BK3432-BLE SDK

库函数介绍

V1.0
2018.12.21

深圳市集贤科技有限公司
T:0755-82571152 F:0755-88373753
http://www.uascent.com
深圳市南山区朗山路同方信息港A栋4楼

目录

1. 版本记录及免责申明和版权公告

1.1 版本记录

Version	Date	Author	Description
V1.0	2018/12/26	Eatun	初始版本

1.2 免责申明和版权公告

本文档中所有信息均按产品现状提供，如有变更，恕不另行通知。

本文档内容不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不对使用本文档内信息产生的任何侵犯专利权的行为负责。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

文中提到的所有商标均属其各自所有者的财产，特此声明。

注 意

由于产品版本升级或其他原因，本手册内容有可能变更，深圳市集贤科技有限公司保留在没有任何通知或者提示的情况下对本手册的内容进行修改的权利，使用者如需获取最新产品信息，请与本公司申请最终文档。本手册仅作为使用指导，深圳市集贤科技有限公司尽力在本手册中提供最新的信息，但不确保手册内容完全准确。本手册中的所有陈述、信息和建议也不构成任何明示或暗示的担保。

一 概述

本文档旨在为用户

二 应用程序接口（API）

（一）GPIO API接口

gpio_config

功能: GPIO口输入输出属性设置
函数定义　　: void gpio_config(uint8_t gpio, Dir_Type dir, Pull_Type pull)
参数:
uint8_t gpio ： 端口号，GPIOA对应GPIO0，GPIOB对应GPIO1，GPIOC对应GPIO2，GPIOD对应GPIO3
Dir_Type dir ： io口状态，OUTPUT：输出，INPUT：输入
Pull_Type pull：是否上下拉。PULL_HIGH：上拉使能，PULL_LOW：下拉使能，PULL_NONE：禁止上下拉
返回: 无

举例:
PB0设置为输出模式：gpio_config(GPIOB_0, OUTPUT, PULL_NONE);
PB2设置为输入上拉: gpio_config(GPIOB_2, INPUT, PULL_HIGH);

gpio_get_input

功能: GPIO 口状态获取
函数定义　　: uint8_t gpio_get_input(uint8_t gpio)
参数:
uint8_t gpio ： GPIO端口号，GPIOA对应GPIO0，GPIOB对应GPIO1，GPIOC对应GPIO2，GPIOD对应GPIO3
返回: 对应GPIO的具体值（0或者1）

举例：读取PB2的数值，gpio_get_input(GPIOB_2);

gpio_set

功能: 设置GPIO 口输出电平的状态
函数定义　　: void gpio_set(uint8_t gpio, uint8_t val)
参数:
uint8_t gpio ： GPIO端口号，GPIOA对应GPIO0，GPIOB对应GPIO1，GPIOC对应GPIO2，GPIOD对应GPIO3
uint8_t val ： 为1，输出高电平， 为0，输出低电平。
返回: 无

举例：读取PB2输出高电平，gpio_set(GPIOB_2,1);

（一）PWM API接口

pwm_init

功能: 初始化PWM
函数定义　　: void pwm_init(PWM_DRV_DESC *pwm_drv_desc)
参数:
PWM_DRV_DESC *pwm_drv_desc：初始化PWM参数的结构体
返回: 无

举例:

```
/*************************************************************
*作用          :PWM功能测试
*参数          :无
*返回值        :无
*其他说明      :
周期计算公式=(end_value/时钟频率)*2
*************************************************************/
static PWM_DRV_DESC ZB_PWMTest;//定义一个结构体
void ZB_BLePwmTest(void)
{
    rwip_prevent_sleep_set(BK_DRIVER_TIMER_ACTIVE);
    ZB_PWMTest.channel          =   3;      //选择GPIOB3作为PWM输出
    ZB_PWMTest.mode             =   0x01;   //选择PWM的模式
    ZB_PWMTest.pre_divid        =   0;
    ZB_PWMTest.end_value        =   64;     //修改PWM输出频率
    ZB_PWMTest.duty_cycle       =   64/2;   //修改PWM占空比
    ZB_PWMTest.p_Int_Handler    =   NULL;
    pwm_init(&ZB_PWMTest);
}
```

pwm_isr

功能: 处理PWM中断事件
函数定义 : void pwm_isr(void)
参数: 无
返回: 无

举例：被中断入口函数调用。

IRQ_Exception

功能: 调用中断处理函数
函数定义 : void IRQ_Exception(void)
参数: 无
返回: 无

举例:

```
void IRQ_Exception(void)
{
    uint32_t IntStat;
    uint32_t irq_status;
    IntStat = intc_status_get();
    #if (SYSTEM_SLEEP) ...
    #endif
    #if (UART_DRIVER)    ...
    #endif
    #if (GPIO_DRIVER) ...
    #endif
    #if (PWM_DRIVER)
    if(IntStat & INT_STATUS_PWM1_bit)
    {
        irq_status |= INT_STATUS_PWM1_bit;
        pwm_isr();
    }
    if(IntStat & INT_STATUS_PWM2_bit) ...
    if(IntStat & INT_STATUS_PWM3_bit) ...
    if(IntStat & INT_STATUS_PWM4_bit) ...
#endif
#if (ADC_DRIVER) ...
#endif
#if RTC_DRIVER ...
#endif
    intc_status_clear(irq_status);
}
```

（一）软件定时器 API接口

ke_timer_set

功能: 10ms定时基准函数
函数定义 : void ke_timer_set(ke_msg_id_t const timer_id, ke_task_id_t const task, uint32_t delay)
参数:
timer_id 定时器消息.
task 任务标识
delay 定时时间，10ms为单位
返回: 无

举例:
ke_timer_set(APP_TIMER_TEST,TASK_APP ,50)： 50*10ms回调一次

ke_timer_clear

功能: 清除软件定时器
函数定义 : ke_timer_clear(ke_msg_id_t const timerid, ke_task_id_t const task);
参数:
timer_id 定时器消息.
task 任务标识
返回: 无

举例：ke_timer_clear (APP_TIMER_TEST,TASK_APP ,50)

（一）ADC API接口

adc_init

作用 ：初始化ADC
函数定义 ： void adc_init(uint8_t chanle,uint8_t mode)
参数 ：
Chanle : 选择ADC通道
Mode ：ADC模式选择 00:休眠模式01:单步模式10:软件控制11:连续模式
返回值 ：无

举例：adc_init(3，1)： 单步模式读取GPIOC3口的ADC值

adc_isr

功能: 处理ADC中断事件
函数定义 ：void adc_isr(void)
参数: 无
返回: 无

举例：被中断入口函数调用。

（一）Flash API接口

flash_init

作用 ：初始化flash
函数定义 ： void flash_init(void)
参数 ：无
返回值 ：无

举例：

flash_erase_sector

功能: 擦除扇区
函数定义 ：void flash_erase_sector(uint8_t type,uint32_t address)
参数 ：
Type ：选择要擦除的flash类型 包括MAIN NVR。
address ：擦除扇区的首地址
返回值 ：无

举例：flash_erase_sector(FLASH_SPACE_TYPE_NVR,0x8000);擦除NVR区的0x8000地址所在的扇区。

flash_erase

功能: 擦除扇区
函数定义 ：void flash_erase(uint8_t flash_type, uint32_t address, uint32_t size)
参数 ：
flash_type ：选择要擦除的flash类型 包括MAIN NVR。
address ：擦除的首地址
size ： 擦除flash的长度
返回值 ：无

举例：flash_erase (FLASH_SPACE_TYPE_NVR,0x8000，60);擦除NVR区的0x8000地址开始的60个字节长度的区域。

flash_writeKey

作用 ：使能flash操作
函数定义 ： static void flash_writeKey(void)
参数 ：无
返回值 ：无

flash_unlock

作用 ： flash写保护使能
函数定义 ： static void flash_unlock(void)
参数 ： 无
返回值 ： 无

flash_start

作用 ： 开始操作flash
函数定义 ： static void flash_start(void)
参数 ： 无
返回值 ： 无

flash_status

作用 ： 判断操作flash是否完成
函数定义 ： static uint8_t flash_status(void)
参数 ： 无
返回值 ： 返回寄存器0x00803004 bit0的状态，该位flash操作完成后硬件自动清零

flash_clearKey

作用 ： 关闭flash操作使能
函数定义 ： static void flash_clearKey(void)
参数 ： 无
返回值 ： 无

flash_write

作用 ： 关闭flash操作使能
函数定义 ： void flash_write(uint8_t flash_space, uint32_t address, uint32_t length, uint8_t *buffer)
参数 ：
flash_space : flash类型。 flash类型:NVR 和MAIN
address ：写入数据的首地址
length ：写入数据的长度
buffer ：写入的数据
返回值 ： 无

举例：flash_write(FLASH_SPACE_TYPE_NVR,0x8000,60,temp_buff);往NVR区域写入60个字节数据，数据存放在temp_buff，首地址：0x8000

flash_read

作用 ： 关闭flash操作使能
函数定义 ： void flash_read(uint8_t flash_space, uint32_t address, uint32_t length, uint8_t *buffer)
参数 ：
flash_space : flash类型。 flash类型:NVR 和MAIN
address ：读出数据的首地址
length ：读出数据的长度
buffer ：读出的数据
返回值 ： 无

举例：flash_read(FLASH_SPACE_TYPE_NVR,0x8000,60,temp_buff);在NVR区域读出60个字节数据，数据存放在temp_buff，首地址：0x8000

write_lmecc_pointq

作用 ： 往flash连续写入数据
函数定义 ： void write_lmecc_pointq(uint8_t *buff1, uint8_t *buff2)
参数 ：
buff1 ：写入的数据
Buff2 ：写入的数据
返回值 ： 无

举例：write_lmecc_pointq(&writedata_one[0],&writedata_one[28]);将writedata_one数组里面的数据写入flash

read_lmecc_pointq_status

作用 ： 对写进去的数据和读出来的数据进行校验
函数定义 ： uint8_t read_lmecc_pointq_status(void)
参数 ： 无
返回值 ： 校验结果 校验正确返回 1 校验错误返回 0

read_lmecc_pointq

作用 ： 读出写入flash的数据
函数定义 ： void read_lmecc_pointq(uint8_t *buff1,uint8_t *buff2)
参数 ：
buff1 ：读出的数据
Buff2 ：读出的数据
返回值 ： 无

举例：read_lmecc_pointq(&readdata_one[0],&readdata_one[30]);读出flash里面的数据，放入数组readdata_one数组里面。

crc16

作用 ： 16为crc校验
函数定义 ： uint16_t crc16 (uint8_t *pbuff, uint32_t len)
参数 ：
pbuff ：校验数据
len ：数据长度
返回值 ： 无

举例：crc16(temp_buff,58);对temp_buff里面的数据进行校验

（一）软件定时器 API接口

ke_timer_set

功能: 10ms定时基准函数
函数定义 : void ke_timer_set(ke_msg_id_t const timer_id, ke_task_id_t const task, uint32_t delay)
参数:
timer_id 定时器消息.
task 任务标识
delay 定时时间，10ms为单位
返回: 无

举例：ke_timer_set(APP_TIMER_TEST,TASK_APP ,50)： 50*10ms回调一次

ke_timer_clear

功能: 清除软件定时器
函数定义 : ke_timer_clear(ke_msg_id_t const timerid, ke_task_id_t const task);
参数:
timer_id 定时器消息.
task 任务标识
返回: 无

举例：ke_timer_clear (APP_TIMER_TEST,TASK_APP ,50)

（一）模拟IIC API接口

iic_init

作用 : 初始化iic
函数定义 ： void iic_init()
参数 : 无
返回值 : 无

iic_start

功能　：模拟iic起始信号函数
函数定义 : void iic_start()
参数　：无
返回　：无

iic_stop

功能　：模拟iic停止信号函数
函数定义 : void iic_stop()
参数　：无
返回　：无

iic_ack

功能　：模拟iic应答信号函数
函数定义 : void iic_ack(void)
参数　：无
返回　：无

iic_no_ack

功能　：模拟iic不应答信号函数
函数定义 : void iic_no_ack(void)
参数　：无
返回　：无

iic_waitask

功能　：CPU产生一个时钟，并读取器件的ACK应答信号
函数定义 : uint8_t iic_waitask(void)
参数　：无
返回　：返回0表示正确应答，1表示无器件响应

iic_tx_byte

功能　：写一个字节数据
函数定义 : void iic_tx_byte(uint8_t dat)
参数　：dat：写入的数据
返回　：无

iic_tx_byte

功能　：读一个字节数据
函数定义 : uint8_t iic_rx_byte()
参数　：
返回　：读出的数据

iic_tx_data

功能　：往AT24C02写数据
函数定义 : void iic_tx_data(uint8_t devAddr,uint8_t addr,uint8_t*buf,uint8_t size)
参数　：
　　devAddr：从机地址
　　addr　：数据写入的首地址
　　buf　：写入的数据
　　size　：写入数据的数量
返回　：

举例：iic_tx_data(0Xa0, 0X01,&writebuf[1],1);往AT24C02的0x01地址写入一个数据

iic_tx_data

功能 ：往AT24C02读数据
函数定义 : void iic_rx_data(uint8_t devAddr,uint8_t addr,uint8_t*buf,uint8_t size)
参数 ：
devAddr：从机地址
addr ： 读出数据的首地址
buf ： 读出的数据
size ： 读出数据的数量
返回 ：

举例：iic_tx_data(0Xa0, 0X01,&writebuf[1],1);往AT24C02的0x01地址读出一个数据

（一）硬件IIC API接口

i2c_init

作用 ：初始化硬件iic
函数定义 ：void i2c_init(uint32_t slaveAddr, uint32_t baudRate)
参数 ：
slaveAddr:从机地址
baudRate :分频 选择0
返回值 ：无

i2c_send_start

功能 ：发送 iic起始信号，只支持主模式
函数定义 ：static void i2c_send_start(void)
参数 ：无
返回 ：无

i2c_send_addr

功能 ：发送从机地址，只支持主模式
函数定义 : static void i2c_send_addr(void)
参数 ：无
返回 ：无

i2c_msg_reset

功能 ：重置i2c消息，必须确保最后一条i2c消息Tx/Rx已经完成
函数定义 : static void i2c_msg_reset(void)
参数 ：无
返回 ：无

is_i2c_busy

功能 ：CPU产生一个时钟，并读取器件的ACK应答信号
函数定义 : static ASK is_i2c_busy(void)
参数 ：无
返回 ：返回0 iic是空闲状态 返回1 iic是忙碌状态

i2c_get_last_msg

功能 ：获取i2c最后一个消息，以便您可以检查错误或处理接收数据
函数定义 : static I2C_MSG * i2c_get_last_msg(void)
参数 ：无
返回 ：读写正确返回i2c最后一个消息 读写错误返回NULL

i2c_msg_init

功能 ：初始化i2c消息，准备读/写数据。调用函数i2c_send_start()启动读/写数据
函数定义 : static STATUS i2c_msg_init(I2C_MSG *p_i2c_msg)
参数 ：信息描述
返回 ：返回 1 表示初始化信息正确 可以开始读/写数据 返回0 表示初始化信息错误 不继续开始读/写数据

i2c_write

功能 ：往AT24C02写数据
函数定义 : STATUS i2c_write(uint8_t devAddr, uint8_t addr, uint8_t*buf, uint8_t size)
参数 ：
devAddr：从机地址
addr ： 数据写入的首地址
buf ： 写入的数据
size ： 写入数据的数量
返回 ：写入成功返回 1 写入失败返回 0

举例：i2c_write(0x50,0x00,&iic_writebuf[1],1);往AT24C02的0x00地址写入一个数据

i2c_read

功能 ：往AT24C02读数据
函数定义 : STATUS i2c_read(uint8_t devAddr, uint8_t addr, uint8_t*buf, uint8_t size)
参数 ：
devAddr：从机地址
addr ： 读出数据的首地址
buf ： 读出的数据
size ： 读出数据的数量
返回 ：读出成功返回 1 读出失败返回 0

举例：i2c_read(0x50,0x00,&iic_readbuf[0],1);往AT24C02的0x00地址读出一个数据

i2c_isr

功能: 处理硬件iic中断事件
函数定义 ：void i2c_isr(void)
参数: 无
返回: 无

举例：被中断入口函数调用。

（一）按键睡眠唤醒

rwip_sleep

功能: 选择睡眠模式
函数定义 ：uint8_t rwip_sleep(void)
参数: 无
返回: 返回 1 进入低速广播模式 返回3 进入低功耗模式

cpu_reduce_voltage_sleep

功能: ARM9 Core时钟关闭和芯片进入低电压模式
函数定义 ：void cpu_reduce_voltage_sleep(void)
参数: 无
返回: 无

cpu_idle_sleep

功能: ARM9 Core时钟关闭
函数定义 ：void cpu_idle_sleep(void)
参数: 无
返回: 无

gpio_isr

功能: 处理硬件gpio中断事件
函数定义 ：void gpio_isr(void)
参数: 无
返回: 无

举例：被中断入口函数调用。