UM321xA API 参考手册

版本：V1.0

UNICMICRO
广芯微电子

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	2022.04.08	初始版

目录

1 应用程序接口（API）

1.1 ADC接口

#include "adc.h"

1.1.1 ADC_IRQHandler

功能	ADC_IRQHandler
描述	ADC中断处理函数
函数定义	void ADC_IRQHandler (void)
参数	none
返回	none

1.1.2 adc_init

功能	adc_init
描述	ADC初始化
函数定义	void adc_init (uint8_t vref_sel, uint32_t speed_div, uint8_t work_mode)
参数	uint8_t vref_sel：ADC参考电压源选择 uint32_t speed_div：ADC内部时钟分频器分频值 uint8_t work_mode：ADC工作模式选择(0:单次模式 1：连续模式)
返回	none

1.1.3 adc_close

功能	adc_close
描述	ADC模块关闭
函数定义	void adc_close(void)
参数	none
返回	none

1.1.4 adc_irq_init

功能	adc_irq_init
描述	ADC中断初始化
函数定义	void adc_irq_init(uint8_t irq_enable, uint32_t irq_type,void (*pfunc)(uint8_t ch))
参数	uint8_t irq_enable：ADC中断使能选择 uint32_t irq_type：ADC中断类型 void (*pfunc)()：ADC中断回调函数
返回	none

1.1.5 adc_channel_gpio_config

功能	adc_channel_gpio_config
描述	ADC通道配置
函数定义	void adc_channel_gpio_config(uint8_t channel, uint32_t gpio)
参数	uint8_t channel：ADC通道选择 uint32_t gpio：ADC通道对应IO管脚
返回	none

1.1.6 adc_convert_start

功能	adc_convert_start
描述	使能ADC转换
函数定义	void adc_convert_start(void)
参数	none
返回	none

1.1.7 adc_convert_stop

功能	adc_convert_stop
描述	停止ADC转换
函数定义	void adc_convert_stop (void)
参数	none
返回	none

1.1.8 adc_channel_cal

功能	adc_channel_cal
描述	通道转换，将当前通道的信息转换为实际的通道编号
函数定义	uint8_t adc_channel_cal(uint8_t ch_num)
参数	uint8_t ch_num：当前完成ADC采样的通道信息
返回	uint8_t ret：实际的通道编号

1.1.9 adc_irq_get_data

功能	adc_irq_get_data
描述	获取ADC通道数据//中断专用
函数定义	uint16_t adc_irq_get_data(uint8_t ch)
参数	uint8_t ch：ADC通道选择
返回	uint16_t ret：ADC通道有效数据及通道编号信息（高4位：通道信息、低12位：有效数据）

1.1.10 adc_get_data

功能	adc_get_data
描述	获取ADC通道数据//查询专用
函数定义	uint16_t adc_get_data(uint8_t ch_num)
参数	uint8_t ch_num：ADC通道选择
返回	uint8_t ret：ADC通道有效数据

1.2 SysTick接口

#include "systick.h"

#include "stdio.h"

1.2.1 systick_init

功能	systick_init
描述	SysTick初始化并使能时钟开始计数
函数定义	void systick_init(void (*pfunc)())
参数	none
返回	none

1.2.2 systick_get_flag

功能	systick_get_flag
描述	获取溢出标志
函数定义	uint32_t systick_get_flag(void)
参数	none
返回	none

1.2.3 systick_get_load

功能	systick_get_load
描述	获取重载值
函数定义	uint32_t systick_get_load(void)
参数	none
返回	none

1.2.4 systick_set_load

功能	systick_set_load
描述	设置重载值
函数定义	void systick_set_load(uint32_t load)
参数	none
返回	none

1.2.5 systick_get_count

功能	systick_get_count
描述	获取当前计数值
函数定义	uint32_t systick_get_count(void)
参数	none
返回	none

1.2.6 systick_clear_count

功能	systick_clear_count
描述	当前计数值清0
函数定义	void systick_clear_count(void)
参数	none
返回	none

1.2.7 SysTick_Handler

功能	SysTick_Handler
描述	中断处理函数
函数定义	void SysTick_Handler(void)
参数	none
返回	none

1.3 UART0接口

#include "uart0.h"

1.3.1 UART0_IRQHandler

功能	UART0_IRQHandler
描述	UART0中断处理函数
函数定义	void UART0_IRQHandler(void)
输出参数	none
返回	none

1.3.2 uart0_init

功能	uart0_init
描述	UART0波特率初始化
函数定义	void uart0_init(uint32_t sys_clk_hz, uint32_t baud_rate)

参数	uint32_t sys_clk_hz：系统时钟 uint32_t baud_rate：Series rate Baud Rate = sys_clk_hz / UARTBRP
返回	none

1.3.3 uart0_irq_init

功能	uart0_irq_init
描述	UART0中断初始化
函数定义	void uart0_irq_init(uint8_t irq_enable,void (*uart_recv)())
参数	uint8_t irq_enable：0 中断失能，1 中断使能 void (*uart_recv)()：接收处理回调函数
返回	none

1.3.4 uart0_set_baud_rate

功能	uart0_set_baud_rate
描述	UART0波特率设置
函数定义	void uart0_set_baud_rate(uint32_t sys_clk_hz, uint32_t baud_rate)
参数	uint32_t clk_hz：cpu frequency uint32_t baud_rate：Series rate Baud Rate = sys_clk_hz / UARTBRP
输出	none
返回	none

函数的调用关系图：

1.3.5 uart0_send_byte

功能	uart0_send_byte
描述	UART0发送一个字节数据
函数定义	void uart0_send_byte(uint8_t c)
参数	uint8_t c：发送的字节数据
返回	none

函数的调用关系图：

1.3.6 uart0_recv_byte

功能	uart0_recv_byte
描述	UART0接收一个字节数据
函数定义	uint8_t uart0_recv_byte(void)
参数	none
返回	data from uart

1.3.7 uart0_send_bytes

功能	uart0_send_bytes
描述	UART0发送多个字节
函数定义	void uart0_send_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buffer数据 uint32_t length：buffer长度
返回	none

1.3.8 uart0_rec_bytes

功能	uart0_rec_bytes
描述	UART0接收多个字节数据
函数定义	void uart0_rec_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buffer数据 uint32_t length：buffer长度
返回	none

1.4 UART1接口

#include "uart1.h"

1.4.1 UART1_IRQHandler

功能	UART1_IRQHandler
描述	UART1中断处理函数
函数定义	void UART1_IRQHandler(void)
输出参数	none
返回	none

1.4.2 uart1_init

功能	uart1_init
描述	UART1波特率初始化
函数定义	void uart1_init(uint32_t sys_clk_hz,uint32_t baud_rate)
参数	uint32_t sys_clk_hz：系统时钟 uint32_t baud_rate：Series rate Baud Rate = sys_clk_hz / UARTBRP
返回	none

1.4.3 uart1_irq_init

功能	uart1_irq_init
描述	UART1中断初始化
函数定义	void uart1_irq_init(uint8_t irq_enable, void (*uart_recv)())
参数	uint8_t irq_enable：0：中断失能，1：中断使能 void (*uart_recv)()：接收回调函数
返回	none

1.4.4 uart1_set_baud_rate

功能	uart1_set_baud_rate
描述	UART1波特率设置
函数定义	void uart1_set_baud_rate(uint32_t clk_hz, uint32_t baud_rate)
参数	uint32_t clk_hz：cpu frequency uint32_t baud_rate：Series rate Baud Rate = sys_clk_hz / UARTBRP
输出	none
返回	none

函数的调用关系图：

1.4.5 uart1_send_byte

功能	uart1_send_byte
描述	UART1发送一个字节数据
函数定义	void uart1_send_byte(uint8_t c)
参数	uint8_t c：发送的字节数据
返回	none

函数的调用关系图：

1.4.6 uart1_recv_byte

功能	uart1_recv_byte
描述	UART1接收一个字节数据
函数定义	uint8_t uart1_recv_byte(void)
参数	none
返回	data from uart

1.4.7 uart1_send_bytes

功能	uart1_send_bytes
描述	UART1发送多个字节
函数定义	void uart1_send_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buffer数据 uint32_t length：buffer长度
返回	none

1.4.8 uart1_rec_bytes

功能	uart1_rec_bytes
描述	UART1接收多个字节数据
函数定义	void uart1_rec_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buffer数据 uint32_t length：buffer长度
返回	none

1.5 CACHE接口

#include "cache.h"

1.5.1 cache_init

功能	cache_init
描述	cache初始化
函数定义	void cache_init(void)
参数	none
返回	none

1.5.2 cache_enable

功能	cache_enable
描述	cache使能

函数定义	void cache_enable(void)
参数	none
返回	none

1.5.3 cache_disable

功能	cache_disable
描述	关闭cache
函数定义	void cache_disable(void)
参数	none
返回	none

1.5.4 cache_clear

功能	cache_clear
描述	cache清除
函数定义	void cache_clear(void)
参数	none
返回	none

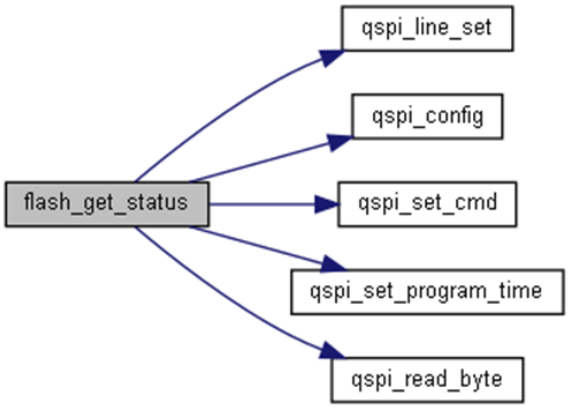
1.6 QSPI接口

#include "qspi.h"
#include "mx25l128.h"

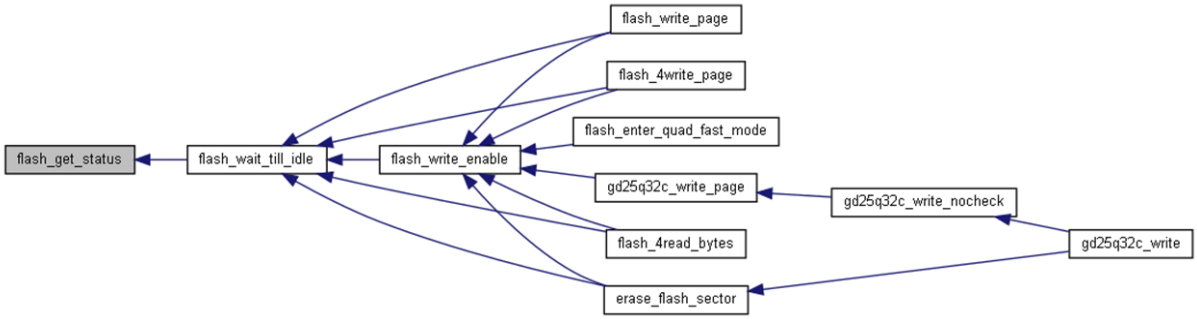
1.6.1 flash_get_status

功能	flash_get_status
描述	get the status of flash(idle/busy)
函数定义	uint8_t flash_get_status(void)
参数	none
返回	status

函数调用图:



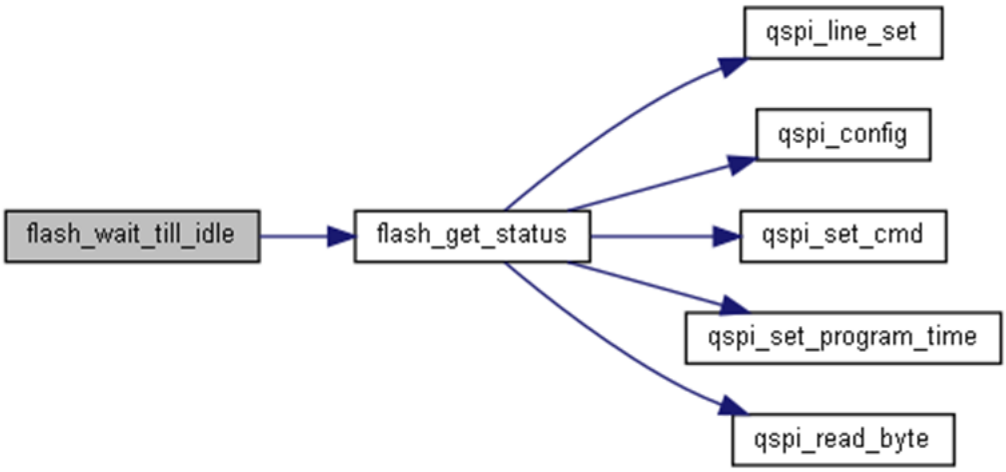
函数的调用关系图:



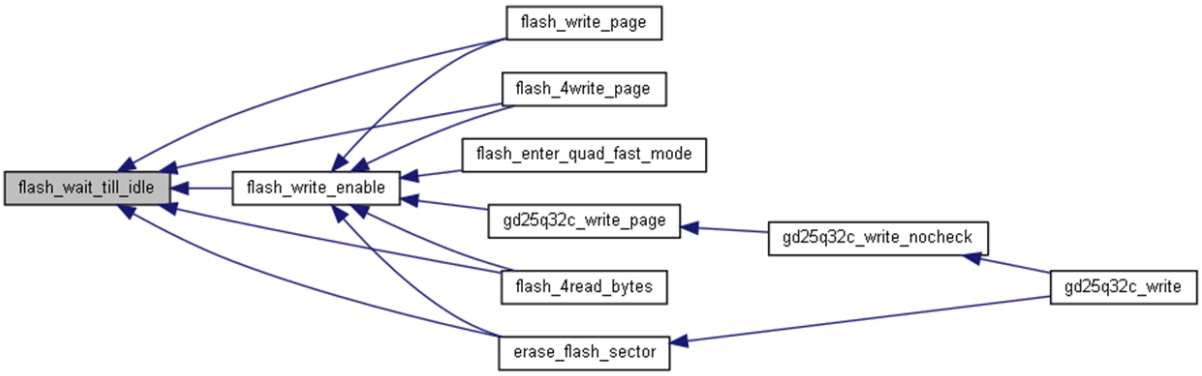
1.6.2 flash_wait_till_idle

功能	flash_wait_till_idle
描述	wait untill flash is idle
函数定义	void flash_wait_till_idle(void)
参数	none
返回	none

函数调用图：



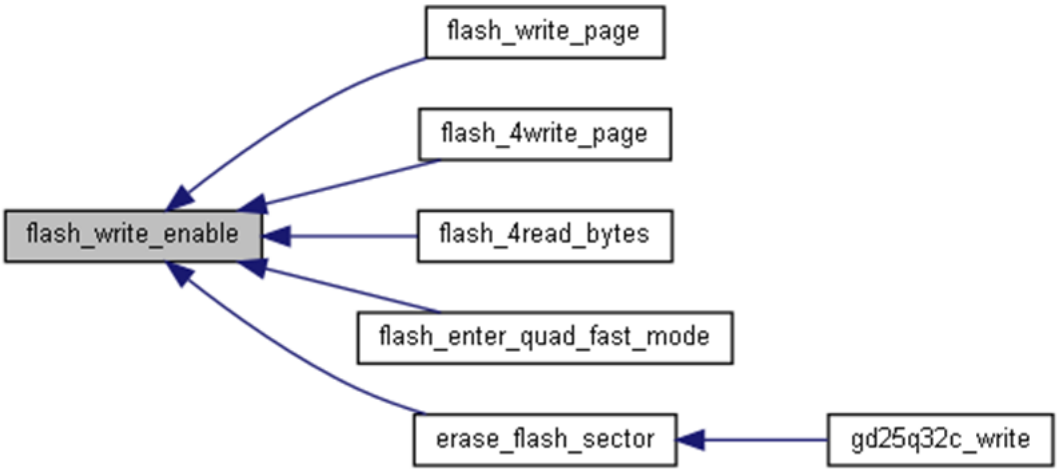
函数的调用关系图:



1.6.3 flash_write_enable

功能	flash_write_enable
描述	enable the write function of flash
函数定义	void flash_write_enable(void)
参数	none
返回	none

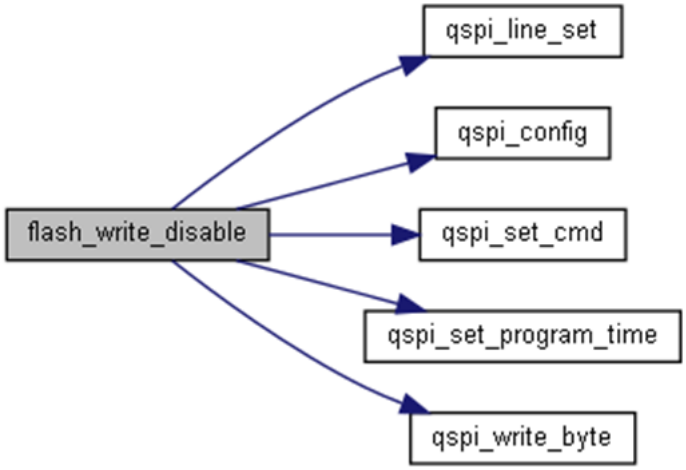
函数的调用关系图:



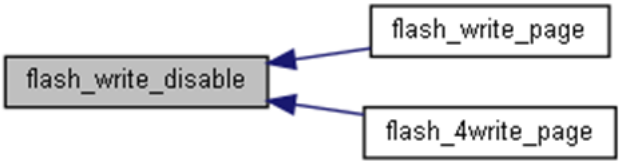
1.6.4 flash_write_disable

功能	flash_write_disable
描述	disable the write function of flash
函数定义	void flash_write_disable(void)
参数	none
返回	none

函数调用图:



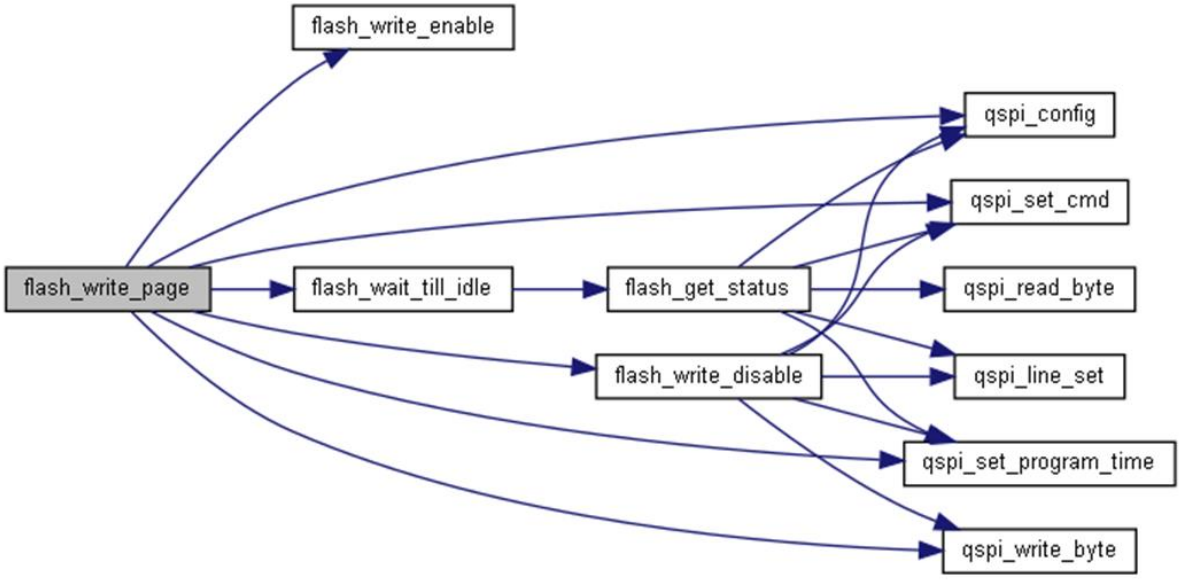
函数的调用关系图:



1.6.5 flash_write_page

功能	flash_write_page
描述	program page of flash using 1-line
函数定义	void flash_write_page(uint8_t* buffer,uint32_t write_addr,uint16_t len)
参数	uint8_t* buffer：buffer which store the data uint32_t write_addr：the starting address of page uint16_t len：length of the buffer to write
返回	none

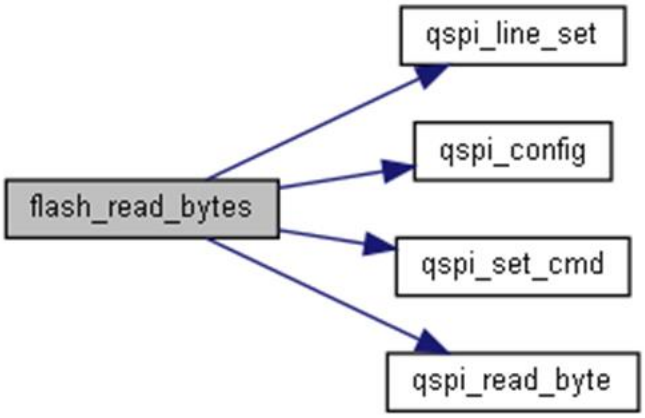
函数调用图:



1.6.6 flash_read_bytes

功能	flash_read_bytes
描述	read data from flash using 1-line
函数定义	void flash_read_bytes(uint8_t* buffer,uint32_t read_addr,uint16_t len)
参数	uint8_t* buffer：buffer which store the data uint32_t read_addr：the starting address of data uint16_t len：length of the buffer to read
返回	none

函数调用图:



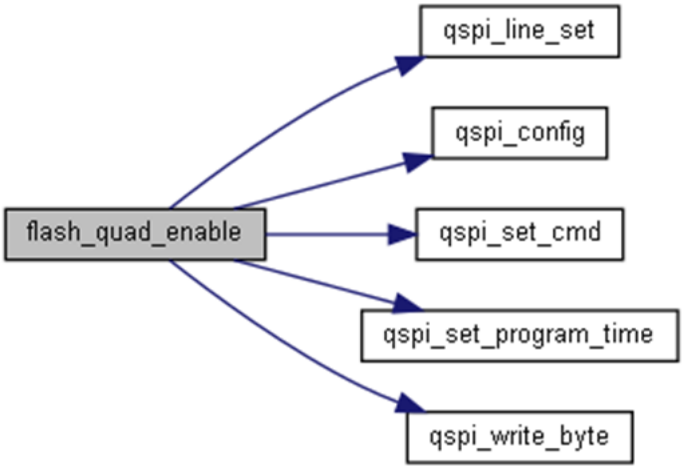
1.6.7 flash_2read_bytes

功能	flash_2read_bytes
描述	read data from flash using 2-line
函数定义	void flash_2read_bytes(uint8_t* buffer,uint32_t read_addr,uint16_t len)
参数	uint8_t* buffer：buffer which store the data uint32_t read_addr：the starting address of data uint16_t len：length of the buffer to read
返回	none

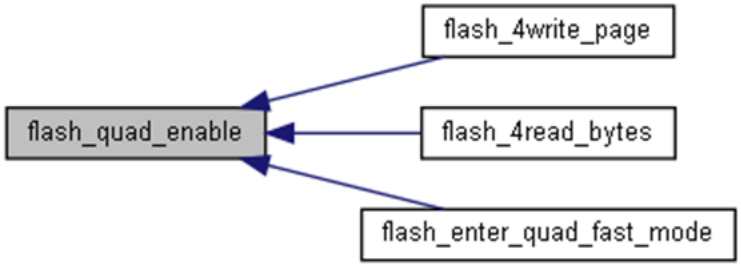
1.6.8 flash_quad_enable

功能	flash_quad_enable
描述	enable the quad mode
函数定义	void flash_quad_enable(void)
参数	none
返回	none

函数调用图:



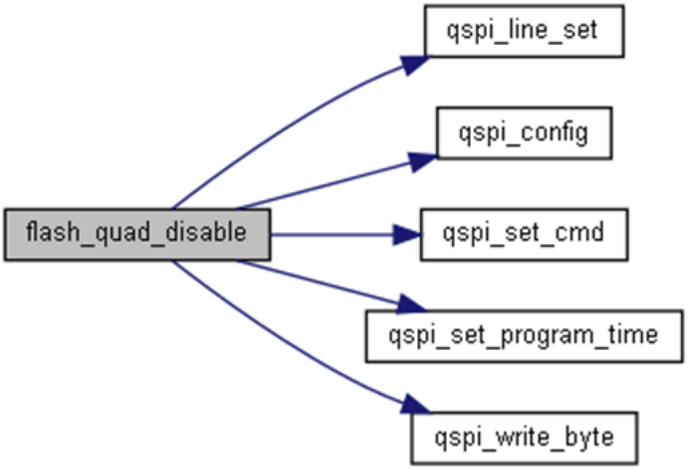
函数的调用关系图:



1.6.9 flash_quad_disable

功能	flash_quad_disable
描述	disable the quad mode
函数定义	void flash_quad_disable(void)
参数	none
返回	none

函数调用图:

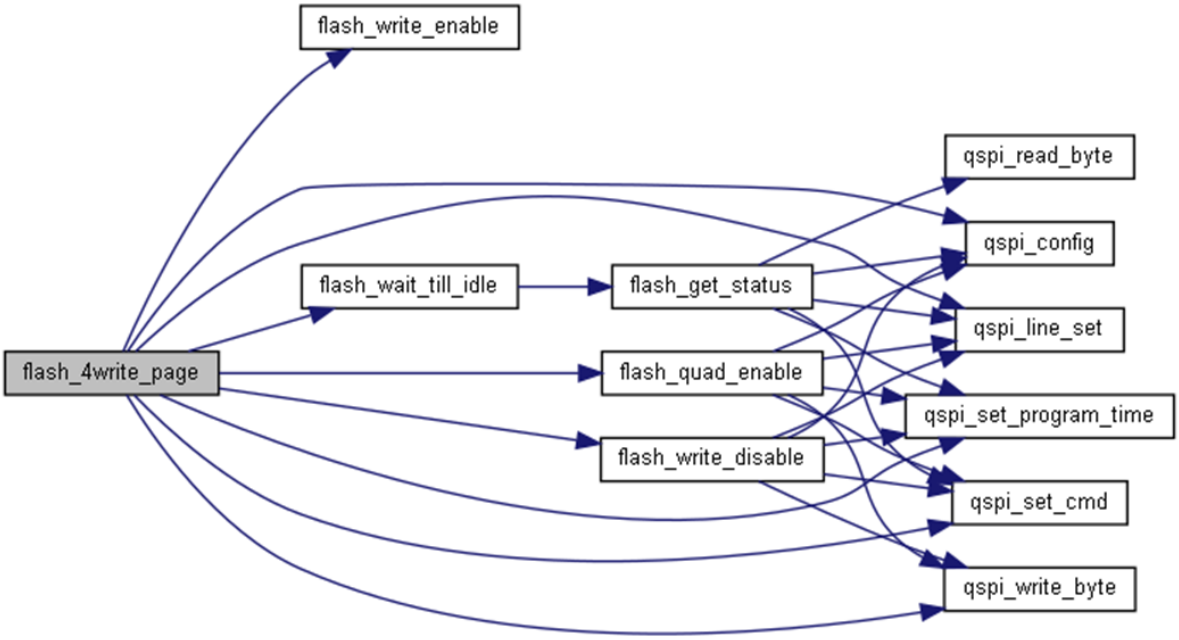


函数的调用关系图:

1.6.10 flash_4write_page

功能	flash_4write_page
描述	program page of flash using 4-line
函数定义	void flash_4write_page(uint8_t* buffer,uint32_t write_addr,uint16_t len)
参数	uint8_t* buffer：buffer which store the data uint32_t write_addr：the starting address of page uint16_t len：length of the buffer to write
返回	none

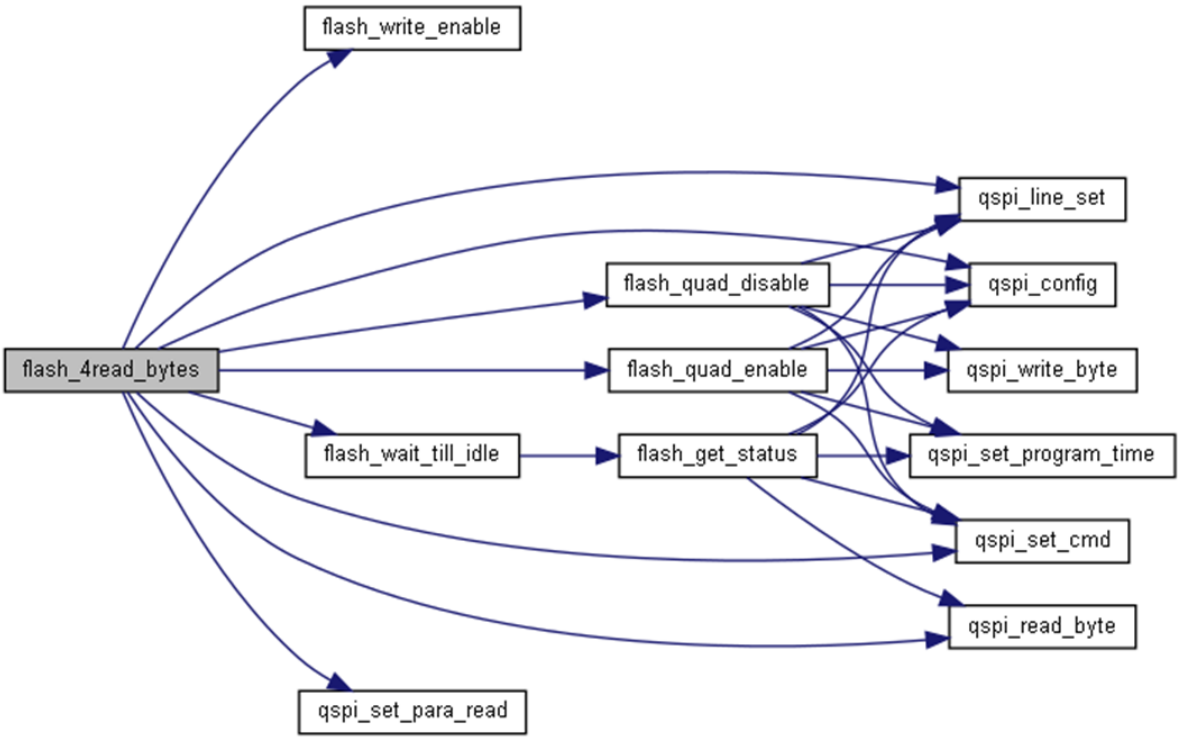
函数调用图：



1.6.11 flash_4read_bytes

功能	flash_4read_bytes
描述	read data from flash using 4-line
函数定义	void flash_4read_bytes(uint8_t* buffer,uint32_t read_addr,uint16_t len)
参数	uint8_t* buffer：buffer which store the data uint32_t read_addr：the starting address of data uint16_t len：length of the buffer to read
返回	none

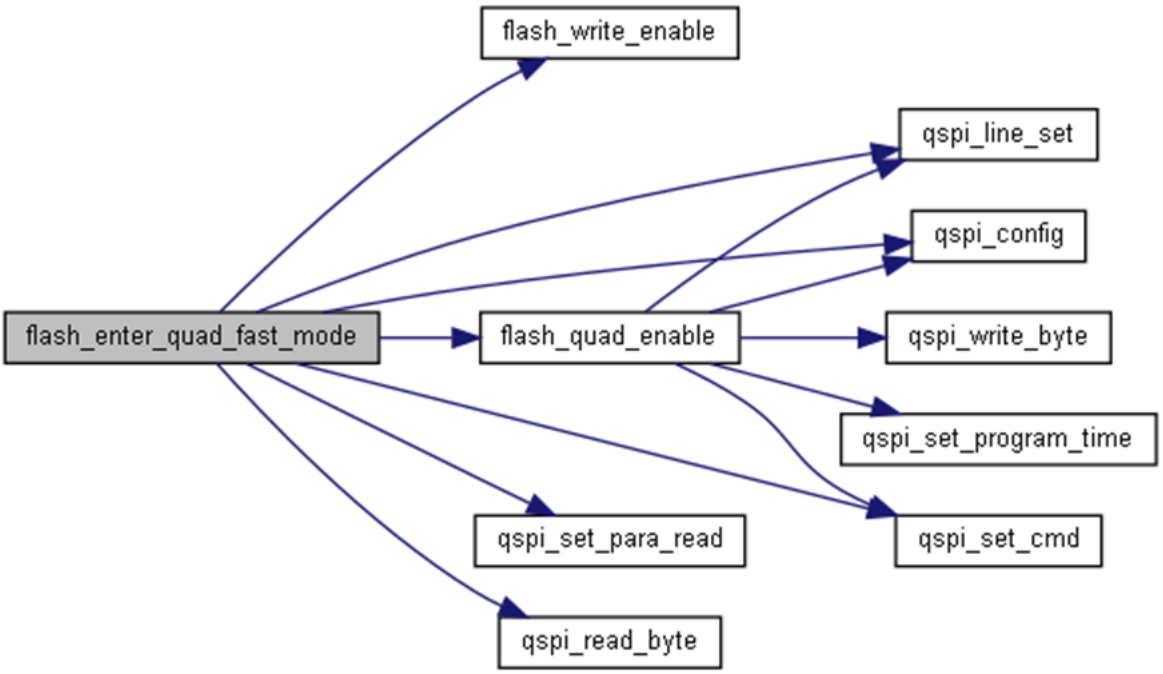
函数调用图：



1.6.12 flash_enter_quad_fast_mode

功能	flash_enter_quad_fast_mode
描述	set flash into 4-line fast read mode
函数定义	void flash_enter_quad_fast_mode(void)
参数	none
返回	none

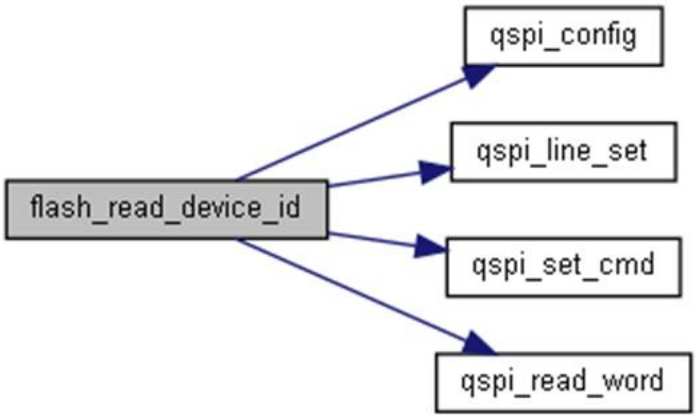
函数调用图:



1.6.13 flash_read_device_id

功能	flash_read_device_id
描述	read flash's deive number
函数定义	uint16_t flash_read_device_id(void)
参数	none
返回	device num

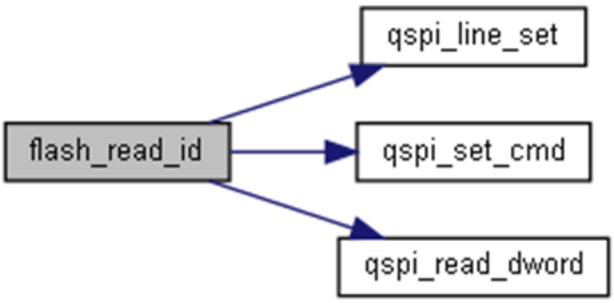
函数调用图:



1.6.14 flash_read_id

功能	flash_read_id
描述	read flash's identifier
函数定义	uint32_t flash_read_id(void)
参数	none
返回	identifier

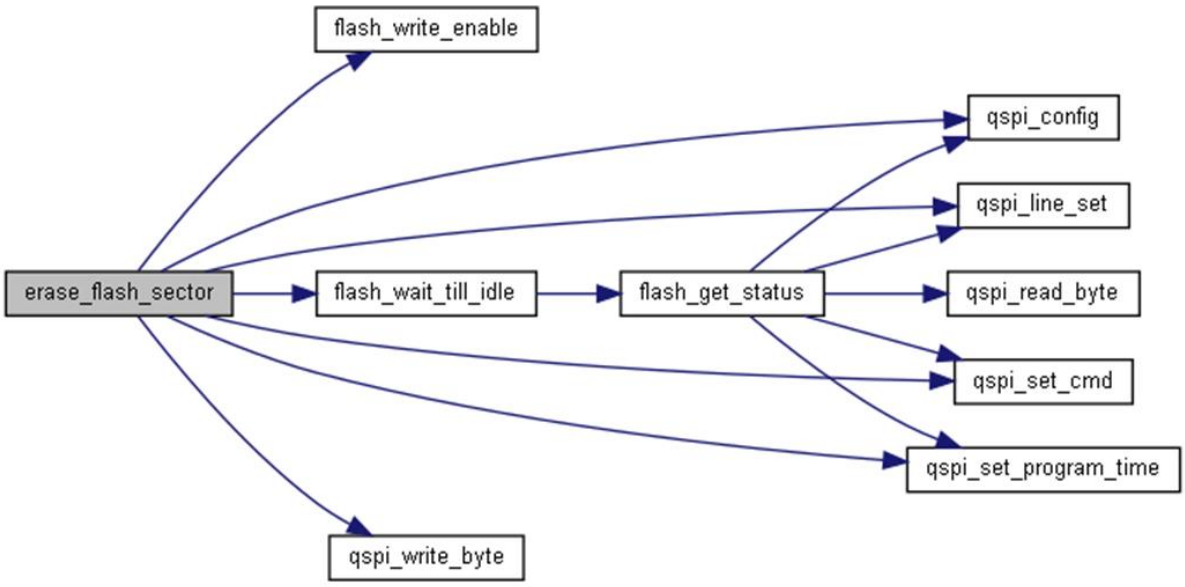
函数调用图:



1.6.15 erase_flash_sector

功能	erase_flash_sector
描述	erase flash sector
函数定义	void erase_flash_sector(uint32_t erase_add)
参数	uint32_t erase_add：starting address of the sector
返回	none

函数调用图:



1.6.16 QSPI_IRQHandler

功能	QSPI_IRQHandler
描述	QSPI中断处理函数
函数定义	void QSPI_IRQHandler(void)
参数	none
返回	none

1.6.17 qspi_init

功能	qspi_init
描述	QSPI初始化
函数定义	void qspi_init(uint8_t qspi_line,uint8_t work_mode)
参数	uint8_t qspi_line：设置qspi为单线、2线or4线传输 uint8_t work_mode: set CPHOL and CPHA，设置时钟极性和时钟相位
返回	none

1.6.18 qspi_irq_init

功能	qspi_irq_init
描述	QSPI中断初始化
函数定义	void qspi_irq_init(uint8_t state, void (*pfunc)())
参数	uint8_t state：enable/disable void (*pfunc)()：回调函数
返回	none

1.6.19 qspi_line_set

功能	qspi_line_set
描述	设置QSPI为单线、2线or4线传输
函数定义	void qspi_line_set(uint8_t line)
参数	uint8_t line：QSPI为单线、2线or4线
返回	none

1.6.20 qspi_config

功能	qspi_config
描述	config the frame of qspi
函数定义	void qspi_config(uint32_t cfg)
参数	uint32_t cfg：parameters
返回	none

1.6.21 qspi_set_cmd

功能	qspi_set_cmd
描述	set the command
函数定义	void qspi_set_cmd(uint8_t type,uint8_t cmd)
参数	uint8_t type：read/write uint8_t cmd：command
返回	none

1.6.22 qspi_set_program_time

功能	qspi_set_program_time
描述	set the waiting time of programing
函数定义	void qspi_set_program_time(uint16_t tim)
参数	uint16_t tim：time to wait,rely on HALF_US
返回	none

1.6.23 qspi_set_para_read

功能	qspi_set_para_read
描述	set the parameter while reading
函数定义	void qspi_set_para_read(uint8_t para1, uint16_t para2)
参数	uint8_t para1：参数1 uint16_t para2：参数2
返回	none

1.6.24 qspi_set_para_write

功能	qspi_set_para_write
描述	set the parameter while writing
函数定义	void qspi_set_para_write(uint8_t para1, uint16_t para2)
参数	uint8_t para1：参数1 uint16_t para2：参数2
返回	void

1.6.25 qspi_read_byte

功能	qspi_read_byte
描述	read 1 byte
函数定义	inline uint8_t qspi_read_byte(uint32_t offset)
参数	uint32_t offset：offset of address to read
返回	void

1.6.26 qspi_read_word

功能	qspi_read_word
描述	read 1 word
函数定义	inline uint16_t qspi_read_word(uint32_t offset)
参数	uint32_t offset：offset
返回	void

1.6.27 qspi_read_dword

功能	qspi_read_dword
描述	read 1 double-word
函数定义	inline uint32_t qspi_read_dword(uint32_t offset)
参数	uint32_t offset：offset of address to read
返回	void

1.6.28 qspi_write_byte

功能	qspi_write_byte
描述	write byte
函数定义	inline void qspi_write_byte(uint32_t offset, uint8_t val)
参数	uint32_t offset：offset of address to write uint8_t val：value to write
返回	void

1.6.29 qspi_write_word

功能	qspi_write_word
描述	write word
函数定义	inline void qspi_write_word(uint32_t offset, uint16_t val)
参数	uint32_t offset：offset of address to write uint16_t val：value to write
返回	void

1.6.30 qspi_write_dword

功能	qspi_write_dword
描述	write double-word
函数定义	inline void qspi_write_dword(uint32_t offset, uint32_t val)
参数	uint32_t offset：offset of address to write uint32_t val：value to write
返回	void

1.7 CAN接口

#include "can.h"

1.7.1 CAN_IRQHandler

功能	CAN_IRQHandler
描述	CAN中断处理
函数定义	void CAN_IRQHandler(void)
参数	none
返回	none

1.7.2 can_init

功能	can_init
描述	CAN初始化
函数定义	void can_init(uint8_t baudrate)
参数	uint8_t baudrate： baudrate 速率可以选择为1M/500k/250k/125k bps APB时钟=PCLK=32Mhz, CAN波特率 BitRate = Fpclk/2*((BRP+1)*(TS1+TS2+3)), 有个约定:TS1>=TS2 设波特率为1M的参数： 设置BRP=1(4分频), BitRate = 1M = 32M/2((1+1)*(TS1+TS2+3)),所以可以设置TS1=3,TS2=2 设波特率为500K的参数： 设置BRP=3(8分频), BitRate = 0.5M = 32M/2((3+1)*(TS1+TS2+3)),所以可以设置TS1=3,TS2=2 设波特率为250K的参数： 设置BRP=7(16分频), BitRate = 0.25M = 32M/2((7+1)*(TS1+TS2+3)),所以可以设置TS1=3,TS2=2 设波特率为125K的参数： 设置BRP=15(32分频), BitRate = 0.125M = 32M/2((15+1)*(TS1+TS2+3)),所以可以设置TS1=3,TS2=2
返回	none

1.7.3 can_filter_config

功能	can_filter_config
描述	CAN过滤器配置
函数定义	void can_filter_config(uint32_t filter_value, uint8_t filter_mode, uint8_t data_mode, uint8_t filter_en)

参数	uint32_t filter_value：filter value uint8_t filter_mode：CAN工作复位模式，使用单过滤器或双过滤器 uint8_t data_mode：数据格式 uint8_t filter_en：CAN过滤器使能
返回	none

1.7.4 can_irq_init

功能	can_irq_init
描述	CAN中断处理
函数定义	void can_irq_init(uint8_t irq_enable, void (*can_int)())
参数	uint8_t irq_enable：interrupt enable/disable void (*can_int)()：中断回调函数
返回	none

1.7.5 can_send_data

功能	can_send_data
描述	CAN发送数据
函数定义	void can_send_data(uint32_t *data)
参数	uint32_t *data：发送的数据
返回	none

1.7.6 can_read_data

功能	can_read_data
描述	获取can数据
函数定义	void can_read_data(uint32_t *buff, uint8_t *ide, uint8_t *rtr, uint8_t *len)
参数	uint32_t *buff：buff uint8_t *ide：IDE uint8_t *rtr：RTR uint8_t *len：数据len
返回	none

1.7.7 can_get_sr_reg

功能	can_get_sr_reg
描述	获取can状态寄存器值
函数定义	uint32_t can_get_sr_reg(void)
参数	none
返回	none

1.7.8 can_get_rmc_reg

功能	can_get_rmc_reg
描述	获取can rx fifo data个数
函数定义	uint8_t can_get_rmc_reg(void)
参数	none
返回	none

1.8 CMP接口

#include "cmp.h"

1.8.1 COMP0_IRQHandler

功能	COMP0_IRQHandler
描述	比较器0中断处理函数
函数定义	void COMP0_IRQHandler(void)
参数	none
返回	none

1.8.2 COMP1_IRQHandler

功能	COMP1_IRQHandler
描述	比较器1中断处理函数
函数定义	void COMP1_IRQHandler(void)
参数	none
返回	none

1.8.3 COMP2_IRQHandler

功能	COMP2_IRQHandler
描述	比较器2中断处理函数
函数定义	void COMP2_IRQHandler(void)
参数	none
返回	none

1.8.4 COMP_init

功能	COMP_init
描述	比较器初始化
函数定义	void COMP_init(uint8_t COMP_select ,void (*COMP0_callback_fun)(), void (*COMP1_callback_fun)(), void (*COMP2_callback_fun)())
参数	uint8_t COMP_select：比较器选择

	void (*COMP0_callback_fun)()：比较器0回调函数 void (*COMP1_callback_fun)()：比较器1回调函数 void (*COMP2_callback_fun)()：比较器1回调函数
返回	none

1.8.5 COMP0_irq_init

功能	COMP0_irq_init
描述	比较器0中断初始化
函数定义	void COMP0_irq_init(uint8_t COMP_irq_mode)
参数	uint8_t COMP_irq_mode：中断配置
返回	none

1.8.6 COMP1_irq_init

功能	COMP1_irq_init
描述	比较器1中断初始化
函数定义	void COMP1_irq_init(uint8_t COMP_irq_mode)
参数	uint8_t COMP_irq_mode: 中断配置
返回	none

1.8.7 COMP2_irq_init

功能	COMP2_irq_init
描述	比较器2中断初始化
函数定义	void COMP2_irq_init(uint8_t COMP_irq_mode)
参数	uint8_t COMP_irq_mode：中断配置
返回	none

1.8.8 get_comp0_status

功能	get_comp0_status
描述	获取比较器0状态
函数定义	uint8_t get_comp0_status(void)
参数	none
返回	none

1.8.9 get_comp1_status

功能	get_comp1_status
描述	获取比较器1状态
函数定义	uint8_t get_comp1_status(void)
参数	none
返回	none

1.8.10 get_comp2_status

功能	get_comp2_status
描述	获取比较器2状态
函数定义	uint8_t get_comp2_status(void)
参数	none
返回	none

1.9 CRC接口

#include "crc.h"

1.9.1 crc16_init

功能	crc16_init
描述	CRC初始化
函数定义	void crc16_init(uint32_t CRC16_RSLT_REV, uint32_t CRC16_DATA_REV, uint32_t CRC16_INITIAL_REV)
参数	uint32_t CRC16_RSLT_REV：CRC计算结果是否进行高低位倒序 uint32_t CRC16_DATA_REV：CRC计算数据是否进行高低位倒序 uint32_t CRC16_INITIAL_REV：CRC初始值是否进行高低位倒序
返回	none

1.9.2 crc16_ccitt

功能	crc16_ccitt
描述	CRC hardware mode硬件CRC算法接口
函数定义	uint16_t crc16_ccitt(uint8_t crc_data[], uint32_t len, uint16_t init_data)
参数	uint8_t crc_data[]：CRC数据数组 uint32_t len：数组长度 uint16_t init_data：CRC初始化数据
返回	REG_CRC16_DAT

1.9.3 crc_soft

功能	crc_soft
描述	CRC soft mode软件CRC算法接口
函数定义	uint16_t crc_soft(uint16_t crc, uint8_t data)
参数	uint16_t crc：CRC值 uint8_t data：计算数据
返回	crc&0xffff

函数的调用关系图：

1.9.4 crc16_ccitt_soft

功能	crc16_ccitt_soft
描述	CRC soft mode，软件多字节CRC算法接口
函数定义	uint16_t crc16_ccitt_soft(uint8_t crc_data[],uint32_t len,uint16_t init_data)
参数	uint8_t crc_data[]：CRC数据数组 uint32_t len：数组长度 uint16_t init_data：CRC初始化数据
返回	reg_crc

函数的调用关系图:

1.10 HRNG接口

#include "hrng.h"

1.10.1 rng_init

功能	rng_init
描述	RNG初始化
函数定义	void rng_init(void)
参数	none
返回	none

1.10.2 rng_write_seed

功能	rng_write_seed
描述	写随机数种子
函数定义	void rng_write_seed(uint32_t rngseed)
参数	uint32_t rngseed：随机数种子
返回	none

1.10.3 get_hrng

功能	get_hrng
描述	获取HRNG值
函数定义	uint32_t get_hrng(void)
参数	none
返回	hrng value

1.11 DMA接口

#include "dma.h"

1.11.1 DMA_IRQHandler

功能	DMA_IRQHandler
描述	DMA中断处理函数
函数定义	void DMA_IRQHandler(void)
参数	none
返回	none

1.11.2 dma_init

功能	dma_init
描述	DMA初始化，配置数据位宽、传输模式等
函数定义	void dma_init(uint8_t channel_index)
参数	uint8_t channel_index：DMA通道号
返回	none

1.11.3 dma_irq_init

功能	dma_irq_init
描述	DMA中断初始化
函数定义	void dma_irq_init(uint8_t channel_index,uint8_t irq_enable,void (*pfunc_tc)())
参数	uint8_t channel_index：DMA通道号 uint8_t irq_enable：DMA中断使能开关 void (*pfunc_tc)()：DMA中断回调函数
返回	none

1.11.4 dma_irq_transfer

功能	dma_irq_transfer
描述	中断方式时使用此函数启动DMA转换
函数定义	void dma_irq_transfer(uint8_t channel_index, uint32_t src_addr, uint32_t dest_addr, uint16_t length)
参数	uint8_t channel_index：DMA通道编号 uint32_t src_addr：源地址 uint32_t dest_addr：目的地址 uint16_t length：传输数据长度，最大为2^15-1
返回	none

1.11.5 dma_poll_transfer

功能	dma_poll_transfer
描述	查询方式时使用此函数启动DMA转换
函数定义	void dma_poll_transfer(uint8_t channel_index, uint32_t src_addr, uint32_t dest_addr, uint16_t length)
参数	uint8_t channel_index：DMA通道编号 uint32_t src_addr：源地址 uint32_t dest_addr：目的地址 uint16_t length：传输数据长度，最大为2^15-1
返回	none

1.11.6 dma_ch0_init

功能	dma_ch0_init
描述	DMA 通道0初始化
函数定义	void dma_ch0_init(void)
参数	none
返回	none

1.11.7 dma_ch1_init

功能	dma_ch1_init
描述	DMA 通道1初始化
函数定义	void dma_ch1_init(void)
参数	none
返回	none

1.11.8 dma_ch0_enable

功能	dma_ch0_enable
描述	DMA通道0使能
函数定义	void dma_ch0_enable(uint32_t dest_addr, uint32_t src_addr, uint32_t length)
参数	uint32_t dest_addr：目的地址 uint32_t src_addr：源地址 uint32_t length：数据长度，单位为byte
返回	none

1.11.9 dma_ch1_enable

功能	dma_ch1_enable
描述	DMA通道1使能
函数定义	void dma_ch1_enable(uint32_t dest_addr, uint32_t src_addr, uint32_t length)
参数	uint32_t dest_addr：目的地址

	uint32_t src_addr：源地址 uint32_t length：数据长度，单位为byte
返回	none

1.12 GPIO接口

#include "gpio.h"

1.12.1 GPIO_PA_IRQHandler

功能	GPIO_PA_IRQHandler
描述	GPIOA中断处理函数
函数定义	void GPIO_PA_IRQHandler(void)
参数	none
返回	none

1.12.2 GPIO_PB_IRQHandler

功能	GPIO_PB_IRQHandler
描述	GPIOB中断处理函数
函数定义	void GPIO_PB_IRQHandler(void)
参数	none
返回	none

1.12.3 GPIO_PC_IRQHandler

功能	GPIO_PC_IRQHandler
描述	GPIOC中断处理函数
函数定义	void GPIO_PC_IRQHandler(void)
参数	none
返回	none

1.12.4 GPIO_PD_IRQHandler

功能	GPIO_PD_IRQHandler
描述	GPIOD中断处理函数
函数定义	void GPIO_PD_IRQHandler(void)
参数	none
返回	none

1.12.5 GPIO_PE_IRQHandler

功能	GPIO_PE_IRQHandler
描述	GPIOE中断处理函数
函数定义	void GPIO_PE_IRQHandler(void)
参数	none
返回	none

1.12.6 gpio_init

功能	gpio_init
描述	GPIO初始化，包括开时钟，模块正常工作
函数定义	void gpio_init(uint8_t port, uint8_t pin)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚
返回	none

1.12.7 gpio_dir

功能	gpio_dir
描述	GPIO方向设置
函数定义	void gpio_dir(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： GPIO_DIR_OUT：输出 GPIO_DIR_IN：输入
返回	none

1.12.8 gpio_in_enable

功能	gpio_in_enable
描述	GPIO输入使能
函数定义	void gpio_in_enable(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： 1：使能 0：禁止
返回	none

1.12.9 gpio_config_pu

功能	gpio_config_pu
描述	GPIO上拉使能设置
函数定义	void gpio_config_pu(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： 1：使能 0：禁止
返回	none

1.12.10 gpio_config_pd

功能	gpio_config_pd
描述	GPIO下拉使能设置
函数定义	void gpio_config_pd(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： 1：使能 0：禁止
返回	none

1.12.11 gpio_config_od

功能	gpio_config_od
描述	GPIO开漏使能设置
函数定义	void gpio_config_od(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： 1：使能 0：禁止
返回	none

1.12.12 gpio_read_io

功能	gpio_read_io
描述	GPIO输入电平获取
函数定义	uint8_t gpio_read_io(uint8_t port, uint8_t pin)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚
返回	管脚电平状态

1.12.13 gpio_write_io

功能	gpio_write_io
描述	GPIO输出电平
函数定义	void gpio_write_io(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： 1：使能 0：禁止
返回	管脚电平状态

1.12.14 gpio_setio

功能	gpio_setio
描述	GPIO电平设置
函数定义	void gpio_setio(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： 1：高电平 0：低电平
返回	none

1.12.15 gpio_irq_init

功能	gpio_irq_init
描述	GPIO中断初始化
函数定义	void gpio_irq_init(uint8_t port, uint8_t pin, uint8_t irq_enable, void (*pfunc)(uint8_t val_ris))
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t irq_enable：GPIO中断使能 void (*pfunc)(uint8_t val_ris)：中断回调函数
返回	none

1.12.16 gpio_set_is

功能	gpio_set_is
描述	GPIO中断触发类型设置
函数定义	void gpio_set_is(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： 1：电平触发

	0：边沿触发
返回	none

1.12.17 gpio_set_iev

功能	gpio_set_iev
描述	GPIO中断触发极性设置
函数定义	void gpio_set_iev(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： 1：上升沿/高电平触发 0：下降沿/低电平触发
返回	none

1.12.18 gpio_set_ibe

功能	gpio_set_ibe
描述	GPIO单双边沿触发
函数定义	void gpio_set_ibe(uint8_t port, uint8_t pin, uint8_t newstate)
参数	uint8_t port：GPIO端口类型 uint8_t pin：GPIO管脚 uint8_t newstate： 1：双边沿触发 0：单边沿触发
返回	none

1.13 GTIMER接口

#include "gtimer.h"

1.13.1 GTIMER0

1.13.1.1 GTIMER0_IRQHandler

功能	GTIMER0_IRQHandler
描述	GTIMER0中断处理函数
函数定义	void GTIMER0_IRQHandler(void)
参数	none
返回	none

1.13.1.2 gtimer0_count_init

功能	gtimer0_count_init
描述	GTIMER0定时器初始化
函数定义	void gtimer0_count_init(uint8_t clk_src,uint16_t arr,uint16_t psc)
参数	uint8_t clk_src：GTIMER0计数时钟源选择 uint16_t arr：GTIMER0重载值设置 uint16_t psc：GTIMER0预分频值设置
返回	none

1.13.1.3 gtimer0_pwm_init

功能	gtimer0_pwm_init
描述	GTIMER0 PWM输出初始化
函数定义	void gtimer0_pwm_init(uint8_t clk_src,uint16_t arr,uint16_t ccr,uint16_t psc)
参数	uint8_t clk_src：GTIMER0计数时钟源选择 uint16_t arr：GTIMER0重载值设置 uint16_t ccr：GTIMER0比较值设置 uint16_t psc：GTIMER0预分频值设置
返回	none

1.13.1.4 gtimer0_capture_init

功能	gtimer0_capture_init
描述	GTIMER0 输入捕获初始化
函数定义	void gtimer0_capture_init(uint8_t clk_src,uint16_t arr,uint16_t psc,uint8_t cap_div)
参数	uint8_t clk_src：GTIMER0计数时钟源选择 uint16_t arr：GTIMER0重载值设置 uint16_t psc：GTIMER0预分频值设置 uint8_t cap_div：GTIMER0捕捉源分频设置
返回	none

1.13.1.5 gtimer0_bke_init

功能	gtimer0_bke_init
描述	GTIMER0 硬件刹车初始化
函数定义	void gtimer0_bke_init(uint32_t bke_src,uint32_t bke_pol)
参数	uint32_t bke_src：GTIMER0刹车源设置 uint32_t bke_pol：GTIMER0刹车信号触发极性
返回	none

1.13.1.6 gtimer0_irq_init

功能	gtimer0_irq_init
描述	GTIMER0 中断初始化
函数定义	void gtimer0_irq_init(uint8_t irq_enable,uint8_t gtimer_irq_type,void (*pfunc)())
参数	uint8_t irq_enable：GTIMER0中断使能开关 1：使能 0：禁止 uint8_t gtimer_irq_type：GTIMER0中断类型选择 void (*pfunc)()：GTIMER0中断回调函数
返回	none

1.13.1.7 gtimer0_start

功能	gtimer0_start
描述	启动GTIMER0计数
函数定义	void gtimer0_start(void)
参数	none
返回	none

1.13.1.8 gtimer0_stop

功能	gtimer0_stop
描述	关闭GTIMER0计数
函数定义	void gtimer0_stop(void)
参数	none
返回	none

1.13.1.9 gtimer0_moe_set

功能	gtimer0_moe_set
描述	GTIMER0通道输出使能开关
函数定义	void gtimer0_moe_set(uint8_t moe_enable)
参数	none
返回	none

1.13.1.10 gtimer0_soft_bke_set

功能	gtimer0_soft_bke_set
描述	GTIMER0软件触发刹车使能开关
函数定义	void gtimer0_soft_bke_set(uint8_t soft_bk_enable)
参数	uint8_t soft_bk_enable： 1：使能软件触发刹车 0：关闭软件触发刹车
返回	none

1.13.2 GTIMER1

1.13.2.1 GTIMER1_IRQHandler

功能	GTIMER1_IRQHandler
描述	GTIMER1中断处理函数
函数定义	void GTIMER1_IRQHandler(void)
参数	none
返回	none

1.13.2.2 gtimer1_count_init

功能	Gtimer1_count_init
描述	GTIMER1定时器初始化
函数定义	void gtimer1_count_init(uint8_t clk_src,uint16_t arr,uint16_t psc)
参数	uint8_t clk_src：GTIMER1计数时钟源选择 uint16_t arr：GTIMER1重载值设置 uint16_t psc：GTIMER1预分频值设置
返回	none

1.13.2.3 gtimer1_pwm_init

功能	gtimer1_pwm_init
描述	GTIMER1 PWM输出初始化
函数定义	void gtimer1_pwm_init(uint8_t clk_src,uint16_t arr,uint16_t ccr,uint16_t psc)
参数	uint8_t clk_src：GTIMER1计数时钟源选择 uint16_t arr：GTIMER1重载值设置 uint16_t ccr：GTIMER1比较值设置 uint16_t psc：GTIMER1预分频值设置
返回	none

1.13.2.4 gtimer1_capture_init

功能	gtimer1_capture_init
描述	GTIMER1 输入捕获初始化
函数定义	void gtimer1_capture_init(uint8_t clk_src,uint16_t arr,uint16_t psc,uint8_t cap_div)
参数	uint8_t clk_src：GTIMER1计数时钟源选择 uint16_t arr：GTIMER1重载值设置 uint16_t psc：GTIMER1预分频值设置 uint8_t cap_div：GTIMER1捕捉源分频设置
返回	none

1.13.2.5 gtimer1_bke_init

功能	gtimer1_bke_init
描述	GTIMER1 硬件刹车初始化
函数定义	void gtimer1_bke_init(uint32_t bke_src,uint32_t bke_pol)
参数	uint32_t bke_src：GTIMER1刹车源设置 uint32_t bke_pol：GTIMER1刹车信号触发极性
返回	none

1.13.2.6 gtimer1_irq_init

功能	gtimer1_irq_init
描述	GTIMER1 中断初始化
函数定义	void gtimer1_irq_init(uint8_t irq_enable,uint8_t gtimer_irq_type,void (*pfunc)())
参数	uint8_t irq_enable：GTIMER1中断使能开关 1：使能 0：禁止 uint8_t gtimer_irq_type：GTIMER1中断类型选择 void (*pfunc)()：GTIMER1中断回调函数
返回	none

1.13.2.7 gtimer1_start

功能	gtimer1_start
描述	启动GTIMER1计数
函数定义	void gtimer0_start(void)
参数	none
返回	none

1.13.2.8 gtimer1_stop

功能	gtimer1_stop
描述	关闭GTIMER1计数
函数定义	void gtimer0_stop(void)
参数	none
返回	none

1.13.2.9 gtimer1_moe_set

功能	gtimer0_moe_set
描述	GTIMER1通道输出使能开关
函数定义	void gtimer1_moe_set(uint8_t moe_enable)
参数	none
返回	none

1.13.2.10 gtimer1_soft_bke_set

功能	gtimer1_soft_bke_set
描述	GTIMER1软件触发刹车使能开关
函数定义	void gtimer1_soft_bke_set(uint8_t soft_bk_enable)
参数	uint8_t soft_bk_enable： 1：使能软件触发刹车 0：关闭软件触发刹车
返回	none

1.13.3 GTIMER2

1.13.3.1 GTIMER2_IRQHandler

功能	GTIMER2_IRQHandler
描述	GTIMER2中断处理函数
函数定义	void GTIMER2_IRQHandler(void)
参数	none
返回	none

1.13.3.2 gtimer2_count_init

功能	gtimer2_count_init
描述	GTIMER2定时器初始化
函数定义	void gtimer2_count_init(uint8_t clk_src,uint16_t arr,uint16_t psc)
参数	uint8_t clk_src：GTIMER2计数时钟源选择 uint16_t arr：GTIMER2重载值设置 uint16_t psc：GTIMER2预分频值设置
返回	none

1.13.3.3 gtimer0_pwm_init

功能	gtimer2_pwm_init
描述	GTIMER2 PWM输出初始化
函数定义	void gtimer0_pwm_init(uint8_t clk_src,uint16_t arr,uint16_t ccr,uint16_t psc)
参数	uint8_t clk_src：GTIMER2计数时钟源选择 uint16_t arr：GTIMER2重载值设置 uint16_t ccr：GTIMER2比较值设置 uint16_t psc：GTIMER2预分频值设置
返回	none

1.13.3.4 gtimer2_capture_init

功能	gtimer2_capture_init
描述	GTIMER2 输入捕获初始化
函数定义	void gtimer2_capture_init(uint8_t clk_src,uint16_t arr,uint16_t psc,uint8_t cap_div)
参数	uint8_t clk_src：GTIMER2计数时钟源选择 uint16_t arr：GTIMER2重载值设置 uint16_t psc：GTIMER2预分频值设置 uint8_t cap_div：GTIMER2捕捉源分频设置
返回	none

1.13.3.5 gtimer2_bke_init

功能	gtimer2_bke_init
描述	GTIMER2 硬件刹车初始化
函数定义	void gtimer2_bke_init(uint32_t bke_src,uint32_t bke_pol)
参数	uint32_t bke_src：GTIMER2刹车源设置 uint32_t bke_pol：GTIMER2刹车信号触发极性
返回	none

1.13.3.6 gtimer2_irq_init

功能	gtimer2_irq_init
描述	GTIMER2 中断初始化
函数定义	void gtimer2_irq_init(uint8_t irq_enable,uint8_t gtimer_irq_type,void (*pfunc)())
参数	uint8_t irq_enable：GTIMER2中断使能开关 1：使能 0：禁止 uint8_t gtimer_irq_type：GTIMER2中断类型选择 void (*pfunc)()：GTIMER2中断回调函数
返回	none

1.13.3.7 gtimer2_start

功能	gtimer2_start
描述	启动GTIMER2计数
函数定义	void gtimer2_start(void)
参数	none
返回	none

1.13.3.8 gtimer2_stop

功能	gtimer2_stop
描述	关闭GTIMER2计数
函数定义	void gtimer2_stop(void)
参数	none
返回	none

1.13.3.9 gtimer2_moe_set

功能	gtimer2_moe_set
描述	GTIMER2通道输出使能开关
函数定义	void gtimer2_moe_set(uint8_t moe_enable)
参数	none
返回	none

1.13.3.10 gtimer2_soft_bke_set

功能	gtimer2_soft_bke_set
描述	GTIMER2软件触发刹车使能开关
函数定义	void gtimer2_soft_bke_set(uint8_t soft_bk_enable)
参数	uint8_t soft_bk_enable： 1：使能软件触发刹车 0：关闭软件触发刹车
返回	none

1.14 I2C接口

#include "i2c.h"

1.14.1 i2c_get_int_status

功能	i2c_get_int_status
描述	获取I2C初始化状态
函数定义	uint8_t i2c_get_int_status(void)
参数	none
返回	1：IFLG status is 1 0：IFLG status is 0

1.14.2 i2c_get_status

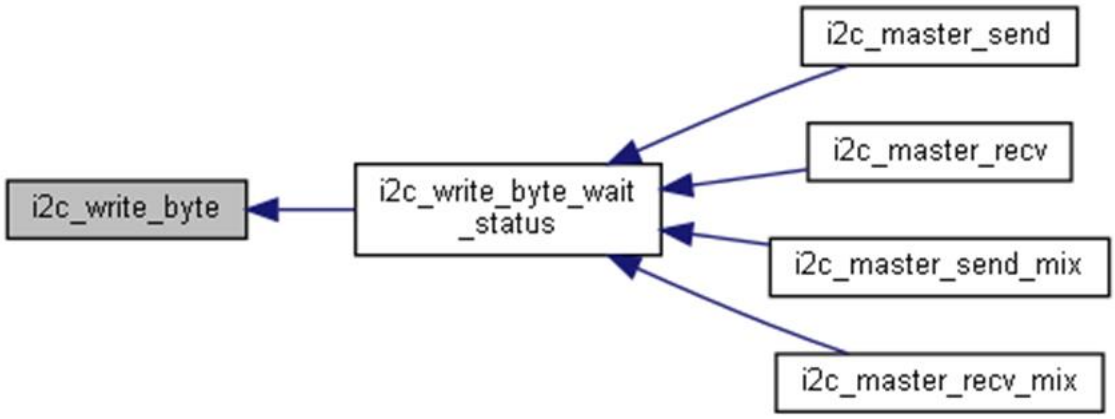
功能	i2c_get_status
描述	获取I2C状态
函数定义	uint8_t i2c_get_status(void)
参数	none
返回	Flag状态

1.14.3 i2c_write_byte

功能	i2c_write_byte
描述	I2C写一个字节

函数定义	static void i2c_write_byte(uint8_t data)
参数	uint8_t data：发送的数据（1 Byte）
返回	rdata: send_result

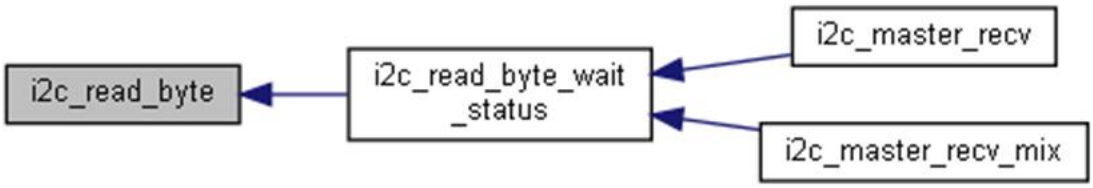
函数的调用关系图:



1.14.4 i2c_read_byte

功能	i2c_read_byte
描述	I2C读一个字节
函数定义	static uint8_t i2c_read_byte(void)
参数	none
返回	rdata: 接收数据（1 Byte）

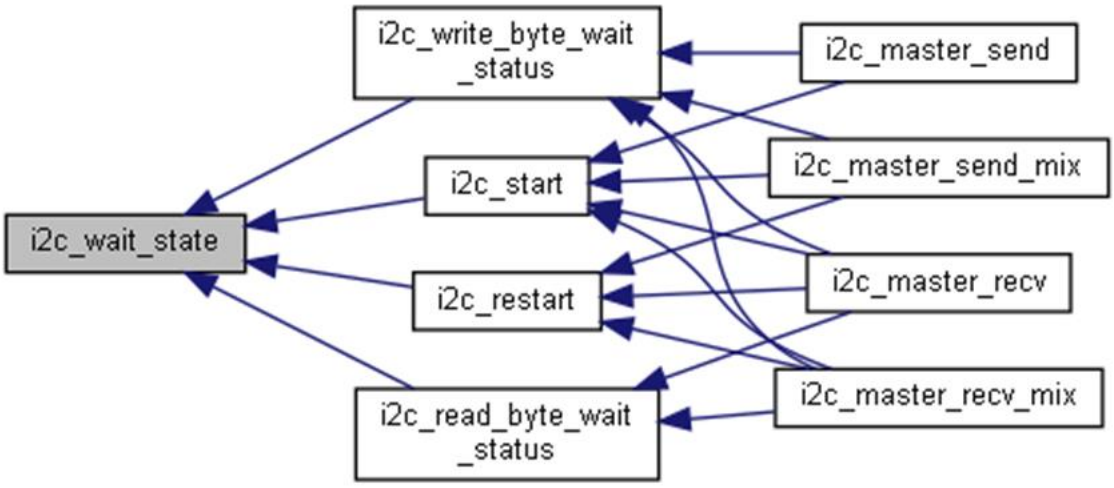
函数的调用关系图:



1.14.5 i2c_wait_state

功能	i2c_wait_state
描述	I2C等待状态
函数定义	static uint8_t i2c_wait_state(uint8_t status,uint32_t wait_time)
参数	uint8_t status：expected status uint32_t wait_time：等待时间
返回	0：TIMEOUT 1：expected status OK 2：unexpected status OK

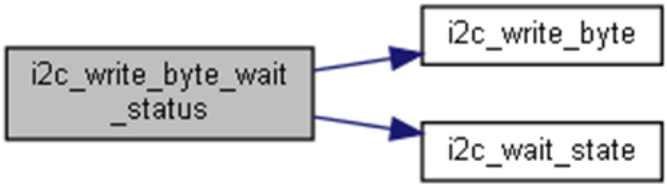
函数的调用关系图:



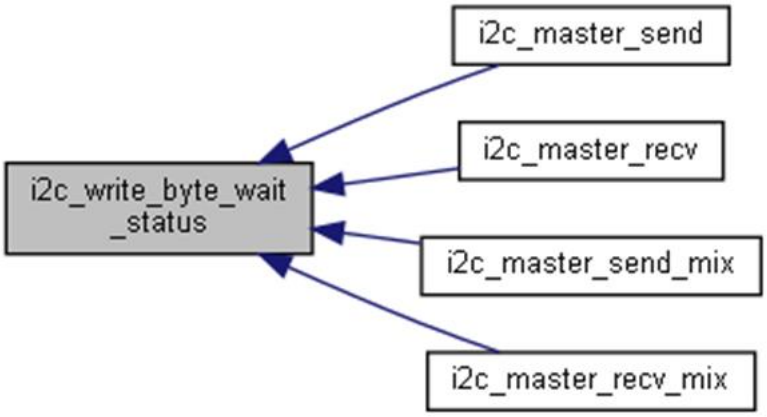
1.14.6 i2c_write_byte_wait_status

功能	i2c_write_byte_wait_status
描述	i2c_write_byte_wait_status
函数定义	static uint8_t i2c_write_byte_wait_status(uint8_t byte,uint8_t status)
参数	uint8_t byte：字节数据 uint8_t status： expected status
返回	0：error other：OK

函数的调用图:



函数的调用关系图:

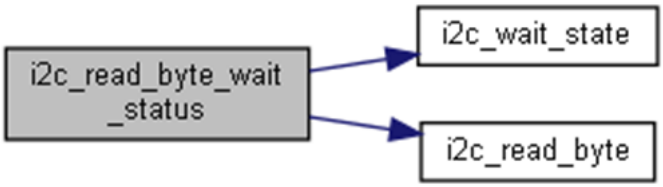


1.14.7 i2c_read_byte_wait_status

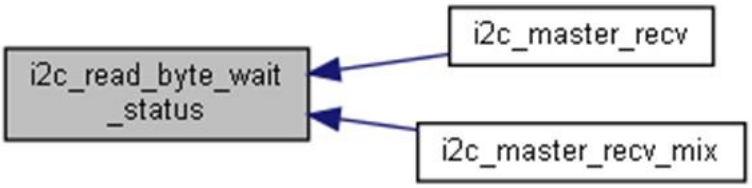
功能	i2c_read_byte_wait_status
描述	i2c_read_byte_wait_status

函数定义	static uint8_t i2c_read_byte_wait_status(uint8_t *byte, uint8_t status)
参数	uint8_t *byte：字节数据 uint8_t status：expected status
返回	0：error other：OK

函数的调用图:



函数的调用关系图:



1.14.8 i2c_speed

功能	i2c_speed
描述	i2c 速率配置： FOSCL = FSCL = PCLK / (2M×(N+1)×10)； 其中，FOSCL是I2C接口输出的SCL的频率。
函数定义	static void i2c_speed(uint8_t speed)
参数	uint8_t speed：1M、400K、100K
返回	rdata: none

函数调用关系图:

1.14.9 i2c_master_init

功能	i2c_master_init
描述	I2C初始化，中断配置
函数定义	void i2c_master_init(uint8_t speed)
参数	uint8_t speed：i2c SCL clk
返回	none

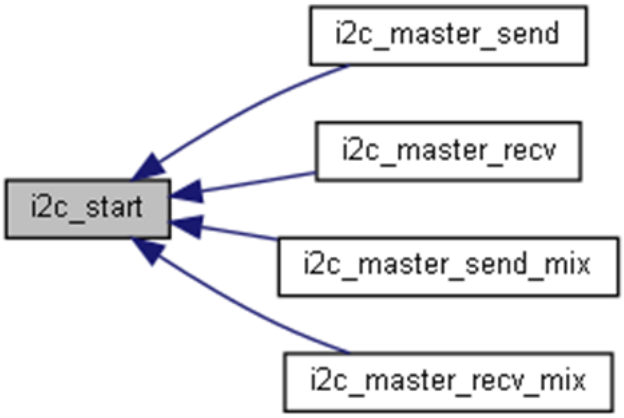
函数调用图:

1.14.10 i2c_start

功能	i2c_start
描述	启动I2C
函数定义	static uint8_t i2c_start(void)
参数	none
返回	0：error other：OK

函数调用图:

函数调用关系图:

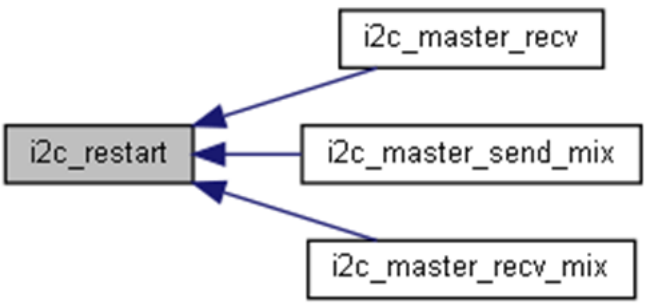


1.14.11 i2c_restart

功能	i2c_restart
描述	重启I2C
函数定义	static uint8_t i2c_restart(void)
参数	none
返回	0：error other：OK

函数的调用图:

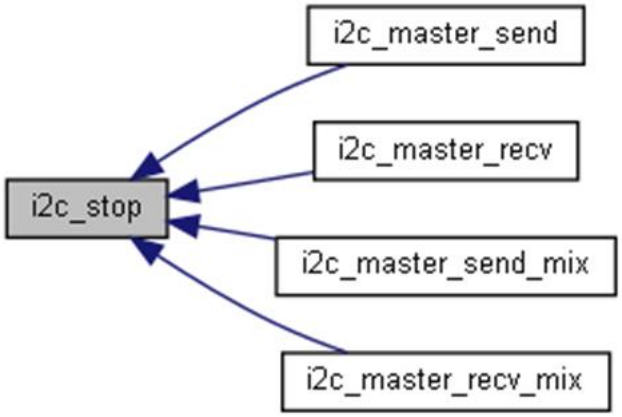
函数的调用关系图:



1.14.12 i2c_stop

功能	i2c_stop
描述	关闭I2C
函数定义	static void i2c_stop(void)
参数	none
返回	none

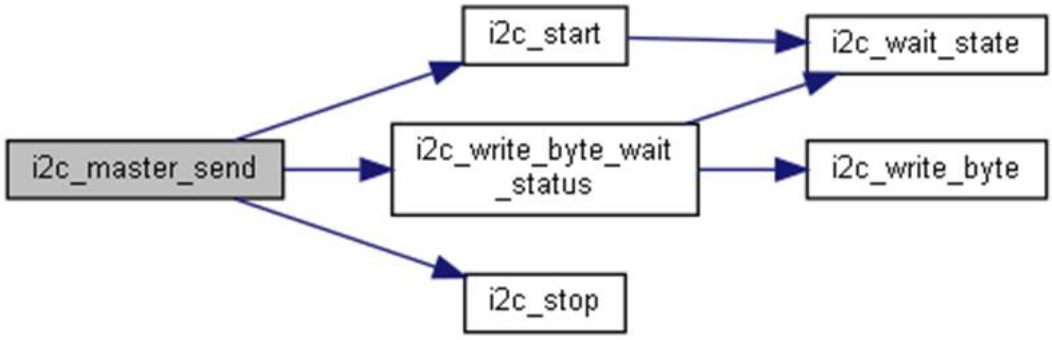
函数的调用关系图:



1.14.13 i2c_master_send

功能	i2c_master_send
描述	i2c master send data by commond mode
函数定义	uint8_t i2c_master_send(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buff数据 uint32_t length：数据长度
返回	0：error 1：OK

函数的调用图:

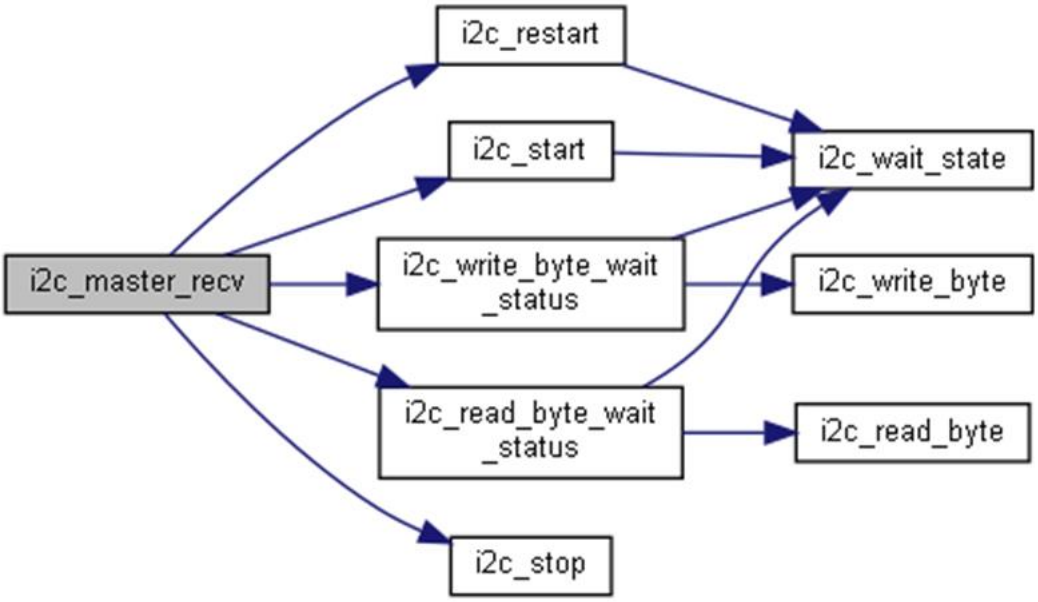


1.14.14 i2c_master_recv

功能	i2c_master_recv
描述	I2C主机接收数据
函数定义	uint8_t i2c_master_recv(uint8_t *buff, uint32_t length)

参数	uint8_t *buff：buff数据 uint32_t length：数据长度
返回	0：error 1：OK

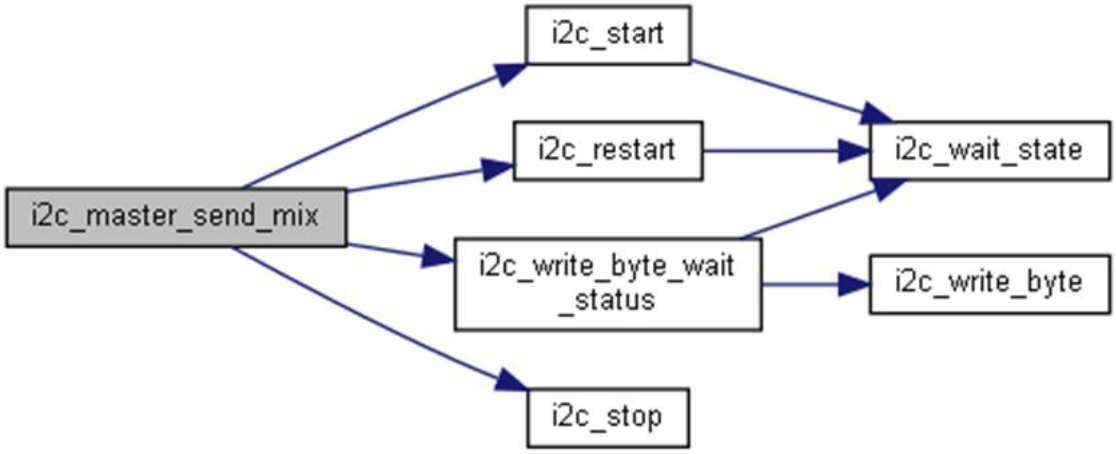
函数的调用图:



1.14.15 i2c_master_send_mix

功能	i2c_master_send_mix
描述	I2C主机发送数据（针对需要单独发memory地址的设备，如I2C EEPROM设备）
函数定义	uint8_t i2c_master_send_mix(uint32_t* write_addr, uint8_t addr_num, uint8_t *buff, uint32_t length)
参数	uint32_t* write_addr：write address uint8_t addr_num：addre size（eg：1,2,3,4） uint8_t *buff：buff数据 uint32_t length：数据长度
返回	0：error 1：OK

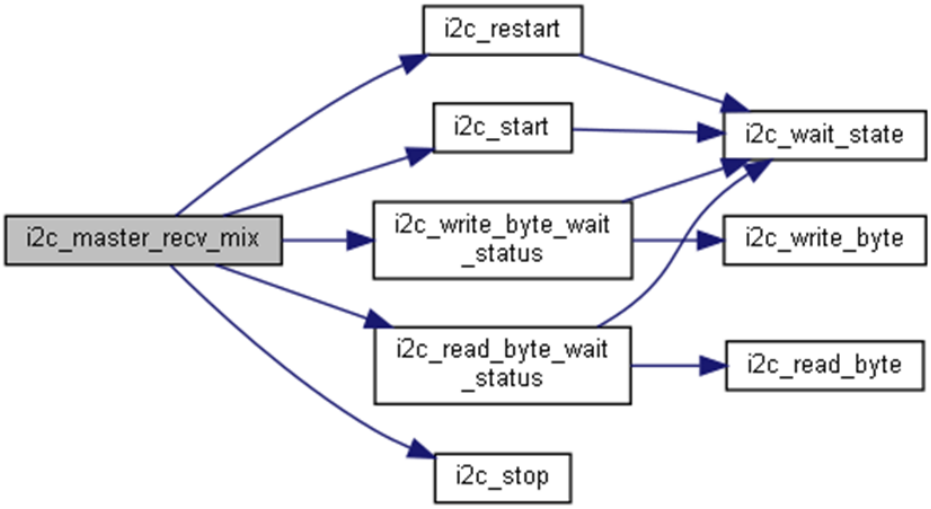
函数的调用图:



1.14.16 i2c_master_recv_mix

功能	i2c_master_receive_mix
描述	I2C主机接收数据（针对需要单独发memory地址的设备，如I2C EEPROM设备）
函数定义	uint8_t i2c_master_recv_mix(uint32_t *read_addr, uint8_t addr_num, uint8_t *buff, uint32_t length)
参数	uint32_t *read_addr：read address uint8_t addr_num：addre size（eg：0,1,2,3,4） uint8_t *buff：buff数据 uint32_t length：数据长度
返回	0：error 1：OK

函数的调用图:



1.14.17 I2C_IRQHandler

功能	I2C_IRQHandler
描述	I2C的中断处理函数
函数定义	void I2C_IRQHandler(void)
参数	none
返回	none

1.14.18 i2c_irq_init

功能	i2c_irq_init
描述	I2C中断初始化
函数定义	void i2c_irq_init(uint8_t irqstate, void (*pfunc_recv)(), void (*pfunc_send)())
参数	uint8_t irqstate： 0：interrupt diable 1：interrupt enable void (*pfunc_recv)()：接收回调函数 void (*pfunc_send)()：发送回调函数
返回	

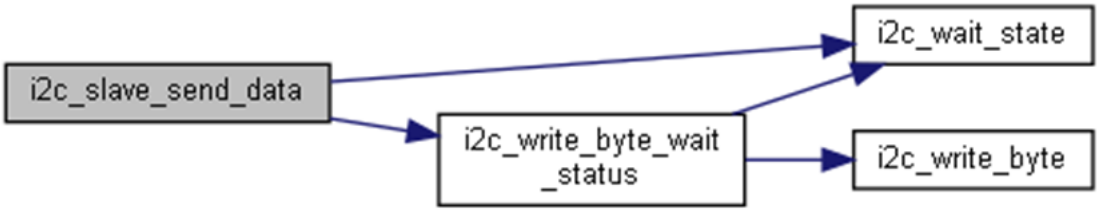
1.14.19 i2c_slave_init

功能	i2c_slave_init
描述	I2C初始化，中断配置
函数定义	void i2c_slave_init(void)
参数	device type: slave or master
返回	none

1.14.20 i2c_slave_send_data

功能	i2c_slave_send_data
描述	I2C从机发送处理函数
函数定义	uint8_t i2c_slave_send_data(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buff数据 uint32_t length：数据长度
返回	status

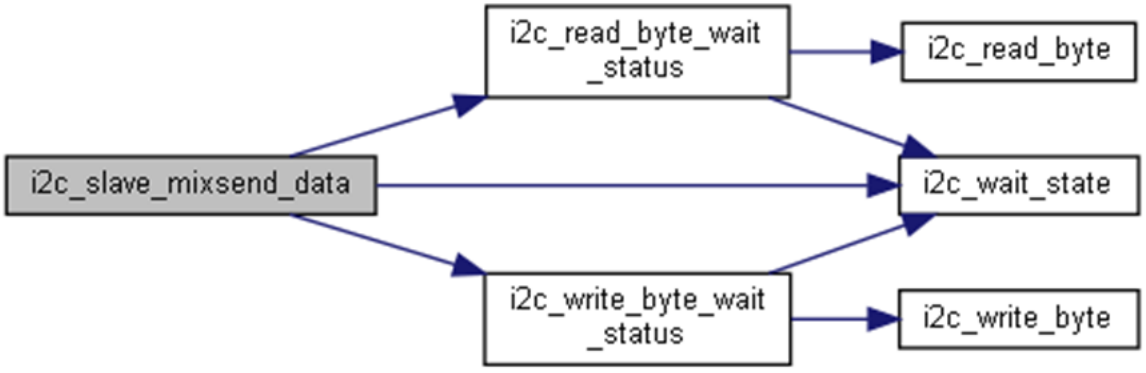
函数的调用图:



1.14.21 i2c_slave_mixsend_data

功能	i2c_slave_mixsend_data
描述	I2C从机发送处理函数（模拟EEPROM的角色示例）
函数定义	uint8_t i2c_slave_mixsend_data(uint8_t addr_len, uint32_t data_len)
参数	uint8_t addr_len：地址长度 uint32_t data_len：数据长度
返回	status

函数的调用图:



1.14.22 i2c_slave_recv_data

功能	i2c_slave_recv_data
描述	I2C从机接收处理函数，查询模式。 如果不知道数据长度，可以输入一个很大的值，比如：256
函数定义	uint8_t i2c_slave_recv_data(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buff数据 uint32_t length：数据长度（传入的length可能大于master实际发送的数据长度）
返回	0：error 1：OK

1.14.23 i2c_slave_mixrecv_data

功能	i2c_slave_mixrecv_data
描述	i2c从机接收处理函数，查询模式。 如果不知道数据长度，可以输入一个很大的值，比如：256
函数定义	uint8_t i2c_slave_mixrecv_data(uint8_t addr_len,uint8_t * buff, uint32_t data_len)
参数	uint8_t addr_len：地址长度 uint8_t * buff：buff数据 uint32_t data_len：数据长度
返回	0：error 1：OK

1.15 LPTIMER接口

#include "lptimer0.h"

1.15.1 LPTIMER0

1.15.1.1 LPTIMER0_IRQHandler

功能	lptimer0_IRQHandler
描述	LPTIMER0中断处理函数
函数定义	void LPTIMER0_IRQHandler(void)
参数	none
返回	none

1.15.1.2 lptim0_count_init

功能	lptim0_count_init
描述	LPTIMER0定时初始化
函数定义	void lptim0_count_init(uint32_t clock_source, uint32_t clock_div, uint16_t time)

参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t time：定时时间
返回	none

1.15.1.3 lptim0_pwm_init

功能	lptim0_pwm_init
描述	LPTIMER0 PWM输出初始化
函数定义	void lptim0_pwm_init(uint16_t cmp, uint16_t target, uint32_t clock_source, uint32_t clock_div, uint8_t level)
参数	uint16_t cmp：比较寄存器的值 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100% uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint8_t level： 1：计数值=比较值时输出上升沿 2：计数值=比较值时输出下降沿
返回	none

1.15.1.4 lptim0_trigger_init

功能	lptim0_trigger_init
描述	Trigger触发计数初始化
函数定义	void lptim0_trigger_init(uint32_t clock_source, uint32_t clock_div, uint16_t target)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100%
返回	none

1.15.1.5 lptim0_extcount_init

功能	lptim0_extcount_init
描述	LPTIMER0外部异步脉冲计数初始化
函数定义	void lptim0_extcount_init(uint32_t clock_source, uint32_t clock_div, uint16_t target)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100%
返回	none

1.15.1.6 lptim0_time_out_init

功能	lptim0_time_out_init
描述	LPTIMER0时钟输出初始化
函数定义	void lptim0_time_out_init(uint32_t clock_source, uint32_t clock_div, uint16_t

	target)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100%
返回	none

1.15.1.7 lptim0_irq_init

功能	lptim0_irq_init
描述	LPTIMER0中断初始化
函数定义	void lptim0_irq_init(uint32_t irq_enable, uint32_t irq_type, void (*pfunc)())
参数	uint32_t irq_enable：中断使能 uint32_t irq_type：中断类型 void (*pfunc)()：中断回调函数
返回	none

1.15.1.8 lptim0_start

功能	lptimer0_start
描述	使能LPTIMER0计数器
函数定义	void lptim0_start(void)
参数	none
返回	none

1.15.1.9 lptim0_stop

功能	lptim0_stop
描述	关闭LPTIMER0计数器
函数定义	void lptim0_stop(void)
参数	none
返回	none

1.15.2 LPTIMER1

1.15.2.1 LPTIMER1_IRQHandler

功能	lptimer1_IRQHandler
描述	LPTIMER1中断处理函数
函数定义	void LPTIMER1_IRQHandler(void)
参数	none
返回	none

1.15.2.2 lptim1_count_init

功能	lptim1_count_init
描述	LPTIMER1定时初始化
函数定义	void lptim1_count_init(uint32_t clock_source, uint32_t clock_div, uint16_t time)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t time：定时时间
返回	none

1.15.2.3 lptim1_pwm_init

功能	lptim1_pwm_init
描述	LPTIMER1 PWM输出初始化
函数定义	void lptim1_pwm_init(uint16_t cmp, uint16_t target, uint32_t clock_source, uint32_t clock_div, uint8_t level)
参数	uint16_t cmp：比较寄存器的值 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100% uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint8_t level： 1：计数值=比较值时输出上升沿 2：计数值=比较值时输出下降沿
返回	none

1.15.2.4 lptim1_trigger_init

功能	lptim1_trigger_init
描述	Trigger触发计数初始化
函数定义	void lptim1_trigger_init(uint32_t clock_source, uint32_t clock_div, uint16_t target)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100%
返回	none

1.15.2.5 lptim1_extcount_init

功能	lptim1_extcount_init
描述	LPTIMER1外部异步脉冲计数初始化
函数定义	void lptim1_extcount_init(uint32_t clock_source, uint32_t clock_div, uint16_t target)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100%
返回	none

1.15.2.6 lptim1_time_out_init

功能	lptim1_time_out_init
描述	LPTIMER1时钟输出初始化
函数定义	void lptim1_time_out_init(uint32_t clock_source, uint32_t clock_div, uint16_t target)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100%
返回	none

1.15.2.7 lptim1_irq_init

功能	lptim1_irq_init
描述	LPTIMER1中断初始化
函数定义	void lptim1_irq_init(uint32_t irq_enable, uint32_t irq_type, void (*pfunc)())
参数	uint32_t irq_enable：中断使能 uint32_t irq_type：中断类型 void (*pfunc)()：中断回调函数
返回	none

1.15.2.8 lptim1_start

功能	lptimer1_start
描述	使能LPTIMER1计数器
函数定义	void lptim1_start(void)
参数	none
返回	none

1.15.2.9 lptim1_stop

功能	lptim1_stop
描述	关闭LPTIMER1计数器
函数定义	void lptim1_stop(void)
参数	none
返回	none

1.15.3 LPTIMER2

1.15.3.1 LPTIMER2_IRQHandler

功能	lptimer2_IRQHandler
描述	LPTIMER2中断处理函数
函数定义	void LPTIMER0_IRQHandler(void)

参数	none
返回	none

1.15.3.2 lptim2_count_init

功能	lptim2_count_init
描述	LPTIMER2定时初始化
函数定义	void lptim2_count_init(uint32_t clock_source, uint32_t clock_div, uint16_t time)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t time：定时时间
返回	none

1.15.3.3 lptim2_pwm_init

功能	lptim2_pwm_init
描述	LPTIMER2 PWM输出初始化
函数定义	void lptim2_pwm_init(uint16_t cmp, uint16_t target, uint32_t clock_source, uint32_t clock_div, uint8_t level)
参数	uint16_t cmp：比较寄存器的值 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100% uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint8_t level： 1：计数值=比较值时输出上升沿 2：计数值=比较值时输出下降沿
返回	none

1.15.3.4 lptim2_trigger_init

功能	lptim2_trigger_init
描述	Trigger触发计数初始化
函数定义	void lptim2_trigger_init(uint32_t clock_source, uint32_t clock_div, uint16_t target)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100%
返回	none

1.15.3.5 lptim2_extcount_init

功能	lptim2_extcount_init
描述	LPTIMER2外部异步脉冲计数初始化
函数定义	void lptim0_extcount_init(uint32_t clock_source, uint32_t clock_div, uint16_t target)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频

	uint16_t target：target duty cyle（占空比） = (cmp / target) * 100%
返回	none

1.15.3.6 lptim2_time_out_init

功能	lptim2_time_out_init
描述	LPTIMER2时钟输出初始化
函数定义	void lptim0_time_out_init(uint32_t clock_source, uint32_t clock_div, uint16_t target)
参数	uint32_t clock_source：时钟源选择 uint32_t clock_div：时钟源分频 uint16_t target：target duty cyle（占空比） = (cmp / target) * 100%
返回	none

1.15.3.7 lptim2_irq_init

功能	lptim2_irq_init
描述	LPTIMER2中断初始化
函数定义	void lptim2_irq_init(uint32_t irq_enable, uint32_t irq_type, void (*pfunc)())
参数	uint32_t irq_enable：中断使能 uint32_t irq_type：中断类型 void (*pfunc)()：中断回调函数
返回	none

1.15.3.8 lptim2_start

功能	lptimer2_start
描述	使能LPTIMER2计数器
函数定义	void lptim2_start(void)
参数	none
返回	none

1.15.3.9 lptim2_stop

功能	lptim2_stop
描述	关闭LPTIMER2计数器
函数定义	void lptim2_stop(void)
参数	none
返回	none

1.16 LPUART接口

#include "lpuart.h"

1.16.1 LPUART_IRQHandler

功能	LPUART_IRQHandler
描述	LPUART中断处理函数
函数定义	void LPUART_IRQHandler(void)
参数	none
返回	none

1.16.2 lpuart_set_baud_rate

功能	lpuart_set_baud_rate
描述	LPUART波特率设置
函数定义	void lpuart_set_baud_rate(uint32_t baud_rate)
参数	uint32_t baud_rate：Series rate
返回	none

函数的调用关系图:

1.16.3 lpuart_init

功能	lpuart_init
描述	LPUART波特率初始化
函数定义	void lpuart_init(uint32_t baud_rate)
参数	uint32_t baud_rate：Series rate
返回	none

函数的调用图:

1.16.4 lpuart_irq_init

功能	lpuart_irq_init
描述	LPUART中断初始化
函数定义	void lpuart_irq_init(uint8_t irq_enable,void (*lpuart_recv)(),void (*lpuart_send)())
参数	uint8_t irq_enable：中断使能或中断失能 void (*lpuart_recv)()：中断接收回调函数 void (*lpuart_send)()：中断发送回调函数
返回	none

1.16.5 lpuart_recv_byte

功能	lpuart_recv_byte
描述	LPUART接收一个字节数据
函数定义	uint8_t lpuart_recv_byte(void)
参数	none
返回	data from UART

函数的调用关系图:

1.16.6 lpuart_rec_bytes

功能	lpuart_rec_bytes
描述	LPUART接收多个字节数据
函数定义	void lpuart_rec_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buff数据 uint32_t length：数据长度
返回	none

函数的调用图:

1.16.7 lpuart_send_byte

功能	lpuart_send_byte
描述	LPUART发送一个字节数据
函数定义	static void lpuart_send_byte(char c)
参数	char c：输出字节
返回	none

函数的调用关系图:

1.16.8 lpuart_send_bytes

功能	lpuart_send_bytes
描述	LPUART发送多个字节数据
函数定义	void lpuart_send_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buff数据 uint32_t length：数据长度
返回	none

函数的调用图:

1.17 OPA接口

#include "opa.h"

1.17.1 OPA_IRQHandler

功能	OPA_IRQHandler
描述	OPA中断处理函数
函数定义	void OPA_IRQHandler(void)
参数	none
返回	none

1.17.2 opa_init

功能	opa_init
描述	OPA初始化
函数定义	void opa_init(void)
参数	none
返回	none

1.17.3 opa_cmp_init

功能	opa_cmp_init
描述	OPA作为CMP比较器初始化
函数定义	void opa_cmp_init(void)
参数	none
返回	none

1.17.4 opa_irq_init

功能	opa_irq_init
描述	OPA中断初始化
函数定义	void opa_irq_init(void (*pfunc)())
参数	void (*pfunc)()：OPA中断回调函数
返回	none

1.18 RTC接口

#include "rtc.h"

1.18.1 RTC_IRQHandler

功能	RTC_IRQHandler
描述	RTC中断处理函数
函数定义	void RTC_IRQHandler(void)
参数	none
返回	none

1.18.2 rtc_irq_init

功能	rtc_irq_init
描述	RTC中断初始化
函数定义	void rtc_irq_init(uint8_t irq_enable, void (*rtc_int)())
参数	uint8_t irq_enable：中断使能/失能 void (*rtc_int)()：中断回调函数
返回	none

1.18.3 rtc_enable

功能	rtc_enable
描述	RTC使能
函数定义	void rtc_enable(void)
参数	none
返回	none

1.18.4 rtc_disable

功能	rtc_disable
描述	RTC关闭
函数定义	void rtc_disable(void)
参数	none
返回	none

1.18.5 rtc_calendar_set

功能	rtc_calendar_set
描述	设置RTC日历
函数定义	void rtc_calendar_set(uint8_t year, uint8_t month, uint8_t day,uint8_t week, uint8_t hour, uint8_t min, uint8_t sec)

参数	请写入十六进制的BCD码，如2019年11月18日星期三15时52分30秒： （0x19,0x11,0x18,0x3,0x52,0x30） uint8_t year：年 uint8_t month：月 uint8_t day：日 uint8_t week：周 uint8_t hour：时 uint8_t min：分 uint8_t sec：秒
返回	none

1.18.6 rtc_calendar_get

功能	rtc_calendar_get
描述	获取RTC日历
函数定义	void rtc_calendar_get(uint8_t *year, uint8_t *month, uint8_t *day,uint8_t *week, uint8_t *hour, uint8_t *min, uint8_t *sec)
参数	uint8_t year：年 uint8_t month：月 uint8_t day：日 uint8_t week：周 uint8_t hour：时 uint8_t min：分 uint8_t sec：秒
返回	none

1.18.7 rtc_alarm_set

功能	rtc_alarm_set
描述	设置RTC闹钟
函数定义	void rtc_alarm_set(uint8_t hour, uint8_t min, uint8_t sec)
参数	uint8_t hour：时 uint8_t min：分 uint8_t sec：秒
返回	none

1.18.8 rtc_fout_init

功能	rtc_fout_init
描述	RTC FOUT初始化
函数定义	void rtc_fout_init(void)
参数	none
返回	none

1.18.9 rtc_ltbc_init

功能	rtc_ltbc_init
描述	RTC_LTBC数字校准
函数定义	void rtc_ltbc_init(void)
参数	none
返回	none

1.18.10 rtc_stamp_init

功能	rtc_stamp_init
描述	RTC时间戳初始化
函数定义	void rtc_stamp_init(void)
参数	none
返回	none

代码示例	PA4,PB7,PD0-STAMP0
	PB3,PB6,PD3-STAMP1
	下面的代码示例：使用PD0,PD3管脚作为触发源
	
	void rtc_stamp_init(void) //RTC时间戳
	{
	REG_SCU_PERIRESET |= ((uint32_t)0x01<<19); //释放GPIOD复位
	REG_SCU_PERICLKEN |= ((uint32_t)0x01<<19); //使能GPIOD时钟
	
	REG_SCU_IOCTRLPROTECT = 0xa5a55a5a;
	
	REG_SCU_PDSEL |= (5<<12); //PD3选择RTC_STAMP1
	REG_SCU_PDSEL |= (5<<0); //PD0选择RTC_STAMP0
	
	REG_SCU_PADIE0 |= (1<<27); //PD3输入使能
	REG_SCU_PADIE0 |= (1<<24); //PD0输入使能
	
	REG_GPIO_DIR(GPIOD) &= ~(1<<3); //PD3设为输入
	REG_GPIO_DIR(GPIOD) &= ~(1<<0); //PD0设为输入
	REG_SCU_IOCTRLPROTECT = 0xffffffff;
	
	REG_RTC_STAMPEN |= (1<<0); //STAMP0触发的时间戳功能使能位
	REG_RTC_STAMPEN |= (1<<1); //STAMP1触发的时间戳功能使能位
	}

1.19 SPI0接口

#include "spi0.h"

#include "gd25q32c.h"

1.19.1 spi_flash_read_device_id

功能	spi_flash_read_device_id
描述	读取设备ID
函数定义	uint32_t spi_flash_read_device_id(void)
参数	none
返回	manufacture_id

1.19.2 spi_flash_read_id

功能	spi_flash_read_id
描述	读取芯片信号ID
函数定义	uint32_t spi_flash_read_id(void)
参数	none
返回	flash_id

1.19.3 erase_flash_sector

功能	erase_flash_sector
描述	擦除芯片扇区
函数定义	void erase_flash_sector(uint32_t erase_add)
参数	uint32_t erase_add: 擦除地址
返回	none

1.19.4 flash_write_enable

功能	flash_write_enable
描述	对flash写使能
函数定义	void flash_write_enable(void)
参数	none
返回	none

1.19.5 wait_flash_idle

功能	wait_flash_idle
描述	wait flash not in write/erase/read mode
函数定义	uint32_t wait_flash_idle(void)
参数	none
返回	Flash_status

1.19.6 wait_idle

功能	wait_flash_idle
描述	wait flash not in write/erase/read mode

函数定义	uint32_t wait_flash_idle(void)
参数	none
返回	Flash_status

1.19.7 gd25q32c_write

功能	gd25q32c_write
描述	写SPI FLASH。在指定地址开始写入指定长度的数据，该函数带擦除操作!
函数定义	void gd25q32c_write(uint8_t* buffer,uint32_t write_addr,uint16_t len)
参数	uint8_t* buffer：数据存储区 uint32_t write_addr：开始写入的地址(24bit) uint16_t len：要写入的字节数(最大65535)
返回	none

1.19.8 gd25q32c_read

功能	gd25q32c_read
描述	读取SPI FLASH，在指定地址开始读取指定长度的数据
函数定义	void gd25q32c_read(uint8_t* buffer,uint32_t read_addr,uint16_t len)
参数	uint8_t* buffer：数据存储区 uint32_t read_addr：开始读取的地址(24bit) uint16_t len：要写入的字节数(最大65535)
返回	none

1.19.9 gd25q32c_write_nocheck

功能	gd25q32c_write_nocheck
描述	无检验写SPI FLASH。 必须确保所写的地址范围内的数据全部为0XFF,否则在非0XFF处写入的数据将失败! 具有自动换页功能，在指定地址开始写入指定长度的数据,但是要确保地址不越界!
函数定义	void gd25q32c_write_nocheck(uint8_t* buffer,uint32_t write_addr,uint16_t len)
参数	uint8_t* buffer：数据存储区 uint32_t write_addr：开始写入的地址(24bit) uint16_t len：要写入的字节数(最大65535)
返回	none

1.19.10 gd25q32c_write_page

功能	gd25q32c_write_page
描述	SPI在一页(0~65535)内写入少于256个字节的数据，在指定地址开始写入最大256字节的数据
函数定义	void gd25q32c_write_page(uint8_t* buffer,uint32_t write_addr,uint16_t len)
参数	uint8_t* buffer：数据存储区

	uint32_t write_addr：开始写入的地址(24bit) uint16_t len：要写入的字节数(最大256),该数不应该超过该页的剩余字节数
返回	none

1.19.11 gd25q32c_release_flash

功能	gd25q32c_release_flash
描述	Release from Deep Power-Down 唤醒flash读不到ID的时候可能误操作进入掉电模式了
函数定义	void gd25q32c_release_flash(void)
参数	none
返回	none

1.19.12 gd25q32c_deep_power_down

功能	gd25q32c_deep_power_down
描述	Release from gd25q32c_deep_power_down，唤醒flash读不到ID的时候可能误操作进入掉电模式了
函数定义	void gd25q32c_deep_power_down(void)
参数	none
返回	none

1.19.13 SPI0_IRQHandler

功能	SPI0_IRQHandler
描述	SPI0中断处理函数
函数定义	void SPI0_IRQHandler(void)
参数	none
返回	none

1.19.14 spi0_init

功能	spi0_init
描述	spi0_init SPI0初始化
函数定义	void spi0_init(uint8_t work_mode,uint8_t spi0_clk_div,uint8_t spi0_cs)
参数	uint8_t work_mode：select spi0 work mode 0,1,2,3 uint8_t spi0_clk_div：spi0 baud rate uint8_t spi0_cs：SPI0_CS0 or SPI0_CS1
返回	none

1.19.15 spi0_irq_init

功能	spi0_irq_init
描述	spi0中断配置
函数定义	void spi0_irq_init(uint8_t irq_enable,uint32_t spi0_intr,void (*pfunc)())
参数	uint8_t irq_enable：中断开关 uint32_t spi0_intr：中断类型 void (*pfunc)()：中断回调函数
返回	none

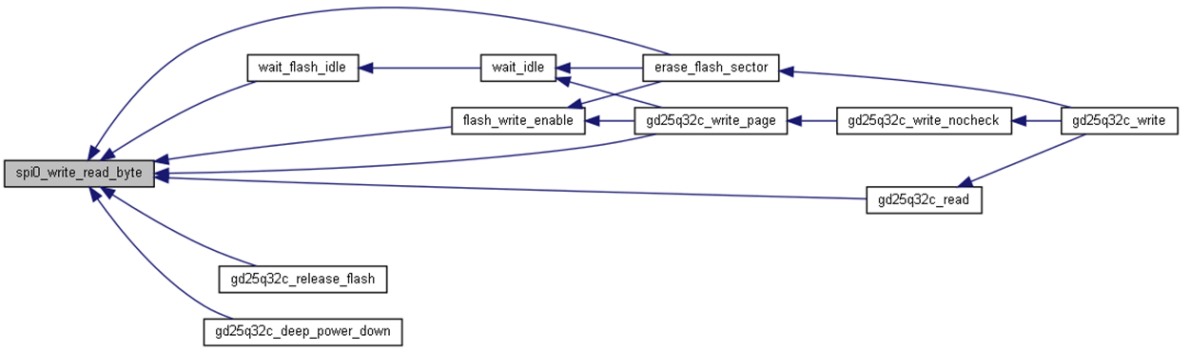
1.19.16 spi0_irq_receive_byte

功能	spi0_irq_receive_byte
描述	在中断接收一个byte数据
函数定义	uint8_t spi0_irq_receive_byte(void)
参数	none
返回	REG_SPI_SPIRXBUF 接收数据寄存器数据

1.19.17 spi0_write_read_byte

功能	spi0_write_read_byte
描述	发送一个byte数据并接收一个byte数据
函数定义	uint8_t spi0_write_read_byte(uint8_t byte)
参数	uint8_t byte：byte数据
返回	rx_buf：正常发送返回 0：异常发送返回

函数的调用关系图:

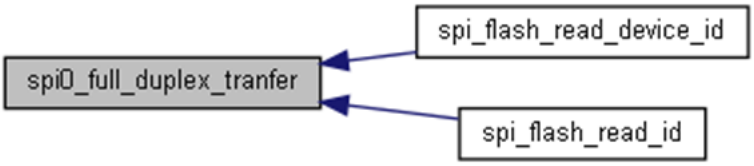


1.19.18 spi0_master _full_duplex_tranfer

功能	spi0_master_full_duplex_tranfer
描述	spi0全双工传输
函数定义	uint8_t spi0_master_full_duplex_tranfer(uint8_t *send_data, uint8_t *rec_buff, uint32_t length)
参数	uint8_t *send_data：发送数据

	uint8_t *rec_buff：接收buff数据 uint32_t length：数据长度
返回	0：正常发送返回 1：异常发送返回

函数的调用关系图:



1.19.19 spi0_master_half_duplex_send_bytes

功能	spi0_master_half_duplex_send_bytes
描述	发送多个byte数据
函数定义	uint8_t spi0_master_half_duplex_send_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buff数据 uint32_t length：数据长度
返回	0：正常发送返回 1：异常发送返回

函数的调用图:

1.19.20 spi0_master_half_duplex_receive_bytes

功能	spi0_master_half_duplex_receive_bytes
描述	接收多个byte数据
函数定义	uint8_t spi0_master_half_duplex_receive_bytes(uint8_t *buff,uint32_t length)
参数	uint8_t *buff：buff数据 uint32_t length：数据长度
返回	0：正常发送返回 1：异常发送返回

函数的调用图:

1.19.21 spi0_slave_half_duplex_send_bytes

功能	spi0_slave_half_duplex_send_bytes
描述	发送多个byte数据
函数定义	uint8_t spi0_slave_half_duplex_send_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：发送buff数据

	uint32_t length：数据长度
返回	none

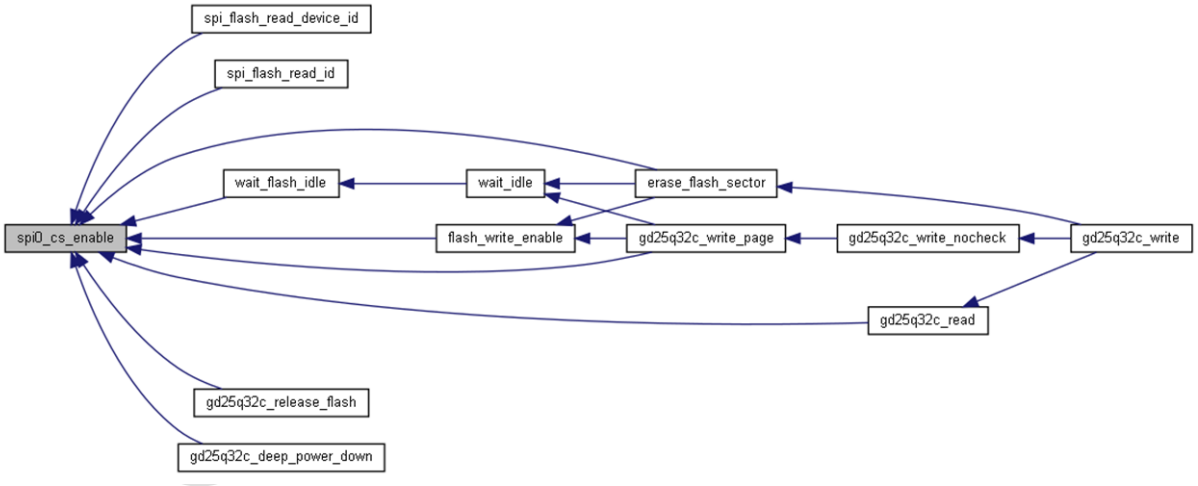
1.19.22 spi0_slave_half_duplex_receive_bytes

功能	spi0_slave_half_duplex_receive_bytes
描述	接收多个byte数据
函数定义	uint8_t spi0_slave_half_duplex_receive_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：接收buff数据 uint32_t length：数据长度
返回	0：正常发送返回 1：异常发送返回

1.19.23 spi0_cs_enable

功能	spi0_cs_enable
描述	CS片选设置
函数定义	void spi0_cs_enable(uint8_t cs_enable, uint8_t cs_select)
参数	uint8_t cs_enable：CS片选开关 uint8_t cs_select：CS片选选择
返回	none

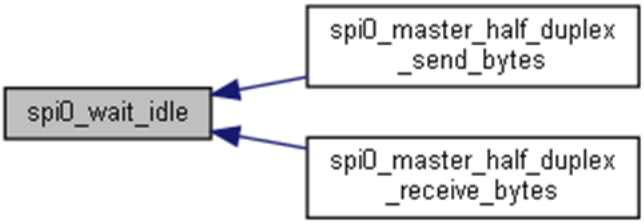
函数的调用关系图:



1.19.24 spi0_wait_idle

功能	spi0_wait_idle
描述	等待SPI0空闲
函数定义	static void spi0_wait_idle(void)
参数	none
返回	none

函数的调用关系图:



1.20 SPI1接口

#include "spi1.h"

1.20.1 SPI1_IRQHandler

功能	SPI1_IRQHandler
描述	SPI1中断处理函数
函数定义	void SPI1_IRQHandler(void)
参数	none
返回	none

1.20.2 spi1_init

功能	spi1_init
描述	SPI1初始化
函数定义	void spi1_init(uint8_t work_mode,uint8_t spi1_clk_div,uint8_t spi1_cs)
参数	uint8_t work_mode：select spi1 work mode 0,1,2,3 uint8_t spi1_clk_div：spi1 baud rate uint8_t spi1_cs：SPI1_CS0 or SPI1_CS1
返回	none

1.20.3 spi1_irq_init

功能	spi1_irq_init
描述	spi1中断配置
函数定义	void spi1_irq_init(uint8_t irq_enable,uint32_t spi1_intr,void (*pfunc)())
参数	uint8_t irq_enable：中断开关 uint32_t spi1_intr：中断类型 void (*pfunc)()：中断回调函数
返回	none

1.20.4 spi1_irq_receive_byte

功能	spi1_irq_receive_byte
描述	在中断接收一个byte数据
函数定义	uint8_t spi1_irq_receive_byte(void)

参数	none
返回	REG_SPI_SPIRXBUF 接收数据寄存器数据

1.20.5 spi1_master _full_duplex_tranfer

功能	spi1_master_full_duplex_tranfer
描述	spi1全双工传输
函数定义	uint8_t spi1_master_full_duplex_tranfer(uint8_t *send_data, uint8_t *rec_buff, uint32_t length)
参数	uint8_t *send_data：发送数据 uint8_t *rec_buff：接收buff数据 uint32_t length：数据长度
返回	0：正常发送返回 1：异常发送返回

1.20.6 spi1_master_half_duplex_send_bytes

功能	spi1_master_half_duplex_send_bytes
描述	发送多个byte数据
函数定义	uint8_t spi1_master_half_duplex_send_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：buff数据 uint32_t length：数据长度
返回	0：正常发送返回 1：异常发送返回

函数的调用图:

1.20.7 spi1_master_half_duplex_receive_bytes

功能	spi1_master_half_duplex_receive_bytes
描述	接收多个byte数据
函数定义	uint8_t spi1_master_half_duplex_receive_bytes(uint8_t *buff,uint32_t length)
参数	uint8_t *buff：buff数据 uint32_t length：数据长度
返回	0：正常发送返回 1：异常发送返回

函数的调用图:

1.20.8 spi1_slave_half_duplex_send_bytes

功能	spi1_slave_half_duplex_send_bytes
描述	发送多个byte数据
函数定义	uint8_t spi1_slave_half_duplex_send_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：发送buff数据 uint32_t length：数据长度
返回	none

1.20.9 spi1_slave_half_duplex_receive_bytes

功能	spi1_slave_half_duplex_receive_bytes
描述	接收多个byte数据
函数定义	uint8_t spi1_slave_half_duplex_receive_bytes(uint8_t *buff, uint32_t length)
参数	uint8_t *buff：接收buff数据 uint32_t length：数据长度
返回	0：正常发送返回 1：异常发送返回

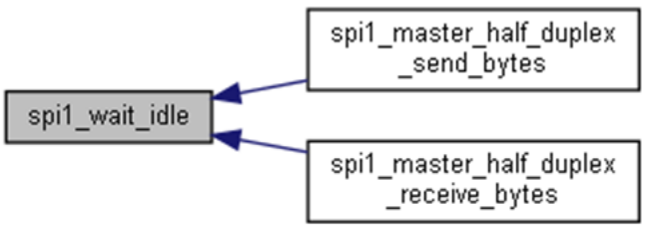
1.20.10 spi1_cs_enable

功能	spi1_cs_enable
描述	CS片选设置
函数定义	void spi1_cs_enable(uint8_t cs_enable, uint8_t cs_select)
参数	uint8_t cs_enable：CS片选开关 uint8_t cs_select：CS片选选择
返回	none

1.20.11 spi1_wait_idle

功能	spi1_wait_idle
描述	等待SPI1空闲
函数定义	static void spi1_wait_idle(void)
参数	none
返回	none

函数的调用关系图:



1.21 WDT接口

#include "wdt.h"

1.21.1 WDT_IRQHandler

功能	WDT_IRQHandler
描述	WDT中断处理函数
函数定义	void WDT_IRQHandler(void)
参数	none
返回	none

1.21.2 wdt_init

功能	wdt_init
描述	WDT初始化
函数定义	void wdt_init(uint16_t clock_div)
参数	uint16_t clock_div：WDT计数器分频
返回	none

1.21.3 wdt_irq_init

功能	wdt_irq_init
描述	WDT中断初始化
函数定义	void wdt_irq_init(uint8_t irq_enable,void (*pfunc)())
参数	uint8_t irq_enable：WDT中断开关 void (*pfunc)()：WDT中断回调函数
返回	none

1.21.4 wdt_reset_set

功能	wdt_reset_set
描述	WDT复位使能设置
函数定义	void wdt_reset_set(uint8_t rst_enable)
参数	none
返回	none

1.21.5 wdt_load_set

功能	wdt_load_set
描述	写入WDT装载值后会开始计数(喂狗)

函数定义	void wdt_load_set(uint32_t load_value)
参数	uint32_t load_value：WDT装载值
返回	none

1.22 WWDT接口

#include "wwdt.h"

1.22.1 WWDT_IRQHandler

功能	WWDT_IRQHandler
描述	WWDT中断处理函数
函数定义	void WWDT_IRQHandler(void)
参数	none
返回	none

1.22.2 wwdt_init

功能	wwdt_init
描述	WWDT初始化
函数定义	void wwdt_init(uint8_t ov_time)
参数	uint8_t ov_time：WWDT溢出时间
返回	none

1.22.3 wwdt_irq_init

功能	wwdt_irq_init
描述	WWDT中断初始化
函数定义	void wwdt_irq_init(uint8_t irq_enable,void (*pfunc)())
参数	uint8_t irq_enable：WWDT中断开关 void (*pfunc)()：中断回调函数
返回	none

1.22.4 wwdt_start

功能	wwdt_start
描述	启动WWDT计数
函数定义	void wwdt_start(void)
参数	none
返回	none

1.22.5 wwdt_feed

功能	wwdt_feed
描述	WWDT喂狗函数
函数定义	void wwdt_feed(void)
参数	none
返回	none