UM324xF API 参考手册

版本：V1.1

广芯微电子（广州）股份有限公司

http://www.unicmicro.com/

条款协议

本文档的所有部分，其著作产权归广芯微电子（广州）股份有限公司（以下简称广芯微电子）所有，未经广芯微电子授权许可，任何个人及组织不得复制、转载、仿制本文档的全部或部分组件。本文档没有任何形式的担保、立场表达或其他暗示，若有任何因本文档或其中提及的产品所有资讯所引起的直接或间接损失，广芯微电子及所属员工恕不为其担保任何责任。除此以外，本文档所提到的产品规格及资讯仅供参考，内容亦会随时更新，恕不另行通知。

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。用户如在设备设计中应用本文档中的电路、软件和相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，广芯微电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，广芯微电子已尽量做到合理注意，但是，广芯微电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，广芯微电子不承担任何责任。
3. 对于因使用本文档中的广芯微电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，广芯微电子不承担任何责任。本文档所记载的内容不应视为对广芯微电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 使用本文档中记载的广芯微电子产品时，应在广芯微电子指定的范围内，特别是在最大额定值、电源工作电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用广芯微电子产品而产生的故障或损失，广芯微电子不承担任何责任。
5. 虽然广芯微电子一直致力于提高广芯微电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，广芯微电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当广芯微电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。

版本修订

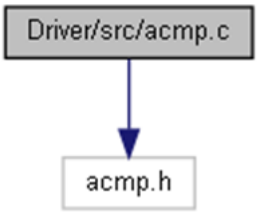
版本	日期	描述
V1.0	2022.11.09	初始版
V1.1	2022.12.08	更新AES接口及USB接口章节

目录

1 应用程序接口（API）

1.1 ACMP接口

ACMP driver source file
#include "acmp.h"
acmp.c 的引用(Include)关系图:



1.1.1 函数

- void **ACMP0_IRQHandler** (void)
  ACMP0 interrupt handling
- void **ACMP1_IRQHandler** (void)
  ACMP1 interrupt handling
- void **ACMP2_IRQHandler** (void)
  ACMP2 interrupt handling
- void **acmp_irq_init** (ACMPX_T ACMPx, IRQ_TYPE_T irq_edge, FUNC_E ir_enable)
  Initializes for ACMP interrupt.
- void **acmp_model_init** (ACMPX_T ACMPx, FUNC_E acmp_en)
  Select whether to turn on the module function.
- void **acmp_input_signal** (ACMPX_T ACMPx, INPUT_SIGNAL_T signal)
  Positive input signal selection setting.
- void **acmp_init** (ACMPX_T ACMPx, INPUT_SIGNAL_T signal, IRQ_TYPE_T irq_edge, FUNC_E ir_enable, FUNC_E acmp_en)
  Initializes for ACMP work.
- BOOL **acmp_get_status** (ACMPX_T ACMPx)
  get ACMP status

1.1.2 函数说明

1.1.2.1 void ACMP0_IRQHandler (void)

ACMP0 interrupt handling

参数:

none	

返回:

none

1.1.2.2 void ACMP1_IRQHandler (void)

ACMP1 interrupt handling

参数:

none	

返回:

none

1.1.2.3 void ACMP2_IRQHandler (void)

ACMP2 interrupt handling

参数:

none	

返回:

none

1.1.2.4 BOOL acmp_get_status (ACMPX_T *ACMPx*)

get ACMP status

参数:

ACMPx	Where x can be 0,1, 2, to select the ACMP peripheral.

返回:

the ouput status of the ACMP

1.1.2.5 void acmp_init (ACMPX_T *ACMPx*, INPUT_SIGNAL_T *signal*, IRQ_TYPE_T *irq_edge*, FUNC_E *ir_enable*, FUNC_E *acmp_en*)

Initializes for ACMP work.

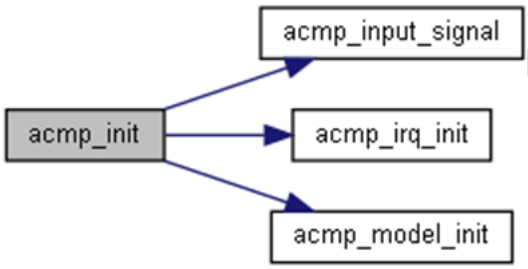
参数:

ACMPx	Where x can be 0,1, 2, to select the ACMP peripheral.
signal	This parameter is the input signal for setting ACMP.
irq_edge	This parameter is to set acmp's interrupt generation edge selection.
ir_enable	This parameter is to set whether the interrupt of ACMP is enabled or not.
acmp_en	This parameter is to select whether to turn on ACMP to work normally.

返回:

none

函数调用图:



1.1.2.6 void acmp_input_signal (ACMPX_T *ACMPx*, INPUT_SIGNAL_T *signal*)

Positive input signal selection setting.

参数:

ACMPx	Where x can be 0,1, 2, to select the ACMP peripheral.
signal	This parameter is the input signal for setting acmp. This parameter can be one of the following values: ● ACMP_CIN External pin input ● ACMP_VDDH Internal vddh input ● ACMP_VREF Vref input ● ACMP_VBC 0.6V internal signal input ● ACMP_DACOUT dac input

返回:

none

函数的调用关系图:

1.1.2.7 void acmp_irq_init (ACMPX_T *ACMPx*, IRQ_TYPE_T *irq_edge*, FUNC_E *ir_enable*)

Initializes for ACMP interrupt.

参数:

ACMPx	Where x can be 0,1, 2, to select the ACMP peripheral.
irq_edge	This parameter is to set acmp's interrupt generation edge selection.
ir_enable	This parameter is to set whether the interrupt of acmp is enabled or not.

返回:

none

函数的调用关系图:

1.1.2.8 void acmp_model_init (ACMPX_T *ACMPx*, FUNC_E *acmp_en*)

Select whether to turn on the module function.

参数:

ACMPx	Where x can be 0,1, 2, to select the ACMP peripheral.
acmp_en	This parameter is to select whether to turn on ACMP to work normally.

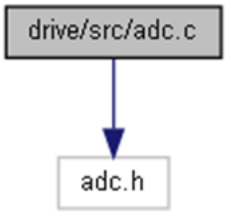
返回:

none

函数的调用关系图:

1.2 ADC接口

ADC driver source file
#include "adc.h"
adc.c 的引用(Include)关系图:



1.2.1 函数

- void **ADC0_IRQHandler** (void)
  ADC0 interrupt handling
- void **ADC1_IRQHandler** (void)
  ADC1 interrupt handling
- void **adc_clk_init** (ADC_T *ADCx, BOOL state)
  ADC clock initial
- void **adc_init** (ADC_T *ADCx, uint8_t vref_sel, adc_sample_rate_t speed_div, uint8_t mode)
  ADC initial function
- void **adc_position_number_config** (ADC_T *ADCx, uint8_t type, uint8_t num)
  ADC position number config
- void **adc_channel_average_config** (ADC_T *ADCx, uint8_t channel, uint8_t num)
  ADC channel average config
- void **adc_fifo_config** (ADC_T *ADCx, BOOL fifo_state, adc_fifo_t watermark)
  ADC fifo config
- void **adc_irq_config** (ADC_T *ADCx, uint8_t irq_type, BOOL newstate, void(*adc_fun)())
  ADC irq config
- void **adc_gpio_config** (uint32_t channel)
  ADC gpio config
- void **adc_start_conversion** (ADC_T *ADCx, uint8_t mode)
  ADC start conversion
- void **adc_stop_conversion** (ADC_T *ADCx)
  ADC stop conversion
- uint32_t **adc_get_value** (ADC_T *ADCx, uint8_t channel)
  ADC get value
- uint32_t **adc_get_fifo** (ADC_T *ADCx)
  ADC get fifo

- void **adc_controler_en** (ADC_T *ADCx, BOOL state)
  ADC controler enable
- void **adc_regular_sequence_position_channel_config** (ADC_T *ADCx, uint8_t position, uint8_t channel)
  ADC regular sequence position channel config
- void **adc_injection_sequence_position_channel_config** (ADC_T *ADCx, uint8_t position, uint8_t channel)
  ADC injection sequence position channel config
- void **adc_watchdog_mode_init** (ADC_T *ADCx, uint8_t channel, float watchdog_max, float watchdog_min, float adc_vref)
  ADC watchdog mode init
- void **adc_differential_mode_init** (ADC_T *ADCx, uint8_t channel)
  ADC differential mode initial
- void **adc_synergy_mode_init** (ADC_T *ADC0, ADC_T *ADC1, adc_synergy_mode_t synergy_mode)
  ADC synergy mode initial
- uint32_t **adc_read_intstat** (ADC_T *ADCx)
  ADC read intstat
- uint32_t **adc_read_stat** (ADC_T *ADCx)
  ADC read stat
- uint32_t **adc_read_dualdat** (ADC_T *ADCx)
  ADC read dualdat
- void **adc_clear_intstat** (ADC_T *ADCx, uint8_t irq_type)
  ADC clear intstat
- void **adc_opa_enable** (ADC_T *ADCx, uint8_t state)
  ADC OPA enable
- void **adc_dma_enable** (ADC_T *ADCx, uint8_t mode)
  ADC DMA mode

1.2.2 函数说明

1.2.2.1 void ADC0_IRQHandler (void)

ADC0 interrupt handling

参数:

none	

返回:

none

1.2.2.2 void ADC1_IRQHandler (void)

ADC1 interrupt handling

参数:

none	

返回:

none

1.2.2.3 void adc_channel_average_config (ADC_T * *ADCx*, uint8_t *channel*, uint8_t *num*)

ADC channel average config

参数:

**ADCx*	pointer to ADC_T structure
channel	set adc conversion channel (ADC_CHANNEL0 - ADC_CHANNEL15)
num	set adc conversion average number

返回:

none

1.2.2.4 void adc_clear_intstat (ADC_T * *ADCx*, uint8_t *irq_type*)

ADC clear intstat

参数:

**ADCx*	pointer to ADC_T structure
**irq_type*	adc interrupt type

返回:

none

1.2.2.5 void adc_clk_init (ADC_T * *ADCx*, BOOL *state*)

ADC clock initial

参数:

**ADCx*	pointer to ADC_T structure
newstate	Clock and reset status. This parameter can be one of the following values: ● ENABLE: enable adc clock and set it into work mode. ● DISABLE：disable adc clock and set adc into reset mode.

返回:

none

1.2.2.6 void adc_controler_en (ADC_T * *ADCx*, BOOL *state*)

ADC controler enable

参数:

**ADCx*	pointer to ADC_T structure
state	enable or disable adc controler

返回:

none

1.2.2.7 void adc_differential_mode_init (ADC_T * *ADCx*, uint8_t *channel*)

ADC differential mode initial

参数:

**ADCx*	pointer to ADC_T structure
channel	differential channel number

返回:

none

1.2.2.8 void adc_dma_enable (ADC_T * *ADCx*, uint8_t *mode*)

ADC DMA mode

参数:

**ADCx*	pointer to ADC_T structure
mode	set dma mode

返回:

none

1.2.2.9 void adc_fifo_config (ADC_T * *ADCx*, BOOL *fifo_state*, adc_fifo_t *watermark*)

ADC FIFO config

参数:

**ADCx*	pointer to ADC_T structure
fifo_state	FIFO status This parameter can be one of the following values: ● ENABLE: enable FIFO ● DISABLE：disable FIFO
watermark	fifo number of available to trigger DMA or fifo_val set bit

返回:

none

1.2.2.10 uint32_t adc_get_fifo (ADC_T * *ADCx*)

ADC get FIFO

参数:

**ADCx*	pointer to ADC_T structure

返回:

temp receiver FIFO data

1.2.2.11 uint32_t adc_get_value (ADC_T * *ADCx*, uint8_t *channel*)

ADC get value

参数:

**ADCx*	pointer to ADC_T structure
channel	chanenl number

返回:

temp channel conversion data

1.2.2.12 void adc_gpio_config (uint32_t *channel*)

ADC GPIO config

参数:

channel	ADC conversion channel

返回:

none

1.2.2.13 void adc_init (ADC_T * *ADCx*, uint8_t *vref_sel*, adc_sample_rate_t *speed_div*, uint8_t *mode*)

ADC initial function

参数:

**ADCx*	pointer to ADC_T structure
vref_sel	set adc reference voltage This parameter can be one of the following values: ● ADC_REF_VDDH : select VDDH as reference voltage ● ADC_REF_VREFIO : select VREF_IN or VREF_OUT as reference voltage
speed_div	set adc sampling rate for example：

	sampling rate = 168M/16/(speed_div + 1)
mode	set adc work mode This parameter can be one of the following values: ● ADC_REGULAR_NONE_MODE : regular scan none mode ● ADC_REGULAR_MODE_SINGLE : regular scan single mode ● ADC_REGULAR_MODE_CONTINUE : regular scan continue mode ● ADC_REGULAR_MODE_OFFON : regular scan off on mode

返回:

none

1.2.2.14 void adc_injection_sequence_position_channel_config (ADC_T * *ADCx*, uint8_t *position*, uint8_t *channel*)

ADC injection sequence position channel config

参数:

**ADCx*	pointer to ADC_T structure
position	number of positions
channel	channel number

返回:

none

1.2.2.15 void adc_irq_config (ADC_T * *ADCx*, uint8_t *irq_type*, BOOL *newstate*, void(*)() *adc_fun*)

ADC IRQ config

参数:

**ADCx*	pointer to ADC_T structure
irq_type	adc interrupt type
newstate	enable or disable adc interrupt
**adc_fun()*	adc interrupt callback function

返回:

none

1.2.2.16 void adc_opa_enable (ADC_T * *ADCx*, uint8_t *state*)

ADC OPA enable

参数:

**ADCx*	pointer to ADC_T structure
state	enable or disable adc + opa

返回:

none

1.2.2.17 void adc_position_number_config (ADC_T * *ADCx*, uint8_t *type*, uint8_t *num*)

ADC position number config

参数:

**ADCx*	pointer to ADC_T structure
type	set adc channel type This parameter can be one of the following values: ● ADC_REGULAR_SHORT_LENG : set regular short sequence conversion length ● ADC_REGULAR_LENG : set regular sequence conversion length ● ADC_INTECTION_LENG : set injection sequence conversion length
num	set adc number of conversion positions

返回:

none

1.2.2.18 uint32_t adc_read_dualdat (ADC_T * *ADCx*)

ADC read dualdat

参数:

**ADCx*	pointer to ADC_T structure

返回:

temp adc dualdat

1.2.2.19 uint32_t adc_read_intstat (ADC_T * *ADCx*)

ADC read intstat

参数:

**ADCx*	pointer to ADC_T structure

返回:

temp adc intstat

1.2.2.20 uint32_t adc_read_stat (ADC_T * *ADCx*)

ADC read stat

参数:

**ADCx*	pointer to ADC_T structure

返回:

temp ADC stat

1.2.2.21 void adc_regular_sequence_position_channel_config (ADC_T * *ADCx*, uint8_t *position*, uint8_t *channel*)

ADC regular sequence position channel config

参数:

**ADCx*	pointer to ADC_T structure
position	number of positions
channel	channel number

返回:

none

1.2.2.22 void adc_start_conversion (ADC_T * *ADCx*, uint8_t *mode*)

ADC start conversion

参数:

**ADCx*	pointer to ADC_T structure
mode	regular conversion or intection conversion

返回:

none

1.2.2.23 void adc_stop_conversion (ADC_T * *ADCx*)

ADC stop conversion

参数:

**ADCx*	pointer to ADC_T structure

返回:

none

1.2.2.24 void adc_synergy_mode_init (ADC_T * *ADC0*, ADC_T * *ADC1*, adc_synergy_mode_t *synergy_mode*)

ADC synergy mode initial

参数:

**ADC0*	pointer to ADC_T structure
**ADC1*	pointer to ADC_T structure
synergy_mode	regular or injection synergy mode

返回:

none

1.2.2.25 void adc_watchdog_mode_init (ADC_T * *ADCx*, uint8_t *channel*, float *watchdog_max*, float *watchdog_min*, float *adc_vref*)

ADC watchdog mode init

参数:

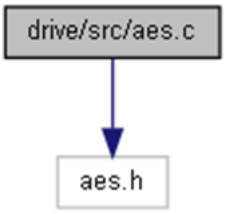
**ADCx*	pointer to ADC_T structure
channel	channel number
watchdog_max	upper limit of voltage
watchdog_min	lower limit of voltage
adc_vref	reference voltage

返回:

none

1.3 AES接口

AES driver source file
#include "aes.h"
aes.c 的引用(Include)关系图:



1.3.1 函数

- void **AES_IRQHandler** (void)
  AES interrupt handling.
- void **aes_init** (AES_T *AES, BOOL newstate)
  AES initial
- void **aes_irq_init** (AES_T *AES, uint8_t irq_enable, void(*pfunc_tc)())
  AES irq init
- uint32_t **write_text_in** (AES_T *AES, char *pText, uint32_t len)
  write text in
- void **read_text_out** (AES_T *AES, char *pText, uint32_t len)
  read text out
- void **write_key** (AES_T *AES, char *pAeskey)
  write key
- void **write_cbc_key** (AES_T *AES, char *pAeskey)
  write cbc key
- void **write_iv_key** (AES_T *AES, char *pAeskey)
  write iv key
- void **write_ctr_key** (AES_T *AES, char *pAeskey)
  write ctr key
- void **write_mac_key** (AES_T *AES, char *pAeskey)
  write mac key
- void **aes_para_config** (AES_T *AES, uint32_t alg_mode, uint32_t data_size, uint8_t key_size, uint32_t dir, uint32_t key_src)
  aes_para_config
- void **wait_aes_done** (void)
  wait aes done

1.3.2 函数说明

1.3.2.1 void aes_init (AES_T * *AES*, BOOL *newstate*)

AES initial

参数:

**AES*	pointer to AES_T structure
newstate	AES_ENABLE / AES_DISABLE

返回:

none

1.3.2.2 void aes_irq_init (AES_T * *AES*, uint8_t *irq_enable*, void(*)() *pfunc_tc*)

AES IRQ init

参数:

**AES*	pointer to AES_T structure
irq_enable	This parameter can be ENABLE or DISABLE.
*(*pfunc_tc)()*	pointer to pfunc_tc

返回:

none

1.3.2.3 void AES_IRQHandler (void)

AES interrupt handling.

参数:

none	

返回:

none

1.3.2.4 void aes_para_config (AES_T * *AES*, uint32_t *alg_mode*, uint32_t *data_size*, uint8_t *key_size*, uint32_t *dir*, uint32_t *key_src*)

aes_para_config

参数:

**AES*	pointer to AES_T structure
alg_mode	mode
data_size	data length

key_size	key length
dir	direction
key_src	key source

返回:

none

1.3.2.5 void read_text_out (AES_T * *AES*, char * *pText*, uint32_t *len*)

read text out

参数:

**AES*	pointer to AES_T structure
**pText*	pointer to output buffer address
len	output data length

返回:

none

1.3.2.6 void wait_aes_done (void)

wait AES done

参数:

none	

返回:

none

1.3.2.7 void write_cbc_key (AES_T * *AES*, char * *pAeskey*)

write cbc key

参数:

**AES*	pointer to AES_T structure
**pAeskey*	pointer to aes_cbc_key adress

返回:

none

1.3.2.8 void write_ctr_key (AES_T * *AES*, char * *pAeskey*)

write ctr key

参数:

**AES*	pointer to AES_T structure

**pAeskey*	pointer to ctr_key adress

返回:

none

1.3.2.9 void write_iv_key (AES_T * *AES*, char * *pAeskey*)

write iv key

参数:

**aes*	pointer to AES_T structure
**pAeskey*	pointer to iv_key adress

返回:

none

1.3.2.10 void write_key (AES_T * *AES*, char * *pAeskey*)

write key

参数:

**AES*	pointer to AES_T structure
**pAeskey*	pointer to aeskey adress

返回:

none

1.3.2.11 void write_mac_key (AES_T * *AES*, char * *pAeskey*)

write mac key

参数:

**AES*	pointer to AES_T structure
**pAeskey*	pointer to mac_key adress

返回:

none

1.3.2.12 uint32_t write_text_in (AES_T * *AES*, char * *pText*, uint32_t *len*)

write text in

参数:

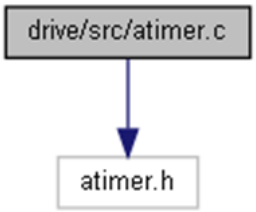
**AES*	pointer to AES_T structure
**pText*	pointer to pText data buffer
len	input data length

返回:

none

1.4 ATIMER接口

ATIMER driver source file.
#include "atimer.h"
atimer.c 的引用(Include)关系图:



1.4.1 函数

- void **TIM0_BRK_TIM8_IRQHandler** (void)
  TIM0 BREAK and TIM8 interrupt handling.
- void **TIM0_UP_TIM9_IRQHandler** (void)
  TIM0 UPDATE and TIM9 interrupt handling.
- void **TIM0_TRG_COM_TIM10_IRQHandler** (void)
  TIM0 TRIGE and COM and TIM10 interrupt handling.
- void **TIM0_CC_IRQHandler** (void)
  TIM0 CC interrupt handling.
- void **TIM7_BRK_TIM11_IRQHandler** (void)
  TIM7 BREAK and TIM11 interrupt handling.
- void **TIM7_UP_TIM12_IRQHandler** (void)
  TIM7 UPDATE and TIM12 interrupt handling.
- void **TIM7_TRG_COM_TIM13_IRQHandler** (void)
  TIM7 TRIGE and COM and TIM13 interrupt handling.
- void **TIM7_CC_IRQHandler** (void)
  TIM7 CC interrupt handling.
- uint8_t **atim_get_status** (ATIM_T *ATIMx, uint8_t status)
  ATIMER get status.
- void **atim_clr_status** (ATIM_T *ATIMx, uint8_t status)
  ATIMER clear update status.
- void **atim_software_event** (ATIM_T *ATIMx, uint8_t events)
  ATIMER software event.
- void **atim_clock_init** (ATIM_T *ATIMx, BOOL atim_enable_type)
  ATIMER clock initial.
- void **atim_active_source_clock_config** (ATIM_T *ATIMx, uint8_t active_source_clock)
  ATIMER active source clock config(apb_clk=HCLK=SYSPLL/2).

- void **atim_irq_init** (ATIM_T *ATIMx, uint8_t atim_irq_type, uint8_t atim_enable_type, void(*pfunc)())
  ATIMER IRQ initial.
- void **atim_dma_init** (ATIM_T *ATIMx, uint8_t atim_dma_type, uint8_t atim_enable_type)
  ATIMER DMA initial.
- void **atim_enable_config** (ATIM_T *ATIMx, uint8_t atim_enable_type)
  ATIMER enable config.
- void **atim_init** (ATIM_T *ATIMx, uint32_t arr, uint16_t psc, uint8_t counter_direction, uint8_t counter_alignment)
  ATIMER initial.
- void **atim_xorinput_config** (ATIM_T *ATIMx)
  ATIMER xor input config CH1、CH2、CH3 input.
- void **atim_slave_config** (ATIM_T *ATIMx, uint8_t slave_mode, uint8_t channel)
  ATIMER slave config.
- void **atim_encoder_config** (ATIM_T *ATIMx, uint8_t encode_mode)
  ATIMER encoder config.
- void **atim_capture_config** (ATIM_T *ATIMx, uint8_t input_mode, uint8_t channel)
  ATIMER capture config.
- void **atim_pwm_config** (ATIM_T *ATIMx, uint8_t output_mode, uint8_t output_behavior, uint8_t channel)
  ATIMER pwm config.
- void **atim_set_compare** (ATIM_T *ATIMx, uint8_t channel, uint32_t compare_value)
  ATIMER set compare.
- uint32_t **atim_get_capture** (ATIM_T *ATIMx, uint8_t channel)
  ATIMER get capture.
- void **atim_dma_config** (ATIM_T *ATIMx, uint8_t length, uint8_t base_addr)
  ATIMER DMA config.
- void **atim_repeat_count_config** (ATIM_T *ATIMx, uint8_t times)
  ATIMER repeat count config.
- uint32_t **atim_get_cnt** (ATIM_T *ATIMx)
  ATIMER get count value.
- void **atim_master_trgo_config** (ATIM_T *ATIMx, uint8_t trgo_type)
  ATIMER master trgo config.

1.4.2 函数说明

1.4.2.1 void atim_active_source_clock_config (ATIM_T * *ATIMx*, uint8_t *active_source_clock*)

ATIMER active source clock config(apb_clk=HCLK=SYSPLL/2)

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
active_source_clock	Atimer source clock. This parameter can be one of the following values: ● ATIM_ACTIVE_SCLOCK_SYSPLL: SYSPLL clock. ● ATIM_ACTIVE_SCLOCK_APB: APB clock.

返回:

none

1.4.2.2 void atim_capture_config (ATIM_T * *ATIMx*, uint8_t *input_mode*, uint8_t *channel*)

ATIMER capture config

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
input_mode	Input mode. This parameter can be one of the following values: ● ATIM_PWMINPUT. ● ATIM_INPUT.
channel	If use pwm input mode, input channel is ATIM_CHANNEL1. This parameter can be one of the following values: ● ATIM_CHANNEL1. ● ATIM_CHANNEL2. ● ATIM_CHANNEL3. ● ATIM_CHANNEL4.

返回:

none

1.4.2.3 void atim_clock_init (ATIM_T * *ATIMx*, BOOL *atim_enable_type*)

ATIMER clock initial

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
atim_enable_type	Clock status. This parameter can be one of the following values: ● ATIM_ENABLE: enable clock. ● ATIM_DISABLE: disable clock.

返回:

none

1.4.2.4 void atim_clr_status (ATIM_T * *ATIMx*, uint8_t *status*)

ATIMER clear update status

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
status	Interrupt flags that you want to clear. This parameter can be one of the following values: ● ATIM_STATUS_ALL. ● ATIM_STATUS_UPDATE. ● ATIM_STATUS_CC1. ● ATIM_STATUS_CC2. ● ATIM_STATUS_CC3. ● ATIM_STATUS_CC4. ● ATIM_STATUS_COM. ● ATIM_STATUS_TRI. ● ATIM_STATUS_BREAK. ● ATIM_STATUS_CC1O. ● ATIM_STATUS_CC2O. ● ATIM_STATUS_CC3O. ● ATIM_STATUS_CC4O.

返回:

none

函数的调用关系图:

1.4.2.5 void atim_dma_config (ATIM_T * *atimx*, uint8_t *length*, uint8_t *base_addr*)

ATIMER DMA config

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
length	Burst length, you can set 1~18.
base_addr	Start base address. This parameter can be one of the following

	values: ● ATIM_CR1. ● ATIM_CR2. ● ATIM_SMCR. ● ATIM_DIER. ● ATIM_SR. ● ATIM_EGR. ● ATIM_CCMR1. ● ATIM_CCMR2. ● ATIM_CCER. ● ATIM_CNT. ● ATIM_PSC. ● ATIM_ARR. ● ATIM_CCR1. ● ATIM_CCR2. ● ATIM_CCR3. ● ATIM_CCR4. ● ATIM_DCR. ● ATIM_DMAR.

返回:

none

1.4.2.6 void atim_dma_init (ATIM_T * *ATIMx*, uint8_t *atim_dma_type*, uint8_t *atim_enable_type*)

ATIMER DMA initial

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
atim_dma_type	Atimer dma type. This parameter can be one of the following values: ● ATIM_DMA_UPDATE. ● ATIM_DMA_CC1. ● ATIM_DMA_CC2. ● ATIM_DMA_CC3. ● ATIM_DMA_CC4. ● ATIM_DMA_COM. ● ATIM_DMA_TRI.
atim_enable_type	Atimer dma status. This parameter can be one of the following values: ● ATIM_ENABLE: enable atimer dma. ● ATIM_DISABLE：disable atimer dma.

返回:

none

1.4.2.7 void atim_enable_config (ATIM_T * *ATIMx*, uint8_t *atim_enable_type*)

ATIMER enable config

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
atim_enable_type	Atimer status. This parameter can be one of the following values: ● ATIM_ENABLE: enable atimer. ● ATIM_DISABLE：disable atimer.

返回:

none

1.4.2.8 void atim_encoder_config (ATIM_T * *ATIMx*, uint8_t *encode_mode*)

ATIMER encoder config

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
encode_mode	Slave mode. This parameter can be one of the following values: ● ATIM_ENCODER1. ● ATIM_ENCODER2. ● ATIM_ENCODER3.

返回:

none

1.4.2.9 uint32_t atim_get_capture (ATIM_T * *ATIMx*, uint8_t *channel*)

ATIMER get capture

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
channel	Atimer channel. This parameter can be one of the following values: ● ATIM_CHANNEL1. ● ATIM_CHANNEL2. ● ATIM_CHANNEL3. ● ATIM_CHANNEL4.

返回:

atim_get_capture Capture value.

1.4.2.10 uint32_t atim_get_cnt (ATIM_T * *ATIMx*)

ATIMER get count value

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.

返回:

atim_get_cnt current count value.

1.4.2.11 uint8_t atim_get_status (ATIM_T * *ATIMx*, uint8_t *status*)

ATIMER get status

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
status	Interrupt flags that you want to check. This parameter can be one of the following values: ● ATIM_STATUS_UPDATE. ● ATIM_STATUS_CC1. ● ATIM_STATUS_CC2. ● ATIM_STATUS_CC3. ● ATIM_STATUS_CC4. ● ATIM_STATUS_COM. ● ATIM_STATUS_TRI. ● ATIM_STATUS_BREAK. ● ATIM_STATUS_CC1O. ● ATIM_STATUS_CC2O. ● ATIM_STATUS_CC3O. ● ATIM_STATUS_CC4O.

返回:

Corresponding status flag.

- 0: Interrupt not setting.
- 1: Interrupt setting.

1.4.2.12 void atim_init (ATIM_T * *ATIMx*, uint32_t *arr*, uint16_t *psc*, uint8_t *counter_direction*, uint8_t *counter_alignment*)

ATIMER initial

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
arr	Automatic reloading value.

psc	Prescaler value.
counter_direction	Direction of counter. This parameter can be one of the following values: ● ATIM_COUNTER_DIRECTION_UP. ● ATIM_COUNTER_DIRECTION_DOWN.
counter_alignment	Alignment of counter. This parameter can be one of the following values: ● ATIM_COUNTER_ALIGNMENT_EDGE. ● ATIM_COUNTER_ALIGNMENT_CENTRE1. ● ATIM_COUNTER_ALIGNMENT_CENTRE2. ● ATIM_COUNTER_ALIGNMENT_CENTRE3.

返回:

none

1.4.2.13 void atim_irq_init (ATIM_T * *ATIMx*, uint8_t *atim_irq_type*, uint8_t *atim_enable_type*, void(*)() *pfunc*)

ATIMER IRQ initial

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
atim_irq_type	Atimer irq type. This parameter can be one of the following values: ● ATIM_IRQ_UPDATE. ● ATIM_IRQ_CC1. ● ATIM_IRQ_CC2. ● ATIM_IRQ_CC3. ● ATIM_IRQ_CC4. ● ATIM_IRQ_COM. ● ATIM_IRQ_TRI. ● ATIM_IRQ_BREAK. ● ATIM_IRQ_CC1O. ● ATIM_IRQ_CC2O. ● ATIM_IRQ_CC3O. ● ATIM_IRQ_CC4O.
atim_enable_type	Interrupt status. This parameter can be one of the following values: ● ATIM_ENABLE: enable interrupt. ● ATIM_DISABLE：disable interrupt.
*void(*pfunc)()*	Interrupt callback function.

返回:

none

函数调用图:

1.4.2.14 void atim_master_trgo_config (ATIM_T * *ATIMx*, uint8_t *trgo_type*)

ATIMER master trgo config

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
trgo_type	Atim trgo type. This parameter can be one of the following values: ● ATIM_TRGO_EGRUG. ● ATIM_TRGO_CNTEN. ● ATIM_TRGO_UPDATE.

返回:

none

1.4.2.15 void atim_pwm_config (ATIM_T * *ATIMx*, uint8_t *output_mode*, uint8_t *output_behavior*, uint8_t *channel*)

ATIMER pwm config

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
output_mode	Output mode. This parameter can be one of the following values: ● ATIM_DEADTIME. ● ATIM_EXTERNALEVENTS. ● ATIM_6STEPPWM. ● ATIM_BRAKE. ● ATIM_PWMOUTPUT.
output_behavior	OCxREF output behavior. ● ATIM_NOTEFFECT. ● ATIM_SETHIGH. ● ATIM_SETLOW. ● ATIM_FLIPLEVEL. ● ATIM_KEEPLOW. ● ATIM_KEEPHIGH. ● ATIM_PWM1. ● ATIM_PWM2.
channel	Output channel. This parameter can be one of the following values: ● ATIM_CHANNEL1. ● ATIM_CHANNEL2. ● ATIM_CHANNEL3. ● ATIM_CHANNEL4.

返回:

none

1.4.2.16 void atim_repeat_count_config (ATIM_T * *ATIMx*, uint8_t *times*)

ATIMER repeat count config

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
times	Repeat count times.

返回:

none

1.4.2.17 oid atim_set_compare (ATIM_T * *ATIMx*, uint8_t *channel*, uint32_t *compare_value*)

ATIMER set compare

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
channel	Atimer channel. This parameter can be one of the following values: ● ATIM_CHANNEL1. ● ATIM_CHANNEL2. ● ATIM_CHANNEL3. ● ATIM_CHANNEL4.
compare_value	Compare value.

返回:

none

1.4.2.18 void atim_slave_config (ATIM_T * *ATIMx*, uint8_t *slave_mode*, uint8_t *channel*)

ATIMER slave config

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
slave_mode	Slave mode. This parameter can be one of the following values: ● ATIM_ENCODER1. ● ATIM_ENCODER2. ● ATIM_ENCODER3. ● ATIM_RESET. ● ATIM_GATING. ● ATIM_TRIGGER. ● ATIM_EXTERNALCLOCK.

channel	Input channel. This parameter can be one of the following values: ● ATIM_CHANNEL1. ● ATIM_CHANNEL2. ● ATIM_CHANNELNULL.

返回:

none

1.4.2.19 void atim_software_event (ATIM_T * *atimx*, uint8_t *events*)

ATIMER software event

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.
events	Events that you want to set. This parameter can be one of the following values: ● ATIM_STATUS_UPDATE. ● ATIM_STATUS_CC1. ● ATIM_STATUS_CC2. ● ATIM_STATUS_CC3. ● ATIM_STATUS_CC4. ● ATIM_STATUS_COM. ● ATIM_STATUS_TRI. ● ATIM_STATUS_BREAK.

返回:

none

1.4.2.20 void atim_xorinput_config (ATIM_T * *ATIMx*)

ATIMER xor input config CH1、CH2、CH3 input

参数:

**ATIMx*	Pointer to ATIM_T structure,where x can be 0 or 7 to select the TIM peripheral.

返回:

none

1.4.2.21 void TIM0_BRK_TIM8_IRQHandler (void)

TIM0 BREAK and TIM8 interrupt handling.

参数:

none	

返回:

none

1.4.2.22 void TIM0_CC_IRQHandler (void)

TIM0 CC interrupt handling.

参数:

none	

返回:

none

1.4.2.23 void TIM0_TRG_COM_TIM10_IRQHandler (void)

TIM0 TRIGE and COM and TIM10 interrupt handling.

参数:

none	

返回:

none

1.4.2.24 void TIM0_UP_TIM9_IRQHandler (void)

TIM0 UPDATE and TIM9 interrupt handling.

参数:

none	

返回:

none

1.4.2.25 void TIM7_BRK_TIM11_IRQHandler (void)

TIM7 BREAK and TIM11 interrupt handling.

参数:

none	

返回:

none

1.4.2.26 void TIM7_CC_IRQHandler (void)

TIM7 CC interrupt handling.

参数:

none	

返回:

none

1.4.2.27 void TIM7_TRG_COM_TIM13_IRQHandler (void)

TIM7 TRIGE and COM and TIM13 interrupt handling.

参数:

none	

返回:

none

1.4.2.28 void TIM7_UP_TIM12_IRQHandler (void)

TIM7 UPDATE and TIM12 interrupt handling.

参数:

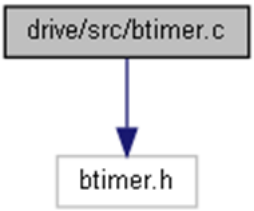
none	

返回:

none

1.5 BTIMER接口

BTIMER driver source file.
#include "btimer.h"
btimer.c 的引用(Include)关系图:



1.5.1 函数

- void **TIM5_IRQHandler** (void)
  TIM5 interrupt handling.
- void **TIM6_IRQHandler** (void)
  TIM6 interrupt handling.
- uint8_t **btim_get_update_status** (BTIM_T *BTIMx)
  BTIMER get update status.
- void **btim_clr_update_status** (BTIM_T *BTIMx)
  BTIMER clear update status.
- void **btim_update_software_event** (BTIM_T *BTIMx)
  BTIMER update software event.
- void **btim_clock_init** (BTIM_T *BTIMx, BOOL btim_enable_type)
  BTIMER clock initial.
- void **btim_update_irq_init** (BTIM_T *BTIMx, uint8_t btim_enable_type, void(*pfunc)())
  BTIMER update irq initial.
- void **btim_update_dma_init** (BTIM_T *BTIMx, uint8_t btim_enable_type)
  BTIMER update dma initial.
- void **btim_enable_config** (BTIM_T *BTIMx, uint8_t btim_enable_type)
  BTIMER enable config.
- void **btim_init** (BTIM_T *BTIMx, uint32_t arr, uint16_t psc)
  BTIMER initial.
- uint32_t **btim_get_cnt** (BTIM_T *BTIMx)
  BTIMER get count value.
- void **btim_master_trgo_config** (BTIM_T *BTIMx, uint8_t trgo_type)
  BTIMER master trgo config.

1.5.2 函数说明

1.5.2.1 void btim_clock_init (BTIM_T * *BTIMx*, BOOL *btim_enable_type*)

BTIMER clock initial

参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.
btim_enable_type	Clock status. This parameter can be one of the following values: ● BTIM_ENABLE: enable clock. ● BTIM_DISABLE: disable clock.

返回:

none

1.5.2.2 void btim_clr_update_status (BTIM_T * *BTIMx*)

BTIMER clear update status

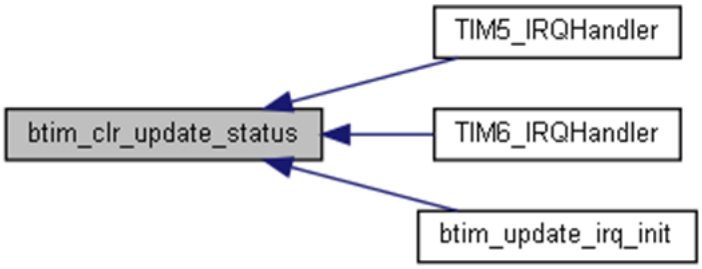
参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.

返回:

none

函数的调用关系图:



1.5.2.3 void btim_enable_config (BTIM_T * *BTIMx*, uint8_t *btim_enable_type*)

BTIMER enable config

参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.
btim_enable_type	Btimer status. This parameter can be one of the following values:

	● BTIM_ENABLE: enable btimer. ● BTIM_DISABLE：disable btimer.

返回:

none

1.5.2.4 uint32_t btim_get_cnt (BTIM_T * *BTIMx*)

BTIMER get count value

参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.

返回:

btim_get_cnt current count value.

1.5.2.5 uint8_t btim_get_update_status (BTIM_T * *BTIMx*)

BTIMER get update status

参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.

返回:

Update status.

- 0: Interrupt not setting.
- 1: Interrupt setting.

1.5.2.6 void btim_init (BTIM_T * *BTIMx*, uint32_t *arr*, uint16_t *psc*)

BTIMER initial

参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.
arr	Automatic reloading value.
psc	Prescaler value.

返回:

none

1.5.2.7 void btim_master_trgo_config (BTIM_T * *BTIMx*, uint8_t *trgo_type*)

BTIMER master trgo config

参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.
trgo_type	Btim trgo type. This parameter can be one of the following values: ● BTIM_TRGO_EGRUG. ● BTIM_TRGO_CNTEN. ● BTIM_TRGO_UPDATE.

返回:

none

1.5.2.8 void btim_update_dma_init (BTIM_T * *BTIMx*, uint8_t *btim_enable_type*)

BTIMER update DMA initial

参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.
btim_enable_type	Btimer dma status. This parameter can be one of the following values: ● BTIM_ENABLE: enable atimer dma. ● BTIM_DISABLE：disable atimer dma.

返回:

none

1.5.2.9 void btim_update_irq_init (BTIM_T * *BTIMx*, uint8_t *btim_enable_type*, void(*)() *pfunc*)

BTIMER update IRQ initial

参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.
btim_enable_type	Interrupt status. This parameter can be one of the following values: ● BTIM_ENABLE: enable interrupt. ● BTIM_DISABLE：disable interrupt.
*void(*pfunc)()*	Interrupt callback function.

返回:

none

函数调用图:

1.5.2.10 void btim_update_software_event (BTIM_T * *BTIMx*)

BTIMER update software event

参数:

**BTIMx*	Pointer to BTIM_T structure,where x can be 5 or 6 to select the TIM peripheral.

返回:

none

1.5.2.11 void TIM5_IRQHandler (void)

TIM5 interrupt handling

参数:

none	

返回:

none

函数调用图:

1.5.2.12 void TIM6_IRQHandler (void)

TIM6 interrupt handling

参数:

none	

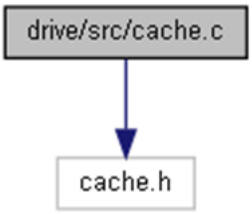
返回:

none

函数调用图:

1.6 CACHE接口

CACHE driver source file
#include "cache.h"
cache.c 的引用(Include)关系图:



1.6.1 函数

- void **cache_enable** (CACHE_T *CACHE, FUNC_E states)
  CACHE_enable
- uint32_t **cache_get_hitcnt** (CACHE_T *CACHE)
  CACHE get hitcnt
- uint32_t **cache_get_misscnt** (CACHE_T *CACHE)
  CACHE get misscnt
- FLAG_E **cache_getcache_status** (CACHE_T *CACHE, uint32_t cache_flag)
  get cache status

1.6.2 函数说明

1.6.2.1 void cache_enable (CACHE_T * *CACHE*, FUNC_E *states*)

CACHE_enable

参数:

**CACHE*	pointer to CACHE_T structure
states	ENABLE / DISABLE

返回:

none

1.6.2.2 uint32_t cache_get_hitcnt (CACHE_T * *CACHE*)

CACHE get hitcnt

参数:

**CACHE*	pointer to CACHE_T structure

返回:

The data which getting in cache_hitcnt register

1.6.2.3 uint32_t cache_get_misscnt (CACHE_T * *CACHE*)

CACHE get misscnt

参数:

**CACHE*	pointer to CACHE_T structure

返回:

The data which getting in cache_misscnt register

1.6.2.4 FLAG_E cache_getcache_status (CACHE_T * *CACHE*, uint32_t *cache_flag*)

CACHE get cache status

参数:

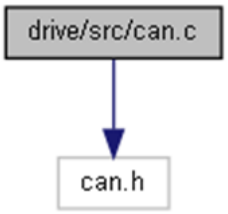
**CACHE*	pointer to CACHE_T structure
cache_flag	The flag of cache status This parameter has one of the following values: ● CACHE_CLOSEED_FLAG Cache off state ● CACHE_OPENING_FLAG Cache is being opened. ● CACHE_OPENED_FLAG Cache open state ● CACHE_CLOSEING_FLAG Cache is being opened.

返回:

The new state of cache_flag (SET or RESET).

1.7 CAN接口

#include "can.h"
can.c 的引用(Include)关系图:



1.7.1 函数

- void **CAN0_IRQHandler** (void)
  CAN0_IRQHandler.
- void **CAN1_IRQHandler** (void)
  CAN1_IRQHandler.
- void **can_clk_init** (CAN_T *CANx, BOOL newstate)
  CAN clk init
- void **can_init** (CAN_T *CANx, uint8_t baudrate)
  CAN_init APB3 clock =(AHB clock)/4 = 42Mhz.
  CAN baudrate = Fpclk/(2(BRP+1)*(TS1+TS2+1)).
  The following parameters of different baud rates are set based on the Fpclk=42M, for reference only. Set the baudrate to 1M：
  BRP=2(3 frequency division),
  baudrate = 1M = 42M/(2*(2+1)*(1+TS1+TS2)),can be set TS1=3,TS2=3
- void **can_irq_init** (CAN_T *CANx, uint8_t irqstate, uint8_t irqtype, void(*pfunc)())
  CAN interrupt enable
- void **can_filter_config** (CAN_T *CANx, uint32_t filter_value, uint8_t filter_mode, uint8_t data_mode)
  CAN filter config
- void **can_send_data** (CAN_T *CANx, uint32_t *data)
  CAN send data
- void **can_read_data** (CAN_T *CANx, uint32_t *buff)
  CAN read data；get can data
- uint8_t **can_get_sr_reg** (CAN_T *CANx)
  CAN get sr reg, get the can status register value

1.7.2 函数说明

1.7.2.1 void CAN0_IRQHandler (void)

CAN0_IRQHandler.

参数:

none	

返回:

none

1.7.2.2 void CAN1_IRQHandler (void)

CAN1_IRQHandler.

参数:

none	

返回:

none

1.7.2.3 void can_clk_init (CAN_T * *CANx*, BOOL *newstate*)

CAN clk init

参数:

**CANx*	pointer to UART_T structure. This parameter can be one of the following values: ● CAN0: CAN0 selected. ● CAN1：CAN1 selected.
newstate	*ENABLE/DISABLE*

返回:

none

1.7.2.4 void can_filter_config (CAN_T * *CANx*, uint32_t *filter_value*, uint8_t *filter_mode*, uint8_t *data_mode*)

CAN filter config

参数:

**CANx*	pointer to UART_T structure. This parameter can be one of the following values:

	● CAN0: CAN0 selected. ● CAN1：CAN1 selected.
filter_value	filter value
filter_mode	This parameter can be CAN_SINGLE_FILTER or CAN_DOUBLE_FILTER.
data_mode	This parameter can be CAN_IDE_STD_FORMAT or CAN_IDE_EXT_FORMAT.

返回:

none

1.7.2.5 uint8_t can_get_sr_reg (CAN_T * *CANx*)

CAN get sr reg, get the can status register value

参数:

**CANx*	pointer to UART_T structure. This parameter can be one of the following values: ● CAN0: CAN0 selected. ● CAN1：CAN1 selected.

返回:

register value

1.7.2.6 void can_init (CAN_T * *CANx*, uint8_t *baudrate*)

CAN_init APB3 clock =(AHB clock)/4 = 42Mhz,

CAN baudrate = Fpclk/(2(BRP+1)*(TS1+TS2+1)) The following parameters of different baud rates are set based on the Fpclk=42M, for reference only.

Set the baudrate to 1M： BRP=2(3 frequency division), baudrate = 1M = 42M/(2*(2+1)*(1+TS1+TS2)),can be set TS1=3,TS2=3

Set the baudrate to 500k： BRP=6(7 frequency division), baudrate = 0.5M = 42M/(2*(6+1)*(1+TS1+TS2)),can be set TS1=3,TS2=2

Set the baudrate to 250k： BRP=13(14 frequency division), baudrate = 0.25M = 42M/(2*(13+1)*(1+TS1+TS2)),can be set TS1=3,TS2=2

Set the baudrate to 125k： BRP=27(28 frequency division), baudrate = 0.125M = 42M/(2*(27+1)*(1+TS1+TS2)),can be set TS1=3,TS2=2

参数:

**CANx*	pointer to UART_T structure. This parameter can be one of the following values: ● CAN0: CAN0 selected. ● CAN1：CAN1 selected.
baudrate	baudrate can chose 1M/500k/250k/125k bps

返回:

none

1.7.2.7 void can_irq_init (CAN_T * *CANx*, uint8_t *irqstate*, uint8_t *irqtype*, void(*)() *pfunc*)

CAN interrupt enable

参数:

**CANx*	pointer to UART_T structure. This parameter can be one of the following values: ● CAN0: CAN0 selected. ● CAN1：CAN1 selected.
irqstate	interrupt status. This parameter can be one of the following values: ● 0: enable interrupt. ● 1：disable interrupt.
irqtype	type of interrupt
void	(*pfunc)() interrupt callback function

返回:

none

1.7.2.8 void can_read_data (CAN_T * *CANx*, uint32_t * *buff*)

CAN read data；get can data

参数:

**CANx*	pointer to UART_T structure. This parameter can be one of the following values: ● CAN0: CAN0 selected. ● CAN1：CAN1 selected.
**buff*	CAN RX data

返回:

none

1.7.2.9 void can_send_data (CAN_T * *CANx*, uint32_t * *data*)

CAN send data

参数:

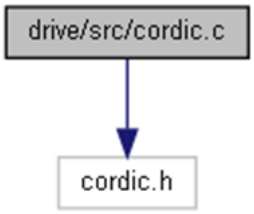
**CANx*	pointer to UART_T structure. This parameter can be one of the following values: ● CAN0: CAN0 selected. ● CAN1：CAN1 selected.
data	CAN TX data

返回:

none

1.8 Cordic接口

cordic driver source file
#include "cordic.h"
cordic.c 的引用(Include)关系图:



1.8.1 函数

- void **cordic_clk_init** (BOOL newstate)
  Initialize the clock of the cordic.
- void **cordic_datain_init** (cordic_data_format_t data_bit, cordic_datain_merg_t data_merg, cordic_addrin_t data_addr)
  Setting the input data format.
- void **cordic_dataout_init** (cordic_data_format_t data_bit, cordic_dataout_merg_t data_merg, cordic_addrout_t data_addr)
  Setting the output data format.
- void **cordic_calculate_times** (uint8_t times)
  cordic_calculate_times
- void **cordic_set_din1** (uint32_t data1)
  cordic_set_din1
- void **cordic_set_din2** (uint32_t data2)
  cordic_set_din2
- uint32_t **cordic_get_dout1** (void)
  cordic_get_dout1
- uint32_t **cordic_get_dout2** (void)
  cordic_get_dout2
- uint8_t **cordic_wait_cal** (void)
  cordic_wait_cal
- void **cordic_trans_mode** (cordic_intrans_mode_t intrans_mode, cordic_outtrans_mode_t outtrans_mode)
  cordic_trans_mode
- void **cordic_init** (cordic_func_mode_t func_mode, uint8_t scale, cordic_intrans_mode_t intrans_mode, cordic_outtrans_mode_t outtrans_mode)
  Initialize the cordic.

1.8.2 函数说明

1.8.2.1 void cordic_calculate_times (uint8_t *times*)

cordic calculate times

参数:

times	This parameter is to set the number of operations

返回:

none

1.8.2.2 void cordic_clk_init (BOOL *newstate*)

Initialize the clock of the cordic.

参数:

newstate	ENABLE/DISABLE

返回:

none

1.8.2.3 void cordic_datain_init (cordic_data_format_t *data_bit*, cordic_datain_merg_t *data_merg*, cordic_addrin_t *data_addr*)

Setting the input data format.

参数:

data_bit	Input data bit number selection
data_merg	Input data merged or not
data_addr	Input data address format setting

返回:

none

1.8.2.4 void cordic_dataout_init (cordic_data_format_t *data_bit*, cordic_dataout_merg_t *data_merg*, cordic_addrout_t *data_addr*)

Setting the output data format.

参数:

data_bit	Output data bit number selection
data_merg	Output data merged or not
data_addr	Output data address format setting

返回:

none

1.8.2.5 uint32_t cordic_get_dout1 (void)

cordic_get_dout1

参数:

none	-

返回:

The output data1 of the calculation result

1.8.2.6 uint32_t cordic_get_dout2 (void)

cordic_get_dout2

参数:

none	-

返回:

The output data2 of the calculation result

1.8.2.7 void cordic_init (cordic_func_mode_t *func_mode*, uint8_t *scale*, cordic_intrans_mode_t *intrans_mode*, cordic_outtrans_mode_t *outtrans_mode*)

Initialize the cordic.

参数:

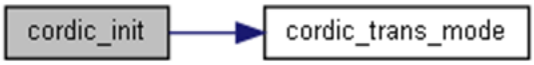
func_mode	set function mode This parameter can be one of the following values: ● CORDIC_FUNCTION_MODE0 CORDIC FUNCTION0 m*sin θ /m*cos θ ● CORDIC_FUNCTION_MODE1 CORDIC FUNCTION1 atan2(y,x) / sqrt(x2+y2) ● CORDIC_FUNCTION_MODE2 CORDIC FUNCTION2 y*x ● CORDIC_FUNCTION_MODE3 CORDIC FUNCTION3 y/x ● CORDIC_FUNCTION_MODE4 CORDIC FUNCTION4 sinhw/coshw ● CORDIC_FUNCTION_MODE5 CORDIC FUNCTION5 tanh-1(y/x) ● CORDIC_FUNCTION_MODE6 CORDIC FUNCTION6 ln(x) ● CORDIC_FUNCTION_MODE7 CORDIC FUNCTION7 sqrt(x)
scale	enter the numeric range SCALE(Please fill in NULL when is not used)
intrans_mode	Selection of input data transmission mode

outtrans_mode	Selection of output data transmission mode

返回:

none

函数调用图:



1.8.2.8 void cordic_set_din1 (uint32_t *data1*)

cordic_set_din1

参数:

data1	This parameter is to set input data 1.

返回:

none

1.8.2.9 void cordic_set_din2 (uint32_t *data2*)

cordic_set_din2

参数:

data2	This parameter is to set input data2.

返回:

none

1.8.2.10 void cordic_trans_mode (cordic_intrans_mode_t *intrans_mode*, cordic_outtrans_mode_t *outtrans_mode*)

cordic_trans_mode

参数:

intrans_mode	Selection of input data transmission mode
outtrans_mode	Selection of output data transmission mode

返回:

none

函数的调用关系图:

1.8.2.11 uint8_t cordic_wait_cal (void)

cordic wait cal

参数:

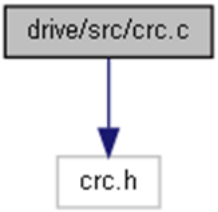
none	

返回:

1 or 0

1.9 CRC接口

CRC driver source file
#include "crc.h"
crc.c 的引用(Include)关系图:



1.9.1 函数

- void **crc_clk_init** (BOOL state)
  CRC clock init
- void **crc_init** (CRC_T *crc, uint8_t pol, uint8_t dw, uint8_t din, uint8_t dout, uint8_t init)
  CRC init
- uint16_t **crc16_count** (CRC_T *CRC, uint16_t crc_data[], uint32_t len, uint16_t init_data, uint16_t xorout)
  CRC16 count
- uint32_t **crc32_count** (CRC_T *CRC, uint32_t crc_data[], uint32_t len, uint32_t init_data, uint32_t xorout)
  CRC32 count
- uint16_t **crc16_dw16_soft** (uint16_t crc, uint16_t data)
  CRC16 data width 16bit use software
- uint32_t **crc32_dw32_bit** (uint32_t crc)
  CRC32 data width 32bit count
- uint32_t **crc32_dw32_soft** (uint32_t crc, uint32_t data)
  CRC32 data width 32bit use software
- uint16_t **crc16_bytes_soft** (uint16_t crc_data[], uint16_t len, uint16_t init_data, uint16_t xorout)
  CRC16 bytes use software
- uint32_t **crc32_bytes_soft** (uint32_t crc_data[], uint32_t len, uint32_t init_data, uint32_t xorout)
  CRC32 bytes use software
- uint8_t **reverse_8** (uint8_t in_data)
  reverse 8 bit
- uint16_t **reverse_16** (uint16_t in_data)
  reverse 16 bit

- uint32_t **reverse_32** (uint32_t in_data)
  reverse 32 bit
- uint16_t **reverse8_16** (uint16_t in_data)
  16bit data is in reverse 8-bit order
- uint32_t **reverse8_32** (uint32_t in_data)
  32bit data is in reverse 8-bit order

1.9.2 函数说明

1.9.2.1 uint16_t crc16_bytes_soft (uint16_t *crc_data*[], uint16_t *len*, uint16_t *init_data*, uint16_t *xorout*)

CRC16 bytes use software

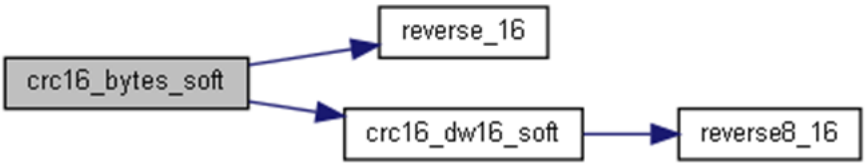
参数:

crc_data[]	CRC data
len	CRC data len
init_data	init data
xorout	the results are XOR

返回:

reg_crc

函数调用图:



1.9.2.2 uint16_t crc16_count (CRC_T * *CRC*, uint16_t *crc_data*[], uint32_t *len*, uint16_t *init_data*, uint16_t *xorout*)

CRC16 count

参数:

**CRC*	pointer to CRC_T structure
crc_data[]	crc data
len	crc data len
init_data	crc init data
xorout	the results are XOR

返回:

crc_data^xorout

1.9.2.3 uint16_t crc16_dw16_soft (uint16_t *crc*, uint16_t *data*)

CRC16 data width 16bit use software

参数:

crc	CRC value
data	CRC data

返回:

crc&0xffff 16bit crc

函数调用图:

函数的调用关系图:

1.9.2.4 uint32_t crc32_bytes_soft (uint32_t *crc_data*[], uint32_t *len*, uint32_t *init_data*, uint32_t *xorout*)

CRC32 bytes use software

参数:

crc_data[]	CRC data
len	CRC data len
init_data	init data
xorout	the results are XOR

返回:

reg_crc

函数调用图:

1.9.2.5 uint32_t crc32_count (CRC_T * *CRC*, uint32_t *crc_data*[], uint32_t *len*, uint32_t *init_data*, uint32_t *xorout*)

CRC32 count

参数:

**CRC*	pointer to CRC_T structure
crc_data[]	crc data
len	crc data len
init_data	crc init data
xorout	the results are XOR

返回:

crc_data^xorout

1.9.2.6 uint32_t crc32_dw32_bit (uint32_t *crc*)

CRC32 data width 32bit count

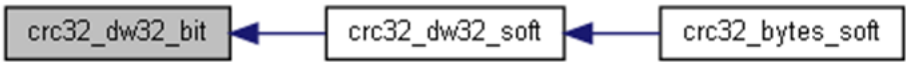
参数:

crc	32 bit CRC

返回:

crc 32 bit crc

函数的调用关系图:



1.9.2.7 uint32_t crc32_dw32_soft (uint32_t *crc*, uint32_t *data*)

crc32 data width 32bit use software

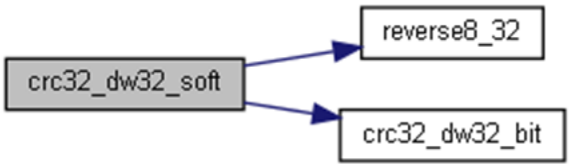
参数:

crc	32 bit CRC
data	CRC data

返回:

crc 32 bit crc

函数调用图:



函数的调用关系图:

1.9.2.8 void crc_clk_init (BOOL *state*)

crc clock init

参数:

state	Clock and reset status. This parameter can be one of the following values: ● CRC_ENABLE: enable crc clock and set it into work mode. ● CRC_DISABLE：disable crc clock and set crc into reset mode.

返回:

none

1.9.2.9 void crc_init (CRC_T * *CRC*, uint8_t *pol*, uint8_t *dw*, uint8_t *din*, uint8_t *dout*, uint8_t *init*)

CRC init

参数:

**CRC*	pointer to CRC_T structure
pol	set polynomial This parameter can be one of the following values: ● POL_CRC16_1021: x16 +x12 + x5 + 1 ● POL_CRC16_8005：x16 +x15 + x2 + 1； ● POL_CRC32_04C11DB7：x32 + x26 + x23 + x22 + x16 + x12 + x11+ x10 + x8 + x7 + x5 + x4 + x2 + x1 + 1
dw	set data width This parameter can be one of the following values: ● DW_8BIT: 8bit data width ● DW_16BIT: 16bit data width ● DW_32BIT: 32bit data width
din	set input data rollover This parameter can be one of the following values: ● DEFAULT: input data default ● DIN_8BITREF: input data in 8bit reverse ● DIN_16BITREF: input data in 16bit reverse ● DIN_32BITREF: input data in 32bit reverse
dout	set output data rollover This parameter can be one of the following values: ● DEFAULT: output data default ● DOUT_REF: output data reverse
init	set init rollover This parameter can be one of the following values: ● DEFAULT: initial value default ● INIT_REF: initial value reverse

返回:

none

1.9.2.10 uint16_t reverse8_16 (uint16_t *in_data*)

16bit data is in reverse 8-bit order

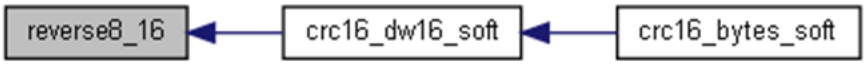
参数:

in_data	input data

返回:

temp

函数的调用关系图:



1.9.2.11 uint32_t reverse8_32 (uint32_t *in_data*)

32bit data is in reverse 8-bit order

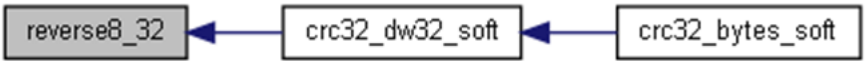
参数:

in_data	input data

返回:

temp

函数的调用关系图:



1.9.2.12 uint16_t reverse_16 (uint16_t *in_data*)

reverse 16 bit

参数:

in_data	input data

返回:

temp

函数的调用关系图:

1.9.2.13 uint32_t reverse_32 (uint32_t *in_data*)

reverse 32 bit

参数:

in_data	input data

返回:

temp

函数的调用关系图:

1.9.2.14 uint8_t reverse_8 (uint8_t *in_data*)

reverse 8 bit

参数:

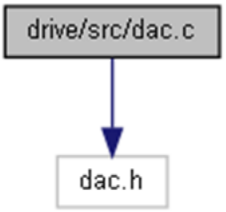
in_data	input data

返回:

temp

1.10 DAC接口

DAC driver source file
#include "dac.h"
dac.c 的引用(Include)关系图:



1.10.1 函数

- void **DAC_IRQHandler** (void)
  DAC interrupt handling.
- void **dac_enable_config** (DAC_T *DACx, uint8_t dac_channelx, uint8_t dac_enable_type)
  DAC enable config.
- void **dac_software_trig** (DAC_T *DACx, uint8_t dac_channelx)
  DAC software trigger.
- void **dac_clock_init** (DAC_T *DACx, uint8_t dac_clk_div)
  DAC clock initial.
- void **dac_init** (DAC_T *DACx, uint8_t dac_channelx, uint8_t reference_voltage, uint8_t trig_source, uint8_t trig_mode_enable_type, uint8_t wave_type, uint8_t masks_amplitudes, uint8_t buffer_enable_type)
  DAC initial.
- void **dac_clr_int** (DAC_T *DACx, uint8_t dac_channelx)
  DAC clear interrupt flag.
- void **dac_irq_init** (DAC_T *DACx, uint8_t dac_channelx, uint8_t irq_enable, void(*pfunc)())
  DAC IRQ initial.
- void **dac_set_channel_data** (DAC_T *DACx, uint8_t dac_channelx, uint8_t data_type, uint16_t channel_data)
  DAC set channel data.
- void **dac_set_double_channel_data** (DAC_T *DACx, uint8_t data_type, uint16_t channel_0_data, uint16_t channel_1_data)
  DAC set double channel data.
- uint16_t **dac_get_channel_data** (DAC_T *DACx, uint8_t dac_channelx)
  DAC get channel data.

1.10.2 函数说明

1.10.2.1 void dac_clock_init (DAC_T * *DACx*, uint8_t *dac_clk_div*)

DAC clock initial

参数:

**DACx*	Pointer to DAC_T structure.
dac_clk_div	0-255. Generally, the value is set to 0.

返回:

none

1.10.2.2 void dac_clr_int (DAC_T * *DACx*, uint8_t *dac_channelx*)

DAC clear interrupt flag

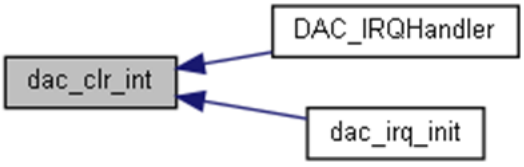
参数:

**DACx*	Pointer to DAC_T structure.
dac_channelx	DAC channel. This parameter can be one of the following values: ● DAC_CHANNEL_0. ● DAC_CHANNEL_1.

返回:

none

函数的调用关系图:



1.10.2.3 void dac_enable_config (DAC_T * *DACx*, uint8_t *dac_channelx*, uint8_t *dac_enable_type*)

DAC enable config

参数:

**DACx*	Pointer to DAC_T structure.
dac_channelx	DAC channel. This parameter can be one of the following values: ● DAC_CHANNEL_0. ● DAC_CHANNEL_1.
dac_enable_type	DAC status. This parameter can be one of the following values: ● DAC_ENABLE: enable DAC.

	● DAC_DISABLE：disable DAC.

返回:

none

1.10.2.4 uint16_t dac_get_channel_data (DAC_T * *DACx*, uint8_t *dac_channelx*)

DAC get channel data

参数:

**DACx*	Pointer to DAC_T structure.
dac_channelx	DAC channel. This parameter can be one of the following values: ● DAC_CHANNEL_0. ● DAC_CHANNEL_1.

返回:

dac_get_channel_data Channelx value.

1.10.2.5 void dac_init (DAC_T * *DACx*, uint8_t *dac_channelx*, uint8_t *reference_voltage*, uint8_t *trig_source*, uint8_t *trig_mode_enable_type*, uint8_t *wave_type*, uint8_t *masks_amplitudes*, uint8_t *buffer_enable_type*)

DAC initial

参数:

**DACx*	Pointer to DAC_T structure.
dac_channelx	DAC channel. This parameter can be one of the following values: ● DAC_CHANNEL_0. ● DAC_CHANNEL_1.
reference_voltage	DAC reference voltage. This parameter can be one of the following values: ● DAC_REFVOL_VDDA. ● VREFIN. ● VREFOUT.
trig_source	DAC trigger source. This parameter can be one of the following values: ● DAC_TRIG_TIM5TRGO. ● DAC_TRIG_TIM7TRGO. ● DAC_TRIG_TIM6TRGO. ● DAC_TRIG_TIM4TRGO. ● DAC_TRIG_TIM1TRGO. ● DAC_TRIG_TIM3TRGO. ● DAC_TRIG_EXTINTPIN. ● DAC_TRIG_SOFTWARE.

trig_mode_enable_type	DAC trigger status. This parameter can be one of the following values: ● DAC_TRIGMODE_ENABLE：enable trigger. ● DAC_TRIGMODE_DISABLE：disable trigger.
wave_type	DAC output mode. This parameter can be one of the following values: ● DAC_WAVE_NOISE. ● TRIANGLE. ● NONE.
masks_amplitudes	Mask in noise mode or Amplitude in triangular wave mode. This parameter can be one of the following values: ● DAC_MASKSORAMPLITUDES_0. ● DAC_MASKSORAMPLITUDES_3. ● DAC_MASKSORAMPLITUDES_7. ● DAC_MASKSORAMPLITUDES_15. ● DAC_MASKSORAMPLITUDES_31. ● DAC_MASKSORAMPLITUDES_63. ● DAC_MASKSORAMPLITUDES_127. ● DAC_MASKSORAMPLITUDES_255. ● DAC_MASKSORAMPLITUDES_511. ● DAC_MASKSORAMPLITUDES_1023. ● DAC_MASKSORAMPLITUDES_2047. ● DAC_MASKSORAMPLITUDES_4095.
buffer_enable_type	DAC buffer status. This parameter can be one of the following values: ● DAC_BUFFER_ENABLE：enable buffer. ● DAC_BUFFER_DISABLE：disable buffer.

返回:

none

1.10.2.6 void dac_irq_init (DAC_T * *DACx*, uint8_t *dac_channelx*, uint8_t *irq_enable*, void(*)() *pfunc*)

DAC IRQ initial

参数:

**DACx*	Pointer to DAC_T structure.
dac_channelx	DAC channel. This parameter can be one of the following values: ● DAC_CHANNEL_0. ● DAC_CHANNEL_1.
irq_enable	DAC irq status. This parameter can be one of the following values: ● DAC_ENABLE: enable irq. ● DAC_DISABLE：disable irq.
*void(*pfunc)()*	Interrupt callback function.

返回:

none

函数调用图:

1.10.2.7 void DAC_IRQHandler (void)

DAC interrupt handling

参数:

none	

返回:

none

函数调用图:

1.10.2.8 void dac_set_channel_data (DAC_T * *DACx*, uint8_t *dac_channelx*, uint8_t *data_type*, uint16_t *channel_data*)

DAC set channel data

参数:

**DACx*	Pointer to DAC_T structure.
dac_channelx	DAC channel. This parameter can be one of the following values: ● DAC_CHANNEL_0. ● DAC_CHANNEL_1.
data_type	DAC data type. This parameter can be one of the following values: ● DAC_DATATYPE_RIGHT_12. ● DAC_DATATYPE_LEFT_12. ● DAC_DATATYPE_RIGHT_8.
channel_data	12bit : 0-4095 output 0-REFVOL / 8bit : 0-255 output 0-REFVOL.

返回:

none

1.10.2.9 void dac_set_double_channel_data (DAC_T * *DACx*, uint8_t *data_type*, uint16_t *channel_0_data*, uint16_t *channel_1_data*)

DAC set double channel data

参数:

**DACx*	Pointer to DAC_T structure.
data_type	DAC data type. This parameter can be one of the following values: ● DAC_DATATYPE_RIGHT_12. ● DAC_DATATYPE_LEFT_12. ● DAC_DATATYPE_RIGHT_8.
channel_1_data	Channel 0, 12bit : 0-4095 output 0-REFVOL / 8bit : 0-255 output 0-REFVOL.
channel_2_data	Channel 1,12bit : 0-4095 output 0-REFVOL / 8bit : 0-255 output 0-REFVOL.

返回:

none

1.10.2.10 void dac_software_trig (DAC_T * *DACx*, uint8_t *dac_channelx*)

DAC software trigger

参数:

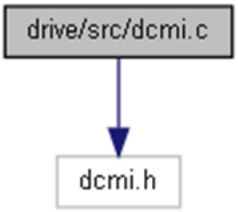
**DACx*	Pointer to DAC_T structure.
dac_channelx	DAC channel. This parameter can be one of the following values: ● DAC_CHANNEL_0. ● DAC_CHANNEL_1.

返回:

none

1.11 DCMI接口

DCMI driver source file
#include "dcmi.h"
dcmi.c 的引用(Include)关系图:



1.11.1 函数

- void **dcmi_clk_init** (FUNC_E newstate)
  enable/disable DCMI clock, meanwhile release/enable dcmi reset status
- void **dcmi_clk_cmd** (BOOL newstate)
  enable/disable DCMI clock
- void **dcmi_reset** (void)
  DCMI reset
- void **dcmi_deinit** (void)
  deinitializes the DCMI registers to default reset values
- DCMI_T * **dcmi_type_read** (void)
  read the DCMI registers values
- uint32_t **dcmi_data_read** (void)
  fetch the data from DCMI fifo
- void * **dcmi_mstadr_read** (void)
  read the data output address
- void **dcmi_mstadr_write** (void *dcmi_mstadr)
  write the data output address
- void **dcmi_input_offset** (FUNC_E newstate)
  data sample from d0~d7 or d2~d9
- void **dcmi_cmd** (FUNC_E newstate)
  enable or disable DCMI interface
- void **dcmi_esc_cmd** (FUNC_E newstate)
  enable synchronization mode, enbeded code or hardware
- void **dcmi_jpeg_cmd** (FUNC_E newstate)
  enable or disable jpeg mode.
- void **dcmi_crop_cmd** (FUNC_E newstate)
  enable or disable crop(window clipping) function

- void **dcmi_sm_cmd** (FUNC_E newstate)
  enable capture mode, single frame or continous frame
- void **dcmi_capture_cmd** (FUNC_E newstate)
  enable or disable DCMI capture function
- void **dcmi_mst_cmd** (FUNC_E newstate)
  enable or disable DCMI data output automatic function
- void **dcmi_vsyncpol_config** (POL_E newstate)
  config vsync polarity
- void **dcmi_hsyncpol_config** (POL_E newstate)
  config hsync polarity
- void **dcmi_pclkpol_config** (POL_E newstate)
  config pixel clock polarity
- void **dcmi_crop_config** (DCMI_CROPINIT_T *dcmi_crop_init)
  config the DCMI crop parameters.
- void **dcmi_data_inwidth_config** (DCMI_DATA_INWIDTH_E data_inwidth)
  config the input data width.
- void **dcmi_data_outwidth_config** (DCMI_DATA_OUTWIDTH_E data_outwidth)
  config the output data width.
- void **dcmi_framerate_config** (DCMI_FRAMERATE_E framerate)
  config the frame capture rate.
- void **dcmi_esc_config** (DCMI_CODESINIT_T *esc_init)
  config the DCMI embedded synchronization codes.
- void **dcmi_escmask_config** (DCMI_CODESINIT_T *escmask_init)
  config the DCMI embedded synchronization codes mask, bit = 1 will be used to detection.
- void **dcmi_addrmode_config** (DCMI_ADDRMODE_E addrmode)
  config the data output address mode.
- void **dcmi_addrst_config** (FUNC_E newstate)
  enable or disable the data output address reset function.
- void **dcmi_it_config** (uint16_t dcmi_it, FUNC_E newstate)
  enable or disable the DCMI interrupts.
- FLAG_E **dcmi_it_flag_get** (uint16_t it_flag)
  get the DCMI it flag status.
- void **dcmi_it_clear** (uint16_t it_clear)
  clear the DCMI interrupt flag(dcmi_intraw register).
- void **dcmi_isr** (void)
  DCMI interrupt service routines.
- void **dcmi_irq_init** (FUNC_E newstate, void(*dcmi_subisr[])())
  initialize the DCMI interrupt.

- void **DCMI_IRQHandler** (void)

  DCMI_IRQHandler.

- void **dcmi_type_init** (void)

  DCMI initialize，this function can be initialized by the user according to requirements. use different macro to switch different transfer methods or different vedio format, for example:rgb,jpeg. following macro can be used: DCMI_DMA_EMC, DCMI_JPEG_MODE.

1.11.2 函数说明

1.11.2.1 void dcmi_addrmode_config (DCMI_ADDRMODE_E *addrmode*)

config the data output address mode.

参数:

addrmode	This parameter can be one of the following value: ● 0x00/0x01: data output to the same address ● 0x02 : data output to the decrease address ● 0x03 : data output to the increase address

返回:

none

函数的调用关系图:

1.11.2.2 void dcmi_addrst_config (FUNC_E *newstate*)

enable or disable the data output address reset function.

参数:

newstate	data output address reset command This parameter can be one of the following value: ● 0:disable ● 1:enable

返回:

none

函数的调用关系图:

1.11.2.3 void dcmi_capture_cmd (FUNC_E *newstate*)

enable or disable DCMI capture function

参数:

newstate	capture command This parameter can be one of the following value: ● 0:disable. ● 1:enable.

返回:

none

函数的调用关系图:

1.11.2.4 void dcmi_clk_cmd (BOOL *newstate*)

enable/disable DCMI clock

参数:

newstate	clock status This parameter can be one of the following value: ● 0:disable. ● 1:enable.

返回:

none

1.11.2.5 void dcmi_clk_init (FUNC_E *newstate*)

enable/disable dcmi clock, meanwhile release/enable dcmi reset status

参数:

newstate	clock and reset status This parameter can be one of the following value: ● 0:disable. ● 1:enable.

返回:

none

1.11.2.6 void dcmi_cmd (FUNC_E *newstate*)

enable or disable DCMI interface

参数:

newstate	dcmi command This parameter can be one of the following value:

	● 0:disable. ● 1:enable.

返回:

none

函数的调用关系图:

1.11.2.7 void dcmi_crop_cmd (FUNC_E *newstate*)

enable or disable crop(window clipping) function

参数:

newstate	crop command This parameter can be one of the following value: ● 0:disable. ● 1:enable.

返回:

none

函数的调用关系图:

1.11.2.8 void dcmi_crop_config (DCMI_CROPINIT_T * *dcmi_crop_init*)

config the DCMI crop parameters.

参数:

dcmi_crop_init	structure pointer point to DCMI_CROPINIT_T This structure contains following member: ● horizonal_startpoint: Specifies the number of pixel clocks to count before starting a capture. This parameter can be a value between 0x00 and 0x3FFF ● vertical_startpoint : Specifies the Vertical start line count from which the image capture will start. This parameter can be a value between 0x00 and 0x1FFF ● horizonal_size : Specifies the number of pixel clocks to be captured from the starting point on the same line.This parameter can be a value between 0x00 and 0x3FFF ● vertical_size : Specifies the number of lines to be captured from the starting point. This parameter can be a value between 0x00 and 0x3FFF

返回:

none

函数的调用关系图:

1.11.2.9 void dcmi_data_inwidth_config (DCMI_DATA_INWIDTH_E *data_inwidth*)

config the input data width.

参数:

data_inwidth	This parameter can be one of the following value: ● 0x00: 8bit ● 0x01: 10bit ● 0x02: 12bit ● 0x03: 14bit

返回:

none

函数的调用关系图:

1.11.2.10 void dcmi_data_outwidth_config (DCMI_DATA_OUTWIDTH_E *data_outwidth*)

config the output data width.

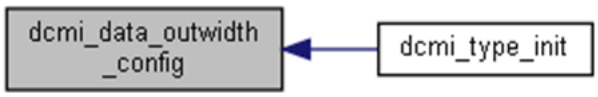
参数:

data_inwidth	This parameter can be one of the following value: ● 0x00: 8bit ● 0x01: 16bit ● 0x02: 32bit

返回:

none

函数的调用关系图:



1.11.2.11 uint32_t dcmi_data_read (void)

fetch the data from DCMI fifo

参数:

none	

返回:

DCMI->DATA read data

1.11.2.12 void dcmi_deinit (void)

deinitializes the DCMI registers to default reset values

参数:

none	

返回:

none

1.11.2.13 void dcmi_esc_cmd (FUNC_E *newstate*)

enable synchronization mode, enbeded code or hardware

参数:

newstate	synchronization command This parameter can be one of the following value: ● 0: hardware synchronization. ● 1: embended code synchronization.

返回:

none

函数的调用关系图:

1.11.2.14 void dcmi_esc_config (DCMI_CODESINIT_T * *esc_init*)

config the DCMI embedded synchronization codes.

参数:

esc_init	structure pointer point to DCMI_CODESINIT_T This structure contains following member: ● framestart_code: Specifies the code of the frame start delimiter. ● linestart_code : Specifies the code of the line start delimiter. ● lineend_code : Specifies the code of the line end delimiter.

	● frameend_code : Specifies the code of the frame end delimiter.

返回:

none

1.11.2.15 void dcmi_escmask_config (DCMI_CODESINIT_T * *escmask_init*)

config the dcmi embedded synchronization codes mask, bit = 1 will be used to detection.

参数:

escmask_init	structure pointer point to DCMI_CODESINIT_T This structure contains following member: ● framestart_code: Specifies the code of the frame start delimiter. ● linestart_code : Specifies the code of the line start delimiter. ● lineend_code : Specifies the code of the line end delimiter. ● frameend_code : Specifies the code of the frame end delimiter.

返回:

none

1.11.2.16 void dcmi_framerate_config (DCMI_FRAMERATE_E *framerate*)

config the frame capture rate.

参数:

framerate	capture rate This parameter can be one of the following value: ● 0x00 : capture all ● 0x01 : capture 1/2 ● 0x02/0x03: capture 1/4

返回:

none

函数的调用关系图:

1.11.2.17 void dcmi_hsyncpol_config (POL_E *newstate*)

config hsync polarity

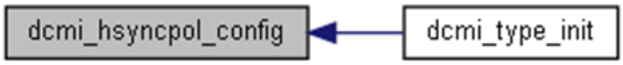
参数:

newstate	hsync polarity This parameter can be one of the following value: ● 0:low. ● 1:high.

返回:

none

函数的调用关系图:



1.11.2.18 void dcmi_input_offset (FUNC_E *newstate*)

data sample from d0~d7 or d2~d9

参数:

newstate	source of sampling data This parameter can be one of the following value: ● 0: d0~d7. ● 1: d2~d9.

返回:

none

函数的调用关系图:

1.11.2.19 void dcmi_irq_init (FUNC_E *newstate*, void(*[])() *dcmi_subisr*)

initialize the DCMI interrupt.

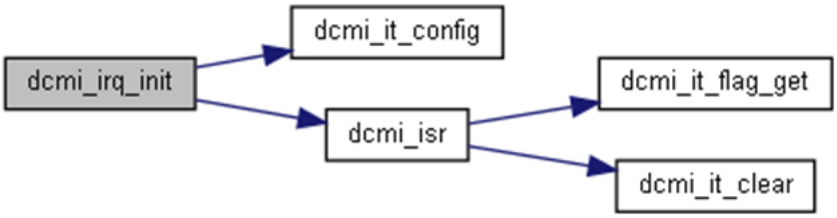
参数:

newstate	interrupt status This parameter can be one of the following value ● 0:disable ● 1:enable
*(*dcmi_subisr)()*	interrupt service runtine subfunction

返回:

none

函数调用图:



1.11.2.20 void DCMI_IRQHandler (void)

DCMI_IRQHandler.

参数:

none	

返回:

none

1.11.2.21 void dcmi_isr (void)

DCMI interrupt service routines.

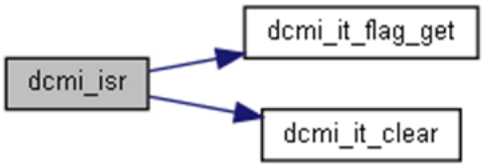
参数:

none	

返回:

none

函数调用图:



函数的调用关系图:

1.11.2.22 void dcmi_it_clear (uint16_t *it_clear*)

clear the dcmi interrupt flag(dcmi_intraw register).

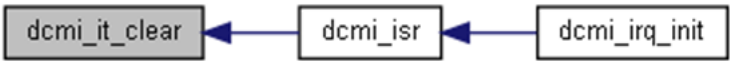
参数:

it_clear	this parameter can be any combination of the following values: DCMI_FRAMEEND: frame capture complete interrupt DCMI_OVERFLOW: overflow interrupt DCMI_ESCERR : synchronization error interrupt DCMI_VSYNC : vSYNC interrupt DCMI_HSYNC : line interrupt DCMI_WRERR : data transfer error interrupt DCMI_INTALL : clear all interrupt

返回:

none

函数的调用关系图:



1.11.2.23 void dcmi_it_config (uint16_t *dcmi_it*, FUNC_E *newstate*)

enable or disable the DCMI interrupts.

参数:

dcmi_it	this parameter can be any combination of the following values: DCMI_FRAMEEND: frame capture complete interrupt DCMI_OVERFLOW: overflow interrupt DCMI_ESCERR : synchronization error interrupt DCMI_VSYNC : vSYNC interrupt DCMI_HSYNC : line interrupt DCMI_WRERR : data transfer error interrupt DCMI_INTALL : enable or disable all interrupt
newstate	This parameter can be one of the following value ● 0:disable ● 1:enable

返回:

none

函数的调用关系图:

1.11.2.24 FLAG_E dcmi_it_flag_get (uint16_t *it_flag*)

get the DCMI it flag status.

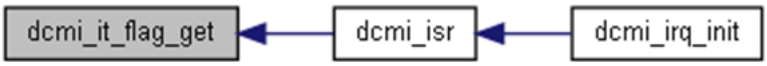
参数:

it_flag	this parameter can be one of the following values: DCMI_STATUS register RO ● DCMI_STATUS_HSYNCDIS ● DCMI_STATUS_VSYNCDIS ● DCMI_FIFO_NONEMPTY DCMI_INTRAW register RO ● DCMI_FLAG_IRFRAMEEND ● DCMI_FLAG_IROVERFLOW ● DCMI_FLAG_IRESCERR ● DCMI_FLAG_IRVSYNC ● DCMI_FLAG_IRHSYNC ● DCMI_FLAG_IRWRERR DCMI_INTEN register ● DCMI_FLAG_IEFRAMEEND ● DCMI_FLAG_IEOVERFLOW ● DCMI_FLAG_IEESCERR ● DCMI_FLAG_IEVSYNC ● DCMI_FLAG_IEHSYNC ● DCMI_FLAG_IEWRERR DCMI_INTMASKED register RO ● DCMI_FLAG_IMFRAMEEND ● DCMI_FLAG_IMOVERFLOW ● DCMI_FLAG_IMESCERR ● DCMI_FLAG_IMVSYNC ● DCMI_FLAG_IMHSYNC ● DCMI_FLAG_IMWRERR

返回:

FLAG_E bit_status 0: reset; 1: set.

函数的调用关系图:



1.11.2.25 void dcmi_jpeg_cmd (FUNC_E *newstate*)

enable or disable jpeg mode.

参数:

newstate	jpeg mode command This parameter can be one of the following value: ● 0:disable. ● 1:enable.

返回:

none

函数的调用关系图:

1.11.2.26 void dcmi_mst_cmd (FUNC_E *newstate*)

enable or disable DCMI data output automatic function

参数:

newstate	output automatic command. This parameter can be one of the following value: ● 0:disable. ● 1:enable.

返回:

none

函数的调用关系图:

1.11.2.27 void* dcmi_mstadr_read (void)

read the data output address

参数:

none	

返回:

DCMI->mstadr read output address

1.11.2.28 void dcmi_mstadr_write (void * *dcmi_mstadr*)

write the data output address

参数:

dcmi_mstadr	write output address

返回:

none

函数的调用关系图:

1.11.2.29 void dcmi_pclkpol_config (POL_E *newstate*)

config pixel clock polarity

参数:

newstate	pixel clock polarity This parameter can be one of the following value: ● 0:falling edge. ● 1:rising edge.

返回:

none

函数的调用关系图:

1.11.2.30 void dcmi_reset (void)

DCMI reset

参数:

none	

返回:

none

1.11.2.31 void dcmi_sm_cmd (FUNC_E *newstate*)

enable capture mode, single frame or continous frame

参数:

newstate	capture mode command This parameter can be one of the following value: ● 0: continous capture. ● 1: single capture.

返回:

none

函数的调用关系图:

1.11.2.32 void dcmi_type_init (void)

DCMI initialize，this function can be initialized by the user according to requirements. use different macro to switch different forwarding methods or different vedio format, for example:rgb,jpeg. following macro can be used: DCMI_DMA_EMC, DCMI_DIR_EMC. DCMI_JPEG_MODE.

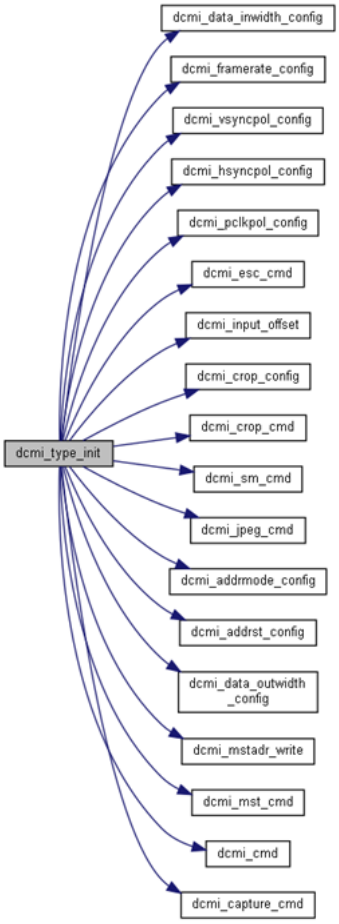
参数:

none	

返回:

none

函数调用图:



1.11.2.33 DCMI_T* dcmi_type_read (void)

read the DCMI registers values

参数:

none	

返回:

DCMI structure pointer point to DCMI_T

1.11.2.34 void dcmi_vsyncpol_config (POL_E *newstate*)

config vsync polarity

参数:

newstate	vsync polarity This parameter can be one of the following value: ● 0:low. ● 1:high.

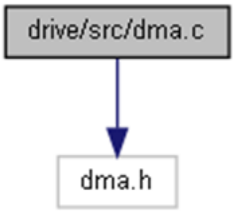
返回:

none

函数的调用关系图:

1.12 DMA接口

DMA driver source file
#include "dma.h"
dma.c 的引用(Include)关系图:



1.12.1 函数

- void **DMAC0CH0_IRQHandler** (void)
  DMAC0 CH0 interrupt handling.
- void **DMAC0CH1_IRQHandler** (void)
  DMAC0 CH1 interrupt handling.
- void **DMAC0CH2_IRQHandler** (void)
  DMAC0 CH2 interrupt handling.
- void **DMAC0CH3_IRQHandler** (void)
  DMAC0 CH3 interrupt handling.
- void **DMAC0CH4_IRQHandler** (void)
  DMAC0 CH4 interrupt handling.
- void **DMAC0CH5_IRQHandler** (void)
  DMAC0 CH5 interrupt handling.
- void **DMAC0CH6_IRQHandler** (void)
  DMAC0 CH6 interrupt handling.
- void **DMAC0CH7_IRQHandler** (void)
  DMAC0 CH7 interrupt handling.
- void **DMAC1CH0_IRQHandler** (void)
  DMAC1 CH0 interrupt handling.
- void **DMAC1CH1_IRQHandler** (void)
  DMAC1 CH1 interrupt handling.
- void **DMAC1CH2_IRQHandler** (void)
  DMAC1 CH2 interrupt handling.
- void **DMAC1CH3_IRQHandler** (void)
  DMAC1 CH3 interrupt handling.
- void **DMAC1CH4_IRQHandler** (void)
  DMAC1 CH4 interrupt handling.

- void **DMAC1CH5_IRQHandler** (void)
  DMAC1 CH5 interrupt handling.
- void **DMAC1CH6_IRQHandler** (void)
  DMAC1 CH6 interrupt handling.
- void **DMAC1CH7_IRQHandler** (void)
  DMAC1 CH7 interrupt handling.
- void **dma_init** (DMA_T *DMAx, BOOL newstate)
  DMA initial
- void **dma_tt_fc_inc_config** (DMA_T *DMAx, uint8_t channel_index, uint32_t tt_fc, uint32_t src_inc, uint32_t dst_inc)
  DMA tt_fc and src&dst inc config
- void **dma_msize_config** (DMA_T *DMAx, uint8_t channel_index, uint32_t src_msize, uint32_t dst_mize)
  DMA src&dst msize config
- void **dma_tr_width_config** (DMA_T *DMAx, uint8_t channel_index, uint32_t src_tr_width, uint32_t dst_tr_width)
  DMA src&dst width config
- void **dma_block_ts_config** (DMA_T *DMAx, uint8_t channel_index, uint32_t block_ts)
  DMA src&dst block ts config
- void **dma_hs_sel_config** (DMA_T *DMAx, uint8_t channel_index, uint32_t src_hs_sel, uint32_t dst_hs_sel)
  DMA src&dst hs_sel config
- void **dma_per_config** (DMA_T *DMAx, uint8_t channel_index, uint32_t src_per, uint32_t dst_per)
  DMA src&dst per config
- void **dma_dma_en_config** (DMA_T *DMAx, uint8_t dma_en)
  DMA en config
- void **dma_irq_init** (DMA_T *DMAx, uint8_t channel_index, uint8_t irq_enable, void(*pfunc_tc)())
  DMA irq init
- void **dma_irq_transfer** (DMA_T *DMAx, uint8_t channel_index, uint32_t src_addr, uint32_t dest_addr, uint16_t length)
  DMA irq transfer
- void **dma_poll_transfer** (DMA_T *DMAx, uint8_t channel_index, uint32_t src_addr, uint32_t dest_addr, uint16_t length)
  DMA poll transfer
- void **dma_ch_susp_transfer** (DMA_T *DMAx, uint8_t channel_index, uint8_t ch_susp)
  DMA ch susp channel transfer

- void **dma_ch_prior_set** (DMA_T *DMAx, uint8_t channel_index, uint8_t ch_prior)

  DMA ch prior set

1.12.2 函数说明

1.12.2.1 void dma_block_ts_config (DMA_T * *DMAx*, uint8_t *channel_index*, uint32_t *block_ts*)

DMA src&dst block ts config

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set dma channel
block_ts	Block data lenth，the unit is sorce size

返回:

none

1.12.2.2 void dma_ch_prior_set (DMA_T * *DMAx*, uint8_t *channel_index*, uint8_t *ch_prior*)

DMA ch prior set

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set dma channel
ch_prior	channel priority（0 is lowest）

返回:

none

1.12.2.3 void dma_ch_susp_transfer (DMA_T * *DMAx*, uint8_t *channel_index*, uint8_t *ch_susp*)

DMA ch susp channel transfer

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set dma channel
ch_susp	1，suspend transfer；0,normal transer

返回:

none

1.12.2.4 void dma_dma_en_config (DMA_T * *DMAx*, uint8_t *dma_en*)

DMA en config

参数:

**DMAx*	pointer to DMA_T structure
dma_en	1,dma enable; 0,dma disable

返回:

none

1.12.2.5 void dma_hs_sel_config (DMA_T * *DMAx*, uint8_t *channel_index*, uint32_t *src_hs_sel*, uint32_t *dst_hs_sel*)

DMA src&dst hs_sel config

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set dma channel
src_hs_sel	source handshake signal select
dst_hs_sel	target handshake signal select

返回:

none

1.12.2.6 void dma_init (DMA_T * *DMAx*, BOOL *newstate*)

DMA initial

参数:

**DMAx*	pointer to DMA_T structure
newstate	DMA_ENABLE / DMA_DISABLE

返回:

none

1.12.2.7 void dma_irq_init (DMA_T * *DMAx*, uint8_t *channel_index*, uint8_t *irq_enable*, void(*)() *pfunc_tc*)

DMA irq init

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set DMA channel
irq_enable	int enable & disable
*(*pfunc_tc)()*	pointer to pfunc_tc

返回:

none

1.12.2.8 void dma_irq_transfer (DMA_T * *DMAx*, uint8_t *channel_index*, uint32_t *src_addr*, uint32_t *dest_addr*, uint16_t *length*)

DMA irq transfer

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set DMA channel
src_addr	source address
dest_addr	target address
length	transfer length

返回:

none

1.12.2.9 void dma_msize_config (DMA_T * *DMAx*, uint8_t *channel_index*, uint32_t *src_msize*, uint32_t *dst_mize*)

DMA src&dst msize config

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set DMA channel
src_msize	source transmission burst length
dst_mize	target transmission burst length

返回:

none

1.12.2.10 void dma_per_config (DMA_T * *DMAx*, uint8_t *channel_index*, uint32_t *src_per*, uint32_t *dst_per*)

DMA src&dst per config

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set DMA channel
src_per	source handshake signal number
dst_per	target handshake signal number

返回:

none

1.12.2.11 void dma_poll_transfer (DMA_T * *DMAx*, uint8_t *channel_index*, uint32_t *src_addr*, uint32_t *dest_addr*, uint16_t *length*)

DMA poll transfer

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set DMA channel
src_addr	source address
dest_addr	target address
length	transfer length

返回:

none

1.12.2.12 void dma_tr_width_config (DMA_T * *DMAx*, uint8_t *channel_index*, uint32_t *src_tr_width*, uint32_t *dst_tr_width*)

DMA src&dst width config

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set DMA channel
src_tr_width	source transmission burst length
dst_tr_width	target transmission burst length

返回:

none

1.12.2.13 void dma_tt_fc_inc_config (DMA_T * *DMAx*, uint8_t *channel_index*, uint32_t *tt_fc*, uint32_t *src_inc*, uint32_t *dst_inc*)

DMA tt_fc and src&dst inc config

参数:

**DMAx*	pointer to DMA_T structure
channel_index	set DMA channel
tt_fc	transmission type and flow control
src_inc	source address control
dst_inc	target address control

返回:

none

1.12.2.14 void DMAC0CH0_IRQHandler (void)

DMAC0 CH0 interrupt handling.

参数:

none	

返回:

none

1.12.2.15 void DMAC0CH1_IRQHandler (void)

DMAC0 CH1 interrupt handling.

参数:

none	

返回:

none

1.12.2.16 void DMAC0CH2_IRQHandler (void)

DMAC0 CH2 interrupt handling.

参数:

none	

返回:

none

1.12.2.17 void DMAC0CH3_IRQHandler (void)

DMAC0 CH3 interrupt handling.

参数:

none	

返回:

none

1.12.2.18 void DMAC0CH4_IRQHandler (void)

DMAC0 CH4 interrupt handling.

参数:

none	

返回:

none

1.12.2.19 void DMAC0CH5_IRQHandler (void)

DMAC0 CH5 interrupt handling.

参数:

none	

返回:

none

1.12.2.20 void DMAC0CH6_IRQHandler (void)

DMAC0 CH6 interrupt handling.

参数:

none	

返回:

none

1.12.2.21 void DMAC0CH7_IRQHandler (void)

DMAC0 CH7 interrupt handling.

参数:

none	

返回:

none

1.12.2.22 void DMAC1CH0_IRQHandler (void)

DMAC1 CH0 interrupt handling.

参数:

none	

返回:

none

1.12.2.23 void DMAC1CH1_IRQHandler (void)

DMAC1 CH1 interrupt handling.

参数:

none	

返回:

none

1.12.2.24 void DMAC1CH2_IRQHandler (void)

DMAC1 CH2 interrupt handling.

参数:

none	

返回:

none

1.12.2.25 void DMAC1CH3_IRQHandler (void)

DMAC1 CH3 interrupt handling.

参数:

none	

返回:

none

1.12.2.26 void DMAC1CH4_IRQHandler (void)

DMAC1 CH4 interrupt handling.

参数:

none	

返回:

none

1.12.2.27 void DMAC1CH5_IRQHandler (void)

DMAC1 CH5 interrupt handling.

参数:

none	

返回:

none

1.12.2.28 void DMAC1CH6_IRQHandler (void)

DMAC1 CH6 interrupt handling.

参数:

none	

返回:

none

1.12.2.29 void DMAC1CH7_IRQHandler (void)

DMAC1 CH7 interrupt handling.

参数:

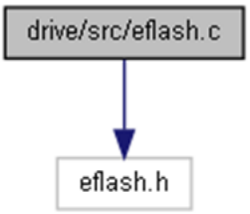
none	

返回:

none

1.13 EFLASH接口

eflash driver source file
#include "eflash.h"
eflash.c 的引用(Include)关系图:



1.13.1 函数

- void **eflash_wait_idle** (void)
  Wait for flash to idle.
- void **eflash_sec_unlock** (void)
  Unlock the access of FLASH security operation control register.
- void **eflash_sec_lock** (void)
  Locks the access of FLASH security operation control register.
- void **eflash_set_time** (uint8_t wait_time, uint8_t efc_freq)
  Set the program time of eflash.
- void **eflash_write_byte** (uint32_t addr, uint8_t value)
  Programs a byte (8-bit) at a specified address.
- void **eflash_write_halfword** (uint32_t addr, uint16_t value)
  Programs a half word (16-bit) at a specified address.
- void **eflash_write_word** (uint32_t addr, uint32_t value)
  Programs a word (32-bit) at a specified address.
- void **eflash_write_data** (uint32_t addr, uint32_t *write_data, uint32_t data_len)
  Continuous programming of multiple words (32 bits) in one-time programming mode.
- void **eflash_read_data** (uint32_t addr, uint32_t *read_data, uint32_t data_len)
  Read multiple words (32 bits) continuously in one-time programming mode.
- void **eflash_continue_write_word** (uint32_t addr, uint32_t *write_data)
  Continuously program 128 words (32 bits) in the specified address space.
- void **eflash_continue_write_data** (uint32_t addr, uint32_t *write_data, uint32_t data_len)
  Continuously program 512 words (32 bits) in the specified address space.
- void **eflash_page_erase** (uint8_t page)
  Erases a specified FLASH Page.
- void **eflash_nvr_addr_erase** (uint32_t addr)
  Erases a specified FLASH_NVR addr.

- void **eflash_multiple_pages_erase** (uint8_t start_page, uint8_t end_page)
  Erase multiple pages continuously.
- void **eflash_chip_erase** (void)
  Erases all flash page.
- FLAG_E **eflash_getflash_status** (uint32_t flash_flag)
  Checks whether the specified FLASH flag is set or not.
- uint8_t **eflash_rewrite_word** (uint32_t addr, uint32_t value)
  Flash writes back a word.
- void **eflash_set_keyctrl** (uint8_t sha_icv, uint8_t aes_key2_position, uint8_t aes_key1_position)
  eflash_set_keyctrl
- FLAG_E **eflash_get_keystatus** (uint32_t status)
  eflash_get_keystatus
- uint16_t **do_crc** (uint32_t addr, uint32_t len, uint16_t crc_init)
  do_crc
- uint16_t **check_crc_sn** (void)
  check_crc_sn
- uint16_t **read_sequence** (uint8_t *buff)
  read_sequence
- uint16_t **read_UID** (uint8_t *buff)
  read_UID

1.13.2 函数说明

1.13.2.1 uint16_t check_crc_sn (void)

check_crc_sn

注解:

read crc for unique SN（16bytes）

参数:

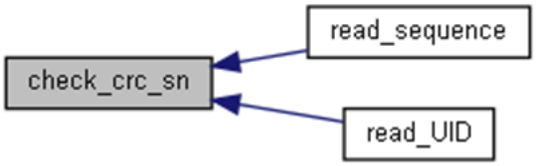
none	

返回:

CRC OK 0xFFFF CRC not write other CRC fail

函数调用图:

函数的调用关系图:



1.13.2.2 uint16_t do_crc (uint32_t *addr*, uint32_t *len*, uint16_t *crc_init*)

do_crc

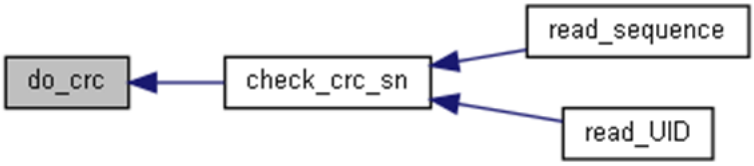
参数:

uint32_t	addr start address
uint32_t	len data length
uint16_t	crc_init initial value for CRC16-CCITT

返回:

CRC value

函数的调用关系图:



1.13.2.3 void eflash_chip_erase (void)

Erases all flash page.

注解:

If both erase and program operations are requested, Erase operation needs to be performed before programming.

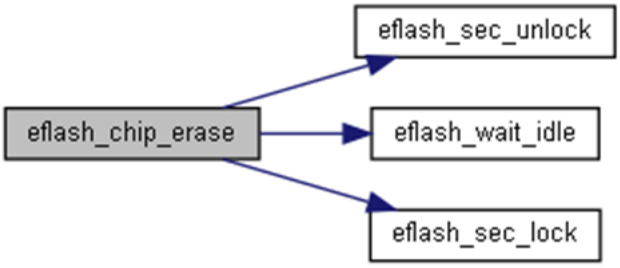
参数:

none	

返回:

none

函数调用图:



1.13.2.4 void eflash_continue_write_data (uint32_t *addr*, uint32_t * *write_data*, uint32_t *data_len*)

Continuously program 512 words (32 bits) in the specified address space.

注解:

If both erase and program operations are requested, erase operation needs to be performed before programming.

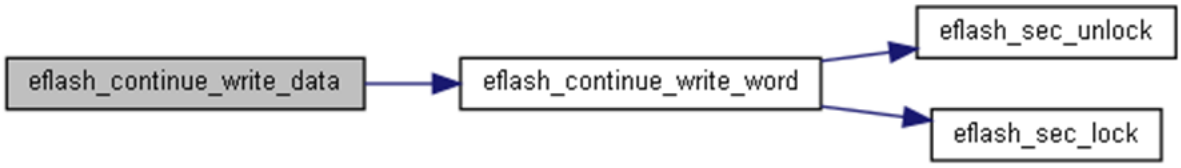
参数:

addr	start address to be continuously programmed. This parameter can be any address in main flash zone.
**write_data*	specify the data to be programmed continuously.
data_len	The length of this operation

返回:

none

函数调用图:



1.13.2.5 void eflash_continue_write_word (uint32_t *addr*, uint32_t * *write_data*)

Continuously program 128 words (32 bits) in the specified address space.

注解:

If both erase and program operations are requested, erase operation needs to be performed before programming.

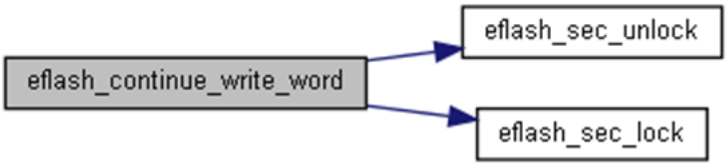
参数:

addr	start address to be continuously programmed. This parameter can be any address in main flash zone.
**write_data*	specify the data to be programmed continuously.

返回:

none

函数调用图:



函数的调用关系图:

1.13.2.6 FLAG_E eflash_get_keystatus (uint32_t *status*)

eflash_get_keystatus

参数:

status	Read status from otp This parameter can be one of the following values: ● FLASH_FLAG_SHAICV SHA_ICV successfully read from otp area flag. ● FLASH_FLAG_AESKEY1 AES_KEY1 successfully read from otp area flag. ● FLASH_FLAG_AESKEY2 AES_KEY2 successfully read from otp area flag.

返回:

The new state of FLAG (SET or RESET).

1.13.2.7 FLAG_E eflash_getflash_status (uint32_t *flash_flag*)

Checks whether the specified FLASH flag is set or not.

参数:

flash_flag	specifies the FLASH flag to check. This parameter can be one of the following values: ● FLASH_FLAG_EOP: FLASH End of Operation flag ● FLASH_FLAG_VDDLVW: FLASH Low voltage warning flag ● FLASH_FLAG_LPAC: FLASH Power down mode flag ● FLASH_FLAG_LPSLEEP: FLASH Sleeping mode flag ● FLASH_FLAG_LPLVDD: FLASH Low voltage operation flag ● FLASH_FLAG_CPRBSY: FLASH is in the state of waiting for the next continuous programming.

返回:

The new state of FLASH_FLAG (SET or RESET).

1.13.2.8 void eflash_multiple_pages_erase (uint8_t *start_page*, uint8_t *end_page*)

Erase multiple pages continuously.

注解:

If both erase and program operations are requested, Erase operation needs to be performed before programming. The flash page size is 8k.

参数:

start_page	Erase the start page number continuously.
end_page	Erase the end page number continuously.

返回:

none

函数调用图:

1.13.2.9 void eflash_nvr_addr_erase (uint32_t *addr*)

Erases a specified FLASH_NVR addr.

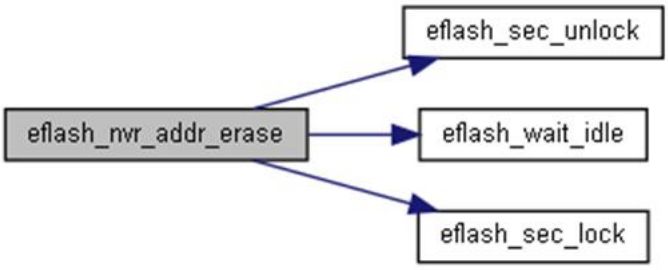
参数:

addr	Page number to be erased.

返回:

none

函数调用图:



函数的调用关系图:

1.13.2.10 void eflash_page_erase (uint8_t *page*)

Erases a specified FLASH Page.

注解:

If both erase and program operations are requested, Erase operation needs to be performed before programming. The flash page size is 8k.

For UM32F4xx devices this parameter can be a value between 0 and 64.

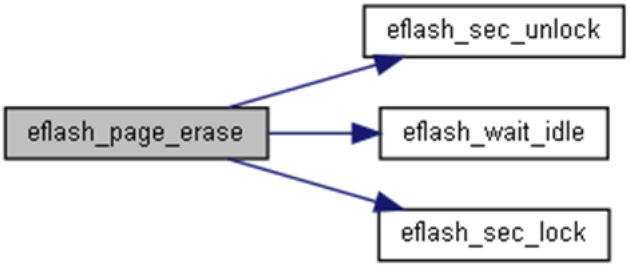
参数:

page	Page number to be erased.

返回:

none

函数调用图:



函数的调用关系图:

1.13.2.11 void eflash_read_data (uint32_t *addr*, uint32_t * *read_data*, uint32_t *data_len*)

Read multiple words (32 bits) continuously in one-time programming mode.

参数:

addr	start address to be continuously read. This parameter can be any address in main flash zone.
**read_data*	specify the data to be read continuously.
data_len	specify the length to be read continuously.

返回:

none

1.13.2.12 uint8_t eflash_rewrite_word (uint32_t *addr*, uint32_t *value*)

Flash writes back a word.

注解:

If both erase and program operations are requested, Erase operation needs to be performed before programming.

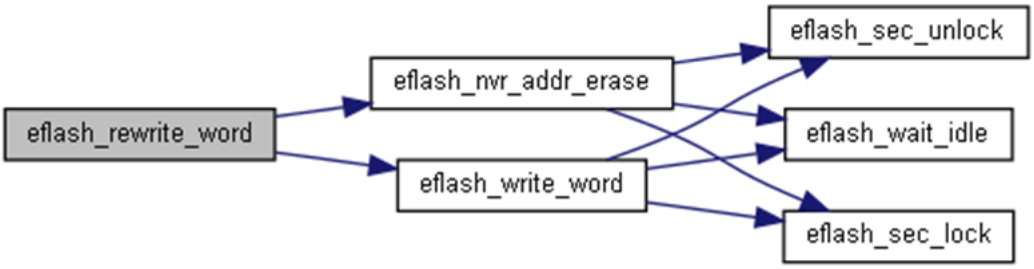
参数:

addr	Address to be written back.
value	Data to be written back.

返回:

1

函数调用图:



1.13.2.13 void eflash_sec_lock (void)

Locks the access of FLASH security operation control register.

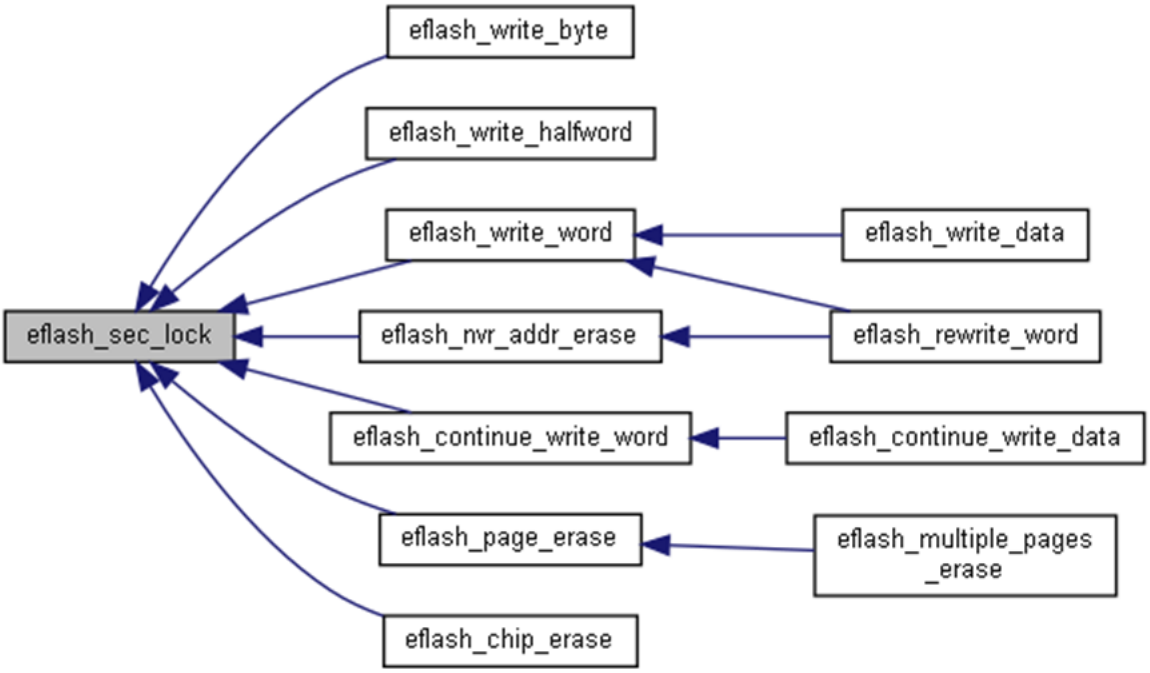
参数:

none	

返回:

none

函数的调用关系图:



1.13.2.14 void eflash_sec_unlock (void)

Unlock the access of FLASH security operation control register.

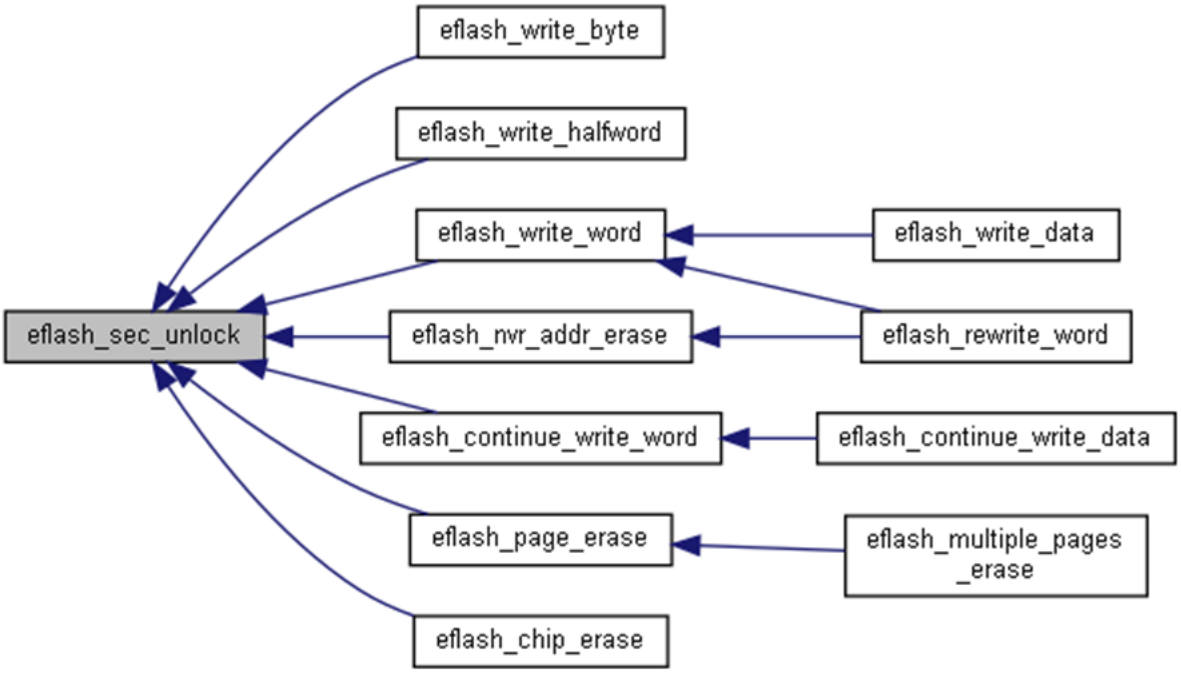
参数:

none	

返回:

none

函数的调用关系图:



1.13.2.15 void eflash_set_keyctrl (uint8_t *sha_icv*, uint8_t *aes_key2_position*, uint8_t *aes_key1_position*)

eflash_set_keyctrl

参数:

sha_icv	location of SHA_ICV
aes_key2_position	location to read AES_KEY2
aes_key1_position	location to read AES_KEY1

返回:

note

函数调用图:

1.13.2.16 void eflash_set_time (uint8_t *wait_time*, uint8_t *efc_freq*)

Set the program time of eflash.

参数:

wait_time	the time of the flash read wait
efc_freq	the frequency of the flash[Computing formula:efc_freq=Fsclk/1000000]

返回:

none

1.13.2.17 void eflash_wait_idle (void)

Wait for flash to idle.

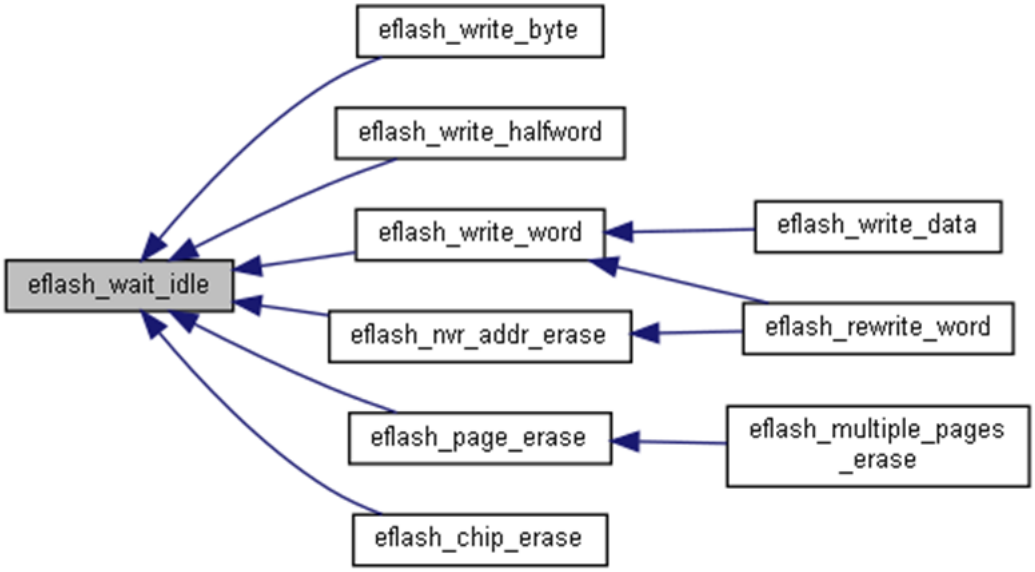
参数:

none	

返回:

none

函数的调用关系图:



1.13.2.18 void eflash_write_byte (uint32_t *addr*, uint8_t *value*)

Programs a byte (8-bit) at a specified address.

注解:

If both erase and program operations are requested, erase operation needs to be performed before programming.

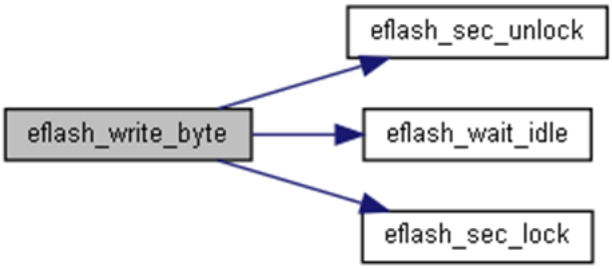
参数:

addr	specifies the address to be programmed. This parameter can be any address in main flash zone.
value	specifies the data to be programmed.

返回:

none

函数调用图:



1.13.2.19 void eflash_write_data (uint32_t *addr*, uint32_t * *write_data*, uint32_t *data_len*)

Continuous programming of multiple words (32 bits) in one-time programming mode.

注解:

If both erase and program operations are requested, erase operation needs to be performed before programming.

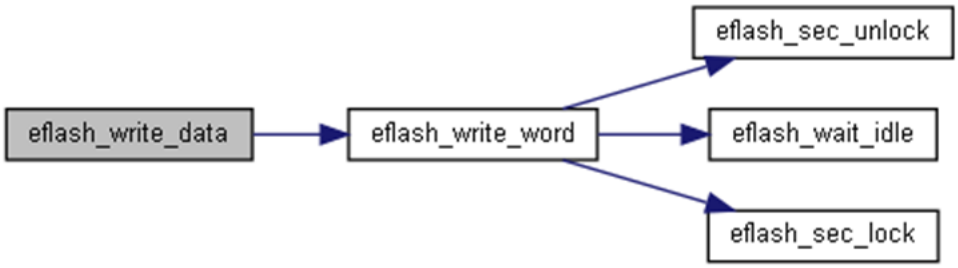
参数:

addr	start address to be continuously programmed. This parameter can be any address in main flash zone.
**write_data*	specify the data to be programmed continuously.
data_len	specify the length to be programmed continuously.

返回:

none

函数调用图:



1.13.2.20 void eflash_write_halfword (uint32_t *addr*, uint16_t *value*)

Programs a half word (16-bit) at a specified address.

注解:

If both erase and program operations are requested, erase operation needs to be performed before programming.

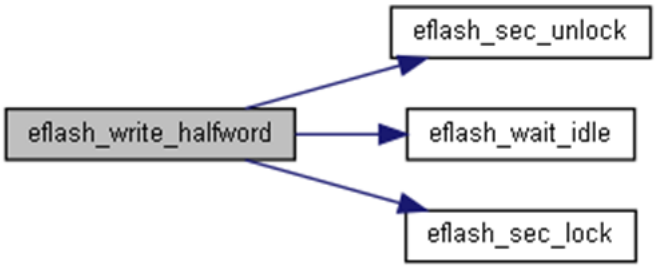
参数:

addr	specifies the address to be programmed. This parameter can be any address in main flash zone.
value	specifies the data to be programmed.

返回:

none

函数调用图:



1.13.2.21 void eflash_write_word (uint32_t *addr*, uint32_t *value*)

Programs a word (32-bit) at a specified address.

注解:

If both erase and program operations are requested, erase operation needs to be performed before programming.

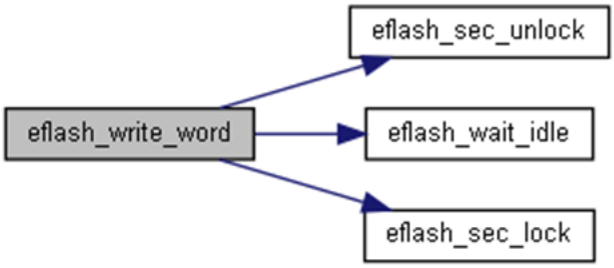
参数:

addr	specifies the address to be programmed. This parameter can be any address in main flash zone.
value	specifies the data to be programmed.

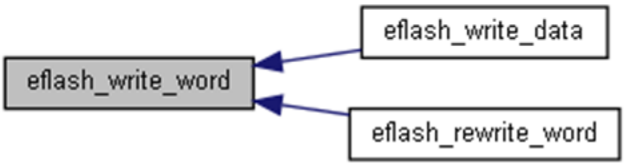
返回:

none

函数调用图:



函数的调用关系图:



1.13.2.22 uint16_t read_sequence (uint8_t * *buff*)

read_sequence

注解:

read unique SN（16 bytes）

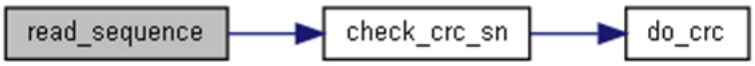
参数:

**buff*	SN buff pointer

返回:

SN CRC result

函数调用图:



1.13.2.23 uint16_t read_UID (uint8_t * *buff*)

read_UID

注解:

read unique UID（8 bytes）

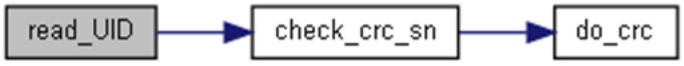
参数:

**buff*	SN buff pointer

返回:

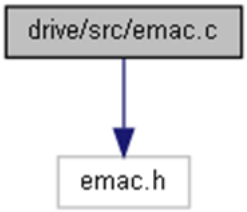
UID CRC result

函数调用图:



1.14 EMAC接口

app source file
#include "emac.h"
emac.c 的引用(Include)关系图:



1.14.1 函数

- void **EMAC_IRQHandler** (void)
  uart0 interrupt handling
- void **emac_clk_init** (EMAC_T *EMAC, BOOL newstate)
  EMAC clk init
- void **emac_set_mode** (SCU_T *SCU, uint32_t mode)
  EMAC set mode
- void **emac_sw_reset** (EMAC_T *EMAC)
  EMAC sw reset
- void **phy_dma_config** (EMAC_T *EMAC)
  phy DMA config
- void **phy_mmc_config** (EMAC_T *EMAC)
  phy MMC config
- void **write_mem** (uint32_t addr, uint32_t val)
  write memory
- uint32_t **read_mem** (uint32_t addr)
  read memory
- void **rdes_init** (uint32_t base_addr, uint32_t buf_addr, uint32_t number)
  rdes init
- void **tdes_init** (uint32_t base_addr, uint32_t buf_addr, uint32_t number)
  tdes init
- void **alternate_rdes_init** (uint32_t base_addr, uint32_t buf_addr, uint32_t number)
  alternate rdes init
- uint32_t **get_trdes_own** (uint32_t base_addr, uint32_t number, uint32_t *len)
  get trdes own
- void **set_trdes_own** (uint32_t base_addr, uint32_t number, uint32_t len)
  set trdes own

- void **tdes_long_pack_set** (uint32_t base_addr, uint32_t number, uint32_t len)
  tdes long pack set
- void * **get_trdes_buf_point** (uint32_t buf_addr, uint32_t number)
  pointer to get_trdes_buf_point
- void **trdes_buf_write** (uint32_t buf_addr, uint32_t number, uint8_t *buf, uint32_t len)
  trdes buf write
- void **emac_description_config** (EMAC_T *EMAC)
  emac description config
- void **emac_macAddrConfig** (EMAC_T *EMAC, uint32_t MacAddr, uint8_t *Addr)
  emac macAddr Config
- void **phy_mac_config** (EMAC_T *EMAC)
  phy MAC config
- void **set_dma_tpd** (EMAC_T *EMAC)
  set DMA tpd reg
- void **emac_mac_transmission_enable** (EMAC_T *EMAC)
  Enables the MAC transmission.
- void **emac_mac_transmission_disable** (EMAC_T *EMAC)
  Disables the MAC transmission.
- void **emac_mac_reception_enable** (EMAC_T *EMAC)
  Enables the MAC reception.
- void **emac_mac_reception_disable** (EMAC_T *EMAC)
  Disables the MAC reception.
- void **emac_dma_transmission_enable** (EMAC_T *EMAC)
  Enables the DMA transmission.
- void **emac_dma_transmission_disable** (EMAC_T *EMAC)
  Disables the DMA transmission.
- void **emac_dma_reception_enable** (EMAC_T *EMAC)
  Enables the DMA reception.
- void **emac_dma_reception_disable** (EMAC_T *EMAC)
  Disables the DMA reception.
- void **emac_dma_irq_init** (EMAC_T *EMAC, uint8_t irq_enable, emac_dma_irq_t irq, void(*pfunc_tc)())
  EMAC dma irq init

1.14.2 函数说明

1.14.2.1 void alternate_rdes_init (uint32_t *base_addr*, uint32_t *buf_addr*, uint32_t *number*)

alternate rdes init

参数:

base_addr	the rdes addr
buf_addr	receive buffer base address
number	rdes number

返回:

none

函数调用图:

1.14.2.2 void emac_clk_init (EMAC_T * *EMAC*, BOOL *newstate*)

EMAC clk init

参数:

**EMAC*	pointer to EMAC_T structure
newstate	EMAC_ENABLE / EMAC_DISABLE

返回:

none

1.14.2.3 void emac_description_config (EMAC_T * *EMAC*)

EMAC description config

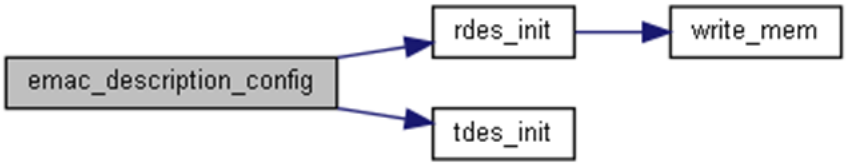
参数:

**EMAC*	pointer to EMAC_T structure

返回:

none

函数调用图:



1.14.2.4 void emac_dma_irq_init (EMAC_T * *EMAC*, uint8_t *irq_enable*, emac_dma_irq_t *irq*, void(*)() *pfunc_tc*)

EMAC dma irq init

参数:

**EMAC*	pointer to EMAC_T structure
irq_enable	enable/disable init
irq	init type
pfunc_tc	pointer to callback

返回:

none

1.14.2.5 void emac_dma_reception_disable (EMAC_T * *EMAC*)

Disables the DMA reception.

参数:

**EMAC*	pointer to a EMAC_T structure

返回值:

none

1.14.2.6 void emac_dma_reception_enable (EMAC_T * *EMAC*)

Enables the DMA reception.

参数:

**EMAC*	pointer to a EMAC_T structure

返回值:

none

1.14.2.7 void emac_dma_transmission_disable (EMAC_T * *EMAC*)

Disables the DMA transmission.

参数:

**EMAC*	pointer to a EMAC_T structure

返回值:

none

1.14.2.8 void emac_dma_transmission_enable (EMAC_T * *EMAC*)

Enables the DMA transmission.

参数:

**EMAC*	pointer to a EMAC_T structure

返回值:

none

1.14.2.9 void EMAC_IRQHandler (void)

uart0 interrupt handling

参数:

none	

返回:

none

1.14.2.10 void emac_mac_reception_disable (EMAC_T * *EMAC*)

Disables the MAC reception.

参数:

**EMAC*	pointer to a EMAC_T structure

返回:

None

1.14.2.11 void emac_mac_reception_enable (EMAC_T * *EMAC*)

Enables the MAC reception.

参数:

**EMAC*	pointer to a EMAC_T structure

返回:

None

1.14.2.12 void emac_mac_transmission_disable (EMAC_T * *EMAC*)

Disables the MAC transmission.

参数:

**EMAC*	pointer to a EMAC_T structure

返回:

None

1.14.2.13 void emac_mac_transmission_enable (EMAC_T * *EMAC*)

Enables the MAC transmission.

参数:

**EMAC*	pointer to a EMAC_T structure

返回:

None

1.14.2.14 void emac_macAddrConfig (EMAC_T * *EMAC*, uint32_t *MacAddr*, uint8_t * *Addr*)

EMAC macAddr Config

参数:

**EMAC*	pointer to EMAC_T structure
MacAddr	mac address index
**Addr*	mac address value

返回:

none

1.14.2.15 void emac_set_mode (SCU_T * *SCU*, uint32_t *mode*)

EMAC set mode

参数:

**SCU*	pointer to SCU_T structure
mode	MII & RGMII & RMII

返回:

none

1.14.2.16 void emac_sw_reset (EMAC_T * *EMAC*)

EMAC sw reset

参数:

**EMAC*	pointer to EMAC_T structure

返回:

none

1.14.2.17 void* get_trdes_buf_point (uint32_t *buf_addr*, uint32_t *number*)

pointer to get_trdes_buf_point

参数:

base_addr	trdes base address
number	trdes index

返回:

none

函数的调用关系图:

1.14.2.18 uint32_t get_trdes_own (uint32_t *base_addr*, uint32_t *number*, uint32_t * *len*)

get trdes own

参数:

base_addr	trdes base address
number	trdes index

返回:

*len data length

函数调用图:

1.14.2.19 void phy_dma_config (EMAC_T * *EMAC*)

phy dma config

参数:

**EMAC*	pointer to EMAC_T structure

返回:

none

1.14.2.20 void phy_mac_config (EMAC_T * *EMAC*)

phy mac config

参数:

**EMAC*	pointer to EMAC_T structure

返回:

none

1.14.2.21 void phy_mmc_config (EMAC_T * *EMAC*)

phy mmc config

参数:

**EMAC*	pointer to EMAC_T structure

返回:

none

1.14.2.22 void rdes_init (uint32_t *base_addr*, uint32_t *buf_addr*, uint32_t *number*)

rdes init

参数:

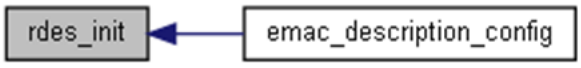
base_addr	the rdes addr
buf_addr	receive buffer base address
number	rdes number

返回:

none

函数调用图:

函数的调用关系图:



1.14.2.23 uint32_t read_mem (uint32_t *addr*)

read memory

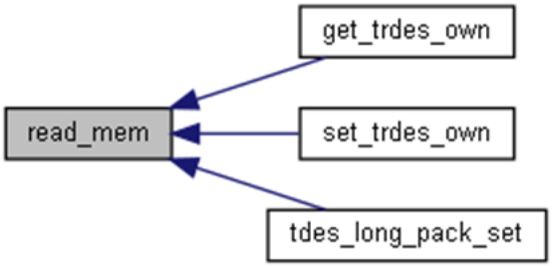
参数:

addr	read address

返回:

(*(uint32_t*)addr) value

函数的调用关系图:



1.14.2.24 void set_dma_tpd (EMAC_T * *EMAC*)

set dma tpd reg

参数:

*EMAC	pointer to EMAC_T structure

返回:

none

1.14.2.25 void set_trdes_own (uint32_t *base_addr*, uint32_t *number*, uint32_t *len*)

set trdes own

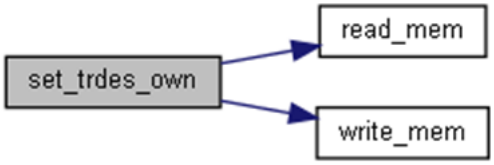
参数:

base_addr	trdes base address
number	trdes index
len	data length

返回:

none

函数调用图:



1.14.2.26 void tdes_init (uint32_t *base_addr*, uint32_t *buf_addr*, uint32_t *number*)

tdes init

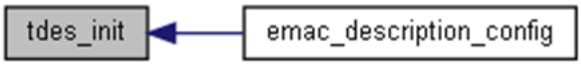
参数:

base_addr	tdes base address
buf_addr	tx buffer base address
number	tdes number

返回:

none

函数的调用关系图:



1.14.2.27 void tdes_long_pack_set (uint32_t *base_addr*, uint32_t *number*, uint32_t *len*)

tdes long pack set

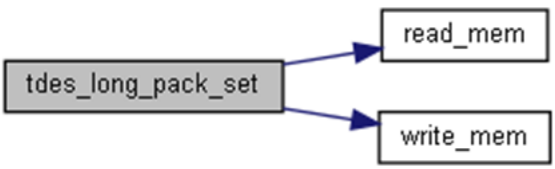
参数:

base_addr	trdes base address
number	trdes index
len	data length

返回:

none

函数调用图:



1.14.2.28 void trdes_buf_write (uint32_t *buf_addr*, uint32_t *number*, uint8_t * *buf*, uint32_t *len*)

trdes buf write

参数:

base_addr	trdes base address
number	trdes index
**buf*	pointer to buf
len	data length

返回:

none

函数调用图:

1.14.2.29 void write_mem (uint32_t *addr*, uint32_t *val*)

write memory

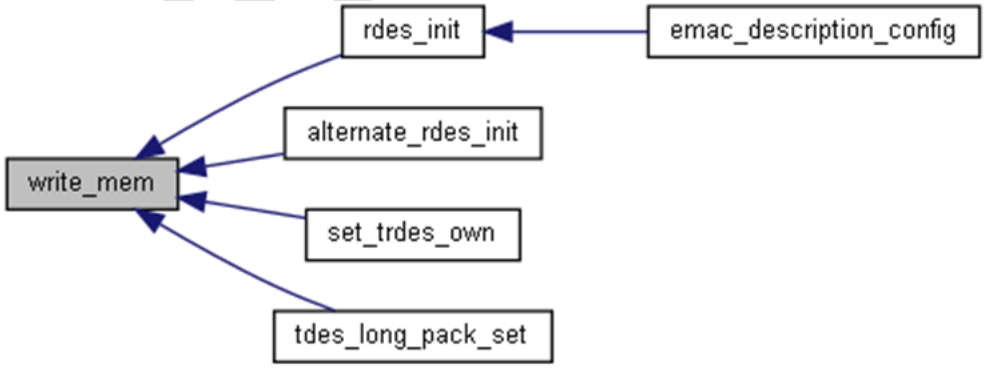
参数:

addr	write address
val	write data

返回:

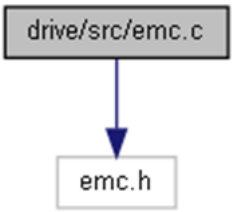
none

函数的调用关系图:



1.15 EMC接口

EMC driver source file
#include "emc.h"
emc.c 的引用(Include)关系图:



1.15.1 函数

- void **emc_clk_init** (BOOL newstate)
  enable/disable emc clock, meanwhile release/enable emc reset status
- void **emc_clk_cmd** (BOOL newstate)
  enable/disable emc clock
- void **emc_reset** (void)
  emc reset
- void **emc_deinit** (void)
  deinitializes the emc registers to default reset values
- EMC_T * **emc_type_read** (void)
  read the emc registers values
- void **emc_csn0_cmd** (BOOL newstate)
  enable csn0
- void **emc_portwidth_cmd** (EMC_WIDTH_E width)
  port data width config
- void **emc_readonly_cmd** (FUNC_E newstate)
  enable read only function
- void **emc_tft_config** (uint8_t tft_mode)
  config csn0 to tft mode
- void **emc_tftcmd_write** (uint32_t tft_cmd)
  tft mode, write this register will start tft cmd operation
- void **emc_tftdata_write** (uint32_t tft_data)
  tft mode, write emc data register
- uint32_t **emc_tftdata_read** (void)
  tft mode, read emc data register
- void **emc_seg0div_config** (EMC_SEGDIV_T *segdiv)

config the csn0 seg0 start address and size. the address unit and the smallest unit of seg0 are both 1KB. the seg0 size has been already minus 1 in this function.

- void **emc_seg0div_get** (EMC_SEGDIV_T *segdiv)
  get the csn0 seg0 start address and size.
- void **emc_waitclk_config** (EMC_WAITCLK_T *waitclk)
  config emc read write wait time.
- void **emc_type_init** (EMC_DEV_E dev_type)
  initialize the emc.

1.15.2 函数说明

1.15.2.1 void emc_clk_cmd (BOOL *newstate*)

enable/disable emc clock

参数:

newstate	clock status This parameter can be one of the following value: ● 0:disable. ● 1:enable.

返回:

none

1.15.2.2 void emc_clk_init (BOOL *newstate*)

enable/disable emc clock, meanwhile release/enable emc reset status

参数:

newstate	clock and reset status This parameter can be one of the following value: ● 0:disable. ● 1:enable.

返回:

none

1.15.2.3 void emc_csn0_cmd (BOOL *newstate*)

enable csn0

参数:

newstate	emc csn0 status This parameter can be one of the following value: ● 0:disable.

	● 1:enable.

返回:

none

函数的调用关系图:

1.15.2.4 void emc_deinit (void)

deinitializes the EMC registers to default reset values

参数:

none	

返回:

none

1.15.2.5 void emc_portwidth_cmd (EMC_WIDTH_E *width*)

set device port data width

参数:

width	emc port width This parameter can be one of the following value: ● 0: 16bit. ● 1: 08bit.

返回:

none

函数的调用关系图:

1.15.2.6 void emc_readonly_cmd (FUNC_E *newstate*)

enable read only function

参数:

newstate	read only status This parameter can be one of the following value: ● 0: both read and write. ● 1: read only.

返回:

none

函数的调用关系图:

1.15.2.7 void emc_seg0div_config (EMC_SEGDIV_T * *segdiv*)

config the csn0 seg0 start address and size. the address unit and the smallest unit of seg0 are both 1KB. the seg0 size has been already minus 1 in this function

参数:

segdiv	structure pointer point to EMC_SEGDIV_T This structure contains following member: ● segdiv_start: segment0 start address ● segdiv_size : segment0 size

返回:

none

1.15.2.8 void emc_seg0div_get (EMC_SEGDIV_T * *segdiv*)

get the csn0 seg0 start address and size.

参数:

segdiv	structure pointer point to EMC_SEGDIV_T This structure contains following member: ● segdiv_start: segment0 start address ● segdiv_size : segment0 size

返回:

none

1.15.2.9 void emc_tft_config (uint8_t *tft_mode*)

config csn0 to tft mode

参数:

tft_mode	0x00: csn0 device is tft mode.

返回:

none

函数的调用关系图:

1.15.2.10 void emc_tftcmd_write (uint32_t *tft_cmd*)

tft mode, write this register will start tft cmd operation.

参数:

tft_cmd	tft command code

返回:

none

1.15.2.11 uint32_t emc_tftdata_read (void)

tft mode, read EMC data register.

参数:

none	

返回:

EMC->DATA data value

1.15.2.12 void emc_tftdata_write (uint32_t *tft_data*)

tft mode, write EMC data register.

参数:

tft_data	data value

返回:

none

1.15.2.13 void emc_type_init (EMC_DEV_E dev_type)

initialize the EMC.

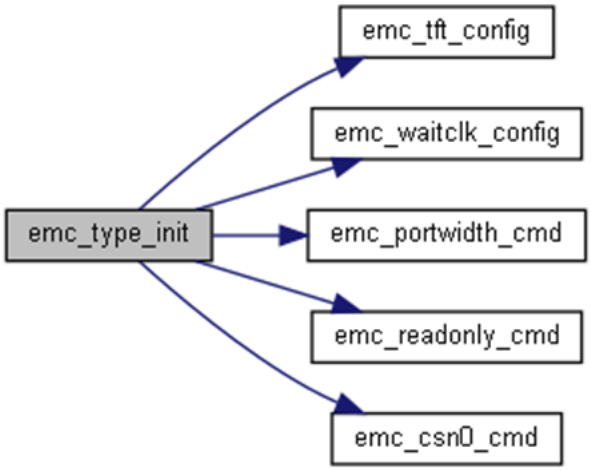
参数:

dev_type	device type This parameter can be one of the following value: ● 0: device tft. ● 1: device sram. ● 2: device norflash.

返回:

none

函数调用图:



1.15.2.14 EMC_T* emc_type_read (void)

read the EMC registers values

参数:

none	

返回:

EMC structure pointer point to EMC_T

1.15.2.15 void emc_waitclk_config (EMC_WAITCLK_T * *waitclk*)

config EMC read and write wait time.

参数:

waitclk	structure pointer point to EMC_WAITCLK_T This structure contains following member: ● wats ● wnts ● rats ● wws ● rws

返回:

none

函数的调用关系图:

1.16 GPIO接口

GPIO driver source file
#include "gpio.h"
gpio.c 的引用(Include)关系图:

1.16.1 函数

- void **EXTI0_IRQHandler** (void)
  EXTI0_IRQHandler.
- void **EXTI1_IRQHandler** (void)
  EXTI1_IRQHandler.
- void **EXTI2_IRQHandler** (void)
  EXTI2_IRQHandler.
- void **EXTI3_IRQHandler** (void)
  EXTI3_IRQHandler.
- void **EXTI4_IRQHandler** (void)
  EXTI4_IRQHandler.
- void **EXTI5TO9_IRQHandler** (void)
  EXTI5TO9_IRQHandler.
- void **EXTI10TO15_IRQHandler** (void)
  EXTI10TO15_IRQHandler.
- void **gpio_clk_init** (GPIO_T *GPIOx, uint8_t newstate)
  GPIO clock initialisation
- void **gpio_set_mux_mode** (GPIO_T *GPIOx, uint8_t pin, uint8_t mode, uint8_t af_type)
  GPIO set multiplex mode
- void **gpio_set_db** (GPIO_T *GPIOx, uint8_t pin, uint32_t db_length, uint8_t newstate)
  GPIO set db
- void **gpio_lock** (GPIO_T *GPIOx, uint8_t pin)
  GPIO lock
- void **gpio_set_im** (GPIO_T *GPIOx, uint8_t pin, uint8_t im_type)
  gpio set import type
- void **gpio_set_pull** (GPIO_T *GPIOx, uint8_t pin, uint8_t pull_type)
  GPIO set pull

- void **gpio_set_sr** (GPIO_T *GPIOx, uint8_t pin, uint8_t sr_type)
  GPIO set slew rate
- void **gpio_set_ds** (GPIO_T *GPIOx, uint8_t pin, uint8_t ds_type)
  GPIO set driving strength
- void **gpio_set_io** (GPIO_T *GPIOx, uint8_t pin)
  GPIO set io
- void **gpio_clr_io** (GPIO_T *GPIOx, uint8_t pin)
  GPIO clear io
- void **gpio_dir_write_io** (GPIO_T *GPIOx, uint8_t pin, uint8_t level_type)
  GPIO direct write io
- uint8_t **gpio_dir_read_io** (GPIO_T *GPIOx, uint8_t pin)
  GPIO direct read io
- void **gpio_set_int** (GPIO_T *GPIOx, uint8_t pin, uint8_t newstate)
  GPIO set interrupt
- void **gpio_set_int_trigger** (GPIO_T *GPIOx, uint8_t pin, uint8_t trigger_type)
  GPIO set interrupt trigger
- void **gpio_clr_int** (GPIO_T *GPIOx, uint8_t pin)
  GPIO clear interrupt
- uint8_t **gpio_get_orig_int_state** (GPIO_T *GPIOx, uint8_t pin)
  GPIO get original interrupt state
- uint8_t **gpio_get_mask_int_state** (GPIO_T *GPIOx, uint8_t pin)
  GPIO get mask interrupt state
- void **gpio_irq_init** (GPIO_T *GPIOx, uint8_t pin, uint8_t newstate, void(*pfunc)(uint8_t))
  GPIO irq initialisation

1.16.2 函数说明

1.16.2.1 void EXTI0_IRQHandler (void)

EXTI0_IRQHandler.

参数:

none	

返回:

none

函数调用图:

1.16.2.2 void EXTI10TO15_IRQHandler (void)

EXTI10TO15_IRQHandler.

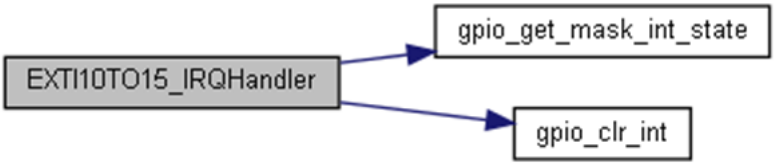
参数:

none	

返回:

none

函数调用图:



1.16.2.3 void EXTI1_IRQHandler (void)

EXTI1_IRQHandler.

参数:

none	

返回:

none

函数调用图:

1.16.2.4 void EXTI2_IRQHandler (void)

EXTI2_IRQHandler.

参数:

none	

返回:

none

函数调用图:

1.16.2.5 void EXTI3_IRQHandler (void)

EXTI3_IRQHandler.

参数:

none	

返回:

none

函数调用图:

1.16.2.6 void EXTI4_IRQHandler (void)

EXTI4_IRQHandler.

参数:

none	

返回:

none

函数调用图:

1.16.2.7 void EXTI5TO9_IRQHandler (void)

EXTI5TO9_IRQHandler.

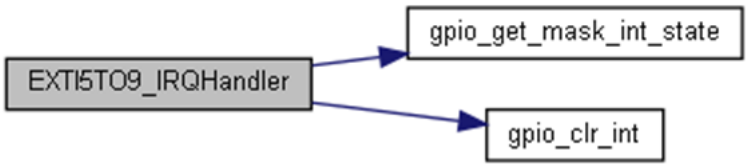
参数:

none	

返回:

none

函数调用图:



1.16.2.8 void gpio_clk_init (GPIO_T * *GPIOx*, uint8_t *newstate*)

GPIO clock initialisation

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
newstate	Clock and reset status This parameter can be one of the following values: ● GPIO_ENABLE: enable GPIOx clock and set it into work mode. ● GPIO_DISABLE：disable GPIOx clock and set it into reset mode. ● GPIO_NULL：the status is null.

返回:

none

1.16.2.9 void gpio_clr_int (GPIO_T * *GPIOx*, uint8_t *pin*)

GPIO clear interrupt

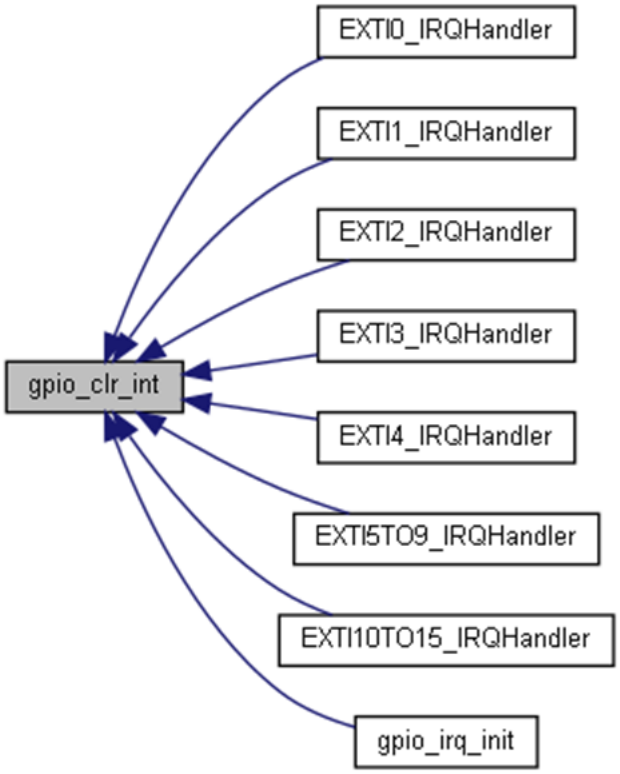
参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)

返回:

none

函数的调用关系图:



1.16.2.10 void gpio_clr_io (GPIO_T * *GPIOx*, uint8_t *pin*)

GPIO clear IO

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)

返回:

none

1.16.2.11 uint8_t gpio_dir_read_io (GPIO_T * *GPIOx*, uint8_t *pin*)

GPIO direct read IO

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)

返回:

1: high level , 0: low level

1.16.2.12 void gpio_dir_write_io (GPIO_T * *GPIOx*, uint8_t *pin*, uint8_t *level_type*)

GPIO direct write io

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
level_type	level type This parameter can be one of the following values: ● GPIO_LEVEL_LOW: Set GPIO into low level. ● GPIO_LEVEL_HIGH: Set GPIO into high level.

返回:

none

1.16.2.13 uint8_t gpio_get_mask_int_state (GPIO_T * *GPIOx*, uint8_t *pin*)

GPIO get mask interrupt state

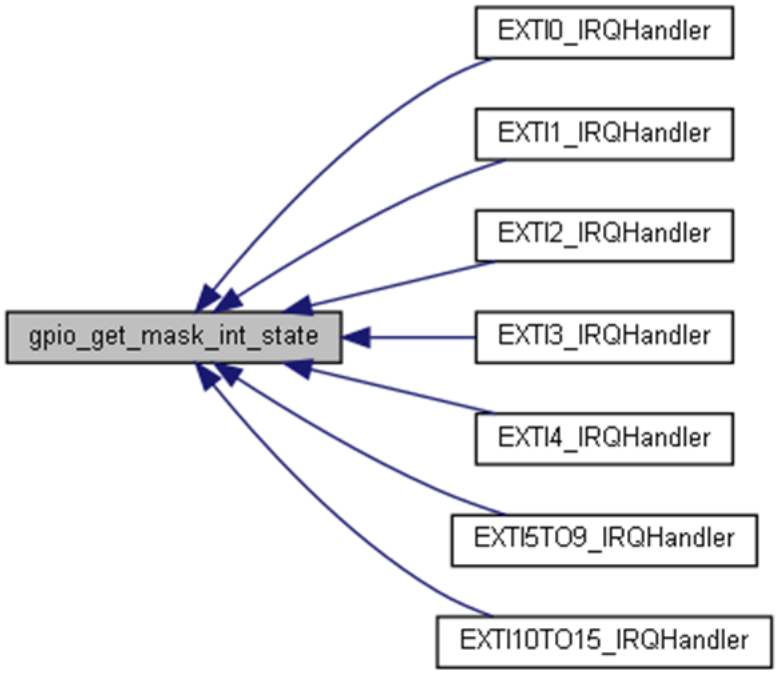
参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)

返回:

1: interrupt happen , 0: interrupt not happen

函数的调用关系图:



1.16.2.14 uint8_t gpio_get_orig_int_state (GPIO_T * *GPIOx*, uint8_t *pin*)

GPIO get original interrupt state

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)

返回:

1: interrupt happen , 0: interrupt not happen

1.16.2.15 void gpio_irq_init (GPIO_T * *GPIOx*, uint8_t *pin*, uint8_t *newstate*, void(*)(uint8_t) *pfunc*)

GPIO IRQ initialisation

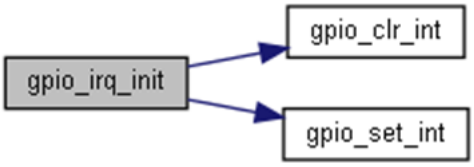
参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
newstate	Clock and reset status This parameter can be one of the following values: ● GPIO_ENABLE: enable GPIOx clock and set it into work mode. ● GPIO_DISABLE：disable GPIOx clock and set it into reset mode. ● GPIO_NULL：the status is null.
void	(*pfunc)() callback function

返回:

none

函数调用图:



1.16.2.16 void gpio_lock (GPIO_T * *GPIOx*, uint8_t *pin*)

GPIO lock

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)

返回:

none

1.16.2.17 void gpio_set_db (GPIO_T * *GPIOx*, uint8_t *pin*, uint32_t *db_length*, uint8_t *newstate*)

GPIO set db

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
db_length	db length,which is a GPIO clock cycle
newstate	Clock and reset status This parameter can be one of the following values: ● GPIO_ENABLE: enable GPIOx clock and set it into work mode. ● GPIO_DISABLE：disable GPIOx clock and set it into reset mode. ● GPIO_NULL：the status is null.

返回:

none

1.16.2.18 void gpio_set_ds (GPIO_T * *GPIOx*, uint8_t *pin*, uint8_t *ds_type*)

GPIO set driving strength

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
ds_type	driving strength type This parameter can be one of the following values: ● GPIO_DS_2MA: can drive 2mA current at most. ● GPIO_DS_4MA: can drive 4mA current at most. ● GPIO_DS_8MA: can drive 8mA current at most. ● GPIO_DS_12MA: can drive 12mA current at most.

返回:

none

1.16.2.19 void gpio_set_im (GPIO_T * *GPIOx*, uint8_t *pin*, uint8_t *im_type*)

GPIO set import type

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
im_type	import type This parameter can be one of the following values: ● GPIO_IM_CMOS: signal directly input via the cmos circuit. ● GPIO_IM_SCHMITT_TRIGGER: signal input via Schmitt trigger.

返回:

none

1.16.2.20 void gpio_set_int (GPIO_T * *GPIOx*, uint8_t *pin*, uint8_t *newstate*)

GPIO set interrupt

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
newstate	Clock and reset status This parameter can be one of the following values: ● GPIO_ENABLE: enable GPIOx clock and set it into work mode. ● GPIO_DISABLE：disable GPIOx clock and set it into reset mode. ● GPIO_NULL：the status is null.

返回:

none

函数的调用关系图:

1.16.2.21 void gpio_set_int_trigger (GPIO_T * *GPIOx*, uint8_t *pin*, uint8_t *trigger_type*)

GPIO set interrupt trigger

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
trigger_type	trigger type This parameter can be one of the following values: ● GPIO_TRIGGER_LEVEL_LOW: Set the trigger mode as low level trigger. ● GPIO_TRIGGER_LEVEL_HIGH: Set the trigger mode as high level trigger. ● GPIO_TRIGGER_EDGE_FALL: Set the trigger mode as fall

	edge trigger. ● GPIO_TRIGGER_EDGE_RISE: Set the trigger mode as rise edge trigger. ● GPIO_TRIGGER_EDGE_FALL_RISE: Set the trigger mode as bilateral edge trigger(including fall edge and rise edge).

返回:

none

1.16.2.22 void gpio_set_io (GPIO_T * *GPIOx*, uint8_t *pin*)

GPIO set io

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)

返回:

none

1.16.2.23 void gpio_set_mux_mode (GPIO_T * *GPIOx*, uint8_t *pin*, uint8_t *mode*, uint8_t *af_type*)

GPIO set multiplex mode

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
mode	functional mode This parameter can be one of the following values: ● GPIO_MODE_INPUT: set gpio mode as input. ● GPIO_MODE_OUTPUT：set gpio mode as output. ● GPIO_MODE_AF：enable the multiplexing of gpio. ● GPIO_MODE_ANALOG：set gpio mode as analog input.
af_type	types of multiplexing, which can be GPIO AF0-AF15/GPIO AF NONE.

返回:

none

1.16.2.24 void gpio_set_pull (GPIO_T * *GPIOx*, uint8_t *pin*, uint8_t *pull_type*)

GPIO set pull

参数:

**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
pull_type	pull type This parameter can be one of the following values: ● GPIO_PULL_DOWN: Set GPIO in pull-down state. ● GPIO_PULL_UP: Set GPIO in pull-up state. ● GPIO_PULL_NONE: the pull state is none.

返回:

none

1.16.2.25 void gpio_set_sr (GPIO_T * *GPIOx*, uint8_t *pin*, uint8_t *sr_type*)

GPIO set slew rate

参数:

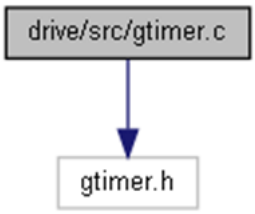
**GPIOx*	pointer to GPIO_T structure ，where x can be A,B,C,D,E,H to select the GPIO peripheral.
pin	PIN Number (pin0~15)
sr_type	slew rate type This parameter can be one of the following values: ● GPIO_SR_HIGH: Set GPIO in high slew rate. ● GPIO_SR_LOW: Set GPIO in low slew rate.

返回:

none

1.17 GTIMER接口

GTIMER driver source file.
#include "gtimer.h"
gtimer.c 的引用(Include)关系图:



1.17.1 函数

- void **TIM1_IRQHandler** (void)
  TIM1 interrupt handling
- void **TIM2_IRQHandler** (void)
  TIM2 interrupt handling
- void **TIM3_IRQHandler** (void)
  TIM3 interrupt handling
- void **TIM4_IRQHandler** (void)
  TIM4 interrupt handling
- void **TIM0_BRK_TIM8_IRQHandler** (void)
  TIM0 BREAK and TIM8 interrupt handling.
- void **TIM0_UP_TIM9_IRQHandler** (void)
  TIM0 UPDATE and TIM9 interrupt handling.
- void **TIM0_TRG_COM_TIM10_IRQHandler** (void)
  TIM0 TRIGE and COM and TIM10 interrupt handling.
- void **TIM7_BRK_TIM11_IRQHandler** (void)
  TIM7 BREAK and TIM11 interrupt handling.
- void **TIM7_UP_TIM12_IRQHandler** (void)
  TIM7 UPDATE and TIM12 interrupt handling.
- void **TIM7_TRG_COM_TIM13_IRQHandler** (void)
  TIM7 TRIGE and COM and TIM13 interrupt handling.
- uint8_t **gtim_get_status** (GTIM_T *GTIMx, uint8_t status)
  GTIMER get status
- void **gtim_clr_status** (GTIM_T *GTIMx, uint8_t status)
  GTIMER clear update status
- void **gtim_software_event** (GTIM_T *GTIMx, uint8_t events)
  GTIMER software event

- void **gtim_clock_init** (GTIM_T *GTIMx, BOOL gtim_enable_type)
  GTIMER clock initial
- void **gtim_active_source_clock_config** (GTIM_T *GTIMx, uint8_t active_source_clock)
  GTIMER active source clock config(apb_clk=HCLK=SYSPLL/2)
- void **gtim_irq_init** (GTIM_T *GTIMx, uint8_t gtim_irq_type, uint8_t gtim_enable_type, void(*pfunc)())
  GTIMER IRQ initial
- void **gtim_dma_init** (GTIM_T *GTIMx, uint8_t gtim_dma_type, uint8_t gtim_enable_type)
  GTIMER DMA initial
- void **gtim_enable_config** (GTIM_T *GTIMx, uint8_t gtim_enable_type)
  GTIMER enable config
- void **gtim_init** (GTIM_T *GTIMx, uint32_t arr, uint16_t psc, uint8_t counter_direction, uint8_t counter_alignment)
  GTIMER initial
- void **gtim_xorinput_config** (GTIM_T *GTIMx)
  GTIMER xor input config CH1、CH2、CH3 input
- void **gtim_slave_config** (GTIM_T *GTIMx, uint8_t slave_mode, uint8_t channel)
  GTIMER slave config
- void **gtim_encoder_config** (GTIM_T *GTIMx, uint8_t encode_mode)
  GTIMER encoder config
- void **gtim_capture_config** (GTIM_T *GTIMx, uint8_t input_mode, uint8_t channel)
  GTIMER encoder config
- void **gtim_pwm_config** (GTIM_T *GTIMx, uint8_t output_mode, uint8_t output_behavior, uint8_t channel)
  GTIMER pwm config
- void **gtim_set_compare** (GTIM_T *GTIMx, uint8_t channel, uint32_t compare_value)
  GTIMER set compare
- uint32_t **gtim_get_capture** (GTIM_T *GTIMx, uint8_t channel)
  GTIMER get compare
- void **gtim_dma_config** (GTIM_T *GTIMx, uint8_t length, uint8_t base_addr)
  GTIMER DMA config
- uint32_t **gtim_get_cnt** (GTIM_T *GTIMx)
  GTIMER get count value
- void **gtim_master_trgo_config** (GTIM_T *GTIMx, uint8_t trgo_type)
  GTIMER master trgo config

1.17.2 函数说明

1.17.2.1 void gtim_active_source_clock_config (GTIM_T * *GTIMx*, uint8_t *active_source_clock*)

GTIMER active source clock config(apb_clk=HCLK=SYSPLL/2)

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
active_source_clock	Gtimer source clock. This parameter can be one of the following values: ● GTIM_ACTIVE_SCLOCK_SYSPLL: SYSPLL clock. ● GTIM_ACTIVE_SCLOCK_APB: APB clock.

返回:

none

1.17.2.2 void gtim_capture_config (GTIM_T * *GTIMx*, uint8_t *input_mode*, uint8_t *channel*)

GTIMER encoder config

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
input_mode	Input mode. This parameter can be one of the following values: ● GTIM_PWMINPUT. ● GTIM_INPUT.
channel	If use pwm input mode, input channel is GTIM_CHANNEL1. This parameter can be one of the following values: ● GTIM_CHANNEL1. ● GTIM_CHANNEL2. ● GTIM_CHANNEL3. ● GTIM_CHANNEL4.

返回:

none

1.17.2.3 void gtim_clock_init (GTIM_T * *GTIMx*, BOOL *gtim_enable_type*)

GTIMER clock initial

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
gtim_enable_type	Clock status. This parameter can be one of the following values: ● GTIM_ENABLE: enable clock. ● GTIM_DISABLE: disable clock.

返回:

none

1.17.2.4 void gtim_clr_status (GTIM_T * *GTIMx*, uint8_t *status*)

GTIMER clear update status

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
status	Interrupt flags that you want to clear. This parameter can be one of the following values: ● GTIM_STATUS_ALL. ● GTIM_STATUS_UPDATE. ● GTIM_STATUS_CC1. ● GTIM_STATUS_CC2. ● GTIM_STATUS_CC3. ● GTIM_STATUS_CC4. ● GTIM_STATUS_TRI. ● GTIM_STATUS_CC1O. ● GTIM_STATUS_CC2O. ● GTIM_STATUS_CC3O. ● GTIM_STATUS_CC4O.

返回:

none

函数的调用关系图:

1.17.2.5 void gtim_dma_config (GTIM_T * *GTIMx*, uint8_t *length*, uint8_t *base_addr*)

GTIMER DMA config

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
length	Burst length, you can set 1~18.
base_addr	Start base address. This parameter can be one of the following values: ● GTIM_CR1.

	● GTIM_CR2. ● GTIM_SMCR. ● GTIM_DIER. ● GTIM_SR. ● GTIM_EGR. ● GTIM_CCMR1. ● GTIM_CCMR2. ● GTIM_CCER. ● GTIM_CNT. ● GTIM_PSC. ● GTIM_ARR. ● GTIM_CCR1. ● GTIM_CCR2. ● GTIM_CCR3. ● GTIM_CCR4. ● GTIM_DCR. ● GTIM_DMAR.

返回:

none

1.17.2.6 void gtim_dma_init (GTIM_T * *GTIMx*, uint8_t *gtim_dma_type*, uint8_t *gtim_enable_type*)

GTIMER DMA initial

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
gtim_dma_type	Gtimer dma type. This parameter can be one of the following values: ● GTIM_DMA_UPDATE. ● GTIM_DMA_CC1. ● GTIM_DMA_CC2. ● GTIM_DMA_CC3. ● GTIM_DMA_CC4. ● GTIM_DMA_TRI.
gtim_enable_type	Gtimer dma status. This parameter can be one of the following values: ● GTIM_ENABLE: enable gtimer dma. ● GTIM_DISABLE：disable gtimer dma.

返回:

none

1.17.2.7 void gtim_enable_config (GTIM_T * *GTIMx*, uint8_t *gtim_enable_type*)

GTIMER enable config

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
gtim_enable_type	Gtimer status. This parameter can be one of the following values: ● GTIM_ENABLE: enable gtimer. ● GTIM_DISABLE：disable gtimer.

返回:

none

1.17.2.8 void gtim_encoder_config (GTIM_T * *GTIMx*, uint8_t *encode_mode*)

GTIMER encoder config

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
encode_mode	Slave mode. This parameter can be one of the following values: ● GTIM_ENCODER1. ● GTIM_ENCODER2. ● GTIM_ENCODER3.

返回:

none

1.17.2.9 uint32_t gtim_get_capture (GTIM_T * *GTIMx*, uint8_t *channel*)

GTIMER get compare

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
channel	Gtimer channel. This parameter can be one of the following values: ● GTIM_CHANNEL1. ● GTIM_CHANNEL2. ● GTIM_CHANNEL3. ● GTIM_CHANNEL4.

返回:

capture value

1.17.2.10 uint32_t gtim_get_cnt (GTIM_T * *GTIMx*)

GTIMER get count value

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.

返回:

current count value

1.17.2.11 uint8_t gtim_get_status (GTIM_T * *GTIMx*, uint8_t *status*)

GTIMER get status

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
status	Interrupt flags that you want to check. ● GTIM_STATUS_UPDATE. ● GTIM_STATUS_CC1. ● GTIM_STATUS_CC2. ● GTIM_STATUS_CC3. ● GTIM_STATUS_CC4. ● GTIM_STATUS_TRI. ● GTIM_STATUS_CC1O. ● GTIM_STATUS_CC2O. ● GTIM_STATUS_CC3O. ● GTIM_STATUS_CC4O.

返回:

Corresponding status flag.

- 0: Interrupt not setting.
- 1: Interrupt setting.

1.17.2.12 void gtim_init (GTIM_T * *GTIMx*, uint32_t *arr*, uint16_t *psc*, uint8_t *counter_direction*, uint8_t *counter_alignment*)

GTIMER initial

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
arr	Automatic reloading value.
psc	Prescaler value.
counter_direction	Direction of counter. This parameter can be one of the following values:

	● GTIM_COUNTER_DIRECTION_UP. ● GTIM_COUNTER_DIRECTION_DOWN.
counter_alignment	Alignment of counter. This parameter can be one of the following values: ● GTIM_COUNTER_ALIGNMENT_EDGE. ● GTIM_COUNTER_ALIGNMENT_CENTRE1. ● GTIM_COUNTER_ALIGNMENT_CENTRE2. ● GTIM_COUNTER_ALIGNMENT_CENTRE3.

返回:

none

1.17.2.13 void gtim_irq_init (GTIM_T * *GTIMx*, uint8_t *gtim_irq_type*, uint8_t *gtim_enable_type*, void(*)() *pfunc*)

GTIMER IRQ initial

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
gtim_irq_type	Gtimer irq type. This parameter can be one of the following values: ● GTIM_IRQ_UPDATE. ● GTIM_IRQ_CC1. ● GTIM_IRQ_CC2. ● GTIM_IRQ_CC3. ● GTIM_IRQ_CC4. ● GTIM_IRQ_TRI. ● GTIM_IRQ_CC1O. ● GTIM_IRQ_CC2O. ● GTIM_IRQ_CC3O. ● GTIM_IRQ_CC4O.
gtim_enable_type	Interrupt status. This parameter can be one of the following values: ● GTIM_ENABLE: enable interrupt. ● GTIM_DISABLE：disable interrupt.
*void(*pfunc)()*	Interrupt callback function.

返回:

none

函数调用图:

1.17.2.14 void gtim_master_trgo_config (GTIM_T * *GTIMx*, uint8_t *trgo_type*)

GTIMER master trgo config

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
trgo_type	Gtim trgo type. This parameter can be one of the following values: ● GTIM_TRGO_EGRUG. ● GTIM_TRGO_CNTEN. ● GTIM_TRGO_UPDATE.

返回:

none

1.17.2.15 void gtim_pwm_config (GTIM_T * *GTIMx*, uint8_t *output_mode*, uint8_t *output_behavior*, uint8_t *channel*)

GTIMER pwm config

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
output_mode	Output mode. ● GTIM_EXTERNALEVENTS. ● GTIM_PWMOUTPUT.
output_behavior	OCxREF output behavior. ● GTIM_NOTEFFECT. ● GTIM_SETHIGH. ● GTIM_SETLOW. ● GTIM_FLIPLEVEL. ● GTIM_KEEPLOW. ● GTIM_KEEPHIGH. ● GTIM_PWM1. ● GTIM_PWM2.
channel	Output channel. This parameter can be one of the following values: ● GTIM_CHANNEL1. ● GTIM_CHANNEL2. ● GTIM_CHANNEL3. ● GTIM_CHANNEL4.

返回:

none

1.17.2.16 void gtim_set_compare (GTIM_T * *GTIMx*, uint8_t *channel*, uint32_t *compare_value*)

GTIMER set compare

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
channel	Gtimer channel. This parameter can be one of the following values: ● GTIM_CHANNEL1. ● GTIM_CHANNEL2. ● GTIM_CHANNEL3. ● GTIM_CHANNEL4.
compare_value	Compare value.

返回:

none

1.17.2.17 void gtim_slave_config (GTIM_T * *GTIMx*, uint8_t *slave_mode*, uint8_t *channel*)

GTIMER slave config

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
slave_mode	Slave mode. This parameter can be one of the following values: ● GTIM_ENCODER1. ● GTIM_ENCODER2. ● GTIM_ENCODER3. ● GTIM_RESET. ● GTIM_GATING. ● GTIM_TRIGGER. ● GTIM_EXTERNALCLOCK.
channel	Input channel. This parameter can be one of the following values: ● GTIM_CHANNEL1. ● GTIM_CHANNEL2. ● GTIM_CHANNELNULL.

返回:

none

1.17.2.18 void gtim_software_event (GTIM_T * *GTIMx*, uint8_t *events*)

GTIMER software event

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.
events	Events that you want to set. This parameter can be one of the following values: ● GTIM_STATUS_UPDATE. ● GTIM_STATUS_CC1. ● GTIM_STATUS_CC2.

	● GTIM_STATUS_CC3. ● GTIM_STATUS_CC4. ● GTIM_STATUS_TRI.

返回:

none

1.17.2.19 void gtim_xorinput_config (GTIM_T * *GTIMx*)

GTIMER xor input config CH1、CH2、CH3 input

参数:

**GTIMx*	Pointer to GTIM_T structure,where x can be 1,2,3,4,8,9,10,11 or 12 to select the TIM peripheral.

返回:

none

1.17.2.20 void TIM0_BRK_TIM8_IRQHandler (void)

TIM0 BREAK and TIM8 interrupt handling.

参数:

none	

返回:

none

1.17.2.21 void TIM0_TRG_COM_TIM10_IRQHandler (void)

TIM0 TRIGE and COM and TIM10 interrupt handling.

参数:

none	

返回:

none

1.17.2.22 void TIM0_UP_TIM9_IRQHandler (void)

TIM0 UPDATE and TIM9 interrupt handling.

参数:

none	

返回:

none

1.17.2.23 void TIM1_IRQHandler (void)

TIM1 interrupt handling

参数:

none	

返回:

none

1.17.2.24 void TIM2_IRQHandler (void)

TIM2 interrupt handling

参数:

none	

返回:

none

1.17.2.25 void TIM3_IRQHandler (void)

TIM3 interrupt handling

参数:

none	

返回:

none

1.17.2.26 void TIM4_IRQHandler (void)

TIM4 interrupt handling

参数:

none	

返回:

none

1.17.2.27 void TIM7_BRK_TIM11_IRQHandler (void)

TIM7 BREAK and TIM11 interrupt handling.

参数:

none	

返回:

none

1.17.2.28 void TIM7_TRG_COM_TIM13_IRQHandler (void)

TIM7 TRIGE and COM and TIM13 interrupt handling.

参数:

none	

返回:

none

1.17.2.29 void TIM7_UP_TIM12_IRQHandler (void)

TIM7 UPDATE and TIM12 interrupt handling.

参数:

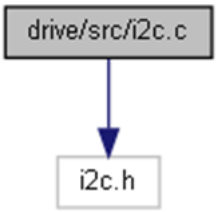
none	

返回:

none

1.18 I2C接口

I2C driver source file.
#include "i2c.h"
i2c.c 的引用(Include)关系图:



1.18.1 函数

- void **I2C0_IRQHandler** (void)
  I2C0 interrupt handling
- void **I2C1_IRQHandler** (void)
  I2C1 interrupt handling
- void **I2C2_IRQHandler** (void)
  I2C2 interrupt handling
- void **i2c_clock_init** (I2C_T *I2Cx, BOOL newstate)
  I2C clock initial
- void **i2c_clear_state** (I2C_T *I2Cx, uint32_t status)
  I2C clear state
- static void **i2c_write** (I2C_T *I2Cx, uint8_t data)
  I2C write data
- uint8_t **i2c_read** (I2C_T *I2Cx)
  I2C read data
- void **i2c_restart** (I2C_T *I2Cx)
  I2C restart
- void **i2c_stop** (I2C_T *I2Cx)
  I2C stop
- void **i2c_master_init** (I2C_T *I2Cx, uint8_t speed, uint8_t masteraddr_bit)
  I2C master initial
- uint8_t **i2c_master_send** (I2C_T *I2Cx, uint8_t slave_addr, uint8_t *data, uint8_t data_length)
  I2C master send data
- uint8_t **i2c_master_receive** (I2C_T *I2Cx, uint8_t slave_addr, uint8_t *data, uint8_t data_length)
  I2C master receive data

- void **i2c_slave_init** (I2C_T *I2Cx, uint8_t slaveaddr_bit, uint16_t slaveaddr)
  I2C slave initial
- uint8_t **i2c_slave_send** (I2C_T *I2Cx, uint8_t *data, uint32_t data_length)
  I2C slave send data
- uint8_t **i2c_slave_receive** (I2C_T *I2Cx, uint8_t *data, uint32_t data_length)
  I2C slave receive data
- uint8_t **i2c_wait_state** (I2C_T *I2Cx, uint8_t status, uint32_t wait_time)
  I2C wait state
- void **i2c_master_dma_init** (I2C_T *I2Cx, uint16_t slaveaddr)
  I2C master DMA initial
- void **i2c_slave_dma_init** (I2C_T *I2Cx)
  I2C slave DMA initial
- void **i2c_irq_init** (I2C_T *I2Cx, uint8_t mode, uint8_t state, void(*pfunc)())
  I2C IRQ initial
- uint8_t **i2c_e2prom_write** (I2C_T *I2Cx, uint8_t slave_addr, uint8_t memory_addr, uint8_t *data, uint8_t data_length)
  I2C write e2prom
- uint8_t **i2c_e2prom_read** (I2C_T *I2Cx, uint8_t slave_addr, uint8_t memory_addr, uint8_t *data, uint8_t data_length)
  I2C read e2prom

1.18.2 函数说明

1.18.2.1 void I2C0_IRQHandler (void)

I2C0 interrupt handling

参数:

none	

返回:

none

函数调用图:

1.18.2.2 void I2C1_IRQHandler (void)

I2C1 interrupt handling

参数:

none	

返回:

none

1.18.2.3 void I2C2_IRQHandler (void)

I2C2 interrupt handling

参数:

none	

返回:

none

1.18.2.4 void i2c_clear_state (I2C_T * *I2Cx*, uint32_t *status*)

I2C clear state

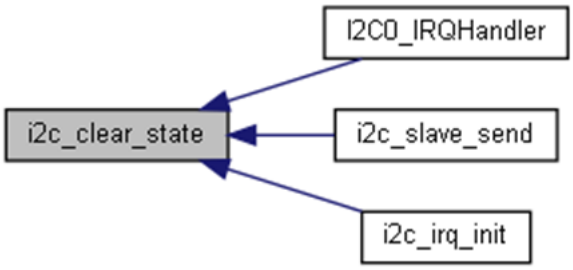
参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
status	I2C interrupt status. This parameter can be one of the following values: ● I2C_STATE_ALL. ● I2C_STATE_RX_UNDER. ● I2C_STATE_RX_OVER. ● I2C_STATE_RX_FULL. ● I2C_STATE_TX_OVER. ● I2C_STATE_TX_EMPTY. ● I2C_STATE_RD_REQ. ● I2C_STATE_TX_ABRT. ● I2C_STATE_RX_DONE. ● I2C_STATE_ACTIVITY. ● I2C_STATE_STOP_DET. ● I2C_STATE_START_DET. ● I2C_STATE_GEN_CALL.

返回:

none

函数的调用关系图:



1.18.2.5 void i2c_clock_init (I2C_T * *I2Cx*, BOOL *newstate*)

I2C clock initial

参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
newstate	Clock status. This parameter can be one of the following values: ● I2C_ENABLE: enable clock. ● I2C_DISABLE: disable clock.

返回:

none

1.18.2.6 uint8_t i2c_e2prom_read (I2C_T * *I2Cx*, uint8_t *slave_addr*, uint8_t *memory_addr*, uint8_t * *data*, uint8_t *data_length*)

I2C read e2prom

参数:

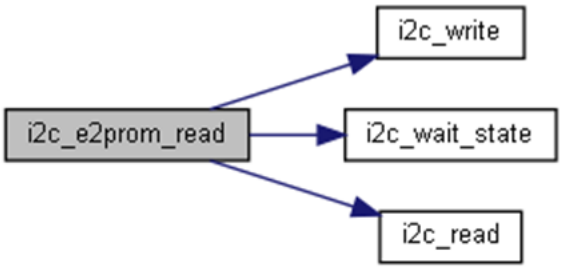
**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
slave_addr	Slave address.
memory_addr	Memory address.
**data*	Pointer to read data.
data_length	The length of data.

返回:

i2c_e2prom_read Read status flag.

- 0: Read success.
- 1: Read error.

函数调用图:



1.18.2.7 uint8_t i2c_e2prom_write (I2C_T * *I2Cx*, uint8_t *slave_addr*, uint8_t *memory_addr*, uint8_t * *data*, uint8_t *data_length*)

I2C write e2prom

参数:

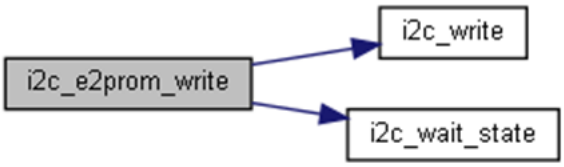
**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
slave_addr	Slave address.
memory_addr	Memnry address.
**data*	Pointer to write data.
data_length	The length of data.

返回:

i2c_e2prom_write Write status flag.

- 0: Write success.
- 1: Write error.

函数调用图:



1.18.2.8 void i2c_irq_init (I2C_T * *I2Cx*, uint8_t *mode*, uint8_t *state*, void(*)() *pfunc*)

I2C IRQ initial

参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
mode	Interrupt mode. This parameter can be one of the following values: ● I2C_RX_UNDER_INT_MODE. ● I2C_RX_OVER_INT_MODE. ● I2C_RX_FULL_INT_MODE. ● I2C_TX_OVER_INT_MODE. ● I2C_TX_EMPTY_INT_MODE. ● I2C_RD_REQ_INT_MODE. ● I2C_TX_ABRT_INT_MODE. ● I2C_RX_DONE_INT_MODE. ● I2C_ACTIVITY_INT_MODE. ● I2C_STOP_DET_INT_MODE. ● I2C_START_DET_INT_MODE. ● I2C_GEN_CALL_INT_MODE.
state	I2C irq status. This parameter can be one of the following values: ● I2C_ENABLE: enable irq. ● I2C_DISABLE: disable irq.
*void(*pfunc)()*	Interrupt callback function.

返回:

none

函数调用图:

1.18.2.9 void i2c_master_dma_init (I2C_T * *I2Cx*, uint16_t *slaveaddr*)

I2C master DMA initial

参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
slaveaddr	Slave address.

返回:

none

1.18.2.10 void i2c_master_init (I2C_T * *I2Cx*, uint8_t *speed*, uint8_t *masteraddr_bit*)

I2C master initial

参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
speed	I2C speed. This parameter can be one of the following values: ● I2C_SPEED_STANDARD. ● I2C_SPEED_FAST.
masteraddr_bit	Master address type. This parameter can be one of the following values: ● I2C_MASTERADDRESS_7BIT. ● I2C_MASTERADDRESS_10BIT.

返回:

none

1.18.2.11 uint8_t i2c_master_receive (I2C_T * *I2Cx*, uint8_t *slave_addr*, uint8_t * *data*, uint8_t *data_length*)

I2C master receive data

参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
slave_addr	I2C slave address.

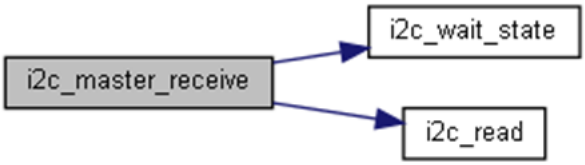
**data*	Pointer to receive data.
data_length	The length of data.

返回:

i2c_master_receive Receive status flag.

- 0: Receive success.
- 1: Receive error.

函数调用图:



1.18.2.12 uint8_t i2c_master_send (I2C_T * *I2Cx*, uint8_t *slave_addr*, uint8_t * *data*, uint8_t *data_length*)

I2C master send data

参数:

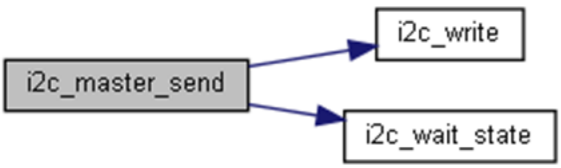
**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
slave_addr	I2C slave address.
**data*	Pointer to send data.
data_length	The length of data.

返回:

i2c_master_send Send status flag.

- 0: Send success.
- 1: Send error.

函数调用图:



1.18.2.13 uint8_t i2c_read (I2C_T * *I2Cx*)

I2C read data

参数:

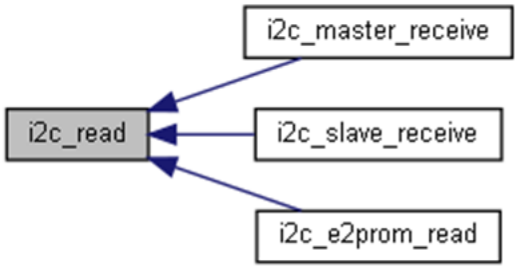
**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C

	peripheral.

返回:

receive data

函数的调用关系图:



1.18.2.14 void i2c_restart (I2C_T * *I2Cx*)

I2C restart

参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.

返回:

none

1.18.2.15 void i2c_slave_dma_init (I2C_T * *I2Cx*)

I2C slave DMA initial

参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.

返回:

none

1.18.2.16 void i2c_slave_init (I2C_T * *I2Cx*, uint8_t *slaveaddr_bit*, uint16_t *slaveaddr*)

I2C slave initial

参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
slaveaddr_bit	Slave address type. This parameter can be one of the following values:

	● I2C_SLAVEADDRESS_7BIT. ● I2C_SLAVEADDRESS_10BIT.
slaveaddr	Slave address.

返回:

none

1.18.2.17 uint8_t i2c_slave_receive (I2C_T * *I2Cx*, uint8_t * *data*, uint32_t *data_length*)

I2C slave receive data

参数:

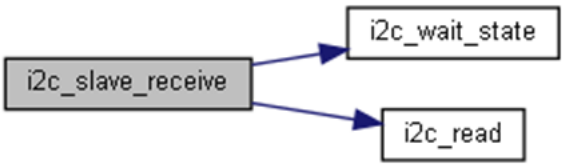
**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
**data*	Pointer to send data.
data_length	The length of data.

返回:

i2c_slave_receive Receive status flag.

- 0: Receive success.
- 1: Receive error.

函数调用图:



1.18.2.18 uint8_t i2c_slave_send (I2C_T * *I2Cx*, uint8_t * *data*, uint32_t *data_length*)

I2C slave send data

参数:

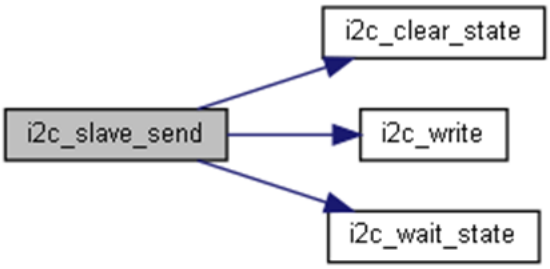
**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
**data*	Pointer to send data.
data_length	The length of data.

返回:

i2c_slave_send Send status flag.

- 0: Send success.
- 1: Send error.

函数调用图:



1.18.2.19 void i2c_stop (I2C_T * *I2Cx*)

I2C stop

参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.

返回:

none

1.18.2.20 uint8_t i2c_wait_state (I2C_T * *I2Cx*, uint8_t *status*, uint32_t *wait_time*)

I2C wait state

参数:

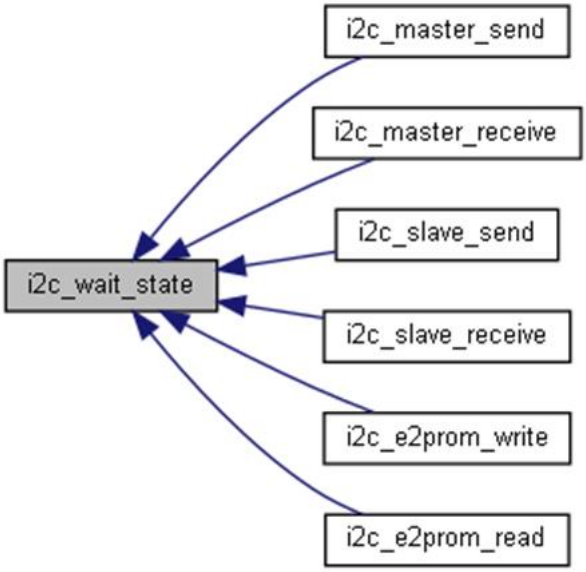
**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
status	Interrupt status. This parameter can be one of the following values: ● I2C_STATE_RX_UNDER. ● I2C_STATE_RX_OVER. ● I2C_STATE_RX_FULL. ● I2C_STATE_TX_OVER. ● I2C_STATE_TX_EMPTY. ● I2C_STATE_RD_REQ. ● I2C_STATE_TX_ABRT. ● I2C_STATE_RX_DONE. ● I2C_STATE_ACTIVITY. ● I2C_STATE_STOP_DET. ● I2C_STATE_START_DET. ● I2C_STATE_GEN_CALL.
wait_time	Set time out value.

返回:

i2c_wait_state Wait interrupt flag.

- 0: time out.
- 1: interrupt flag be set.

函数的调用关系图:



1.18.2.21 static void i2c_write (I2C_T * *I2Cx*, uint8_t *data*)[static]

I2C write data

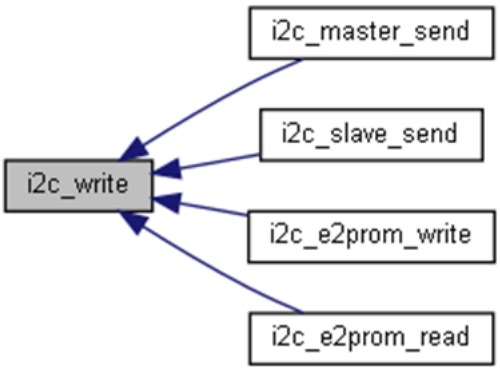
参数:

**I2Cx*	Pointer to I2C_T structure,where x can be 0,1 or 2 to select the I2C peripheral.
data	The 8bit data that need to be send.

返回:

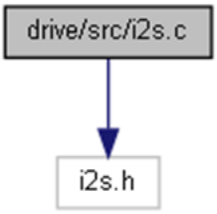
none

函数的调用关系图:



1.19 I2S

I2S driver source file
#include "i2s.h"
i2s.c 的引用(Include)关系图:



1.19.1 函数

- void **I2S0_IRQHandler** (void)
  I2S0 interrupt handling.
- void **I2S1_IRQHandler** (void)
  I2S1 interrupt handling.
- void **i2s_clk_init** (I2S_T *I2Sx, BOOL newstate)
  I2S clk init.
- void **i2s_para_config** (I2S_T *I2Sx)
  I2S para config
- void **i2s_fs_rate_init** (uint32_t fs_rate)
  I2S fs rate init
- void **i2s_enable** (I2S_T *I2Sx)
  I2S enable.
- void **i2s_disable** (I2S_T *I2Sx)
  I2S disable
- void **i2s_transmit_enable** (I2S_T *I2Sx)
  I2S transmit enable
- void **i2s_transmit_disable** (I2S_T *I2Sx)
  I2S transmit enable
- void **i2s_recive_enable** (I2S_T *I2Sx)
  I2S recive enable
- void **i2s_recive_disable** (I2S_T *I2Sx)
  I2S recive disable
- void **i2s_irq_init** (i2s_t i2s_en, uint8_t irq_enable, I2S_T *I2Sx, i2s_irq_t irq, void(*pfunc_tc)())
  I2S irq init
- void **i2s_write_buf** (I2S_T *I2Sx, uint32_t *buf, uint32_t size)

I2S write buf

- void **i2s_write_fifo** (I2S_T *I2Sx, uint32_t data)

  I2S write fifo
- uint32_t **i2s_read_fifo** (I2S_T *I2Sx)

  I2S write fifo
- uint32_t **i2s_read_CSR** (I2S_T *I2Sx)

  read CSR register
- uint32_t **i2s_read_wr_addr** (I2S_T *I2Sx)

  get I2S wr register address
- uint32_t **i2s_read_rd_addr** (I2S_T *I2Sx)

  get I2S rd register address

1.19.2 函数说明

1.19.2.1 void I2S0_IRQHandler (void)

I2S0 interrupt handling.

参数:

none	

返回:

none

1.19.2.2 void I2S1_IRQHandler (void)

I2S1 interrupt handling.

参数:

none	

返回:

none

1.19.2.3 void i2s_clk_init (I2S_T * *I2Sx*, BOOL *newstate*)

I2S clk init.

参数:

**I2Sx*	pointer to I2S_T structure
newstate	I2S_ENABLE / I2S_DISABLE

返回:

none

1.19.2.4 void i2s_disable (I2S_T * *I2Sx*)

I2S disable

参数:

**I2Sx*	pointer to I2S_T structure

返回:

none

1.19.2.5 void i2s_enable (I2S_T * *I2Sx*)

I2S enable.

参数:

**I2Sx*	pointer to I2S_T structure

返回:

none

1.19.2.6 void i2s_fs_rate_init (uint32_t *fs_rate*)

I2S fs_rate init 设置I2S的采样率

参数:

fs_rate	I2S的采样率

返回:

none

1.19.2.7 void i2s_irq_init (i2s_t *i2s_en*, uint8_t *irq_enable*, I2S_T * *I2Sx*, i2s_irq_t *irq*, void(*)() *pfunc_tc*)

i2s irq init

参数:

i2s_en	select I2S0/I2S1
irq_enable	irq enable & disable
**I2Sx*	pointer to I2S_T structure
irq	irq mode
*(*pfunc_tc)()*	pointer to pfunc_tc

返回:

none

1.19.2.8 void i2s_para_config (I2S_T * *I2Sx*)

I2S para config

参数:

**I2Sx*	pointer to I2S_T structure

返回:

none

1.19.2.9 uint32_t i2s_read_CSR (I2S_T * *I2Sx*)

read CSR register

参数:

**I2Sx*	pointer to I2S_T structure

返回:

I2S->CSR state register

1.19.2.10 uint32_t i2s_read_fifo (I2S_T * *I2Sx*)

I2S write fifo

参数:

**I2Sx*	pointer to I2S_T structure

返回:

I2S->rd 数据寄存器

1.19.2.11 uint32_t i2s_read_rd_addr (I2S_T * *I2Sx*)

get I2S rd register address

参数:

**I2Sx*	pointer to I2S_T structure

返回:

(uint32_t)&I2Sx->RD RD register address

1.19.2.12 uint32_t i2s_read_wr_addr (I2S_T * *I2Sx*)

Get I2S wr register address

参数:

**I2Sx*	pointer to I2S_T structure

返回:

(uint32_t)&I2Sx->WR WR register address

1.19.2.13 void i2s_recive_disable (I2S_T * *I2Sx*)

I2S recive disable

参数:

**I2Sx*	pointer to I2S_T structure

返回:

none

1.19.2.14 void i2s_recive_enable (I2S_T * *I2Sx*)

I2S recive enable

参数:

**I2Sx*	pointer to I2S_T structure

返回:

none

1.19.2.15 void i2s_transmit_disable (I2S_T * *I2Sx*)

i2s transmit enable

参数:

**I2Sx*	pointer to I2S_T structure

返回:

none

1.19.2.16 void i2s_transmit_enable (I2S_T * *I2Sx*)

i2s transmit enable

参数:

**I2Sx*	pointer to I2S_T structure

返回:

none

1.19.2.17 void i2s_write_buf (I2S_T * *I2Sx*, uint32_t * *buf*, uint32_t *size*)

I2S write buf

参数:

**I2Sx*	pointer to I2S_T structure
**buf*	pointer to buf
size	data len

返回:

none

1.19.2.18 void i2s_write_fifo (I2S_T * *I2Sx*, uint32_t *data*)

I2S write fifo

参数:

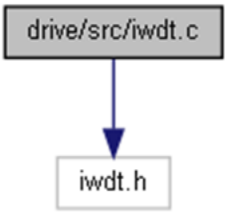
**I2Sx*	pointer to I2S_T structure
data	data

返回:

none

1.20 IWDT接口

IWDT driver source file
#include "iwdt.h"
iwdt.c 的引用(Include)关系图:



1.20.1 函数

- void **IWDT_IRQHandler** (void)
  IWDT interrupt handling
- void **iwdt_init** (IWDT_T *IWDT, uint16_t clock_div)
  IWDT initial
- void **iwdt_irq_init** (IWDT_T *IWDT, uint8_t irq_enable, void(*pfunc)())
  IWDT IRQ initial
- uint32_t **iwdt_cnt_read** (IWDT_T *IWDT)
  IWDT cnt read
- void **iwdt_reset_set** (IWDT_T *IWDT, uint8_t rst_mode, uint8_t rst_enable)
  IWDT reset set
- void **iwdt_load_set** (IWDT_T *IWDT, uint32_t load_value)
  IWDT load set

1.20.2 函数说明

1.20.2.1 uint32_t iwdt_cnt_read (IWDT_T * *IWDT*)

IWDT cnt read

参数:

**IWDT*	pointer to IWDT_T structure

返回:

IWDT->CNT IWDT counter value

1.20.2.2 void iwdt_init (IWDT_T * *IWDT*, uint16_t *clock_div*)

IWDT initial

参数:

**IWDT*	Pointer to IWDT_T structure
clock_div	Clock frequency division, range: 1~65536

返回:

none

1.20.2.3 void iwdt_irq_init (IWDT_T * *IWDT*, uint8_t *irq_enable*, void(*)() *pfunc*)

IWDT IRQ initial

参数:

**IWDT*	Pointer to IWDT_T structure
irq_enable	Interrupt status This parameter can be one of the following values: ● IWDT_IRQ_ENABLE Enable interrupt ● IWDT_IRQ_DISABLE Disable interrupt
*(*pfunc)()*	Callback func

返回:

none

1.20.2.4 void IWDT_IRQHandler (void)

IWDT interrupt handling

参数:

none	

返回:

none

1.20.2.5 void iwdt_load_set (IWDT_T * *IWDT*, uint32_t *load_value*)

IWDT load set

参数:

**IWDT*	Pointer to IWDT_T structure
load_value	Load value, range: 0x0000_0000 ~ 0xffff_ffff

返回:

none

1.20.2.6 void iwdt_reset_set (IWDT_T * *IWDT*, uint8_t *rst_mode*, uint8_t *rst_enable*)

IWDT reset set

参数:

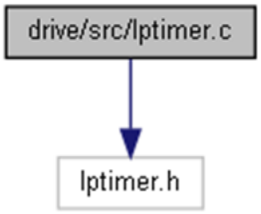
**IWDT*	Pointer to IWDT_T structure
rst_mode	Select reset count times This parameter can be one of the following values: ● IWDT_RESET_MODE_TWICE Reset after counting overflow twice ● IWDT_RESET_MODE_ONCE Reset after counting overflow once
rst_enable	Reset enable status This parameter can be one of the following values: ● IWDT_RESET_ENABLE Enable reset ● IWDT_RESET_DISABLE Disable reset

返回:

none

1.21 LPTIMER接口

LPTIMER driver source file
#include "lptimer.h"
lptimer.c 的引用(Include)关系图:



1.21.1 函数

- void **LPTIMER0_IRQHandler** (void)
  LPTIMER0 interrupt service
- void **LPTIMER1_IRQHandler** (void)
  LPTIMER1 interrupt service
- void **lptimer_irq_init** (LPTIMER_T *LPTIMERx, uint32_t irq_enable, uint32_t irq_type, void(*pfunc)())
  LPTIMER interrupt initialization
- void **lptimer_start** (LPTIMER_T *LPTIMERx)
  LPTIMER enable
- void **lptimer_stop** (LPTIMER_T *LPTIMERx)
  LPTIMER disable
- void **lptimer_count_init** (LPTIMER_T *LPTIMERx, uint32_t clock_source, uint16_t clock_div, uint16_t time)
  LPTIMER count function initialization
- void **lptimer_pwm_init** (LPTIMER_T *LPTIMERx, uint16_t cmp, uint16_t target, uint32_t clock_source, uint32_t clock_div, uint8_t level)
  LPTIMER pwm function initialization
- void **lptimer_pwm_setcompare** (LPTIMER_T *LPTIMERx, uint16_t cmp)
  pwm output duty cycle set
- void **lptimer_trigger_init** (LPTIMER_T *LPTIMERx, uint32_t clock_source, uint32_t clock_div, uint32_t trigger_edge, uint16_t target)
  LPTIMER trigger function initialization
- void **lptimer_timeout_init** (LPTIMER_T *LPTIMERx, uint32_t clock_source, uint32_t clock_div, uint32_t trigger_edge, uint16_t target)
  LPTIMER output function initialization
- void **lptimer_extcount_init** (LPTIMER_T *LPTIMERx, uint32_t clock_source, uint32_t

clock_div, uint16_t target)

LPTIMER external trigger function initialization

1.21.2 函数说明

1.21.2.1 void LPTIMER0_IRQHandler (void)

LPTIMER0 interrupt service

参数:

none	

返回:

none

1.21.2.2 void LPTIMER1_IRQHandler (void)

LPTIMER1 interrupt service

参数:

none	

返回:

none

1.21.2.3 void lptimer_count_init (LPTIMER_T * *LPTIMERx*, uint32_t *clock_source*, uint16_t *clock_div*, uint16_t *time*)

LPTIMER count function initialization

参数:

**LPTIMERx*	Pointer to LPTIMER_T structure
clock_source	LPTIMER counting clock source
clock_div	LPTIMER counting clock source frequency division
time	LPTIMER count target value, range: 0 ~ 65535

返回:

none

1.21.2.4 void lptimer_extcount_init (LPTIMER_T * *LPTIMERx*, uint32_t *clock_source*, uint32_t *clock_div*, uint16_t *target*)

LPTIMER external trigger function initialization

参数:

**LPTIMERx*	Pointer to LPTIMER_T structure
clock_source	LPTIMER counting clock source
clock_div	LPTIMER counting clock source frequency division
target	LPTIMER count target value

返回:

none

1.21.2.5 void lptimer_irq_init (LPTIMER_T * *LPTIMERx*, uint32_t *irq_enable*, uint32_t *irq_type*, void(*)() *pfunc*)

LPTIMER interrupt initialization

参数:

**LPTIMERx*	Pointer to LPTIMER_T structure
irq_enable	Interrupt status This parameter can be one of the following values: ● LPTIM_IRQ_ENABLE Enable interrupt ● LPTIM_IRQ_DISABLE Disable interrupt
irq_type	Interrupt type
*(*pfunc)()*	Interrupt callback function

返回:

none

1.21.2.6 void lptimer_pwm_init (LPTIMER_T * *LPTIMERx*, uint16_t *cmp*, uint16_t *target*, uint32_t *clock_source*, uint32_t *clock_div*, uint8_t *level*)

LPTIMER pwm function initialization

参数:

**LPTIMERx*	Pointer to LPTIMER_T structure
cmp	LPTIMER count comparison value
target	LPTIMER count target value
clock_source	LPTIMER counting clock source
clock_div	LPTIMER counting clock source frequency division
level	LPTIMER compare output polarity

返回:

none

1.21.2.7 void lptimer_pwm_setcompare (LPTIMER_T * *LPTIMERx*, uint16_t *cmp*)

pwm output duty cycle set

参数:

**LPTIMERx*	Pointer to LPTIMER_T structure
cmp	LPTIMER comparison value

返回:

none

1.21.2.8 void lptimer_start (LPTIMER_T * *LPTIMERx*)

LPTIMER enable

参数:

**LPTIMERx*	Pointer to LPTIMER_T structure

返回:

none

1.21.2.9 void lptimer_stop (LPTIMER_T * *LPTIMERx*)

LPTIMER disable

参数:

**LPTIMERx*	Pointer to LPTIMER_T structure

返回:

none

1.21.2.10 void lptimer_timeout_init (LPTIMER_T * *LPTIMERx*, uint32_t *clock_source*, uint32_t *clock_div*, uint32_t *trigger_edge*, uint16_t *target*)

LPTIMER output function initialization

参数:

**LPTIMER*	Pointer to LPTIMER_T structure
clock_source	LPTIMER counting clock source
clock_div	LPTIMER counting clock source frequency division
trigger_edge	LPTIMER trigger edge
target	LPTIMER count target value

返回:

none

1.21.2.11 void lptimer_trigger_init (LPTIMER_T * *LPTIMERx*, uint32_t *clock_source*, uint32_t *clock_div*, uint32_t *trigger_edge*, uint16_t *target*)

LPTIMER trigger function initialization

参数:

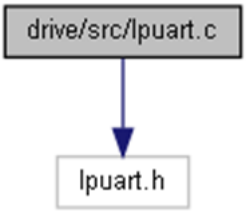
**LPTIMERx*	Pointer to LPTIMER_T structure
clock_source	LPTIMER counting clock source
clock_div	LPTIMER counting clock source frequency division
trigger_edge	LPTIMER trigger edge
target	LPTIMER count target value

返回:

none

1.22 LPUART接口

LPUART driver source file
#include "lpuart.h"
lpuart.c 的引用(Include)关系图:



1.22.1 函数

- void **LPUART_IRQHandler** (void)
  LPUART interrupt handling
- void **lpuart_set_baud_rate** (LPUART_T *LPUART, uint32_t baud_rate)
  set LPUART baud rate
- void **lpuart_clk_init** (BOOL newstate)
  LPUART clk init
- void **lpuart_init** (LPUART_T *LPUART, uint32_t baud_rate)
  LPUART initial
- void **lpuart_irq_init** (LPUART_T *LPUART, uint8_t irq_enable, void(*pfunc_rec)())
  LPUART IRQ initial
- void **lpuart_send_byte** (LPUART_T *LPUART, char c)
  LPUART send byte
- void **lpuart_send_bytes** (LPUART_T *LPUART, uint8_t *buff, uint32_t length)
  LPUART send bytes
- uint8_t **lpuart_recv_byte** (LPUART_T *LPUART)
  LPUART rece byte

1.22.2 函数说明

1.22.2.1 void lpuart_clk_init (BOOL *newstate*)

LPUART clk init

参数:

newstate	Clock and reset status. This parameter can be one of the following values: ● LPUART_ENABLE: enable lpuart clock and set it into work

	mode. ● LPUART_DISABLE：disable lpuart clock and set lpuart into reset mode.

返回:

none

1.22.2.2 void lpuart_init (LPUART_T * *LPUART*, uint32_t *baud_rate*)

LPUART initial

参数:

**LPUART*	pointer to LPUART_T structure
baud_rate	set lpuart communication data rate

返回:

none

函数调用图:

1.22.2.3 void lpuart_irq_init (LPUART_T * *LPUART*, uint8_t *irq_enable*, void(*)() *pfunc_rec*)

LPUART IRQ initial

参数:

**LPUART*	pointer to LPUART_T structure
irq_enable	Interrupt status. This parameter can be one of the following values: ● ENABLE: enable interrupt. ● DISABLE：disable interrupt.
void	(*pfunc_rec)() receive interrupt callback function

返回:

none

1.22.2.4 void LPUART_IRQHandler (void)

LPUART interrupt handling

参数:

none	

返回:

none

1.22.2.5 uint8_t lpuart_recv_byte (LPUART_T * *LPUART*)

LPUART rece byte

参数:

**LPUART*	pointer to LPUART_T structure

返回:

temp data from LPUART

1.22.2.6 void lpuart_send_byte (LPUART_T * *LPUART*, char *c*)

LPUART send byte

参数:

**LPUART*	pointer to LPUART_T structure
c	set lpuart transfer data

返回:

none

函数的调用关系图:

1.22.2.7 void lpuart_send_bytes (LPUART_T * *LPUART*, uint8_t * *buff*, uint32_t *length*)

LPUART send bytes

参数:

**LPUART*	pointer to LPUART_T structure
**buff*	pointer to transfer buff
length	set lpuart transfer data length

返回:

none

函数调用图:

1.22.2.8 void lpuart_set_baud_rate (LPUART_T * *LPUART*, uint32_t *baud_rate*)

set LPUART baud rate

参数:

**LPUART*	pointer to LPUART_T structure
baud_rate	set lpuart communication data rate

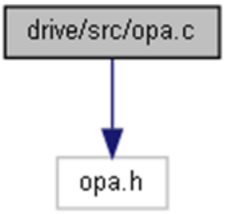
返回:

none

函数的调用关系图:

1.23 OPA接口

OPA driver source file
#include "opa.h"
opa.c 的引用(Include)关系图:



1.23.1 函数

- void **OPA0_IRQHandler** (void)
  opa0 interrupt handling
- void **OPA1_IRQHandler** (void)
  opa1 interrupt handling
- void **opa_uintbufffer_mode** (OPAX_T OPAx, OPA_SELP_OPT_T selp)
  Initializes for opa's uintbufffer mode.
- void **opa_opa_mode** (OPAX_T OPAx, OPA_SELP_OPT_T selp)
  Initializes for opa's opa mode.
- void **opa_pga_mode** (OPAX_T OPAx, OPA_GAIN_T times, OPA_SELP_OPT_T selp)
  Initializes for opa's pga mode.
- void **opa_irq_init** (OPAX_T OPAx, OPA_EDGE_T irq_edge, FUNC_E irq_enable)
  Initializes for opa interrupt.
- void **opa_comp_mode** (OPAX_T OPAx, OPA_EDGE_T irq_edge, FUNC_E irq_enable)
  Initializes for opa's comp mode.
- void **opa_external_gain_mode** (OPAX_T OPAx, OPA_SELP_OPT_T selp)
  Opa is amplified by external feedback resistor.
- FLAG_E **opa_getopa_status** (OPAX_T OPAx, uint32_t opa_flag)
  Checks whether the specified opa flag is set or not.

1.23.2 函数说明

1.23.2.1 void OPA0_IRQHandler (void)

OPA0 interrupt handling

参数:

none	

返回:

none

1.23.2.2 void OPA1_IRQHandler (void)

OPA1 interrupt handling

参数:

none	

返回:

none

1.23.2.3 void opa_comp_mode (OPAX_T *OPAx*, OPA_EDGE_T *irq_edge*, FUNC_E *irq_enable*)

Initializes for opa's comp mode.

参数:

OPAx	where x can be 0, 1,to select the OPA peripheral.
irq_edge	Setting the generate interrupt edge . This parameter can be one of the following values: ● OPA_ARBITRARY_EDGE Any edge generates an interrupt. ● OPA_TRAILING_EDGE The falling edge generates an interrupt. ● OPA_RISING_EDGE The rising edge generates interrupt.
irq_enable	This parameter can be OPA_IRQ_ENABLE or OPQ_IRQ_DISABLE.

返回:

none

函数调用图:

1.23.2.4 void opa_external_gain_mode (OPAX_T *OPAx*, OPA_SELP_OPT_T *selp*)

OPA is amplified by external feedback resistor.

注解:

The amplification factor is determined by the external feedback resistor.

参数:

OPAx	where x can be 0, 1,to select the OPA peripheral.
selp	Positive signal

返回:

none

1.23.2.5 FLAG_E opa_getopa_status (OPAX_T *OPAx*, uint32_t *opa_flag*)

Checks whether the specified opa flag is set or not.

参数:

OPAx	where x can be 0, 1,to select the OPA peripheral.
opa_flag	specifies the opa flag to check. ● OPA_CMP_OUTPUT_FLAG The output flag of OPA as a comparator.

返回:

The new state of OPA flag (SET or RESET).

1.23.2.6 void opa_irq_init (OPAX_T *OPAx*, OPA_EDGE_T *irq_edge*, FUNC_E *irq_enable*)

Initializes for opa interrupt.

参数:

OPAx	where x can be 0, 1,to select the OPA peripheral.
irq_edge	Setting the generate interrupt edge . This parameter can be one of the following values: ● OPA_ARBITRARY_EDGE Any edge generates an interrupt. ● OPA_TRAILING_EDGE The falling edge generates an interrupt. ● OPA_RISING_EDGE The rising edge generates interrupt.
irq_enable	This parameter can be OPA_IRQ_ENABLE or OPQ_IRQ_DISABLE.

返回:

none

函数的调用关系图:

1.23.2.7 void opa_opa_mode (OPAX_T *OPAx*, OPA_SELP_OPT_T *selp*)

Initializes for opa's opa mode.

参数:

OPAx	where x can be 0, 1,to select the OPA peripheral.
selp	Positive signal

返回:

none

1.23.2.8 void opa_pga_mode (OPAX_T *OPAx*, OPA_GAIN_T *times*, OPA_SELP_OPT_T *selp*)

Initializes for opa's pga mode.

参数:

OPAx	where x can be 0, 1,to select the OPA peripheral.
times	This parameter is to set the gain multiple of pga This parameter can be one of the following values: ● OPA_GAIN_2x Gain times equal to 2 times. ● OPA_GAIN_4x Gain times equal to 4 times. ● OPA_GAIN_8x Gain times equal to 8 times. ● OPA_GAIN_16x Gain times equal to 16 times. ● OPA_GAIN_32x Gain times equal to 32 times. ● OPA_GAIN_64x Gain times equal to 64 times.
selp	Positive signal

返回:

none

1.23.2.9 void opa_uintbufffer_mode (OPAX_T *OPAx*, OPA_SELP_OPT_T *selp*)

Initializes for opa's uintbufffer mode.

参数:

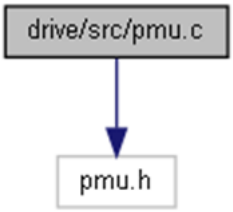
OPAx	where x can be 0, 1,to select the OPA peripheral.
selp	Positive signal

返回:

none

1.24 PMU接口

PMU driver source file
#include "pmu.h"
pmu.c 的引用(Include)关系图:



1.24.1 函数

- void **pmu_standby_wakeup_events_config** (uint8_t wake_event, BOOL newstate)
  wake up event configuration for the standby mode(standbymode0、standbymode1、deepstandbymode0、deepstandbymode1)
- void **pmu_standby_wakeup_io_polarity_config** (uint8_t wakeup_io, BOOL io_polarity)
  I/O polarity configuration for the standby mode(standbymode0、standbymode1、deepstandbymode0、deepstandbymode1)
- uint8_t **pmu_standby_get_wakeup_status** (uint8_t status)
  gets the status of the wake up event for the standby mode(standbymode0、standbymode1、deepstandbymode0、deepstandbymode1)
- void **pmu_standby_clr_wakeup_status** (void)
  clears all of the status of the wake up event for the standby mode(standbymode0、standbymode1、deepstandbymode0、deepstandbymode1)
- void **pmu_stop_exti_mode_config** (uint8_t exti_mode)
  exti mode configuration for the stop mode
- void **pmu_stop_wakeup_exti_events_config** (uint8_t wake_event, BOOL newstate)
  exti wake up event configuration for the stop mode
- void **pmu_stop_wakeup_exti_polarity_config** (uint8_t wakeup_exti, BOOL exti_polarity)
  exti polarity configuration for the stop mode
- uint8_t **pmu_stop_get_exti_wakeup_status** (uint8_t exti_status)
  gets the status of the exti wake up event for the stop mode
- void **pmu_stop_clr_exti_wakeup_status** (uint8_t exti_status)
  clears the status of the exti wake up event for the stop mode

1.24.2 函数说明

1.24.2.1 void pmu_standby_clr_wakeup_status (void)

clears all of the status of the wake up event for the standby mode(standbymode0、standbymode1、deepstandbymode0、deepstandbymode1)

参数:

none	

返回:

none

1.24.2.2 uint8_t pmu_standby_get_wakeup_status (uint8_t *status*)

gets the status of the wake up event for the standby mode(standbymode0、standbymode1、deepstandbymode0、deepstandbymode1)

参数:

status	Status of wake up event. This parameter can be one of the following values: ● PMU_STANDBY_WAKEUP_STATUS_PA0. ● PMU_STANDBY_WAKEUP_STATUS_PA2. ● PMU_STANDBY_WAKEUP_STATUS_PC0. ● PMU_STANDBY_WAKEUP_STATUS_PC2. ● PMU_STANDBY_WAKEUP_STATUS_PC3. ● PMU_STANDBY_WAKEUP_STATUS_PC13. ● PMU_STANDBY_WAKEUP_STATUS_RESETN.

返回:

pmu_standby_get_wakeup_status.

- 0: Flag event not happen.
- 1: Flag event happen.

1.24.2.3 void pmu_standby_wakeup_events_config (uint8_t *wake_event*, BOOL *newstate*)

wake up event configuration for the standby mode(standbymode0、standbymode1、deepstandbymode0、deepstandbymode1)

参数:

wake_event	Wake up events in standby mode. This parameter can be one of the following values:

	● PMU_STANDBY_WAKEUP_EVENT_RESETN. ● PMU_STANDBY_WAKEUP_EVENT_PA0. ● PMU_STANDBY_WAKEUP_EVENT_RTCALARM. ● PMU_STANDBY_WAKEUP_EVENT_RTCTAMP. ● PMU_STANDBY_WAKEUP_EVENT_IWDT. ● PMU_STANDBY_WAKEUP_EVENT_LPUART. ● PMU_STANDBY_WAKEUP_EVENT_LPTIM0. ● PMU_STANDBY_WAKEUP_EVENT_LPTIM1. ● PMU_STANDBY_WAKEUP_EVENT_PA2. ● PMU_STANDBY_WAKEUP_EVENT_PC0. ● PMU_STANDBY_WAKEUP_EVENT_PC2. ● PMU_STANDBY_WAKEUP_EVENT_PC3. ● PMU_STANDBY_WAKEUP_EVENT_PC13.
newstate	Wake up event status. This parameter can be one of the following values: ● PMU_ENABLE: enable wake event. ● PMU_DISABLE: disable wake event.

返回:

none

1.24.2.4 void pmu_standby_wakeup_io_polarity_config (uint8_t *wakeup_io*, BOOL *io_polarity*)

I/O polarity configuration for the standby mode(standbymode0 、 standbymode1 、 deepstandbymode0、deepstandbymode1)

参数:

wakeup_io	Wake up io. This parameter can be one of the following values: ● PMU_STANDBY_WAKEUP_IO_PA0. ● PMU_STANDBY_WAKEUP_IO_PA2. ● PMU_STANDBY_WAKEUP_IO_PC0. ● PMU_STANDBY_WAKEUP_IO_PC2. ● PMU_STANDBY_WAKEUP_IO_PC3. ● PMU_STANDBY_WAKEUP_IO_PC13.
io_polarity.	This parameter can be one of the following values: ● PMU_STANDBY_WAKEUP_IO_POLARITY_UP. ● PMU_STANDBY_WAKEUP_IO_POLARITY_DOWN.

返回:

none

1.24.2.5 void pmu_stop_clr_exti_wakeup_status (uint8_t *exti_status*)

clears the status of the exti wake up event for the stop mode

参数:

status	Status of exti wake up event. This parameter can be one of the following values: ● PMU_STANDBY_WAKEUP_EVENT_EXTI0. ● PMU_STANDBY_WAKEUP_EVENT_EXTI1. ● PMU_STANDBY_WAKEUP_EVENT_EXTI2. ● PMU_STANDBY_WAKEUP_EVENT_EXTI3. ● PMU_STANDBY_WAKEUP_EVENT_EXTI4. ● PMU_STANDBY_WAKEUP_EVENT_EXTI5. ● PMU_STANDBY_WAKEUP_EVENT_EXTI6. ● PMU_STANDBY_WAKEUP_EVENT_EXTI7. ● PMU_STANDBY_WAKEUP_EVENT_EXTI8. ● PMU_STANDBY_WAKEUP_EVENT_EXTI9. ● PMU_STANDBY_WAKEUP_EVENT_EXTI10. ● PMU_STANDBY_WAKEUP_EVENT_EXTI11. ● PMU_STANDBY_WAKEUP_EVENT_EXTI12. ● PMU_STANDBY_WAKEUP_EVENT_EXTI13. ● PMU_STANDBY_WAKEUP_EVENT_EXTI14. ● PMU_STANDBY_WAKEUP_EVENT_EXTI15.

返回:

none

1.24.2.6 void pmu_stop_exti_mode_config (uint8_t *exti_mode*)

exti mode configuration for the stop mode

参数:

exti_mode	When you want to use exti inperrupt to wake stop mode, you must set EXTI_MODE_STOP. After wake stop mode, you should set EXTI_MODE_ACTIVE. This parameter can be one of the following values: ● EXTI_MODE_STOP. ● EXTI_MODE_ACTIVE.

返回:

none

1.24.2.7 uint8_t pmu_stop_get_exti_wakeup_status (uint8_t *exti_status*)

gets the status of the exti wake up event for the stop mode

参数:

status	Status of exti wake up event. This parameter can be one of the following values: ● PMU_STANDBY_WAKEUP_EVENT_EXTI0. ● PMU_STANDBY_WAKEUP_EVENT_EXTI1. ● PMU_STANDBY_WAKEUP_EVENT_EXTI2. ● PMU_STANDBY_WAKEUP_EVENT_EXTI3. ● PMU_STANDBY_WAKEUP_EVENT_EXTI4. ● PMU_STANDBY_WAKEUP_EVENT_EXTI5.

	● PMU_STANDBY_WAKEUP_EVENT_EXTI6. ● PMU_STANDBY_WAKEUP_EVENT_EXTI7. ● PMU_STANDBY_WAKEUP_EVENT_EXTI8. ● PMU_STANDBY_WAKEUP_EVENT_EXTI9. ● PMU_STANDBY_WAKEUP_EVENT_EXTI10. ● PMU_STANDBY_WAKEUP_EVENT_EXTI11. ● PMU_STANDBY_WAKEUP_EVENT_EXTI12. ● PMU_STANDBY_WAKEUP_EVENT_EXTI13. ● PMU_STANDBY_WAKEUP_EVENT_EXTI14. ● PMU_STANDBY_WAKEUP_EVENT_EXTI15.

返回:

pmu_stop_get_exti_wakeup_status.

- 0: Flag event not happen.
- 1: Flag event happen.

1.24.2.8 void pmu_stop_wakeup_exti_events_config (uint8_t *wake_event*, BOOL *newstate*)

exti wake up event configuration for the stop mode

参数:

wake_event	Wake up events in stop mode. This parameter can be one of the following values: ● PMU_STANDBY_WAKEUP_EVENT_EXTI0. ● PMU_STANDBY_WAKEUP_EVENT_EXTI1. ● PMU_STANDBY_WAKEUP_EVENT_EXTI2. ● PMU_STANDBY_WAKEUP_EVENT_EXTI3. ● PMU_STANDBY_WAKEUP_EVENT_EXTI4. ● PMU_STANDBY_WAKEUP_EVENT_EXTI5. ● PMU_STANDBY_WAKEUP_EVENT_EXTI6. ● PMU_STANDBY_WAKEUP_EVENT_EXTI7. ● PMU_STANDBY_WAKEUP_EVENT_EXTI8. ● PMU_STANDBY_WAKEUP_EVENT_EXTI9. ● PMU_STANDBY_WAKEUP_EVENT_EXTI10. ● PMU_STANDBY_WAKEUP_EVENT_EXTI11. ● PMU_STANDBY_WAKEUP_EVENT_EXTI12. ● PMU_STANDBY_WAKEUP_EVENT_EXTI13. ● PMU_STANDBY_WAKEUP_EVENT_EXTI14. ● PMU_STANDBY_WAKEUP_EVENT_EXTI15.
newstate	Wake up event status. This parameter can be one of the following values: ● PMU_ENABLE: enable wake event. ● PMU_DISABLE: disable wake event.

返回:

none

1.24.2.9 void pmu_stop_wakeup_exti_polarity_config (uint8_t *wakeup_exti*, BOOL *exti_polarity*)

exti polarity configuration for the stop mode

参数:

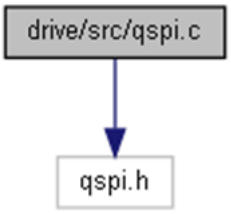
wakeup_exti	Wake up exti. This parameter can be one of the following values: ● PMU_STANDBY_WAKEUP_EVENT_EXTI0. ● PMU_STANDBY_WAKEUP_EVENT_EXTI1. ● PMU_STANDBY_WAKEUP_EVENT_EXTI2. ● PMU_STANDBY_WAKEUP_EVENT_EXTI3. ● PMU_STANDBY_WAKEUP_EVENT_EXTI4. ● PMU_STANDBY_WAKEUP_EVENT_EXTI5. ● PMU_STANDBY_WAKEUP_EVENT_EXTI6. ● PMU_STANDBY_WAKEUP_EVENT_EXTI7. ● PMU_STANDBY_WAKEUP_EVENT_EXTI8. ● PMU_STANDBY_WAKEUP_EVENT_EXTI9. ● PMU_STANDBY_WAKEUP_EVENT_EXTI10. ● PMU_STANDBY_WAKEUP_EVENT_EXTI11. ● PMU_STANDBY_WAKEUP_EVENT_EXTI12. ● PMU_STANDBY_WAKEUP_EVENT_EXTI13. ● PMU_STANDBY_WAKEUP_EVENT_EXTI14. ● PMU_STANDBY_WAKEUP_EVENT_EXTI15.
io_polarity.	This parameter can be one of the following values: ● PMU_STOP_WAKEUP_EXTI_POLARITY_UP. ● PMU_STOP_WAKEUP_EXTI_POLARITY_DOWN.

返回:

none

1.25 QSPI接口

QSPI driver source file
#include "qspi.h"
qspi.c 的引用(Include)关系图:



1.25.1 函数

- void **QSPI_IRQHandler** (void)
  QSPI interrupt handling
- void **qspi_clk_init** (BOOL newstate)
  Initializs the clock of QSPI.
- void **qspi_set_drir** (QSPI_DUMMY_T dummy_clks, ADDR_MODE_T addr_io_mode, DATA_MODE_T data_io_mode, uint8_t rd_cmd)
  Setting the QSPI_DRIR register
- void **qspi_set_dwir** (ADDR_MODE_T addr_io_mode, DATA_MODE_T data_io_mode, uint8_t wr_cmd)
  Setting the QSPI_DWIR register
- void **qspi_set_program_time** (uint8_t csda, uint8_t cseot, uint8_t cssot)
  Set the communication delay time of QSPI.
  sclk_out=width of pulse ref_clk=1/sysclk.
- void **qspi_set_dscr** (QSPI_ADDR_BYTES_T addr_bytes, uint32_t page_bytes, uint8_t blk_bytes)
  Setting the QSPI_DSCR register
- void **qspi_set_cfgr** (QSPI_WAY_T qspi_way, QSPI_BAUD_T brdiv, QSPI_WORK_MODE_T work_mode)
  Setting the QSPI_CFGR register
- void **qspi_set_rdcr** (uint8_t dlyt, uint8_t smes, uint8_t dlyr)
  Setting the QSPI_RDCR register
- void **qspi_select_func** (QSPI_FUNC_EN_T func, FUNC_E func_enable)
  Setting the function of the QSPI
- void **qspi_wait_idle** (void)
  QSPI_wait_idle
- void **qspi_command_idle** (void)

qspi wait command run idle

- void **qspi_indac_write_set** (uint32_t write_start_addr, uint32_t num_bytes, uint32_t ahb_addr, uint8_t trg_addr_range)
  Setting the INDAC mode of QSPI to write
- void **qspi_indac_read_set** (uint32_t read_start_addr, uint32_t num_bytes, uint32_t ahb_addr, uint8_t trg_addr_range)
  Setting the INDAC mode of QSPI to read
- void **qspi_set_fcar** (uint32_t addr)
  Setting the QSPI_FCAR register
- uint32_t **qspi_get_fcrlr** (void)
  Setting the QSPI_FCRLR register
- uint32_t **qspi_get_fcrhr** (void)
  Setting the QSPI_FCRHR register
- void **qspi_ind_write_start** (void)
  wait for qspi indac write start
- void **qspi_set_fcr** (uint32_t opcode, uint8_t addr_en, uint8_t modb_en, QSPI_ADDR_BYTES_T add_num, QSPI_DUMMY_T dum_num, uint8_t rd_en, QSPI_RDNUM_T rd_num, uint8_t wd_en, QSPI_WDNUM_T wd_num)
  Setting the QSPI_FCR register
- void **qpspi_wait_indwr_cmpl** (void)
  wait for the qpspi indac write completely
- void **qspi_ind_read_start** (void)
  wait for the qpspi indac read start
- void **qpspi_wait_indrd_cmpl** (void)
  wait for the qpspi indac read completely
- void **qspi_set_dmacr** (uint8_t burst_num, uint8_t single_num)
  Setting the QSPI_DMACR register
- void **qspi_irq_init** (FUNC_E irq_enable, QSPI_IRQ_T irtype, void(*pfunc)())
  Initializs the interrupt of qspi.
- void **qspi_set_per** (uint32_t pcycn)
  Setting the QSPI_PER register
- void **qspi_set_rxhr** (uint8_t rxhr)
  Setting the QSPI_RXHR register
- void **qspi_set_txhr** (uint8_t txhr)
  Setting the QSPI_TXHR register
- void **qspi_set_fcwlr** (uint32_t cmd_dl)
  Setting the QSPI_FCWLR register
- void **qspi_set_mbr** (uint8_t modeb)

Setting the QSPI_MBR register

- void **qspi_set_fcwhr** (uint32_t cmd_dh)

  Setting the QSPI_FCWHR register
- void **qspi_polling_delay** (uint16_t prepd)

  Setting the time of the qspi polling delay
- void **qspi_polling_set** (uint8_t pplt, uint8_t pbind)

  Setting the QSPI's polling mode
- void **qspi_set_wcr** (uint8_t poll_exp_en, uint8_t polling_en, uint8_t pcnt, uint16_t opcode)

  Setting the QSPI_WCR register
- void **qspi_set_wpcr** (QSPI_WPINVERT_T wpinv, FUNC_E wp_en)

  Setting the QSPI_WPCR register
- void **qspi_set_wplr** (uint32_t addr)

  Setting the QSPI_WPLR register
- void **qspi_set_wphr** (uint32_t addr)

  Setting the QSPI_WPHR register
- void **qspi_set_spr** (uint8_t sprx)

  Setting the QSPI_SPR register
- BOOL **qspi_get_status** (uint32_t qspi_flag)

  Checks whether the specified QSPI irq flag is set or not.

1.25.2 函数说明

1.25.2.1 void qpspi_wait_indrd_cmpl (void)

wait for the qpspi indac read completely

参数:

none	

返回:

none

1.25.2.2 void qpspi_wait_indwr_cmpl (void)

wait for the qpspi indac write completely

参数:

none	

返回:

none

1.25.2.3 void qspi_clk_init (BOOL *newstate*)

Initializs the clock of QSPI.

参数:

newstate	ENABLE/DISABLE

返回:

none

函数调用图:

1.25.2.4 void qspi_command_idle (void)

QSPI wait command run idle

参数:

none	

返回:

none

1.25.2.5 uint32_t qspi_get_fcrhr (void)

Setting the QSPI_FCRHR register.

参数:

none	

返回:

data

1.25.2.6 uint32_t qspi_get_fcrlr (void)

Setting the QSPI_FCRLR register.

参数:

none	

返回:

data

1.25.2.7 BOOL qspi_get_status (uint32_t *qspi_flag*)

Checks whether the specified QSPI irq flag is set or not.

参数:

flash_flag	specifies the QSPI flag to check. This parameter can be one of the following values: ● QSPI_POLLF_FLAG Maximum number of polling cycles flag ● QSPI_IND_RDFF_FLAG The indirect read area in SRAM is full, and cannot be completed immediately flag . ● QSPI_STRFFF_FLAG Small capacity RXFIFO full flag[Current FIFO status] ● QSPI_SRFNEF_FLAG Small capacity RXFIFO non-empty flag[Current FIFO status] ● QSPI_STFFF_FLAG Small capacity TXFIFO full flag[Current FIFO status] ● QSPI_STFNFF_FLAG Small capacity TXFIFO non-empty flag[Current FIFO status] ● QSPI_ROVF_FLAG Receive overflow flag ● QSPI_IND_TWF_FLAG Exceeds the indirect transmission depth threshold flag ● QSPI_AHB_AEF_FLAG Illegal AHB access flag ● QSPI_WPAF_FLAG Attempt to write protected area denied flag ● QSPI_IND_RRF_FLAG Not received the indirect operation request flag ● QSPI_IND_CF_FALG The controller has completed the last indirect operation flag ● QSPI_UDFF_FLAG Check underflow flag

返回:

The new state of QSPI_FLAG (SET or RESET).

1.25.2.8 void qspi_ind_read_start (void)

wait for the qpspi indac read start

参数:

none	

返回:

none

1.25.2.9 void qspi_ind_write_start (void)

wait for QSPI indac write start

参数:

none	

返回:

none

1.25.2.10 void qspi_indac_read_set (uint32_t *read_start_addr*, uint32_t *num_bytes*, uint32_t *ahb_addr*, uint8_t *trg_addr_range*)

Setting the INDAC mode of QSPI to read.

参数:

read_start_addr	The start address of the indac mode
num_bytes	Number of bytes of the indac mode
ahb_addr	Trigger address of the indac mode
trg_addr_range	Indirect access to range width

返回:

none

1.25.2.11 void qspi_indac_write_set (uint32_t *write_start_addr*, uint32_t *num_bytes*, uint32_t *ahb_addr*, uint8_t *trg_addr_range*)

Setting the INDAC mode of QSPI to write.

参数:

write_start_addr	The start address of the indac mode
num_bytes	Number of bytes of the indac mode
ahb_addr	Trigger address of the indac mode
trg_addr_range	Indirect access to range width

返回:

none

1.25.2.12 void qspi_irq_init (FUNC_E *irq_enable*, QSPI_IRQ_T *irtype*, void(*)() *pfunc*)

Initializs the interrupt of QSPI.

参数:

irq_enable	ENABLE/DISABLE
irtype	Interrupt type
void	(*pfunc)() Callback handler function

返回:

none

1.25.2.13 void QSPI_IRQHandler (void)

qspi interrupt handling

参数:

none	

返回:

none

1.25.2.14 void qspi_polling_delay (uint16_t *prepd*)

Setting the time of the QSPI polling delay.

参数:

prepd	Polling repeat delay

返回:

none

1.25.2.15 void qspi_polling_set (uint8_t *pplt*, uint8_t *pbind*)

Setting the QSPI's polling mode.

参数:

pplt	Polling polarity
pbind	Polling bit retrieval

返回:

none

1.25.2.16 void qspi_select_func (QSPI_FUNC_EN_T *func*, FUNC_E *func_enable*)

Setting the function of the QSPI.

参数:

func	qspi function selection This parameter can be one of the following values: ● QSPI_DTR_EN Enabled the DTR protocol ● QSPI_AHB_DECODER_EN Enabled the AHB decoder ● QSPI_ENTER_XIP_DIR Enter XIP mode immediately. ● QSPI_ENTER_XIP_NEXT Enter XIP mode at the next read instruction. ● QSPI_AHB_ADDR_REMAP AHB address remapping enabled ● QSPI_WRITE_PROTECT Enabled the Write-protect
func_enable	ENABLE / DISABLE

返回:

none

1.25.2.17 void qspi_set_cfgr (QSPI_WAY_T *qspi_way*, QSPI_BAUD_T *brdiv*, QSPI_WORK_MODE_T *work_mode*)

Setting the QSPI_CFGR register.

参数:

qspi_way	Qspi communicate mode
brdiv	Main mode baud rate frequency division
work_mode	Clock phase and polarity

返回:

none

1.25.2.18 void qspi_set_dmacr (uint8_t *burst_num*, uint8_t *single_num*)

Setting the QSPI_DMACR register.

参数:

burst_num	Number of bytes of Burst type in DMA request
single_num	Number of bytes of type Single in DMA request

返回:

none

1.25.2.19 void qspi_set_drir (QSPI_DUMMY_T *dummy_clks*, ADDR_MODE_T *addr_io_mode*, DATA_MODE_T *data_io_mode*, uint8_t *rd_cmd*)

Setting the QSPI_DRIR register.

参数:

dummy_clks	This parameter is setting the number of the dummy
addr_io_mode	Read address transmission line width
data_io_mode	Read data transmission line width
rd_cmd	Read instructions

返回:

none

1.25.2.20 void qspi_set_dscr (QSPI_ADDR_BYTES_T *addr_bytes*, uint32_t *page_bytes*, uint8_t *blk_bytes*)

Setting the QSPI_DSCR register.

参数:

addr_bytes	Number of address bytes
page_bytes	Page number of bytes

blk_bytes	Block of bytes

返回:

none

1.25.2.21 void qspi_set_dwir (ADDR_MODE_T *addr_io_mode*, DATA_MODE_T *data_io_mode*, uint8_t *wr_cmd*)

Setting the QSPI_DWIR register.

参数:

addr_io_mode	Write address transmission line width
data_io_mode	Write data transmission line width
wr_cmd	Written instructions

返回:

none

1.25.2.22 void qspi_set_fcar (uint32_t *addr*)

Setting the QSPI_FCAR register.

参数:

addr	Command address

返回:

none

1.25.2.23 void qspi_set_fcr (uint32_t *opcode*, uint8_t *addr_en*, uint8_t *modb_en*, QSPI_ADDR_BYTES_T *add_num*, QSPI_DUMMY_T *dum_num*, uint8_t *rd_en*, QSPI_RDNUM_T *rd_num*, uint8_t *wd_en*, QSPI_WDNUM_T *wd_num*)

Setting the QSPI_FCR register.

注解:

This function is a common configuration option for setting stig mode.

参数:

opcode	The command used by the operation
addr_en	Command address enable setting bit
modb_en	Mode bit enable setting bit
add_num	The number of address bytes[ADDR_BYTES_1 is required by default.] This parameter can be one of the following values: ● ADDR_BYTES_1 The number of address bytes is 1 ● ADDR_BYTES_2 The number of address bytes is 2

	● ADDR_BYTES_3 The number of address bytes is 3 ● ADDR_BYTES_4 The number of address bytes is 4
dum_num	Dummy clock period[DUMMY_CLKS_0 is required by default.]
rd_en	Enable read data
rd_num	Number of data bytes of read[RD_NUM1 is required by default.]
wd_en	Enable write data
wd_num	Number of data bytes of write[WD_NUM1 is required by default.]

返回:

none

1.25.2.24 void qspi_set_fcwhr (uint32_t *cmd_dh*)

Setting the QSPI_FCWHR register.

参数:

cmd_dh	Write command high data

返回:

none

1.25.2.25 void qspi_set_fcwlr (uint32_t *cmd_dl*)

Setting the QSPI_FCWLR register.

参数:

cmd_dl	Write command low data

返回:

none

1.25.2.26 void qspi_set_mbr (uint8_t *modeb*)

Setting the QSPI_MBR register.

参数:

modeb	mode bit

返回:

none

1.25.2.27 void qspi_set_per (uint32_t *pcycn*)

Setting the QSPI_PER register.

参数:

pcycn	This parameter is setting the number of polling cycles

返回:

none

1.25.2.28 void qspi_set_program_time (uint8_t *csda*, uint8_t *cseot*, uint8_t *cssot*)

Set the communication delay time of QSPI.

注解:

sclk_out=width of pulse ref_clk=1/sysclk

参数:

csda	Set the slice selection to invalid time
cseot	Slice selection end time of transmission
cssot	Select the start time of the transmission

返回:

none

1.25.2.29 void qspi_set_rdcr (uint8_t *dlyt*, uint8_t *smes*, uint8_t *dlyr*)

Setting the QSPI_RDCR register.

参数:

dlyt	Transmission data delay (the delay time is the set ref_clk cycle number)
smes	Sampling edge selection (Flash memory data output)
dlyr	Read data capture delay (the delay time is the set number of ref_clk cycles)

返回:

none

1.25.2.30 void qspi_set_rxhr (uint8_t *rxhr*)

Setting the QSPI_RXHR register.

参数:

rxhr	Setting value of receive threshold register

返回:

none

1.25.2.31 void qspi_set_spr (uint8_t *sprx*)

Setting the QSPI_SPR register.

参数:

sprx	Divide the address range of indirect write and indirect read

注解:

Avoid setting sprx to 0xFF or 0x00.

返回:

none

1.25.2.32 void qspi_set_txhr (uint8_t *txhr*)

Setting the QSPI_RXHR register.

参数:

txhr	Setting value of send threshold register

返回:

none

1.25.2.33 void qspi_set_wcr (uint8_t *poll_exp_en*, uint8_t *polling_en*, uint8_t *pcnt*, uint16_t *opcode*)

Setting the QSPI_WCR register.

参数:

poll_exp_en	Enable polling to end interruption
polling_en	Whether the automatic polling function is enabled or not
pcnt	Define the number of polling times
opcode	Set the polling command

返回:

none

1.25.2.34 void qspi_set_wpcr (QSPI_WPINVERT_T *wpinv*, FUNC_E *wp_en*)

Setting the QSPI_WCR register.

参数:

wpinv	Write protect invert control
wp_en	ENABLE / DISABLE

返回:

none

1.25.2.35 void qspi_set_wphr (uint32_t *addr*)

Setting the QSPI_WPHR register.

参数:

addr	End addr of the write protected

返回:

none

1.25.2.36 void qspi_set_wplr (uint32_t *addr*)

Setting the QSPI_WPLR register.

参数:

addr	Start addr of the write protected

返回:

none

1.25.2.37 void qspi_wait_idle (void)

QSPI wait idle

参数:

none	

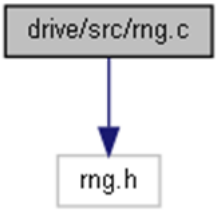
返回:

none

函数的调用关系图:

1.26 RNG接口

RNG driver source file
#include "rng.h"
rng.c 的引用(Include)关系图:



1.26.1 函数

- void **rng_init** (RNG_T *RNG)
  RNG initialisation
- void **rng_write_seed** (RNG_T *RNG, uint32_t rngseed)
  RNG write seed
- uint32_t **get_rng** (RNG_T *RNG)
  get RNG value

1.26.2 函数说明

1.26.2.1 uint32_t get_rng (RNG_T * *RNG*)

get RNG value

参数:

**RNG*	pointer to RNG_T structure

返回:

RNG value

1.26.2.2 void rng_init (RNG_T * *RNG*)

RNG initialisation

参数:

**RNG*	pointer to RNG_T structure

返回:

none

1.26.2.3 void rng_write_seed (RNG_T * *RNG*, uint32_t *rngseed*)

RNG write seed

参数:

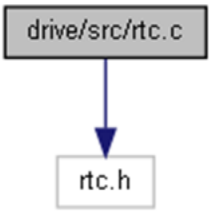
**RNG*	pointer to RNG_T structure
rngseed	set RNG seed transfer data

返回:

none

1.27 RTC接口

RTC driver source file
#include "rtc.h"
rtc.c 的引用(Include)关系图:



1.27.1 函数

- void **TAMPSTAMP_IRQHandler** (void)
  tamper interrupt service function
- void **RTC_ALARM_IRQHandler** (void)
  alarm interrupt service function
- uint8_t **rtc_int_state_read** (RTC_T *RTC, uint8_t irq_mode)
  interrupt status reading after bbu enable
- uint8_t **rtc_int_raw_state_read** (RTC_T *RTC, uint8_t irq_mode)
  bbu original interrupt status reading
- void **rtc_int_state_clear** (RTC_T *RTC, uint8_t irq_mode)
  bbu interrupt status clear
- void **rtc_irq_init** (RTC_T *RTC, uint8_t irq_mode, uint8_t irq_enable, void(*pfunc)())
  RTC interrupt initialization
- void **rtc_enable** (void)
  RTC enable
- void **rtc_disable** (void)
  RTC disable
- void **rtc_up_test** (RTC_T *RTC)
  RTC count up test
- void **rtc_calendat_time_set** (uint16_t year, uint8_t month, uint8_t day, uint8_t week, uint8_t hour, uint8_t min, uint8_t sec, uint8_t centisec, uint32_t *date, uint32_t *time)
  RTC calendar time setting
- void **rtc_calendar_time_init** (RTC_T *RTC, uint16_t year, uint8_t month, uint8_t day, uint8_t week, uint8_t hour, uint8_t min, uint8_t sec, uint8_t centisec)
  RTC calendar time initialization
- void **rtc_calendar_time_get** (RTC_T *RTC, uint8_t *year, uint8_t *month, uint8_t *day, uint8_t *week, uint8_t *hour, uint8_t *min, uint8_t *sec, uint8_t *centisec, uint8_t

*am_pm_centisec)
RTC calendar time acquisition

- void **rtc_alarm1_time_set** (RTC_T *RTC, uint16_t year, uint8_t month, uint8_t day, uint8_t week, uint8_t hour, uint8_t min, uint8_t sec, uint8_t centisec)
  RTC alarm clock 1 time setting
- void **rtc_alarm1_en_set** (RTC_T *RTC, uint32_t condition, uint8_t alarm_enable)
  RTC alarm clock 1 enable setting
- void **rtc_alarm2_time_set** (RTC_T *RTC, uint32_t time)
  RTC alarm clock 2 time setting
- void **rtc_alarm2_en_set** (RTC_T *RTC, uint8_t alarm_enable)
  RTC alarm clock 2 enable setting
- void **rtc_bk_wirte_word** (RTC_T *RTC, uint32_t word, uint8_t bk_area)
  write a character to the rtc backup register
- uint32_t **rtc_bk_read_word** (RTC_T *RTC, uint8_t bk_area)
  RTC backup register reads a character
- void **rtc_bk_wirte_words** (RTC_T *RTC, uint32_t *word, uint8_t bk_start_area, uint8_t length)
  data written by RTC backup register
- void **rtc_tamper_en_set** (RTC_T *RTC, uint8_t tamper_enable)
  enable tamper detection
- void **rtc_tamper_set** (RTC_T *RTC, uint8_t edge)
  tamper trigger edge setting
- uint8_t **rtc_tamper_cnt_read** (RTC_T *RTC)
  read the value of the tamper counter
- void **rtc_tamper_cnt_clear** (RTC_T *RTC)
  clear tamper counter
- void **rtc_tamper_time_get** (RTC_T *RTC, uint8_t tamperx, uint8_t *year, uint8_t *month, uint8_t *day, uint8_t *hour, uint8_t *min, uint8_t *sec)
  tamper event event acquisition
- void **rtc_adjust_mode_set** (RTC_T *RTC, uint8_t adjust_frequency, uint8_t adjust_value)
  adjustment compensation setting
- void **rtc_adjust_en_set** (RTC_T *RTC, uint8_t adjust_enable)
  adjustment compensation enable
- void **rtc_check_mode_set** (RTC_T *RTC, uint8_t check_mode)
  compensation calibration settings
- void **rtc_check_en_set** (RTC_T *RTC, uint8_t check_enable)
  check pulse(vld_on_o) output enable

1.27.2 函数说明

1.27.2.1 void rtc_adjust_en_set (RTC_T * *RTC*, uint8_t *adjust_enable*)

adjustment compensation enable

参数:

**RTC*	Pointer to RTC_T structure
adjust_enable	Adjustment compensation enable or disable

返回:

none

1.27.2.2 void rtc_adjust_mode_set (RTC_T * *RTC*, uint8_t *adjust_frequency*, uint8_t *adjust_value*)

adjustment compensation setting

参数:

**RTC*	Pointer to RTC_T structure
adjust_frequency	Adjust compensation frequency
adjust_value	Adjustment compensation value setting

返回:

none

1.27.2.3 void rtc_alarm1_en_set (RTC_T * *RTC*, uint32_t *condition*, uint8_t *alarm_enable*)

rtc alarm clock 1 enable setting

参数:

**RTC*	Pointer to RTC_T structure
condition	Alarm 1 setting enable
alarm_enable	Alarm 1 enable or disable

返回:

none

1.27.2.4 void rtc_alarm1_time_set (RTC_T * *RTC*, uint16_t *year*, uint8_t *month*, uint8_t *day*, uint8_t *week*, uint8_t *hour*, uint8_t *min*, uint8_t *sec*, uint8_t *centisec*)

RTC alarm clock 1 time setting

参数:

**RTC*	pointer to RTC_T structure
year	00 ~ 99
month	1 ~ 12
day	1 ~ 31
week	1 ~ 7
hour	0 ~ 23
min	0 ~ 59
sec	0 ~ 59
centisec	0 ~ 99

返回:

none

函数调用图:

1.27.2.5 void rtc_alarm2_en_set (RTC_T * *RTC*, uint8_t *alarm_enable*)

RTC alarm clock 2 enable setting

参数:

**RTC*	Pointer to RTC_T structure
alarm_enable	Alarm clock 2 enable or disable

返回:

none

1.27.2.6 void rtc_alarm2_time_set (RTC_T * *RTC*, uint32_t *time*)

RTC alarm clock 2 time setting

参数:

**RTC*	Pointer to RTC_T structure
time	Alarm 2 interrupt output cycle

返回:

none

1.27.2.7 void RTC_ALARM_IRQHandler (void)

alarm interrupt service function

参数:

none	

返回:

none

1.27.2.8 uint32_t rtc_bk_read_word (RTC_T * *RTC*, uint8_t *bk_area*)

RTC backup register reads a character

参数:

**RTC*	Pointer to RTC_T structure
bk_area	Backup register to operate

返回:

rtc->bkreg[bk_area] value of read backup register

1.27.2.9 void rtc_bk_wirte_word (RTC_T * *RTC*, uint32_t *word*, uint8_t *bk_area*)

write a character to the rtc backup register

参数:

**RTC*	Pointer to RTC_T structure
word	Characters written
bk_area	Backup register to operate

返回:

none

1.27.2.10 void rtc_bk_wirte_words (RTC_T * *RTC*, uint32_t * *word*, uint8_t *bk_start_area*, uint8_t *length*)

data written by rtc backup register

参数:

**RTC*	Pointer to RTC_T structure
**word*	Data to be written
bk_start_area	Start writing from the specified backup register
length	Length of written data

返回:

none

1.27.2.11 void rtc_calendar_time_get (RTC_T * *RTC*, uint8_t * *year*, uint8_t * *month*, uint8_t * *day*, uint8_t * *week*, uint8_t * *hour*, uint8_t * *min*, uint8_t * *sec*, uint8_t * *centisec*, uint8_t * *am_pm_centisec*)

rtc calendar time acquisition

参数:

**RTC*	Pointer to RTC_T structure
**year*	00 ~ 99
**month*	1 ~ 12
**day*	1 ~ 31
**week*	1 ~ 7
**hour*	0 ~ 23
**min*	0 ~ 59
**sec*	0 ~ 59
**centisec*	0 ~ 99
**am_pm_centisec*	It is used to express morning, afternoon and centisecond

返回:

none

1.27.2.12 void rtc_calendar_time_init (RTC_T * *RTC*, uint16_t *year*, uint8_t *month*, uint8_t *day*, uint8_t *week*, uint8_t *hour*, uint8_t *min*, uint8_t *sec*, uint8_t *centisec*)

RTC calendar time initialization

参数:

**RTC*	Pointer to RTC_T structure
year	00 ~ 99
month	1 ~ 12
day	1 ~ 31
week	1 ~ 7
hour	0 ~ 23
min	0 ~ 59
sec	0 ~ 59
centisec	0 ~ 99

返回:

none

函数调用图:

1.27.2.13 void rtc_calendat_time_set (uint16_t *year*, uint8_t *month*, uint8_t *day*, uint8_t *week*, uint8_t *hour*, uint8_t *min*, uint8_t *sec*, uint8_t *centisec*, uint32_t * *date*, uint32_t * *time*)

RTC calendar time setting

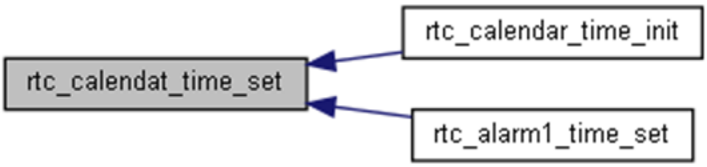
参数:

year	00 ~ 99
month	1 ~ 12
day	1 ~ 31
week	1 ~ 7
hour	0 ~ 23
min	0 ~ 59
sec	0 ~ 59
centisec	0 ~ 99
**date*	Write the date register after storing the date
**time*	Write the time register after storing the time

返回:

none

函数的调用关系图:



1.27.2.14 void rtc_check_en_set (RTC_T * *RTC*, uint8_t *check_enable*)

check pulse(vld_on_o) output enable

参数:

**RTC*	Pointer to RTC_T structure
check_enable	Check output enable or disable

返回:

none

1.27.2.15 void rtc_check_mode_set (RTC_T * *RTC*, uint8_t *check_mode*)

compensation calibration settings

参数:

**RTC*	Pointer to RTC_T structure
check_mode	Check cycle selection

返回:

none

1.27.2.16 void rtc_disable (void)

RTC disable

参数:

none	

返回:

none

1.27.2.17 void rtc_enable (void)

RTC enable

参数:

none	

返回:

none

1.27.2.18 uint8_t rtc_int_raw_state_read (RTC_T * *RTC*, uint8_t *irq_mode*)

bbu original interrupt status reading

参数:

**RTC*	Pointer to RTC_T structure
irq_mode	Interrupt mode

返回:

rtc->bbu_int_raw & irq_mode return to interrupt status

1.27.2.19 void rtc_int_state_clear (RTC_T * *RTC*, uint8_t *irq_mode*)

bbu interrupt status clear

参数:

**RTC*	Pointer to RTC_T structure
irq_mode	Interrupt mode

返回:

none

函数的调用关系图:

1.27.2.20 uint8_t rtc_int_state_read (RTC_T * *RTC*, uint8_t *irq_mode*)

interrupt status reading after bbu enable

参数:

**RTC*	Pointer to RTC_T structure
irq_mode	Interrupt mode

返回:

rtc->bbu_int_sta & irq_mode return to interrupt status

1.27.2.21 void rtc_irq_init (RTC_T * *RTC*, uint8_t *irq_mode*, uint8_t *irq_enable*, void(*)() *pfunc*)

RTC interrupt initialization

参数:

**RTC*	Pointer to RTC_T structure
irq_mode	Interrupt mode
irq_enable	Interrupt enable or disable
*(*pfunc)()*	Interrupt callback function

返回:

none

1.27.2.22 void rtc_tamper_cnt_clear (RTC_T * *RTC*)

clear tamper counter

参数:

**RTC*	Pointer to RTC_T structure

返回:

none

1.27.2.23 uint8_t rtc_tamper_cnt_read (RTC_T * *RTC*)

read the value of the tamper counter

参数:

**RTC*	Pointer to RTC_T structure

返回:

(rtc->tamp_cnt & 0x00000003f) times of tampering

1.27.2.24 void rtc_tamper_en_set (RTC_T * *RTC*, uint8_t *tamper_enable*)

enable tamper detection

参数:

**RTC*	Pointer to RTC_T structure
tamper_enable	RTC tamper enable or disable

返回:

none

1.27.2.25 void rtc_tamper_set (RTC_T * *RTC*, uint8_t *edge*)

tamper trigger edge setting

参数:

**RTC*	Pointer to RTC_T structure
edge	Tamper detection edge setting

返回:

none

1.27.2.26 void rtc_tamper_time_get (RTC_T * *RTC*, uint8_t *tamperx*, uint8_t * *year*, uint8_t * *month*, uint8_t * *day*, uint8_t * *hour*, uint8_t * *min*, uint8_t * *sec*)

tamper event event acquisition

参数:

**RTC*	Pointer to RTC_T structure
tamperx	Tamper event selection
**year*	00 ~ 99
**month*	1 ~ 12
**day*	1 ~ 31
**hour*	0 ~ 23
**min*	0 ~ 59
**sec*	0 ~ 59

返回:

none

1.27.2.27 void rtc_up_test (RTC_T * *RTC*)

rtc count up test

参数:

**RTC*	Pointer to RTC_T structure

返回:

none

1.27.2.28 void TAMPSTAMP_IRQHandler (void)

tamper interrupt service function

参数:

none	

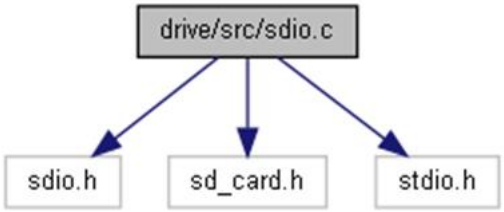
返回:

none

函数调用图:

1.28 SDIO接口

SDIO driver source file
#include "sdio.h"
#include "sd_card.h"
#include "stdio.h"
sdio.c 的引用(Include)关系图:



1.28.1 函数

- void **SDIO0_IRQHandler** (void)
  SDIO interrupt handling.
- void **sdio_clk_init** (SDIO_T *SDIO, BOOL newstate)
  SDIO clk init
- void **sdio_initial** (SDIO_T *SDIO)
  SDIO initial
- void **sdio_irq_init** (SDIO_T *SDIO, uint8_t irq_enable, void(*sdio_int)())
  SDIO irq init
- void **sdio_width_config** (SDIO_T *SDIO, uint8_t sdio_index, uint32_t wid_bus)
  SDIO width config
- void **sdio_set_clock** (SDIO_T *SDIO, uint8_t sdio_index, uint32_t clk_src, uint32_t div)
  SDIO set clock
- void **sdio_power_config** (SDIO_T *SDIO, uint8_t sdio_index, uint32_t pwr_sta)
  SDIO config power register
- void **sdio_clock_config** (SDIO_T *SDIO, uint8_t sdio_index, uint32_t clock_sta)
  SDIO config clock register
- void **sdio_send_cmd** (SDIO_T *SDIO, uint8_t sdio_index, uint32_t cmd_index, uint32_t cmd_arg, uint8_t waitresp)
  SDIO send cmd
- void **sdio_send_data_config** (SDIO_T *SDIO, uint32_t data_len, uint16_t block_size)
  SDIO send data config
- uint32_t **sdio_recv_byte** (SDIO_T *SDIO)
  SDIO recv byte

1.28.2 函数说明

1.28.2.1 void SDIO0_IRQHandler (void)

SDIO interrupt handling.

参数:

none	

返回:

none

1.28.2.2 void sdio_clk_init (SDIO_T * *SDIO*, BOOL *newstate*)

SDIO clk init

参数:

**SDIO*	pointer to SDIO_T structure
newstate	SDIO_ENABLE / SDIO_DISABLE

返回:

none

1.28.2.3 void sdio_clock_config (SDIO_T * *SDIO*, uint8_t *sdio_index*, uint32_t *clock_sta*)

SDIO config clock register

参数:

**SDIO*	pointer to SDIO_T structure
sdio_index	SDIO index
clock_sta	clock register

返回:

none

1.28.2.4 void sdio_initial (SDIO_T * *SDIO*)

SDIO initial

参数:

**SDIO*	pointer to SDIO_T structure

返回:

none

1.28.2.5 void sdio_irq_init (SDIO_T * *SDIO*, uint8_t *irq_enable*, void(*)() *sdio_int*)

SDIO irq init

参数:

**SDIO*	pointer to SDIO_T structure
irq_enable	interrupt enable/disable
*(*sdio_int)()*	interrupt callback function

返回:

none

1.28.2.6 void sdio_power_config (SDIO_T * *SDIO*, uint8_t *sdio_index*, uint32_t *pwr_sta*)

SDIO config power register

参数:

**SDIO*	pointer to SDIO_T structure
sdio_index	SDIO index
pwr_sta	power register

返回:

none

1.28.2.7 uint32_t sdio_recv_byte (SDIO_T * *SDIO*)

SDIO recv byte

参数:

**SDIO*	pointer to SDIO_T structure

返回:

none

1.28.2.8 void sdio_send_cmd (SDIO_T * *SDIO*, uint8_t *sdio_index*, uint32_t *cmd_index*, uint32_t *cmd_arg*, uint8_t *waitresp*)

SDIO send cmd

参数:

**SDIO*	pointer to SDIO_T structure
sdio_index	SDIO index
cmd_index	cmd index
cmd_arg	clock cmd para

waitresp	respond type

返回:

none

1.28.2.9 void sdio_send_data_config (SDIO_T * *SDIO*, uint32_t *data_len*, uint16_t *block_size*)

SDIO send data config

参数:

**SDIO*	pointer to SDIO_T structure
data_len	data lenth
block_size	block_size block size

返回:

none

1.28.2.10 void sdio_set_clock (SDIO_T * *SDIO*, uint8_t *sdio_index*, uint32_t *clk_src*, uint32_t *div*)

SDIO set clock

参数:

**SDIO*	pointer to SDIO_T structure
sdio_index	SDIO index
clk_src	reserved
div	clk div

返回:

none

1.28.2.11 void sdio_width_config (SDIO_T * *SDIO*, uint8_t *sdio_index*, uint32_t *wid_bus*)

SDIO width config

参数:

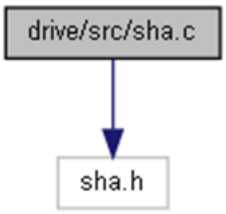
**SDIO*	pointer to SDIO_T structure
sdio_index	SDIO index
wid_bus	bus width

返回:

none

1.29 SHA接口

SHA driver source file
#include "sha.h"
sha.c 的引用(Include)关系图:



1.29.1 函数

- void **SHA_IRQHandler** (void)
  SHA_IRQHandler.
- void **sha_irq_init** (BOOL newstate, void(*sha_isr)())
  SHA interrupt initial
- void **sha_clk_init** (BOOL newstate)
  enable/disable SHA clock, meanwhile release/enable sha reset status
- void **sha_clk_cmd** (BOOL newstate)
  enable/disable SHA clock
- void **sha_reset** (void)
  SHA reset

1.29.2 函数说明

1.29.2.1 void sha_clk_cmd (BOOL *newstate*)

enable/disable SHA clock

参数:

newstate	clock status This parameter can be one of the following value ● 0:disable. ● 1:enable.

返回:

none

1.29.2.2 void sha_clk_init (BOOL *newstate*)

enable/disable SHA clock, meanwhile release/enable SHA reset status

参数:

newstate	clock and reset status This parameter can be one of the following value: ● 0:disable. ● 1:enable.

返回:

none

1.29.2.3 void sha_irq_init (BOOL *newstate*, void(*)() *sha_isr*)

SHA interrupt initial

参数:

newstate	interrupt status This parameter can be one of the following value ● 0:disable ● 1:enable
*(*sha_isr)()*	interrupt service runtine function

返回:

none

1.29.2.4 void SHA_IRQHandler (void)

SHA_IRQHandler.

参数:

none	

返回:

none

1.29.2.5 void sha_reset (void)

SHA reset

参数:

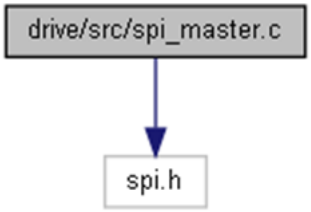
none	

返回:

none

1.30 SPI_master接口

SPI master driver source file
#include "spi.h"
spi_master.c 的引用(Include)关系图:



1.30.1 函数

- void **SPI0_IRQHandler** (void)
  SPI0 interrupt handling
- void **SPI1_IRQHandler** (void)
  SPI1 interrupt handling
- void **SPI2_IRQHandler** (void)
  SPI2 interrupt handling
- void **SPI3_IRQHandler** (void)
  SPI3 interrupt handling
- void **spi_clock_init** (SPI_T *SPIx, BOOL newstate)
  Initializes the clock of the spi.
- void **spi_work_mode_init** (SPI_T *SPIx, SPI_WORK_WAY_T work_way, SPI_WORK_MODE_T work_mode)
  Initializes the work mode for spi.
- void **spi_master_init** (SPI_T *SPIx, SPI_WORK_WAY_T work_way, SPI_WORK_MODE_T mode, SPI_FIRSTBIT_T firstbit, uint16_t sclk_div)
  Initializes the SPI master.
- void **spi_send_ctrl** (SPI_T *SPIx, FUNC_E enable)
  spi_send_ctrl
- void **spi_receive_ctrl** (SPI_T *SPIx, FUNC_E enable)
  spi_receive_ctrl
- void **spi_en_ctrl** (SPI_T *SPIx, FUNC_E enable)
  spi_en_ctrl
- void **spi_irq_init** (SPI_T *SPIx, IRQn_Type irqn_type, uint32_t spi_intr, void(*pfunc)(), FUNC_E irq_enable)
  Initializes for thr spi interrupt.
- void **spi_cs_enable** (SPI_T *SPIx, SPI_CS_LEVEL_T level)

Set cs signal of spi.

- uint8_t **spi_send_byte** (SPI_T *SPIx, uint8_t data)

  SPI send byte

- uint8_t **spi_receive_byte** (SPI_T *SPIx)

  SPI receive byte

- uint8_t **spi_write_read_byte** (SPI_T *SPIx, uint8_t byte)

  spi_write_read_byte

1.30.2 函数说明

1.30.2.1 void SPI0_IRQHandler (void)

SPI0 interrupt handling

参数:

none	

返回:

none

1.30.2.2 void SPI1_IRQHandler (void)

SPI1 interrupt handling

参数:

none	

返回:

none

1.30.2.3 void SPI2_IRQHandler (void)

SPI2 interrupt handling

参数:

none	

返回:

none

1.30.2.4 void SPI3_IRQHandler (void)

SPI3 interrupt handling

参数:

none	

返回:

none

1.30.2.5 void spi_clock_init (SPI_T * *SPIx*, BOOL *newstate*)

Initializes the clock of the spi.

参数:

**SPIx*	pointer to SPI_T structure
newstate	ENABLE / DISABLE

返回:

none

1.30.2.6 void spi_cs_enable (SPI_T * *SPIx*, SPI_CS_LEVEL_T *level*)

Set cs signal of spi.

参数:

**SPIx*	pointer to SPI_T structure
level	This parameter is to set the level of cs signal

返回:

none

1.30.2.7 void spi_en_ctrl (SPI_T * *SPIx*, FUNC_E *enable*)

spi_en_ctrl

参数:

**SPIx*	pointer to SPI_T structure
enable	This parameter can be ENABLE or DISABLE.

返回:

none

1.30.2.8 void spi_irq_init (SPI_T * *SPIx*, IRQn_Type *irqn_type*, uint32_t *spi_intr*, void(*)() *pfunc*, FUNC_E *irq_enable*)

Initializes for thr spi interrupt.

参数:

*SPIx	pointer to SPI_T structure
irqn_type	Enable spi interrupt
spi_intr	Enable spi interrupt type
void	(*pfunc)() Interrupt callback function
irq_enable	This parameter can be ENABLE or DISABLE.

返回:

none

1.30.2.9 void spi_master_init (SPI_T * *SPIx*, SPI_WORK_WAY_T *work_way*, SPI_WORK_MODE_T *mode*, SPI_FIRSTBIT_T *firstbit*, uint16_t *sclk_div*)

Initializes the spi master.

参数:

*SPIx	pointer to SPI_T structure
work_way	spi work in TI mode or MOTOROLA
mode	select of working mode under TI or MOTOROLA
firstbit	MSB / LSB
sclk_div	This parameter is to set the working rate of spi.

返回:

none

函数调用图:

1.30.2.10 uint8_t spi_receive_byte (SPI_T * *SPIx*)

spi receive byte

参数:

*SPIx	pointer to SPI_T structure

返回:

rxreg

1.30.2.11 void spi_receive_ctrl (SPI_T * *SPIx*, FUNC_E *enable*)

spi_receive_ctrl

参数:

*SPIx	pointer to SPI_T structure
enable	This parameter can be ENABLE or DISABLE.

返回:

none

1.30.2.12 uint8_t spi_send_byte (SPI_T * *SPIx*, uint8_t *data*)

SPI send byte

参数:

**SPIx*	pointer to SPI_T structure
data	This parameter is the data to be sent.

返回:

none

1.30.2.13 void spi_send_ctrl (SPI_T * *SPIx*, FUNC_E *enable*)

spi_send_ctrl

参数:

**SPIx*	pointer to SPI_T structure
enable	This parameter can be ENABLE or DISABLE.

返回:

none

1.30.2.14 void spi_work_mode_init (SPI_T * *SPIx*, SPI_WORK_WAY_T *work_way*, SPI_WORK_MODE_T *work_mode*)

Initializes the work mode for spi.

参数:

**SPIx*	pointer to SPI_T structure
work_way	spi work in TI mode or MOTOROLA
work_mode	select of working mode under TI or MOTOROLA

返回:

none

函数的调用关系图:

1.30.2.15 uint8_t spi_write_read_byte (SPI_T * *SPIx*, uint8_t *byte*)

spi_write_read_byte

参数:

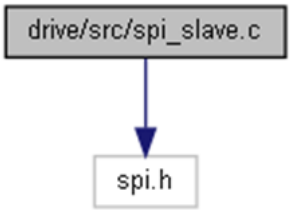
**SPIx*	pointer to SPI_T structure
byte	Bytes to write.

返回:

return the read data

1.31 SPI_slave接口

SPI slave driver source file
#include "spi.h"
spi_slave.c 的引用(Include)关系图:



1.31.1 函数

- void **SPI0_IRQHandler** (void)
  SPI0 interrupt handling
- void **SPI1_IRQHandler** (void)
  SPI1 interrupt handling
- void **SPI2_IRQHandler** (void)
  SPI2 interrupt handling
- void **SPI3_IRQHandler** (void)
  SPI3 interrupt handling
- void **spi_clock_init** (SPI_T *SPIx, BOOL newstate)
  Initializes the clock of the spi.
- void **spi_work_mode_init** (SPI_T *SPIx, SPI_WORK_WAY_T work_way, SPI_WORK_MODE_T work_mode)
  Initializes the work mode for SPI.
- void **spi_slave_init** (SPI_T *SPIx, SPI_WORK_WAY_T work_way, SPI_WORK_MODE_T mode, SPI_FIRSTBIT_T firstbit)
  Initializes the SPI slave.
- void **spi_irq_init** (SPI_T *SPIx, IRQn_Type irqn_type, uint32_t spi_intr, void(*pfunc)(), FUNC_E irq_enable)
  Initializes the SPI interrupt.
- uint8_t **spi_send_byte** (SPI_T *SPIx, uint8_t data)
  SPI send byte
- uint8_t **spi_receive_byte** (SPI_T *SPIx)
  spi receive byte
- uint8_t **spi_write_read_byte** (SPI_T *SPIx, uint8_t byte)
  spi_write_read_byte
- void **spi_send_ctrl** (SPI_T *SPIx, FUNC_E enable)

spi_send_ctrl

- void **spi_receive_ctrl** (SPI_T *SPIx, FUNC_E enable)

  spi_receive_ctrl

- void **spi_en_ctrl** (SPI_T *SPIx, FUNC_E enable)

  spi_en_ctrl

1.31.2 函数说明

1.31.2.1 void SPI0_IRQHandler (void)

SPI0 interrupt handling

参数:

none	

返回:

none

1.31.2.2 void SPI1_IRQHandler (void)

SPI1 interrupt handling

参数:

none	

返回:

none

1.31.2.3 void SPI2_IRQHandler (void)

SPI2 interrupt handling

参数:

none	

返回:

none

1.31.2.4 void SPI3_IRQHandler (void)

SPI3 interrupt handling

参数:

none	

返回:

none

1.31.2.5 void spi_clock_init (SPI_T * *SPIx*, BOOL *newstate*)

Initializes the clock of the SPI.

参数:

**SPIx*	pointer to SPI_T structure
newstate	ENABLE / DISABLE

返回:

none

1.31.2.6 void spi_en_ctrl (SPI_T * *SPIx*, FUNC_E *enable*)

spi_en_ctrl

参数:

**SPIx*	pointer to SPI_T structure
enable	This parameter can be ENABLE or DISABLE.

返回:

none

1.31.2.7 void spi_irq_init (SPI_T * *SPIx*, IRQn_Type *irqn_type*, uint32_t *spi_intr*, void(*)() *pfunc*, FUNC_E *irq_enable*)

Initializes the spi interrupt.

参数:

**SPIx*	pointer to SPI_T structure
irqn_type	Enable spi interrupt
spi_intr	Enable spi interrupt type
void	(*pfunc)() Interrupt callback function
irq_enable	This parameter can be ENABLE or DISABLE.

返回:

none

1.31.2.8 uint8_t spi_receive_byte (SPI_T * *SPIx*)

SPI receive byte

参数:

*SPIx	pointer to SPI_T structure

返回:

rxreg

1.31.2.9 void spi_receive_ctrl (SPI_T * *SPIx*, FUNC_E *enable*)

spi_receive_ctrl

参数:

*SPIx	pointer to SPI_T structure
enable	This parameter can be ENABLE or DISABLE.

返回:

none

1.31.2.10 uint8_t spi_send_byte (SPI_T * *SPIx*, uint8_t *data*)

SPI send byte

参数:

*SPIx	pointer to SPI_T structure
data	This parameter is the data to be sent.

返回:

none

1.31.2.11 void spi_send_ctrl (SPI_T * *SPIx*, FUNC_E *enable*)

spi_send_ctrl

参数:

*SPIx	pointer to SPI_T structure
enable	This parameter can be ENABLE or DISABLE.

返回:

none

1.31.2.12 void spi_slave_init (SPI_T * *SPIx*, SPI_WORK_WAY_T *work_way*, SPI_WORK_MODE_T *mode*, SPI_FIRSTBIT_T *firstbit*)

Initializes the spi slave.

参数:

*SPIx	pointer to SPI_T structure
work_way	spi work in TI mode or MOTOROLA

mode	select of working mode under TI or MOTOROLA
firstbit	MSB / LSB

返回:

none

函数调用图:

1.31.2.13 void spi_work_mode_init (SPI_T * *SPIx*, SPI_WORK_WAY_T *work_way*, SPI_WORK_MODE_T *work_mode*)

Initializes the work mode for spi.

参数:

**SPIx*	pointer to SPI_T structure
work_way	spi work in TI mode or MOTOROLA
work_mode	select of working mode under TI or MOTOROLA

返回:

none

函数的调用关系图:

1.31.2.14 uint8_t spi_write_read_byte (SPI_T * *SPIx*, uint8_t *byte*)

spi_write_read_byte

参数:

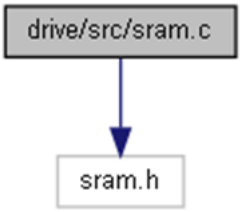
**SPIx*	pointer to SPI_T structure
byte	Bytes to write.

返回:

return the read data

1.32 SRAM接口

SRAM driver source file
#include "sram.h"
sram.c 的引用(Include)关系图:



1.32.1 函数

- BOOL **sram_erw_32bit** (uint32_t startpage, uint32_t endpage, uint32_t data)
  sram_erw_32bit
- BOOL **sram_erw_16bit** (uint32_t startpage, uint32_t endpage, uint16_t data)
  sram_erw_16bit
- BOOL **sram_erw_8bit** (uint32_t startpage, uint32_t endpage, uint8_t data)
  sram_erw_8bit
- void **sram_erw_init** (rw_mode_t rw_mode, uint32_t startpage, uint32_t endpage, uint32_t data)
  sram_erw_init

1.32.2 函数说明

1.32.2.1 BOOL sram_erw_16bit (uint32_t *startpage*, uint32_t *endpage*, uint16_t *data*)

sram_erw_16bit

参数:

startpage	This parameter is to set which page to start reading and writing from.
endpage	This parameter is to set which page to end reading and writing.
data	Data to be written

返回:

true or false

函数的调用关系图:

1.32.2.2 BOOL sram_erw_32bit (uint32_t *startpage*, uint32_t *endpage*, uint32_t *data*)

sram_erw_32bit

参数:

startpage	This parameter is to set which page to start reading and writing from.
endpage	This parameter is to set which page to end reading and writing.
data	Data to be written

返回:

true or false

函数的调用关系图:

1.32.2.3 BOOL sram_erw_8bit (uint32_t *startpage*, uint32_t *endpage*, uint8_t *data*)

sram_erw_8bit

参数:

startpage	This parameter is to set which page to start reading and writing from.
endpage	This parameter is to set which page to end reading and writing.
data	Data to be written

返回:

true or false

函数的调用关系图:

1.32.2.4 void sram_erw_init (rw_mode_t *rw_mode*, uint32_t *startpage*, uint32_t *endpage*, uint32_t *data*)

sram_erw_init

参数:

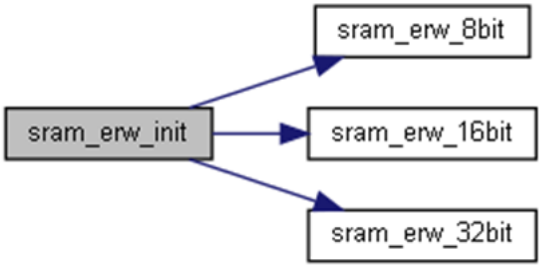
rw_mode	This parameter is to select the number of read and write bits of sram.

startpage	This parameter is to set which page to start reading and writing from.
endpage	This parameter is to set which page to end reading and writing.
data	Data to be written

返回:

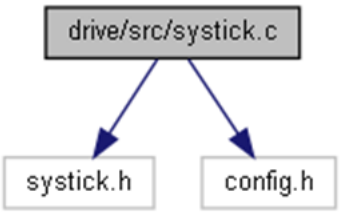
none

函数调用图:



1.33 SYSTICK接口

SYSTICK source file
#include "systick.h"
#include "config.h"
systick.c 的引用(Include)关系图:



1.33.1 函数

- void **systick_init** (SYSTICK_T *SYSTICK)
  SYSTICK init
- void **systick_irq_en** (SYSTICK_T *SYSTICK, void(*pfunc)())
  SYSTICK IRQ enable
- uint32_t **systick_get_flag** (SYSTICK_T *SYSTICK)
  get SYSTICK flag
- uint32_t **systick_get_load** (SYSTICK_T *SYSTICK)
  get SYSTICK load
- void **systick_set_load** (SYSTICK_T *SYSTICK, uint32_t load)
  set SYSTICK load
- uint32_t **systick_get_count** (SYSTICK_T *SYSTICK)
  get SYSTICK count
- void **systick_clear_count** (SYSTICK_T *SYSTICK)
  clear SYSTICK count
- void **SysTick_Handler** (void)
  SYSTICK Handler.

1.33.2 函数说明

1.33.2.1 void systick_clear_count (SYSTICK_T * *SYSTICK*)

clear SYSTICK count

参数:

**SYSTICK*	pointer to SYSTICK_T structure

返回:

none

1.33.2.2 uint32_t systick_get_count (SYSTICK_T * *SYSTICK*)

get SYSTICK count

参数:

**SYSTICK*	pointer to SYSTICK_T structure

返回:

cvr get SYSTICK cvr

1.33.2.3 uint32_t systick_get_flag (SYSTICK_T * *SYSTICK*)

get SYSTICK flag

参数:

**SYSTICK*	pointer to SYSTICK_T structure

返回:

csr

1.33.2.4 uint32_t systick_get_load (SYSTICK_T * *SYSTICK*)

get SYSTICK load

参数:

**SYSTICK*	pointer to SYSTICK_T structure

返回:

rvr get systick rvr

1.33.2.5 void SysTick_Handler (void)

SYSTICK Handler.

参数:

none	

返回:

none

1.33.2.6 void systick_init (SYSTICK_T * *SYSTICK*)

SYSTICK init

参数:

**SYSTICK*	pointer to SYSTICK_T structure

返回:

none

函数调用图:

1.33.2.7 void systick_irq_en (SYSTICK_T * *SYSTICK*, void(*)() *pfunc*)

SYSTICK IRQ enable

参数:

**SYSTICK*	pointer to SYSTICK_T structure
*(*pfunc)()*	This parameter is interrupt callback function

返回:

none

函数的调用关系图:

1.33.2.8 void systick_set_load (SYSTICK_T * *SYSTICK*, uint32_t *load*)

set SYSTICK load

参数:

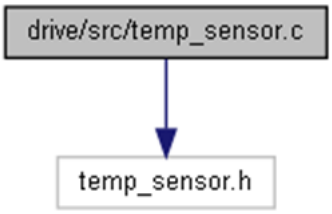
**SYSTICK*	pointer to SYSTICK_T structure
load	set systick load

返回:

none

1.34 temp_sensor接口

temp_sensor driver source file
#include "temp_sensor.h"
temp_sensor.c 的引用(Include)关系图:



1.34.1 函数

- void **TS_IRQHandler** (void)
  temp_sensor interrupt handling
- void **temp_sensor_irq_init** (FUNC_E irq_enable)
  Initializes for the temp_sensor interrupt.
- void **temp_sensor_init** (TS_MODE_T ts_mode, uint32_t clk_data)
  Initializes for temp_sensor.
- float **calculate_temp** (uint16_t code)
  Calculated temperature sensor calculates temperature.
- uint16_t **temp_sensor_wait_data** (TS_MODE_T ts_mode)
  temp_sensor_wait_data

1.34.2 函数说明

1.34.2.1 float calculate_temp (uint16_t *code*)

Calculated temperature sensor calculates temperature.

参数:

code	: Real-time data acquired by temperature sensor

返回:

Temperature sensor converts temperature value.

1.34.2.2 void temp_sensor_init (TS_MODE_T *ts_mode*, uint32_t *clk_data*)

Initializes for temp_sensor.

参数:

ts_mode	Selection of temperature sensor working mode
clk_data	Setting of the chopping clock frequency

返回:

none

1.34.2.3 void temp_sensor_irq_init (FUNC_E *irq_enable*)

Initializes for the temp_sensor interrupt.

参数:

irq_enable	This parameter can be DISABLE or ENABLE.

返回:

none

1.34.2.4 uint16_t temp_sensor_wait_data (TS_MODE_T *ts_mode*)

temp_sensor_wait_data

参数:

ts_mode	Selection of temperature sensor working mode

返回:

The value of the Temperature sensor gain .

1.34.2.5 void TS_IRQHandler (void)

temp_sensor interrupt handling

参数:

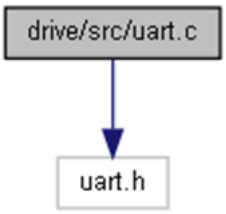
none	

返回:

none

1.35 UART接口

UART driver source file
#include "uart.h"
uart.c 的引用(Include)关系图:



1.35.1 函数

- void **UART0_IRQHandler** (void)
  UART0 interrupt handling
- void **UART1_IRQHandler** (void)
  UART1 interrupt handling
- void **UART2_IRQHandler** (void)
  UART2 interrupt handling
- void **UART3_IRQHandler** (void)
  UART3 interrupt handling
- void **UART4_IRQHandler** (void)
  UART4 interrupt handling
- void **UART5_IRQHandler** (void)
  UART5 interrupt handling
- void **uart_clk_init** (UART_T *UARTx, BOOL newstate)
  UART clk init
- void **uart_init** (UART_T *UARTx, uint32_t sys_clk_hz, uint32_t baud_rate)
  UART initial for baud_rate
- void **uart_irq_init** (UART_T *UARTx, uint8_t irq_enable, void(*uart_recv)())
  Enables or disables the specified UART interrupts
- void **uart_set_baud_rate** (UART_T *UARTx, uint32_t clk_hz, uint32_t baud_rate)
  set UART baud rate
- void **uart_send_byte** (UART_T *UARTx, uint8_t c)
  uart send byte
- void **uart_send_bytes** (UART_T *UARTx, uint8_t *buff, uint32_t length)
  UART send bytes
- uint8_t **uart_recv_byte** (UART_T *UARTx)
  UART receive byte

- void **uart_rec_bytes** (UART_T *UARTx, uint8_t *buff, uint32_t length)

  UART receive bytes

1.35.2 函数说明

1.35.2.1 void UART0_IRQHandler (void)

UART0 interrupt handling

参数:

none	

返回:

none

1.35.2.2 void UART1_IRQHandler (void)

UART1 interrupt handling

参数:

none	

返回:

none

1.35.2.3 void UART2_IRQHandler (void)

UART2 interrupt handling

参数:

none	

返回:

none

1.35.2.4 void UART3_IRQHandler (void)

UART3 interrupt handling

参数:

none	

返回:

none

1.35.2.5 void UART4_IRQHandler (void)

UART4 interrupt handling

参数:

none	

返回:

none

1.35.2.6 void UART5_IRQHandler (void)

UART5 interrupt handling

参数:

none	

返回:

none

1.35.2.7 void uart_clk_init (UART_T * *UARTx*, BOOL *newstate*)

UART clk init

参数:

**UARTx*	Pointer to UART_T structure.
newstate	Clock and reset status. This parameter can be one of the following values: ● UART_ENABLE: enable uartx clock and set it into work mode. ● UART_DISABLE: disable uartx clock and set uartx into reset mode.

返回:

none

1.35.2.8 void uart_init (UART_T * *UARTx*, uint32_t *sys_clk_hz*, uint32_t *baud_rate*)

UART initial for baud_rate

参数:

**UARTx*	pointer to UART_T structure
sys_clk_hz	set UART used system clk
baud_rate	set UART communication data rate

返回:

none

函数调用图:

1.35.2.9 void uart_irq_init (UART_T * *UARTx*, uint8_t *irq_enable*, void(*)() *uart_recv*)

Enables or disables the specified UART interrupts.

参数:

**UARTx*	Pointer to UART_T structure,where x can be 0,1,2,3,4 or 5 to select the UART peripheral.
irq_enable	Interrupt status. This parameter can be one of the following values: ● ENABLE: enable interrupt. ● DISABLE：disable interrupt.
*void(*uart_recv)()*	Interrupt callback function.

返回:

none

1.35.2.10 void uart_rec_bytes (UART_T * *UARTx*, uint8_t * *buff*, uint32_t *length*)

UART receive bytes

参数:

**UARTx*	pointer to UART_T structure
**buff*	pointer to received data buff
length	set received data length

返回:

none

1.35.2.11 uint8_t uart_recv_byte (UART_T * *UARTx*)

UART receive byte

参数:

**UARTx*	pointer to UART_T structure

返回:

none

1.35.2.12 void uart_send_byte (UART_T * *UARTx*, uint8_t *c*)

UART send byte

参数:

**UARTx*	pointer to UART_T structure
c	set UART transfer data

返回:

none

函数的调用关系图:

1.35.2.13 void uart_send_bytes (UART_T * *UARTx*, uint8_t * *buff*, uint32_t *length*)

UART send bytes

参数:

**UARTx*	pointer to UART_T structure
**buff*	pointer to transfer buff
length	set UART transfer data length

返回:

none

函数调用图:

1.35.2.14 void uart_set_baud_rate (UART_T * *UARTx*, uint32_t *clk_hz*, uint32_t *baud_rate*)

set UART baud rate

参数:

**UARTx*	pointer to UART_T structure
clk_hz	set UART used system clk
baud_rate	set UART communication data rate

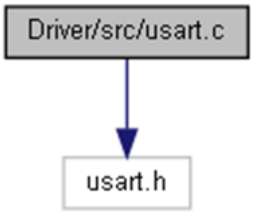
返回:

none

函数的调用关系图:

1.36 USART接口

USART driver source file
#include "usart.h"
usart.c 的引用(Include)关系图:



1.36.1 函数

- void USART6_IRQHandler (void)
  USART6 interrupt handling
- void **USART7_IRQHandler** (void)
  USART7 interrupt handling
- void **usart_clk_init** (USART_T *USARTx, BOOL newstate)
  USART clk init
- void **usart_init** (USART_T *USARTx, uint32_t sys_clk_hz, uint32_t baud_rate)
  USART initial
- void **usart_lin_init** (USART_T *USARTx, uint8_t work_mode, uint32_t sys_clk_hz, uint32_t baud_rate)
  USART lin initial
- void **usart_lin_master_send_data** (USART_T *USARTx, uint8_t idchr, uint8_t *buff, uint8_t length)
  USART lin master send data
- void **usart_lin_master_rece_data** (USART_T *USARTx, uint8_t idchr, uint8_t *buff, uint8_t length)
  USART lin master receive data
- uint8_t **usart_lin_slave_get_id** (USART_T *USARTx)
  USART lin slave get id
- void **usart_lin_slave_get_frame** (USART_T *USARTx, uint8_t *rece_data, uint8_t length)
  USART lin slave get frame
- void **usart_lin_slave_send_data** (USART_T *USARTx, uint8_t *send_data, uint8_t length)
  USART lin slave send data
- void **usart_spi_init** (USART_T *USARTx, uint8_t work_way, spi_work_mode_t mode, uint16_t sclk_div)
  USART SPI initial

- void **usart_irq_init** (USART_T *USARTx, uint8_t irq_enable, void(*usart_recv)())
  USART IRQ initial
- uint8_t **usart_spi_send_byte** (USART_T *USARTx, uint8_t data)
  USART SPI send data
- uint8_t **usart_spi_receive_byte** (USART_T *USARTx)
  USART SPI receive data
- void **usart_spi_cs_enable** (USART_T *USARTx, BOOL newstate)
  USART SPI cs enable
- void **usart_set_baud_rate** (USART_T *USARTx, uint32_t clk_hz, uint32_t baud_rate)
  USART set baud rate
- void **usart_send_byte** (USART_T *USARTx, uint8_t c)
  USART send byte
- void **usart_send_bytes** (USART_T *USARTx, uint8_t *buff, uint32_t length)
  USART send bytes
- uint8_t **usart_recv_byte** (USART_T *USARTx)
  USART receive byte

1.36.2 函数说明

1.36.2.1 void USART6_IRQHandler (void)

USART6 interrupt handling

参数:

none	

返回:

none

1.36.2.2 void USART7_IRQHandler (void)

USART7 interrupt handling

参数:

none	

返回:

none

1.36.2.3 void usart_clk_init (USART_T * *USARTx*, BOOL *newstate*)

USART clk init

参数:

**USARTx*	pointer to USART_T structure
newstate	Clock and reset status. This parameter can be one of the following values: ● USART_ENABLE: enable usartx clock and set it into work mode. ● USART_DISABLE: disable usartx clock and set usartx into reset mode.

返回:

none

1.36.2.4 void usart_init (USART_T * *USARTx*, uint32_t *sys_clk_hz*, uint32_t *baud_rate*)

USART initial

参数:

**USARTx*	pointer to USART_T structure
sys_clk_hz	set uart used system clk
baud_rate	set uart communication data rate

返回:

none

函数调用图:

1.36.2.5 void usart_irq_init (USART_T * *USARTx*, uint8_t *irq_enable*, void(*)() *usart_recv*)

USART IRQ initial

参数:

**USARTx*	pointer to USART_T structure
irq_enable	Interrupt status. This parameter can be one of the following values: ● ENABLE: enable interrupt. ● DISABLE：disable interrupt.
*void(*usart_recv)()*	call back function

返回:

none

1.36.2.6 void usart_lin_init (USART_T * *USARTx*, uint8_t *work_mode*, uint32_t *sys_clk_hz*, uint32_t *baud_rate*)

USART lin initial

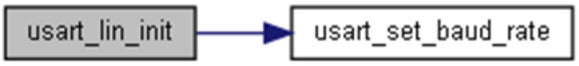
参数:

**USARTx*	pointer to USART_T structure
work_mode	set usart work mode
sys_clk_hz	set usart used system clk
baud_rate	set usart communication data rate

返回:

none

函数调用图:



1.36.2.7 void usart_lin_master_rece_data (USART_T * *USARTx*, uint8_t *idchr*, uint8_t * *buff*, uint8_t *length*)

USART lin master receive data

参数:

**USARTx*	pointer to USART_T structure
idchr	id to be sent
**buff*	receive data buff
length	receive data length

返回:

none

1.36.2.8 void usart_lin_master_send_data (USART_T * *USARTx*, uint8_t *idchr*, uint8_t * *buff*, uint8_t *length*)

USART lin master send data

参数:

**USARTx*	pointer to USART_T structure
idchr	id to be sent
**buff*	data buff to be sent
length	send data length

返回:

none

1.36.2.9 void usart_lin_slave_get_frame (USART_T * *USARTx*, uint8_t * *rece_data*, uint8_t *length*)

USART lin slave get frame

参数:

**USARTx*	pointer to USART_T structure
**rece_data*	receive data buff
length	receive data length

返回:

none

1.36.2.10 uint8_t usart_lin_slave_get_id (USART_T * *USARTx*)

USART lin slave get id

参数:

**USARTx*	pointer to USART_T structure

返回:

id

1.36.2.11 void usart_lin_slave_send_data (USART_T * *USARTx*, uint8_t * *send_data*, uint8_t *length*)

USART lin slave send data

参数:

**USARTx*	pointer to USART_T structure
**send_data*	data to send
length	send data length

返回:

none

1.36.2.12 uint8_t usart_recv_byte (USART_T * *USARTx*)

USART receive byte

参数:

**USARTx*	pointer to USART_T structure

返回:

receive byte

1.36.2.13 void usart_send_byte (USART_T * *USARTx*, uint8_t *c*)

USART send byte

参数:

**USARTx*	pointer to USART_T structure
c	char to send

返回:

none

函数的调用关系图:

1.36.2.14 void usart_send_bytes (USART_T * *USARTx*, uint8_t * *buff*, uint32_t *length*)

USART send bytes

参数:

**USARTx*	pointer to USART_T structure
**buff*	data
length	data length

返回:

none

函数调用图:

1.36.2.15 void usart_set_baud_rate (USART_T * *USARTx*, uint32_t *clk_hz*, uint32_t *baud_rate*)

USART set baud rate

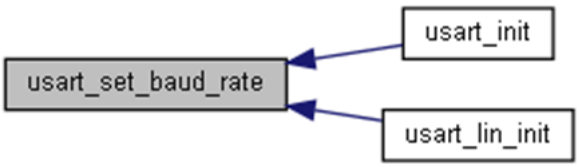
参数:

**USARTx*	pointer to USART_T structure
clk_hz	cpu fre
baud_rate	communication baud rate

返回:

none

函数的调用关系图:



1.36.2.16 void usart_spi_cs_enable (USART_T * *USARTx*, BOOL *newstate*)

USART SPI cs enable

参数:

**USARTx*	pointer to USART_T structure
newstate	USART_ENABLE / USART_DISABLE

返回:

none

1.36.2.17 void usart_spi_init (USART_T * *USARTx*, uint8_t *work_way*, spi_work_mode_t *mode*, uint16_t *sclk_div*)

USART SPI initial

参数:

**USARTx*	pointer to USART_T structure
work_way	master or slave
mode	mode 0-3
sclk_div	set clk div

返回:

none

1.36.2.18 uint8_t usart_spi_receive_byte (USART_T * *USARTx*)

USART SPI receive data

参数:

**USARTx*	pointer to USART_T structure

返回:

receive data or time out

1.36.2.19 uint8_t usart_spi_send_byte (USART_T * *USARTx*, uint8_t *data*)

USART SPI send data

参数:

**USARTx*	pointer to USART_T structure

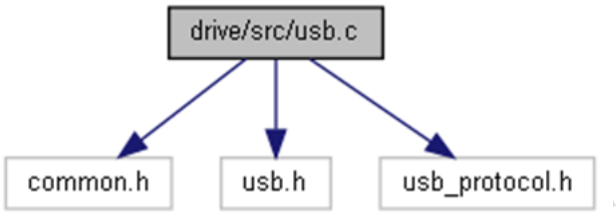
data	data to send

返回:

0 or 0xff

1.37 USB接口

USB driver source file
#include "common.h"
#include "usb.h"
#include "usb_protocol.h"
usb.c 的引用(Include)关系图:



1.37.1 函数

- void **USB0_IRQHandler** (void)
  USB interrupt handling
- void **usb_clk_init** (USB_T *USB, BOOL enable)
  USB clock initial
- void **usb_initial** (USB_T *USB)
  USB initial
- void **usb_irq_init** (USB_T *USB, uint8_t irq_enable, void(*callback)())
  USB irq initial
- void **usb_connect** (USB_T *USB)
  USB connect
- void **usb_disconnect** (USB_T *USB)
  USB disconnect
- void **usb_interrupt_disable** (USB_T *USB, uint32_t int_src)
  USB interrupt disable
- void **usb_interrupt_enable** (USB_T *USB, uint32_t int_src)
  USB interrupt enable
- void **usb_bus_reset** (USB_T *USB)
  USB bus reset
- void **usb_resume** (USB_T *USB)
  USB resume
- void **usb_suspend** (USB_T *USB)
  USB suspend
- void **usb_remote_wakeup** (USB_T *USB)

USB remote wakeup

- void **usb_start_ep_transfer** (USB_T *USB, uint32_t length, uint8_t ep_index)
  USB start endpoint transfer
- void **usb_ep0_send_empty_packet** (USB_T *USB)
  USB ep0 send empty packet
- uint16_t **usb_get_fifo_length** (USB_T *USB, uint8_t ep_index)
  USB get fifo length
- void **usb_clear_fifo** (USB_T *USB, uint8_t ep_index, uint8_t ep_dir)
  USB clear fifo
- void **usb_ep0_send_stall** (USB_T *USB)
  USB ep0 send stall
- void **usb_clear_stall** (USB_T *USB, uint8_t ep_index)
  USB clear stall
- void **usb_send_stall** (USB_T *USB, uint8_t ep_index)
  USB *send stall*
- void **usb_read_ep_mem8** (USB_T *USB, uint8_t *dst, uint32_t length, uint32_t fifo_offset, uint8_t ep_index)
  USB read ep memory
- void **usb_write_ep_mem8** (USB_T *USB, uint8_t *src, uint32_t length, uint32_t fifo_offset, uint8_t ep_index)
  USB write ep memory
- void **usb_set_rx_ready** (USB_T *USB, uint8_t ep_index)
  USB set receive ready
- void **usb_send_data** (USB_T *USB, uint8_t *buff, uint32_t length, uint8_t ep_index)
  USB send data
- void **usb_receive_data** (USB_T *USB, uint8_t *buff, uint32_t length, uint8_t ep_index)
  USB receive data

1.37.2 函数说明

1.37.2.1 void USB0_IRQHandler (void)

USB interrupt handling

参数:

none	

返回:

none

1.37.2.2 void usb_bus_reset (USB_T * *USB*)

USB bus reset

参数:

usb	pointer to USB_T structure

返回:

none

函数调用图:

1.37.2.3 void usb_clear_fifo (USB_T * *USB*, uint8_t *ep_index*, uint8_t *ep_dir*)

USB clear fifo

参数:

usb	pointer to USB_T structure
ep_index	endpoint index
ep_dir	endpoint dir

返回:

none

1.37.2.4 void usb_clear_stall (USB_T * *USB*, uint8_t *ep_index*)

USB clear stall

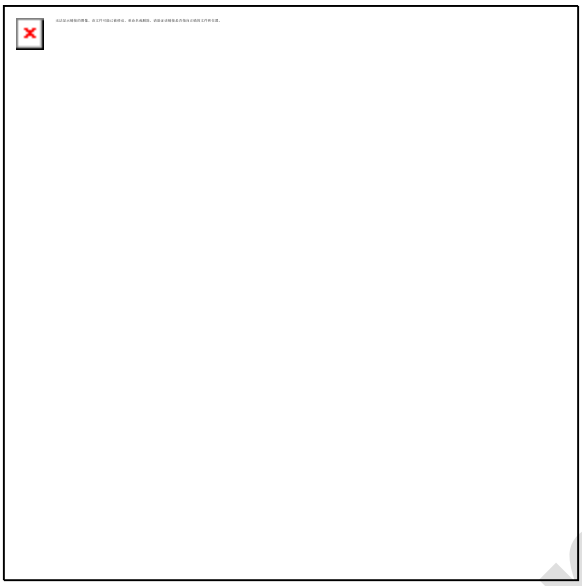
参数:

usb	pointer to USB_T structure
ep_index	endpoint index

返回:

none

函数的调用关系图:



1.37.2.5 void usb_clk_init (USB_T * *USB*, BOOL *enable*)

USB clock initial

参数:

usb	pointer to USB_T structure
enable	ENABLE or DISABLE This parameter can be one of the following values: ● USB_ENABLE: enable USB clock and set it into work mode. ● USB_DISABLE：disable USB clock and set uartx into reset mode.

返回:

none

1.37.2.6 void usb_connect (USB_T * *USB*)

USB connect

参数:

USB	pointer to USB_T structure

返回:

none

1.37.2.7 void usb_disconnect (USB_T * *USB*)

USB disconnect

参数:

USB	pointer to USB_T structure

返回:

none

1.37.2.8 void usb_ep0_send_empty_packet (USB_T * *USB*)

USB ep0 send empty packet

参数:

USB	pointer to USB_T structure

返回:

none

函数调用图:

1.37.2.9 void usb_ep0_send_stall (USB_T * *USB*)

USB ep0 send stall

参数:

USB	pointer to USB_T structure

返回:

none

1.37.2.10 uint16_t usb_get_fifo_length (USB_T * *USB*, uint8_t *ep_index*)

USB get fifo length

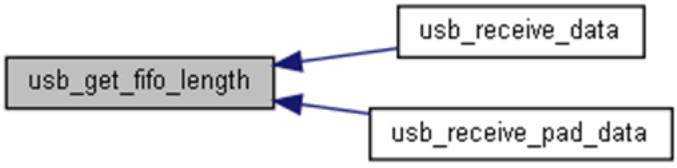
参数:

usb	pointer to USB_T structure
ep_index	endpoint index

返回:

fifo length

函数的调用关系图:



1.37.2.11 void usb_initial (USB_T * *USB*)

USB initial

参数:

usb	pointer to USB_T structure

返回:

none

1.37.2.12 void usb_interrupt_disable (USB_T * *USB*, uint32_t *int_src*)

USB interrupt disable

参数:

USB	pointer to USB_T structure
int_src	usb interrupt source

返回:

none

1.37.2.13 void usb_interrupt_enable (USB_T * *USB*, uint32_t *int_src*)

USB interrupt enable

参数:

USB	pointer to USB_T structure
int_src	usb interrupt source

返回:

none

1.37.2.14 void usb_irq_init (USB_T * *USB*, uint8_t *irq_enable*, void(*)() *callback*)

USB irq initial

参数:

USB	Pointer to USB_T structure
irq_enable	Interrupt enable/disable This parameter can be one of the following values: ● ENABLE: enable interrupt. ● DISABLE：disable interrupt.
callback	Interrupt callback function

返回:

none

1.37.2.15 void usb_read_ep_mem8 (USB_T * *USB*, uint8_t * *dst*, uint32_t *length*, uint32_t *fifo_offset*, uint8_t *ep_index*)

USB read ep memory

参数:

usb	pointer to USB_T structure
dst	data buff
length	data length
fifo_offset	fifo offset (unit:byte)
ep_index	endpoint index

返回:

none

函数的调用关系图:

1.37.2.16 void usb_receive_data (USB_T * *USB*, uint8_t * *buff*, uint32_t *length*, uint8_t *ep_index*)

USB receive data

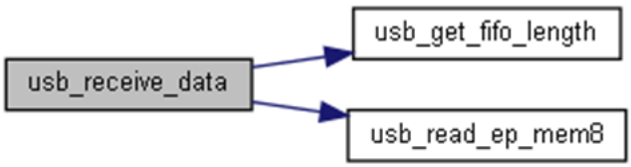
参数:

usb	pointer to USB_T structure
buff	data buff
length	data length
ep_index	endpoint index

返回:

none

函数调用图:



1.37.2.17 void usb_remote_wakeup (USB_T * *USB*)

USB remote wakeup

参数:

usb	pointer to USB_T structure

返回:

none

1.37.2.18 void usb_resume (USB_T * *USB*)

USB resume

参数:

USB	pointer to USB_T structure

返回:

none

1.37.2.19 void usb_send_data (USB_T * *USB*, uint8_t * *buff*, uint32_t *length*, uint8_t *ep_index*)

USB send data

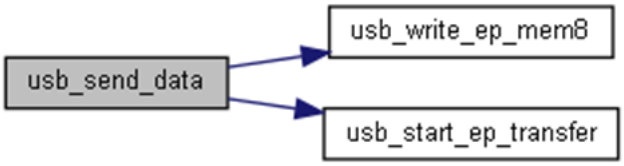
参数:

USB	pointer to USB_T structure
buff	data buff
length	data length
ep_index	endpoint index

返回:

none

函数调用图:



1.37.2.20 void usb_send_stall (USB_T * *USB*, uint8_t *ep_index*)

USB send stall

参数:

USB	pointer to USB_T structure
ep_index	endpoint index (EP1~4)

返回:

none

1.37.2.21 void usb_set_rx_ready (USB_T * *USB*, uint8_t *ep_index*)

USB set receive ready

参数:

USB	pointer to USB_T structure
ep_index	endpoint index

返回:

none

1.37.2.22 void usb_start_ep_transfer (USB_T * *USB*, uint32_t *length*, uint8_t *ep_index*)

USB start endpoint transfer

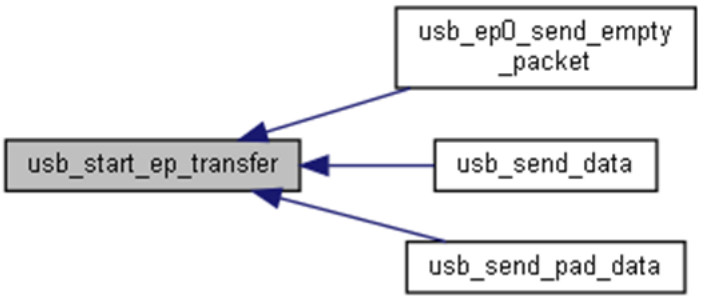
参数:

usb	pointer to USB_T structure
length	transfer data length
ep_index	endpoint index

返回:

none

函数的调用关系图:



1.37.2.23 void usb_suspend (USB_T * *USB*)

USB suspend

参数:

usb	pointer to USB_T structure

返回:

none

1.37.2.24 void usb_write_ep_mem8 (USB_T * *USB*, uint8_t * *src*, uint32_t *length*, uint32_t *fifo_offset*, uint8_t *ep_index*)

USB write ep memory

参数:

USB	pointer to USB_T structure
src	data buff
length	data length
fifo_offset	fifo offset (unit:byte)
ep_index	endpoint index

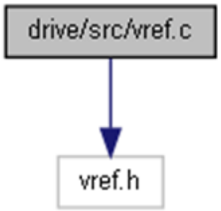
返回:

none

函数的调用关系图:

1.38 VREF接口

VREF driver source file
#include "vref.h"
vref.c 的引用(Include)关系图:



1.38.1 函数

- void **vref_init** (VREF_SEL_T vref_sel, uint32_t chop_data)

  Initializes for the vref.

1.38.2 函数说明

1.38.2.1 void vref_init (VREF_SEL_T *vref_sel*, uint32_t *chop_data*)

Initializes for the vref.

参数:

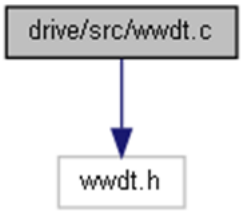
vref_sel	This parameter is to set the output gear of vref. This parameter can be one of the following values: ● VREF_SEL_0 Output voltage is equal to 1.5V ● VREF_SEL_1 Output voltage is equal to 2.0V ● VREF_SEL_2 Output voltage is equal to 2.5V ● VREF_SEL_3 Output voltage is equal to 3.0V
chop_data	This parameter is to set the frequency of the chopping clock of vref.

返回:

none

1.39 WWDT接口

WWDT driver source file
#include "wwdt.h"
wwdt.c 的引用(Include)关系图:



1.39.1 函数

- void **WWDT_IRQHandler** (void)
  WWDT interrupt handling
- void **wwdt_init** (WWDT_T *WWDT, uint8_t ov_time)
  WWDT initial
- void **wwdt_irq_init** (WWDT_T *WWDT, uint8_t irq_enable, void(*pfunc)())
  WWDT IRQ init
- void **wwdt_start** (WWDT_T *WWDT)
  WWDT start
- void **wwdt_feed** (WWDT_T *WWDT)
  WWDT feed
- uint16_t **wwdt_cnt_read** (WWDT_T *WWDT)
  WWDT cnt read
- uint16_t **pclk_div_cnt_read** (WWDT_T *WWDT)
  pclk div cnt read

1.39.2 函数说明

1.39.2.1 uint16_t pclk_div_cnt_read (WWDT_T * *WWDT*)

pclk div cnt read

参数:

**WWDT*	pointer to WWDT_T structure

返回:

wwdt->div_cnt pclk prescaler counter value

1.39.2.2 uint16_t wwdt_cnt_read (WWDT_T * *WWDT*)

WWDT cnt read

参数:

**WWDT*	pointer to WWDT_T structure

返回:

WWDT ->cnt WWDT counter value

1.39.2.3 void wwdt_feed (WWDT_T * *WWDT*)

WWDT feed

参数:

**WWDT*	pointer to WWDT_T structure

返回:

none

1.39.2.4 void wwdt_init (WWDT_T * *WWDT*, uint8_t *ov_time*)

WWDT initial

参数:

**WWDT*	pointer to WWDT_T structure
ov_time	WWDT overflow time

返回:

none

1.39.2.5 void wwdt_irq_init (WWDT_T * *WWDT*, uint8_t *irq_enable*, void(*)() *pfunc*)

WWDT IRQ init

参数:

**wwdt*	Pointer to WWDT_T structure
irq_enable	Interrupt status This parameter can be one of the following values: ● WWDT_IRQ_ENABLE Enable interrupt ● WWDT_IRQ_DISABLE Disable interrupt
*(*pfunc)()*	Interrupt callback function

返回:

none

1.39.2.6 void WWDT_IRQHandler (void)

WWDT interrupt handling

参数:

none	

返回:

none

1.39.2.7 void wwdt_start (WWDT_T * *WWDT*)

WWDT start

参数:

**WWDT*	pointer to WWDT_T structure

返回:

none