YC31xx FLASH 接口说明

V1.4

Yichip Microelectronics

Revision History

Version	Date	Author	Description
V1.0	2020-2-19	Dengzhiqian	Initial version
V1.1	2020-04-11	Kiwen	增加接口
V1.2	2020-04-28	Kiwen	明确部分函数参数限制
V1.3	2020-05-11	Kiwen	修正 flash_blank_check 函数返回值描述
V1.4	2021-11-05	Kiwen	添加注意事项

Confidentiality Level:

confidential

目录

1 文档说明

1.1 编写目的

为使用 FLASH API 接口函数、相关 demo 及 FLASH 应用分区提供指南

1.2 适用范围

31xx 系列芯片

1.3 文件说明

Demo 路径为： ModuleDemo\QSPI
库文件路径为： Librarier\sdk
yc_qspi.h：FLASH 加密及非加密操作库函数头文件
yc_qspi.lib：FLASH 加密及非加密操作函数 keil 环境库文件
libyc_qspi.a：FLASH 加密及非加密操作函数 gcc 环境库文件
yc_encflash_bulk.h：FLASH 大块(bulk)加密写库函数头文件
yc_encflash_bulk.lib：FLASH 大块(bulk)加密写函数 keil 环境库文件
libyc_encflash_bulk.a：FLASH 大块(bulk)加密写函数 gcc 环境库文件

1.4 名词解释

Flash 物理地址：内部 flash 物理存储地址，地址参数要求物理地址的函数为明文操作函数
CPU 地址：CPU 访问 flash 的映射地址，地址参数要求 CPU 地址的函数为加密操作函数
A、因加密关系，物理地址与 CPU 地址不是一一对应关系，即 CPU 地址 0x1008000 不对应物理地址 0x8000.
B、加密算法中有乱序规则，乱序单元为 CPU 地址 32k，所以分配加密区域时大小必须为 32k 整数倍

1.5 库使用注意事项

yc_qspi.lib 库对 RAM 有要求，0x245fc-0x24974(888byte)这段 RAM 地址不能给 yc_qspi.lib 库作为 buf 使用，包括 buf 传参及库里面申明局部 buf，所以：
A、当工程中只用到了 yc_qspi.lib 且没有使用 yc_encflash_bulk.lib 时需要修改 yc_uart.c 的 UART DMA 申明，将其中一个 DMA BUF 指定到以上区域，具体代码如下：

```
#define uart_DMA_buf_len    1024
uint8_t uart0_DMA_buf[uart_DMA_buf_len] __attribute__((at(0x000245fc))) = {0};//keil

uint8_t uart0_DMA_buf[uart_DMA_buf_len] __attribute__((section(".ARM.__at_0x000245fc"))) = {0};//gcc
```

注意：gcc 工程还需在链接脚本中做处理才能将 buf 链接到指定地址
B、当工程中同时用到 yc_qspi.lib 和 yc_encflash_bulk.lib 或者都没有使用这两个 lib 则不用做以上处理

2 接口说明

2.1 qspi_flash_sectorerase

函数原型：uint8_t qspi_flash_sectorerase(uint32_t flash_addr);
说明： qspi flash 4k 扇区擦除函数。

参数	方向	说明
uint32_t flash_addr	IN	待擦除的地址(flash 物理地址 4k 对齐)

返回值	说明
uint8_t	1：SUCCESS 0：ERROR

2.2 qspi_flash_blockerase32k

函数原型：uint8_t qspi_flash_sectorerase(uint32_t flash_addr);
说明： qspi flash 32k 块擦除函数。

参数		方向	说明
uint32_t flash_addr		IN	待擦除的地址(flash 物理地址 32k 对齐)

返回值	说明
uint8_t	1：SUCCESS 0：ERROR

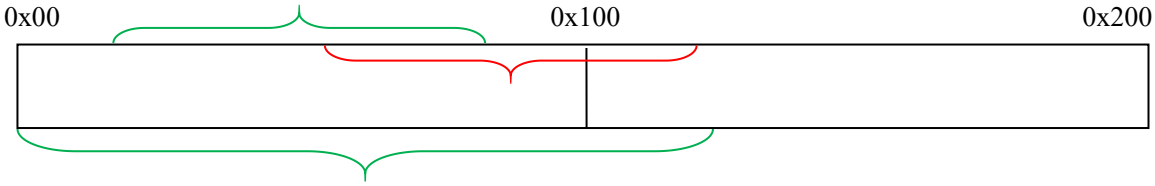
2.3 qspi_flash_write

函数原型：uint8_t qspi_flash_write(uint32_t flash_addr, uint8_t *buf, uint32_t len);
说明： qspi flash 非加密写函数.

参数	方向	说明
uint32_t flash_addr	IN	待写入的地址(flash 物理地址)
uint8_t *buf	IN	待写入的数据
uint32_t len	IN	待写入的长度

返回值	说明
uint8_t	1：SUCCESS 0：ERROR

qspi_flash_write 函数页对齐说明：



如上图，绿色图示的两种写 flash 方式是允许的，红色图示的方式不允许，即起始地址不是页(256bytes)的首地址时不允许跨页写。

2.4 qspi_flash_read

函数原型：uint8_t qspi_flash_write(uint32_t flash_addr, uint8_t *buf, uint32_t len);
说明： qspi flash 非加密读函数。

参数	方向	说明
uint32_t flash_addr	IN	待读取的地址(flash 物理地址)
uint8_t *buf	OUT	读取数据数据存入的首地址
uint32_t len	IN	待读取的长度

返回值	说明
uint8_t	1：SUCCESS 0：ERROR

2.5 enc_erase_flash_32byte

函数原型：void enc_erase_flash_32byte(uint32_t flash_addr);
说明：qspi flash 加密擦除 32 字节函数，用于少量数据区域擦除，列如改写带加密的参数区

参数	方向	说明
uint32_t flash_addr	IN	待擦除地址，基于 CPU 地址 32 字节对齐

返回值	说明
None	None

2.6 enc_erase_flash_32k

函数原型：void enc_erase_flash_32k(uint32_t flash_addr)
说明：qspi flash 加密擦除 32k 函数。

参数	方向	说明

uint32_t flash_addr	IN	待擦除地址，基于 CPU 地址 32k 对齐

返回值	说明
None	None

2.7 enc_earse_flash_app_area

函数原型：uint8_t enc_earse_flash_app_area(uint32_t addr,uint32_t len);
说明：qspi flash code 区擦除函数，用于擦除连续加密区域，列如升级固件前擦除原固件

参数	方向	说明
uint32_t addr	IN	待擦除地址，基于 CPU 地址 32k 对齐
uint32_t len	IN	

返回值	说明
None	None

2.8 enc_write_flash

函数原型：void enc_write_flash (uint32_t flash_addr, uint8_t *buf, uint32_t len);
说明： qspi flash 加密写函数，同一个 32 byte 区域，擦除之后只能写一次，如果 32 byte 都要写满，则需要一次性写入。

参数	方向	说明
uint32_t flash_addr	IN	待写入地址，基于 CPU 地址 32 字节对齐
uint8_t *buf	IN	待写入数据
uint32_t len	IN	待写入长度(必须为 32 的整数倍)

返回值	说明
None	None

2.9 enc_read_flash

函数原型：void enc_read_flash(uint32_t flash_addr, uint8_t *buf, uint32_t len);
说明： qspi flash 加密读函数。

参数	方向	说明
uint32_t flash_addr	IN	待读取地址(CPU 地址)
uint8_t *buf	OUT	读取数据存入的首地址
uint32_t len	IN	待读取数据长度

返回值	说明
None	None

2.10flash_blank_check

函数原型：Boolean flash_blank_check(uint32_t startaddr,uint32_t len);
说明： 检测 qspi flash 指定区域数据是否符合加密规则，在通过总线直接访问 flash 数据前，需要对访问的区域进行检测，否则可能触发硬件错误。enc_read_flash 已集成此操作，可直接使用。

参数	方向	说明
uint32_t startaddr	IN	待检测的起始地址(CPU 地址)
uint32_t len	IN	检测长度

返回值	说明
Boolean	返回 TRUE 表示此区域为空，即此区域为无效数据，返回 FALSE 则此区域为有效数据，可进行加密读操作或通过总线直接访问

2.11prefetch

函数原型：void prefetch(void *start_addr, void *end_addr)
说明：将 qspi flash 指定区域的数据更新到 cache，在通过总线直接访问 flash 数据前，需要用此函数刷新 cache。enc_read_flash 已集成此操作，可直接使用。

参数	方向	说明
void *start_addr	IN	起始地址(CPU 地址)
void *end_addr	IN	结束地址(CPU 地址)

返回值	说明
None	None

2.12app_enable_download

函数原型：void app_enable_download();
说明：清除应用信息头，清除后复位即进入 ccid boot 下载

参数	说明
None	None

返回值	说明
None	None

2.13app_clear_disable_download_flag

函数原型：void　app_clear_disable_download_flag();
说明：清除禁用下载标记位，清除后复位需要 gpio1(uart0_tx)接地才会进入 ccid boot 下载

参数	说明
None	None

返回值	说明
None	None

2.14enc_write_flash_bulk_init

函数原型：void enc_write_flash_bulk_init();
说明：加密 bulk 写初始化函数，函数在 yc_encflash_bulk 库中。

参数	说明
None	None

返回值	说明
None	None

2.15enc_write_flash_bulk

函数原型：void enc_write_flash_bulk(uint32_t flash_addr, uint8_t *buf, uint32_t len,uint8_t isend);
说明：　qspi flash 大块(bulk)加密写函数，bulk 加密写方式能大幅提升速度，为 BOOT 升级应用专用，同时此方式会分配 36k RAM 作为加密缓冲 buf，满 32k 或结束(isend==1)进行写入，在应用中对少量数据(32k 以下)的加密写建议使用 enc_write_flash 函数，enc_write_flash_bulk 函数在 yc_encflash_bulk 库中。
注意：1、整个下载过程只需调用一次 enc_write_flash_bulk_init()
　　2、enc_write_flash_bulk_init()之后第一次使用 enc_write_flash_bulk 时，起始地址必须为 MCU 地址的 32k 对齐地址

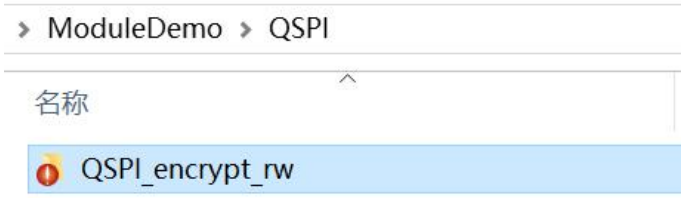
参数	方向	说明
uint32_t flash_addr	IN	待写入地址，基于 CPU 地址 32 字节对齐
uint8_t *buf	IN	待写入数据
uint32_t len	IN	待写入长度(非最后一包时，长度必须为 32 的整数倍)

uint8_t isend	IN	传入升级过程最后一包数据时为 1，其余情况为 0

返回值	说明
None	None

3 示例代码及说明

示例代码存放在 ModuleDemo\QSPI 目录下（如下图）



① QSPI_encrypt_rw : QSPI 加密读写示例

3.1 示例代码

```
int main(void)
{
    UART_Configuration(); // 串口初始化，初始化配置参考 UART 应用说明

    MyPrintf("YC3121 QSPI Encrypt read write Demo !\n\n");

    test_enc_write_flash_32byte(0x1020000); // 加密写 32byte 数据

    test_enc_write_flash_32K(0x1030000); // 加密写 32k 数据

    MyPrintf("TEST END!\r\n");

    while (1)
    {

    }
}

void test_enc_write_flash_32byte(uint32_t base_addr)
{
    uint8_t wbuf[32] = {0};
    Boolean isblank = FALSE;
    // check flash is blank
    isblank = flash_blank_check(base_addr&0xffffff, (base_addr+0x20)&0xffffff);
```

```
    MyPrintf("\r\nflash blank check(%x~%x)= %d", base_addr, base_addr + 0x20, isblank);

    enc_erase_flash_32byte(base_addr); //  擦除起始地址为 base_addr 的 32byte 数据

    for (int i = 0; i < 32; i++)
    {
        wbuf[i] = i;
    }
    printv(wbuf, 32, "wbuf:");

    enc_write_flash(base_addr,wbuf,32); //  加密写
    prefetch((volatile uint32_t *)base_addr, (volatile uint32_t *)(base_addr+32)); //  更新 cache 数据

    //read new data
    printv((volatile uint8_t*)(base_addr), 32, "base_addr NEW data:");
    for (int i = 0; i < 32; i++)
    {
        if(wbuf[i] != *(volatile uint8_t*)(base_addr+i))
        {
            MyPrintf("Error wbuf[%d] = %x\r\n", i, *(volatile uint8_t*)(base_addr+i));
        }
    }
}
```

4 FLASH 应用分区说明

4.1 FLASH 加密说明

YC31xx 系列芯片提供 flash 数据加密及不加密两种操作方式。因为 CPU 取指令执行时是带解密操作的，所以 code 区的数据必须使用加密写入，参数区则可以加密也可以不加密。

- **加密最小单元为 32byte：**CPU 地址 0x1000000-0x100001f 为 flash 第一个 32byte 单元，往后依此类推。加密写入时，同一个 32 byte 区域，擦除之后只能写一次，如果 32 byte 都要写满，则需要一次性写入。
- **加密数据以 32byte 为一个单元在 32k 区域内乱序存放：**乱序规则在每次使用 ROM 自带的串口下载接口更新固件时更新一次，CPU 地址 0x1000000-0x1007fff 为 flash 第一个 32k 单元，往后依此类推

4.2 FLASH 应用分区参考

基于 4.1 所述特性，flash 中的应用分区应跟 CPU 地址 32k 对齐，即起始地址和大小是 32k 整数倍，下面是基于 512kflash 的分区参考：

- 0x1000000-0x1007fff:二次 BOOT(32k)，CPU 地址，由 ROM BOOT 加密方式写入

- 0x1010000-0x1067fff:应用区(352k) ，CPU 地址，由二次 BOOT 加密方式写入,因相连的两个 32kCPU 地址交界处，对于 flash 物理地址不是 4k 对齐的，因此建议 app 与 boot 之间预留 32k CPU 地址空间，避免 boot 升级 app 时出现正在擦除 app 与 boot 连接处，还没来得及写回 boot 尾部数据情况下设备掉电，导致 boot 损坏。
- 0x1076000-0x107ffff:参数区(40k) ，物理地址，由应用通过不加密方式读写，建议参数区放在 boot 及 app 等 code 区之后，参数区的起始地址(物理地址)基于 app 区的结束地址换算，列如 app 区域起始地址为 0x1010000，app 区域大小为 352k，则 app 区结束地址(app_end_addr)=0x1010000+352*1024 =0x1068000，则参数区物理地址(param_start_addr)换算如下:

  param_start_addr=((((app_end_addr&0xffffff)+(0x8000-1))/0x8000)*0x9000+0x1000)|0x1000000

  param_start_addr=0x1076000

  公式说明:

  app_end_addr&0xffffff ： 去掉最高位 1

  ((app_end_addr&0xffffff)+(0x8000-1))/0x8000 ： 32k 对齐

  *0x9000 ： 每 32k CPU 地址空间占用 36k 物理空间

  +0x1000 ： flash 头部内部使用区域

  |0x1000000 ： 恢复最高位 1

4.3FLASH 接口使用说明

BOOT 下载及升级应用时，属于大块(bulk)连续操作，推荐使用以下接口函数:

擦除函数：enc_earse_flash_app_area，连续大块擦除，此函数效率较高

写入函数：enc_write_flash_bulk，bulk 加密写函数，需先使用 enc_write_flash_bulk_init 函数初始化

读取函数：enc_read_flash

应用中对加密参数区的操作，使用以下接口函数:

擦除函数：enc_erase_flash_32byte

写入函数：enc_write_flash

读取函数：enc_read_flash

应用中对非加密参数区的操作，使用以下接口函数:

擦除函数：qspi_flash_sectorerase、qspi_flash_blockerase32k

写入函数：qspi_flash_write

读取函数：qspi_flash_read