. 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 TIMER channel output parameter struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
ocpara	TIMER channel output parameter struct，refer to***Table 3-572. Structure timer_oc_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize the TIMER channel output parameter structure */

timer_oc_parameter_struct ocpara;

ocpara->outputstate = TIMER_CCX_DISABLE;

ocpara->outputnstate = TIMER_CCXN_DISABLE;

ocpara->ocpolarity = TIMER_OC_POLARITY_HIGH;

ocpara->ocnpolarity = TIMER_OCN_POLARITY_HIGH;

ocpara->ocidlestate = TIMER_OC_IDLE_STATE_LOW;

ocpara->ocnidlestate = TIMER_OCN_IDLE_STATE_LOW;

timer_channel_output_struct_para_init(&ocpara);

timer_channel_output_config

The description of timer_channel_output_config is shown as below:

Table 3-609. 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..4,7..13))
TIMER_CH_1	TIMER channel 1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2(TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel 3(TIMERx(x=0..4,7))
Input parameter{in}	
ocpara	TIMER channel output parameter struct，refer to ***Table 3-572. Structure timer_oc_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output function */
timer_oc_parameter_struct timer_ocintpara;
timer_ocintpara.outputstate = TIMER_CCX_ENABLE;
```

```
timer_ocintpara.outputnstate = TIMER_CCXN_ENABLE;
timer_ocintpara.ocpolarity = TIMER_OC_POLARITY_HIGH;
timer_ocintpara.ocnpolarity = TIMER_OCN_POLARITY_HIGH;
timer_ocintpara.ocidlestate = TIMER_OC_IDLE_STATE_HIGH;
timer_ocintpara.ocnidlestate = TIMER_OCN_IDLE_STATE_LOW;
timer_channel_output_config(TIMER0,TIMER_CH_0,&timer_ocintpara);
```

timer_channel_output_mode_config

The description of timer_channel_output_mode_config is shown as below:

Table 3-610. 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 channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel3(TIMERx(x=0..4,7))
Input parameter{in}	
ocmode	channel output compare mode
TIMER_OC_MODE_TIMING	frozen mode
TIMER_OC_MODE_AC	set the channel output

TIVE	
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 3-611. 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, uint16_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 channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel3(TIMERx(x=0..4,7))
Input parameter{in}	
pulse	channel output pulse value（0~65535）
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 output pulse value */

timer_channel_output_pulse_value_config(TIMER0,TIMER_CH_0,399);

timer_channel_output_shadow_config

The description of timer_channel_output_shadow_config is shown as below:

Table 3-612. 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 channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel3(TIMERx(x=0..4,7))
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 3-613. 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 channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel3(TIMERx(x=0..4,7))
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 3-614. 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(x=0..4,7)	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel0
TIMER_CH_1	TIMER channel1
TIMER_CH_2	TIMER channel2
TIMER_CH_3	TIMER channel3
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 3-615. 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 channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel3(TIMERx(x=0..4,7))
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 3-616. 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	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
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 3-617. 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 channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel3(TIMERx(x=0..4,7))
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 3-618. 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=0,7)	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel0
TIMER_CH_1	TIMER channel1
TIMER_CH_2	TIMER channel2
Input parameter{in}	
ocnstate	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 3-619. 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 TIMER channel input parameter struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
icpara	TIMER channel intput parameter struct, refer to***Table 3-573. Structure timer_ic_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the TIMER channel input parameter structure */
timer_ic_parameter_struct icpara;
icpara->icpolarity = TIMER_IC_POLARITY_RISING;
icpara->icselection = TIMER_IC_SELECTION_DIRECTTI;
icpara->icprescaler = TIMER_IC_PSC_DIV1;
icpara->icfilter = 0U;
timer_channel_input_struct_para_init(&icpara);
```

timer_input_capture_config

The description of timer_input_capture_config is shown as below:

Table 3-620. 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 channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel3(TIMERx(x=0..4,7))
Input parameter{in}	
icpara	TIMER channel intput parameter struct, refer to***Table 3-573. 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 3-621. 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 channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel3(TIMERx(x=0..4,7))
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 3-622. 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 channel0(TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel1(TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel2(TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel3(TIMERx(x=0..4,7))
Output parameter{out}	
-	-
Return value	
uint32_t	channel capture compare register value(0x0000~0xFFFF)

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 3-623. 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..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel0
TIMER_CH_1	TIMER channel1
Input parameter{in}	
icpwm	channel intput pwm parameter struct, refer to ***Table 3-573. 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 3-624. 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..4,7)	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 3-625. 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	please refer to the following parameters
Input parameter{in}	
intrigger	trigger selection
TIMER_SMCFG_TRGSEL_ITI0	Internal trigger input 0 (ITI0) , TIMERx(x=0..4,7,8,11)
TIMER_SMCFG_TRGSEL_ITI1	Internal trigger input 1 (ITI1) , TIMERx(x=0..4,7,8,11)
TIMER_SMCFG_TRGSEL_ITI2	Internal trigger input 2 (ITI2) , TIMERx(x=0..4,7,8,11)
TIMER_SMCFG_TRGSEL_ITI3	Internal trigger input 3 (ITI3) , TIMERx(x=0..4,7,8,11)
TIMER_SMCFG_TRGSEL_CI0F_ED	TI0 edge detector , TIMERx(x=0..4,7,8,11)
TIMER_SMCFG_TRGSEL_CI0FE0	filtered TIMER input 0 , TIMERx(x=0..4,7,8,11)
TIMER_SMCFG_TRGSEL_CI1FE1	filtered TIMER input 1 , TIMERx(x=0..4,7,8,11)
TIMER_SMCFG_TRGSEL_ETIFP	external trigger , TIMERx(x=0..4,7)
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 3-626. 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..7)	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	a capture or a compare match occurred in channel 0 as trigger output TRGO.
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 3-627. Function timer_slave_mode_select

Function name	timer_slave_mode_select
Function prototype	void timer_slave_mode_select(uint32_t timer_periph, uint32_t slavemode);
Function descriptions	select TIMER slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	
slavemode	slave mode
TIMER_SLAVE_MODE_DISABLE	slave mode disable
TIMER_ENCODER_MODE0	encoder mode 0
TIMER_ENCODER_MODE1	encoder mode 1

TIMER_ENCODER_MODE2	encoder mode 2
TIMER_SLAVE_MODE_RESTART	restart mode
TIMER_SLAVE_MODE_PAUSE	pause mode
TIMER_SLAVE_MODE_EVENT	event mode
TIMER_SLAVE_MODE_EXTERNAL0	external clock mode 0
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TIMER0 slave mode */
timer_slave_mode_select(TIMER0, TIMER_ENCODER_MODE0);
```

timer_master_slave_mode_config

The description of timer_master_slave_mode_config is shown as below:

Table 3-628. Function timer_master_slave_mode_config

Function name	timer_master_slave_mode_config
Function prototype	void timer_master_slave_mode_config(uint32_t timer_periph, uint32_t masterslave);
Function descriptions	configure TIMER master slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	

masterslave	master slave mode state
TIMER_MASTER_SLAVE_MODE_ENABLE	master slave mode enable
TIMER_MASTER_SLAVE_MODE_DISABLE	master slave mode disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 master slave mode  */
timer_master_slave_mode_config(TIMER0, TIMER_MASTER_SLAVE_MODE_ENABLE);
```

timer_external_trigger_config

The description of timer_external_trigger_config is shown as below:

Table 3-629. 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, uint8_t extfilter);
Function descriptions	configure TIMER external trigger input
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7)	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 3-630. 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..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	
decomode	quadrature decoder mode
TIMER_ENCODER_MODE0	counter counts on CI0FE0 edge depending on CI1FE1 level
TIMER_ENCODER_MODE1	counter counts on CI1FE1 edge depending on CI0FE0 level
TIMER_ENCODER_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_ENCODER_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 3-631. 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..4,7,8,11)	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 3-632. 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..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	

intrigger	trigger selection
TIMER_SMCFG_TRGSEL_ITI0	Internal trigger input 0 (ITI0)
TIMER_SMCFG_TRGSEL_ITI1	Internal trigger input 0 (ITI1)
TIMER_SMCFG_TRGSEL_ITI2	Internal trigger input 0 (ITI2)
TIMER_SMCFG_TRGSEL_ITI3	Internal trigger input 0 (ITI3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 the internal trigger ITI0 as external clock input */

timer_internal_trigger_as_external_clock_config(TIMER0, TIMER_SMCFG_TRGSEL_ITI0);

timer_external_trigger_as_external_clock_config

The description of timer_external_trigger_as_external_clock_config is shown as below:

Table 3-633. 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, uint8_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..4,7,8,11)	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
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 3-634. 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, uint8_t extfilter);
Function descriptions	configure TIMER the external clock mode0
Precondition	-

The called functions	timer_external_trigger_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	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 3-635. 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, uint8_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..4,7)	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 3-636. 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..4,7)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable TIMER0 the external clock mode1 */

timer_external_clock_mode1_disable(TIMER0);

timer_interrupt_enable

The description of timer_interrupt_enable is shown as below:

Table 3-637. 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..13)
TIMER_INT_CH0	channel 0 interrupt enable, TIMERx(x=0..4,7..13)
TIMER_INT_CH1	channel 1 interrupt enable, TIMERx(x=0..4,7,8,11)
TIMER_INT_CH2	channel 2 interrupt enable, TIMERx(x=0..4,7)
TIMER_INT_CH3	channel 3 interrupt enable , TIMERx(x=0..4,7)
TIMER_INT_CMT	commutation interrupt enable, TIMERx(x=0,7)
TIMER_INT_TRG	trigger interrupt enable, TIMERx(x=0..4,7,8,11)
TIMER_INT_BRK	break interrupt enable, TIMERx(x=0,7)
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 3-638. 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..13)
TIMER_INT_CH0	channel 0 interrupt disable, TIMERx(x=0..4,7..13)
TIMER_INT_CH1	channel 1 interrupt disable, TIMERx(x=0..4,7,8,11)
TIMER_INT_CH2	channel 2 interrupt disable, TIMERx(x=0..4,7)
TIMER_INT_CH3	channel 3 interrupt disable, TIMERx(x=0..4,7)
TIMER_INT_CMT	commutation interrupt disable, TIMERx(x=0,7)
TIMER_INT_TRG	trigger interrupt disable, TIMERx(x=0..4,7,8,11)
TIMER_INT_BRK	break interrupt disable, TIMERx(x=0,7)
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 3-639. 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..13)
TIMER_INT_FLAG_CH0	channel 0 interrupt flag,TIMERx(x=0..4,7..13)
TIMER_INT_FLAG_CH1	channel 1 interrupt flag,TIMERx(x=0..4,7,8,11)
TIMER_INT_FLAG_CH2	channel 2 interrupt flag,TIMERx(x=0..4,7)
TIMER_INT_FLAG_CH3	channel 3 interrupt flag,TIMERx(x=0..4,7)
TIMER_INT_FLAG_CMT	channel commutation interrupt flag,TIMERx(x=0,7)
TIMER_INT_FLAG_TRG	trigger interrupt flag,TIMERx(x=0,7,8,11)
TIMER_INT_FLAG_BRK	break interrupt flag,TIMERx(x=0,7)
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 3-640. 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..13)
TIMER_INT_FLAG_CH0	channel 0 interrupt flag,TIMERx(x=0..4,7..13)
TIMER_INT_FLAG_CH1	channel 1 interrupt flag,TIMERx(x=0..4,7,8,11)
TIMER_INT_FLAG_CH2	channel 2 interrupt flag,TIMERx(x=0..4,7)
TIMER_INT_FLAG_CH3	channel 3 interrupt flag,TIMERx(x=0..4,7)
TIMER_INT_FLAG_CMT	channel commutation interrupt flag,TIMERx(x=0,7)
TIMER_INT_FLAG_TRG	trigger interrupt flag,TIMERx(x=0,7,8,11)
TIMER_INT_FLAG_BRK	break interrupt flag,TIMERx(x=0,7)
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 3-641. 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..13)
TIMER_FLAG_CH0	channel 0 flag,TIMERx(x=0..4,7..13)
TIMER_FLAG_CH1	channel 1 flag,TIMERx(x=0..4,7,8,11)
TIMER_FLAG_CH2	channel 2 flag,TIMERx(x=0..4,7)
TIMER_FLAG_CH3	channel 3 flag,TIMERx(x=0..4,7)
TIMER_FLAG_CMT	channel commutation flag,TIMERx(x=0,7)
TIMER_FLAG_TRG	trigger flag,TIMERx(x=0,7,8,11)
TIMER_FLAG_BRK	break flag,TIMERx(x=0,7)
TIMER_FLAG_CH0O	channel 0 overcapture flag,TIMERx(x=0..4,7..11)
TIMER_FLAG_CH1O	channel 1 overcapture flag,TIMERx(x=0..4,7,8,11)
TIMER_FLAG_CH2O	channel 2 overcapture flag,TIMERx(x=0..4,7)
TIMER_FLAG_CH3O	channel 3 overcapture flag,TIMERx(x=0..4,7)
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 3-642. 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..13)
TIMER_FLAG_CH0	channel 0 flag,TIMERx(x=0..4,7..13)
TIMER_FLAG_CH1	channel 1 flag,TIMERx(x=0..4,7,8,11)
TIMER_FLAG_CH2	channel 2 flag,TIMERx(x=0..4,7)
TIMER_FLAG_CH3	channel 3 flag,TIMERx(x=0..4,7)
TIMER_FLAG_CMT	channel commutation flag,TIMERx(x=0,7)
TIMER_FLAG_TRG	trigger flag,TIMERx(x=0,7,8,11)
TIMER_FLAG_BRK	break flag,TIMERx(x=0,7)
TIMER_FLAG_CH0O	channel 0 overcapture flag,TIMERx(x=0..4,7..11)
TIMER_FLAG_CH1O	channel 1 overcapture flag,TIMERx(x=0..4,7,8,11)
TIMER_FLAG_CH2O	channel 2 overcapture flag,TIMERx(x=0..4,7)
TIMER_FLAG_CH3O	channel 3 overcapture flag,TIMERx(x=0..4,7)
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* clear TIMER0 update flags */

timer_flag_clear(TIMER0, TIMER_FLAG_UP);

3.23. USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter ***0***, the USART firmware functions are introduced in chapter ***3.23.1*** Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-643. USART Registers

Registers	Descriptions
USART_STAT	Status register
USART_DATA	Data register
USART_BAUD	Baud rate register
USART_CTL0	Control register 0
USART_CTL1	Control register 1
USART_CTL2	Control register 2
USART_GP	Guard time and prescaler register

3.23.1. Descriptions of Peripheral functions

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

Table 3-644. USART firmware function

Function name	Function description
usart_deinit	reset USART/UART
usart_baudrate_set	configure USART baud rate value
usart_parity_config	configure USART parity

Function name	Function description
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_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_mute_mode_enable	receiver in mute mode
usart_mute_mode_disable	receiver in active 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_send_break	send break frame
usart_halfduplex_enable	enable half duplex mode
usart_halfduplex_disable	disable half duplex mode
usart_synchronous_clock_enable	enable CK pin in synchronous mode
usart_synchronous_clock_disable	disable CK pin in synchronous mode
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

Function name	Function description
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
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_dma_receive_config	configure USART DMA reception
usart_dma_transmit_config	configure USART DMA transmission
usart_flag_get	get flag in STAT 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 in STAT register

usart_deinit

The description of usart_deinit is shown as below:

Table 3-645. Function usart_deinit

Function name	usart_deinit
Function prototype	void usart_deinit(uint32_t usart_periph);
Function descriptions	reset USART/UART
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4

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 3-646. 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/UART baud rate value
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-647. 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/UART parity
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-648. 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/UART word length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-649. 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/UART stop bit length
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
stblen	USART stop bit configure
USART_STB_1BIT	1 bit
USART_STB_0_5BIT	0.5 bit, not available for UARTx(x=3,4)
USART_STB_2BIT	2 bits
USART_STB_1_5BIT	1.5 bits, not available for UARTx(x=3,4)
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 3-650. Function usart_enable

Function name	usart_enable
Function prototype	void usart_enable(uint32_t usart_periph);
Function descriptions	enable USART/UART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable USART0 */

usart_enable(USART0);

usart_disable

The description of usart_disable is shown as below:

Table 3-651. Function usart_disable

Function name	usart_disable
Function prototype	void usart_disable(uint32_t usart_periph);
Function descriptions	disable USART/UART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-652. 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/UART transmitter
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-653. 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/UART receiver

Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
rxconfig	enable or disable USART receiver
USART_RECEIVE_ENABLE	enable USART reception
USART_RECEIVE_DISABLE	disable USART reception
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* Configure USART0 receive */

usart_receive_config(USART0, USART_RECEIVE_ENABLE);

usart_data_transmit

The description of usart_data_transmit is shown as below:

Table 3-654. Function usart_data_transmit

Function name	usart_data_transmit
Function prototype	void usart_data_transmit(uint32_t usart_periph, uint16_t data);
Function descriptions	USART/UART transmit data function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral

USARTx	x=0,1,2
UARTx	x=3,4
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_receiveis shown as below:

Table 3-655. Function usart_data_receive

Function name	usart_data_receive
Function prototype	void usart_receive_config(uint32_t usart_periph, uint32_t rxconfig);
Function descriptions	USART/UART receive data function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-656. 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/UART in wake up by address match mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
addr	address of USART/UART
0-0xFF	address of USART/UART
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address of the USART0 */

usart_address_config(USART0, 0x00);
```

usart_mute_mode_enable

The description of usart_mute_mode_enable is shown as below:

Table 3-657. Function usart_mute_mode_enable

Function name	usart_mute_mode_enable
Function prototype	void usart_mute_mode_enable(uint32_t usart_periph);
Function descriptions	enable mute mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-658. Function usart_mute_mode_disable

Function name	usart_mute_mode_disable
Function prototype	void usart_mute_mode_disable(uint32_t usart_periph);
Function descriptions	disable mute mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2

UARTx	x=3,4
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 3-659. Function usart_mute_mode_wakeup_config

Function name	usart_mute_mode_wakeup_config
Function prototype	void usart_mute_mode_wakeup_config(uint32_t usart_periph, uint32_t wmethod);
Function descriptions	configure wakeup method in mute mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-660. Function usart_lin_mode_enable

Function name	usart_lin_mode_enable
Function prototype	void usart_lin_mode_enable(uint32_t usart_periph);
Function descriptions	enable LIN mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-661. Function usart_lin_mode_disable

Function name	usart_lin_mode_disable
Function prototype	void usart_lin_mode_disable(uint32_t usart_periph);

Function descriptions	disable LIN mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-662. 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	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	

lblen	lin break frame length
USART_LBLEN_10B	10 bits
USART_LBLEN_11B	11 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure LIN break frame length */

usart_lin_break_dection_length_config(USART0, USART_LBLEN_10B);

usart_send_break

The description of usart_send_break is shown as below:

Table 3-663. Function usart_send_break

Function name	usart_send_break
Function prototype	void usart_send_break(uint32_t usart_periph);
Function descriptions	send break frame
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 send break frame */

usart_send_break(USART0);

usart_halfduplex_enable

The description of usart_halfduplex_enable is shown as below:

Table 3-664. Function usart_halfduplex_enable

Function name	usart_halfduplex_enable
Function prototype	void usart_halfduplex_enable(uint32_t usart_periph);
Function descriptions	enable half duplex mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-665. Function usart_halfduplex_disable

Function name	usart_halfduplex_disable
Function prototype	void usart_halfduplex_disable(uint32_t usart_periph);
Function descriptions	disable half duplex mode
Precondition	-
The called functions	-

Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 half duplex mode*/
usart_halfduplex_disable(USART0);
```

usart_synchronous_clock_enable

The description of usart_synchronous_clock_enable is shown as below:

Table 3-666. Function usart_synchronous_clock_enable

Function name	usart_synchronous_clock_enable
Function prototype	void usart_synchronous_clock_enable(uint32_t usart_periph);
Function descriptions	enable CK pin in synchronous mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 CK pin in synchronous mode */
```

usart_synchronous_clock_enable(USART0);

usart_synchronous_clock_disable

The description of usart_synchronous_clock_disable is shown as below:

Table 3-667. Function usart_synchronous_clock_disable

Function name	usart_synchronous_clock_disable
Function prototype	void usart_synchronous_clock_disable(uint32_t usart_periph);
Function descriptions	disable CK pin in synchronous mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 CK pin in synchronous mode */

usart_synchronous_clock_disable(USART0);

usart_synchronous_clock_config

The description of usart_synchronous_clock_config is shown as below:

Table 3-668. Function usart_synchronous_clock_config

Function name	usart_synchronous_clock_config
Function prototype	void usart_synchronous_clock_config(uint32_t usart_periph, uint32_t clen, uint32_t cph, uint32_t cpl);
Function descriptions	configure USART synchronous mode parameters
Precondition	-
The called functions	-

Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Input parameter{in}	
clen	CK length
USART_CLEN_NONE	there are 7 CK pulses for an 8 bit frame and 8 CK pulses for a 9 bit frame
USART_CLEN_EN	there are 8 CK pulses for an 8 bit frame and 9 CK pulses for a 9 bit frame
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 3-669. Function usart_guard_time_config

Function name	usart_guard_time_config
Function prototype	void usart_guard_time_config(uint32_t usart_periph,uint32_t gaut);

Function descriptions	configure guard time value in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Input parameter{in}	
gaut	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 3-670. Function usart_smartcard_mode_enable

Function name	usart_smartcard_mode_enable
Function prototype	void usart_smartcard_mode_enable(uint32_t usart_periph);
Function descriptions	enable smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
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 3-671. Function usart_smartcard_mode_disable

Function name	usart_smartcard_mode_disable
Function prototype	void usart_smartcard_mode_disable(uint32_t usart_periph);
Function descriptions	disable smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
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 3-672. Function usart_smartcard_mode_nack_enable

Function name	usart_smartcard_mode_nack_enable

Function prototype	void usart_smartcard_mode_nack_enable(uint32_t usart_periph);
Function descriptions	enable NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
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 3-673. Function usart_smartcard_mode_nack_disable

Function name	usart_smartcard_mode_nack_disable
Function prototype	void usart_smartcard_mode_nack_disable(uint32_t usart_periph);
Function descriptions	disable NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* disable USART0 NACK in smartcard mode */

usart_smartcard_mode_nack_disable(USART0);

usart_irda_mode_enable

The description of usart_irda_mode_enable is shown as below:

Table 3-674. Function usart_irda_mode_enable

Function name	usart_irda_mode_enable
Function prototype	void usart_irda_mode_enable(uint32_t usart_periph);
Function descriptions	enable IrDA mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-675. Function usart_irda_mode_disable

Function name	usart_irda_mode_disable
Function prototype	void usart_irda_mode_disable(uint32_t usart_periph);

Function descriptions	disable IrDA mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-676. 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
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-677. Function usart_irda_lowpower_config

Function name	usart_irda_lowpower_config
Function prototype	void usart_irda_lowpower_config(uint32_t usart_periph, uint32_t irlp);
Function descriptions	configure IrDA low-power
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
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 3-678. Function usart_hardware_flow_rts_config

Function name	usart_hardware_flow_rts_config
Function prototype	void usart_hardware_flow_rts_config(uint32_t usart_periph, uint32_t rtsconfig);
Function descriptions	configure hardware flow control RTS
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
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_cts_config(USART0, USART_RTS_ENABLE);

usart_hardware_flow_cts_config

The description of usart_hardware_flow_cts_config is shown as below:

Table 3-679. 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 RTS
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
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_dma_receive_config

The description of usart_dma_receive_config is shown as below:

Table 3-680. Function usart_dma_receive_config

Function name	usart_dma_receive_config
Function prototype	void usart_dma_receive_config(uint32_t usart_periph, uint32_t dmacmd);

Function descriptions	configure USART/UART DMA reception
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3
Input parameter{in}	
dmacmd	enable or disable DMA for reception
USART_DENR_ENABLE	DMA enable for reception
USART_DENR_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 3-681. Function usart_dma_transmit_config

Function name	usart_dma_transmit_config
Function prototype	void usart_dma_transmit_config(uint32_t usart_periph, uint32_t dmacmd);
Function descriptions	configure USART/UART DMA transmission
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3
Input parameter{in}	
dmacmd	enable or disable DMA for transmission
USART_DENT_ENABLE	DMA enable for transmission
USART_DENT_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_flag_get

The description of usart_flag_get is shown as below:

Table 3-682. 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 register
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	

flag	USART flags, refer to usart_flag_enum
USART_FLAG_CTSF	CTS change flag
USART_FLAG_LBDF	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_IDLEF	IDLE frame 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
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 3-683. Function usart_flag_clear

Function name	usart_flag_clear
Function prototype	void usart_flag_clear(uint32_t usart_periph, usart_flag_enum flag);
Function descriptions	clear flag in STAT register
Precondition	-
The called functions	-

Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
flag	USART flags, refer to usart_flag_enum
USART_FLAG_CTSF	CTS change flag
USART_FLAG_LBDF	LIN break detected flag
USART_FLAG_TC	transmission complete
USART_FLAG_RBNE	read data buffer not empty
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 3-684. Function usart_interrupt_enable

Function name	usart_interrupt_enable
Function prototype	void usart_interrupt_enable(uint32_t usart_periph, uint32_t int_flag);
Function descriptions	enable USART/UART interrupt
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2

UARTx	x=3,4
Input parameter{in}	
int_flag	USART interrupt flag
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_ERR	error interrupt
USART_INT_CTS	CTS 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 3-685. Function usart_interrupt_disable

Function name	usart_interrupt_disable
Function prototype	void usart_interrupt_disable(uint32_t usart_periph, uint32_t int_flag);
Function descriptions	disable USART/UART interrupt
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral

USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
int_flag	USART interrupt flag
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_ERR	error interrupt
USART_INT_CTS	CTS 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 3-686. Function usart_interrupt_flag_get

Function name	usart_interrupt_flag_get
Function prototype	FlagStatus usart_interrupt_flag_get(uint32_t usart_periph, uint32_t int_flag);
Function descriptions	get USART/UART interrupt and flag status
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
int_flag	USART interrupt 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_CTS	CTS interrupt and flag
USART_INT_FLAG_ERR_ORERR	error interrupt and overrun error
USART_INT_FLAG_ERR_NERR	error interrupt and noise error flag
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 3-687. Function usart_interrupt_flag_clear

Function name	usart_interrupt_flag_clear
Function prototype	void usart_interrupt_flag_clear(uint32_t usart_periph, uint32_t flag);
Function descriptions	clear USART/UART interrupt flag in STAT register
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
flag	USART interrupt flag
USART_INT_FLAG_CTS	CTS change flag
USART_INT_FLAG_LBD	LIN break detected flag
USART_INT_FLAG_TC	transmission complete
USART_INT_FLAG_RBNE	read data buffer not empty
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear the USART0 interrupt enable bit status */

usart_interrupt_flag_clear(USART0, USART_INT_FLAG_RBNE);

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 WWDGT 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 3-688. 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 3-689. WWDGT firmware function

Function name	Function description
wwdgt_deinit	reset the window watchdog timer configuration
wwdgt_enable	start the window watchdog timer counter
wwdgt_counter_update	configure the window watchdog timer counter value
wwdgt_config	configure counter value, window value, and prescaler divider value
wwdgt_interrupt_enable	enable early wakeup interrupt of WWDGT
wwdgt_flag_get	check early wakeup interrupt state of WWDGT
wwdgt_flag_clear	clear early wakeup interrupt state of WWDGT

wwdgt_deinit

The description of wwdgt_deinit is shown as below:

Table 3-690. Function wwdgt_deinit

Function name	wwdgt_deinit

Function prototype	void wwdgt_deinit(void);
Function descriptions	reset the window watchdog timer configuration
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the window watchdog timer configuration */
wwdgt_deinit( );
```

wwdgt_enable

The description of wwdgt_enable is shown as below:

Table 3-691. Function wwdgt_enable

Function name	wwdgt_enable
Function prototype	void wwdgt_enable(void);
Function descriptions	start the window watchdog timer counter
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* start the window watchdog timer counter */

wwdgt_enable( );

wwdgt_counter_update

The description of wwdgt_counter_update is shown as below:

Table 3-692. Function wwdgt_counter_update

Function name	wwdgt_counter_update
Function prototype	void wwdgt_counter_update(uint16_t counter_value);
Function descriptions	configure the window watchdog timer counter value
Precondition	-
The called functions	-
Input parameter{in}	
counter_value	0x00 - 0x7F
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 3-693. Function wwdgt_config

Function name	wwdgt_config
Function prototype	void wwdgt_config(uint16_t counter, uint16_t window, uint32_t prescaler);
Function descriptions	configure counter value, window value, and prescaler divider value
Precondition	-
The called functions	-
Input parameter{in}	

counter	0x00 - 0x7F
Input parameter{in}	
window	0x00 - 0x7F
Input parameter{in}	
prescaler	wwdgt prescaler value
WWDGT_CFG_PSC_DIV1	the time base of window watchdog counter = (PCLK1/4096)/1
WWDGT_CFG_PSC_DIV2	the time base of window watchdog counter = (PCLK1/4096)/2
WWDGT_CFG_PSC_DIV4	the time base of window watchdog counter = (PCLK1/4096)/4
WWDGT_CFG_PSC_DIV8	the time base of window watchdog 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 3-694. Function wwdgt_interrupt_enable

Function name	wwdgt_interrupt_enable
Function prototype	void wwdgt_interrupt_enable(void);
Function descriptions	enable early wakeup interrupt of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

```
/* enable early wakeup interrupt of WWDGT */
wwdgt_interrupt_enable( );
```

wwdgt_flag_get

The description of wwdgt_flag_get is shown as below:

Table 3-695. Function wwdgt_flag_get

Function name	wwdgt_flag_get
Function prototype	FlagStatus wwdgt_flag_get(void);
Function descriptions	check early wakeup interrupt state of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* test if the counter value update has reached the 0x40 */
FlagStatus status;
status = wwdgt_flag_get ( );
if(status == RESET)
{
```

```
...
}else
{
...
}
```

wwdgt_flag_clear

The description of wwdgt_flag_clear is shown as below:

Table 3-696. Function wwdgt_flag_clear

Function name	wwdgt_flag_clear
Function prototype	void wwdgt_flag_clear(void);
Function descriptions	clear early wakeup interrupt state of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear early wakeup interrupt state of WWDGT */
wwdgt_flag_clear( );
```

3.25. USBD

Firmware function description of USBD refers to document ***GD32F10x-Firmware-Library-USB User Guide_V1.0***.

3.26. USBFS

Firmware function description of USBFS refers to document ***GD32F10x-Firmware-Library-***

USB User Guide_V1.0.

4. Revision history

Table 4-1. Revision history

Revison No.	Description	Date
1.0	Initial Release	Mar.26, 2018

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.