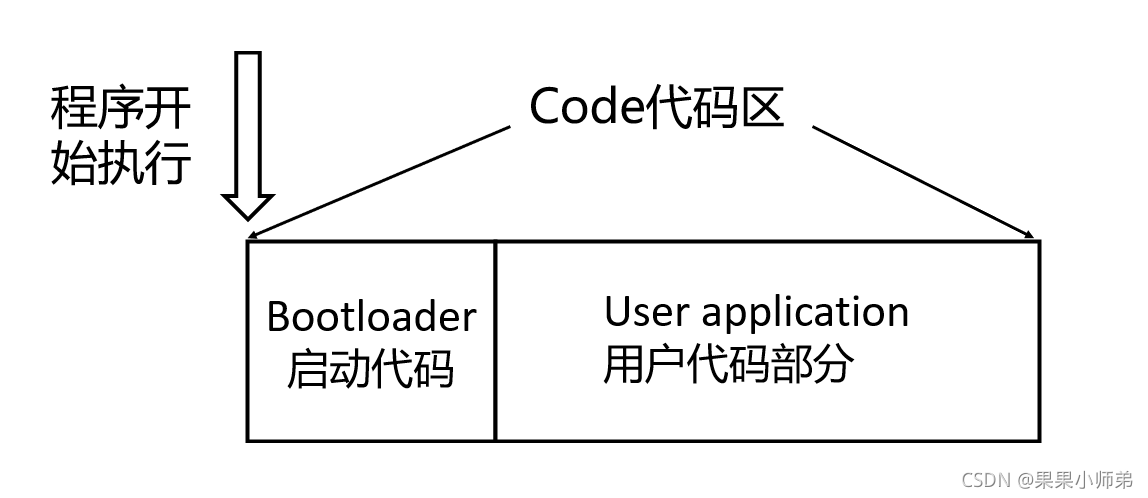
## IAP技术

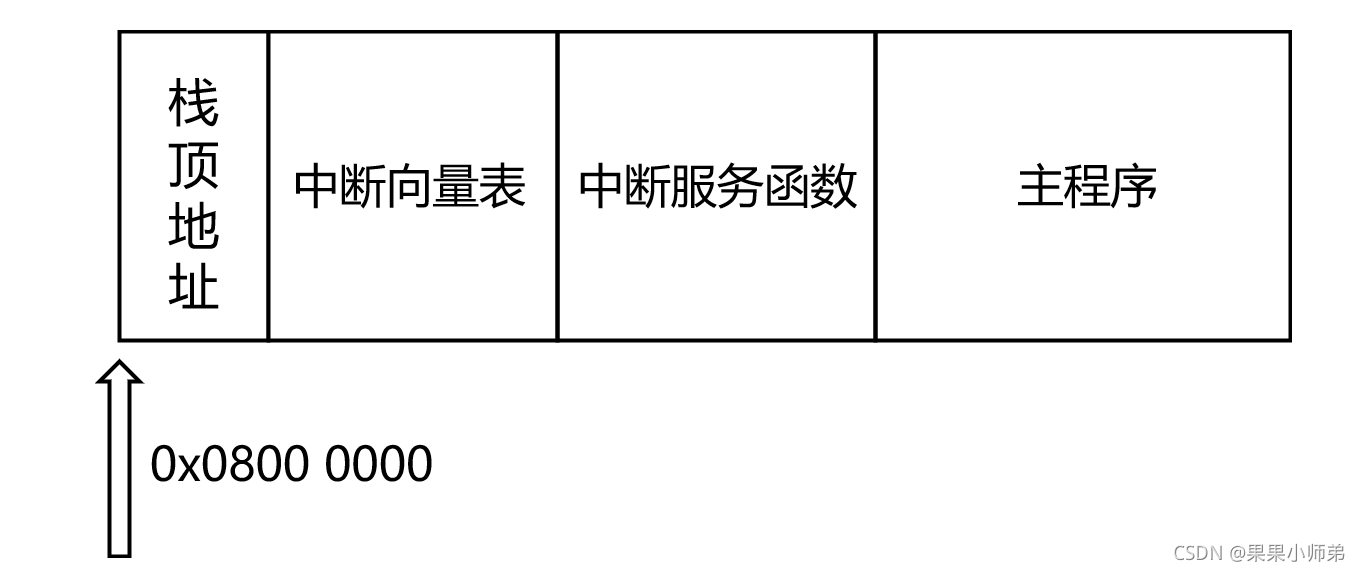
IAP即为In Application Programming(在应用中编程)，一片STM32芯片的Code(代码)区内一般只有一个用户程序。而IAP方案则是将代码区划分为两部分，两部分区域各存放一个程序，一个叫bootloader(引导加载程序)，另一个叫user application(用户应用程序)。bootloader在出厂时就固定下来了，在需要变更user application时只需要通过触发bootloader对userapplication的擦除和重新写入即可完成用户应用的更换。如图所示



单片机上电，程序执行初始进入bootloader，在bootloader里面检测条件是否被触发(可通过按键是否被按下、串口是否接收到特定的数据、U盘是否插入等等)，如果没有则直接跳转到user application执行应用；如果有则进行擦除用户代码并重新写入新的用户代码。

## 普通程序的内部FLASH分配（bin文件）

内置Flash的分配情况大致如下图。



## STM32启动流程概述

1.内核初始化;

1.内核复位和NVIC寄存器部分清零；

2.内核设置堆栈：内核从向量表0地址读出堆栈地址，并**设置主堆栈指针（SP\_main）**；

3.设置PC和LR寄存器

a. LR设置未初始复位值0xFFFF FFFF

b. 单片机的内部硬件机制自动将PC指针定位到中断向量表的复位中断向量处，把复位中断函数Reset\_Handler的地址赋值给PC指针，然后跳转执行Reset\_Handler。

2.强制**PC指针指向中断向量表**的复位中断向量执行复位中断函数;

3.在复位中断函数中**调用SystemInit函数，初始化时钟，配置中断向量表**等

4.调用 \_\_main 函数完成全局/静态变量的初始化和重定位工作，**初始化堆栈**和库函数

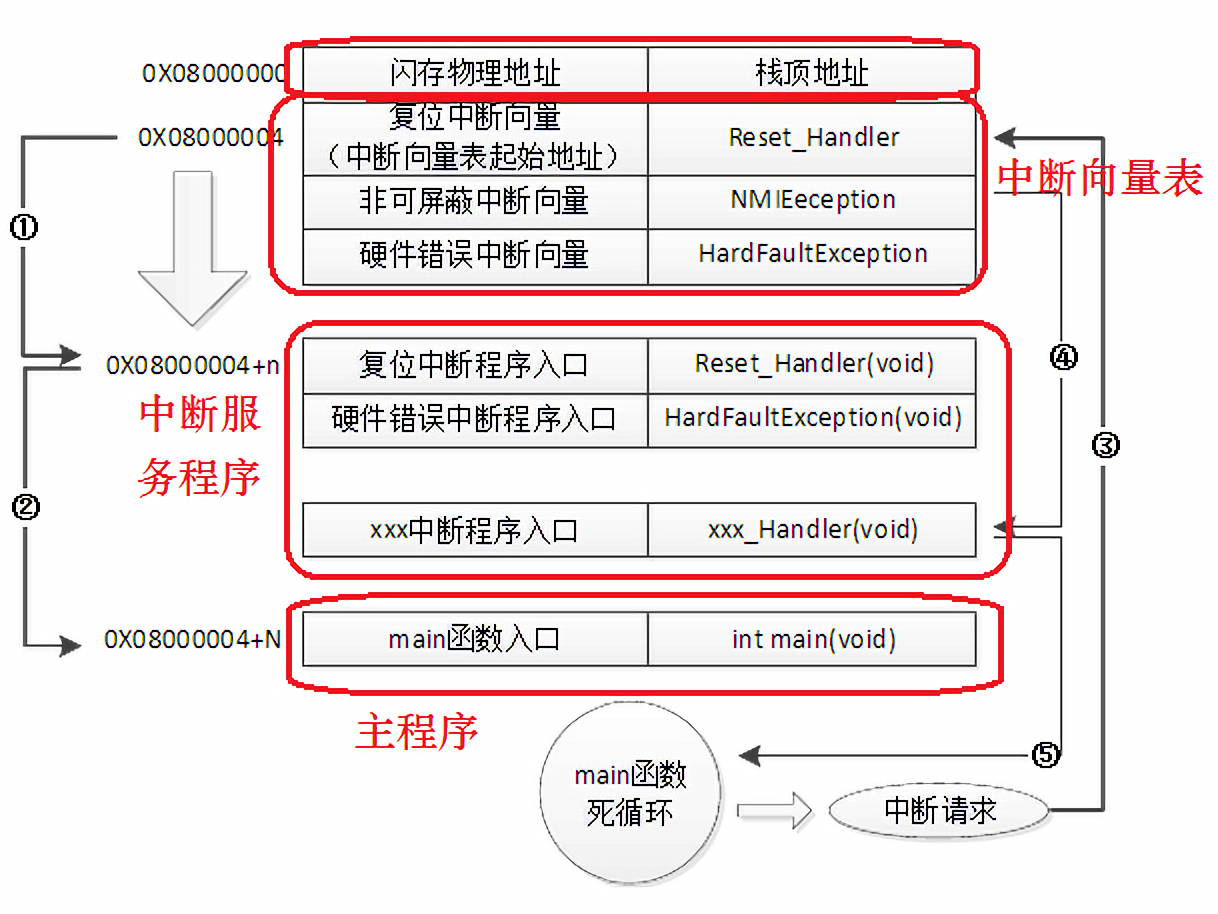
5.**跳转到main函数**中执行

【\_\_main函数详解】

\_\_main 其实不是我们定义的，当编译器编译时，只要遇到这个标号就会定义这个函数，该函数的主要功能是：负责初始化栈、堆，配置系统境，并在最后跳转到用户自定义的main函数，从此来到C的世界。

## 中断的发生

正常情况下发生中断的过程为：发生中断(中断请求)到中断向量表查找中断函数入口地址跳转到中断函数执行中断函数中断返回。也就是说在STM32的内置的Flash中有一个中断向量表来存放各个中断服务函数的入口地址。



STM32F10x有一个中断向量表，这个中断向量表存放在代码开始部分的后4个字节处(即0x0800 0004)，代码开始的4个字节存放的是堆栈栈顶的地址，当发生中断后程序通过查找该表得到相应的中断服务程序入口地址，然后再跳到相应的中断服务程序中执行。

上电后从0x08000004处取出复位中断向量的地址，然后跳转到复位中断程序的入口(标号①所示)，

执行结束后跳转到main函数中(标号②所示)。

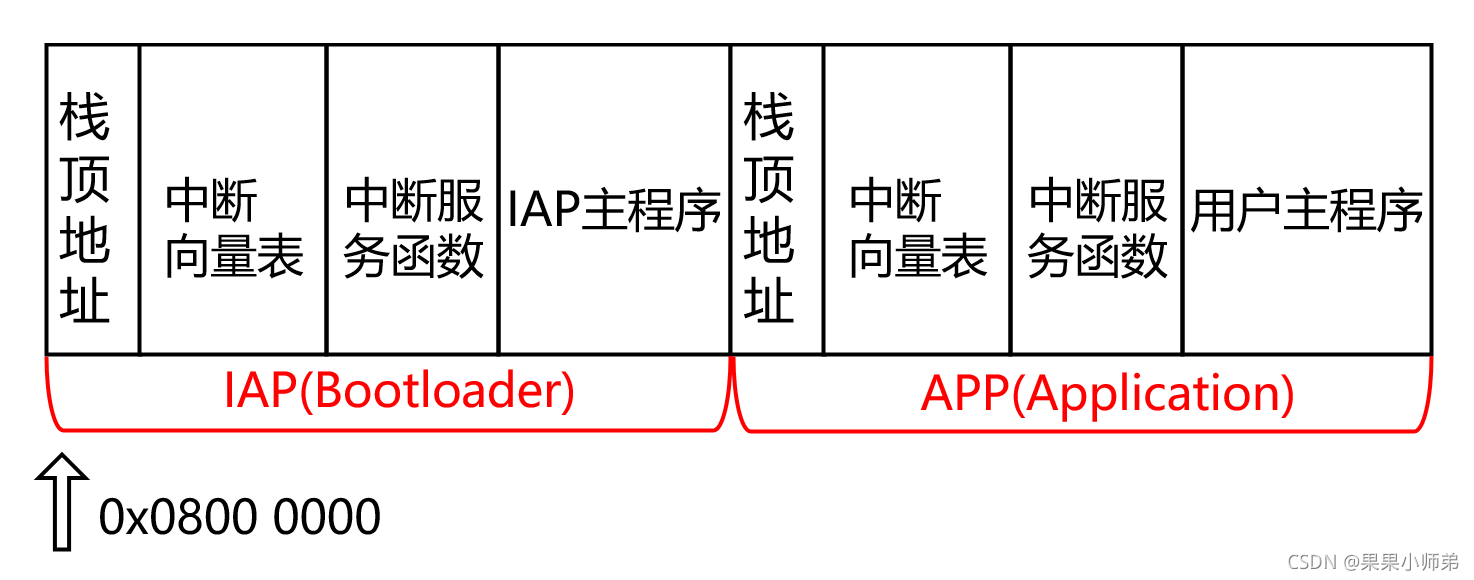
在执行main函数的过程中发生中断，则STM32强制将PC指针指回中断向量表处(标号③所示)，

从中断向量表中找到相应的中断函数入口地址，跳转到相应的中断服务函数(标号④所示)，

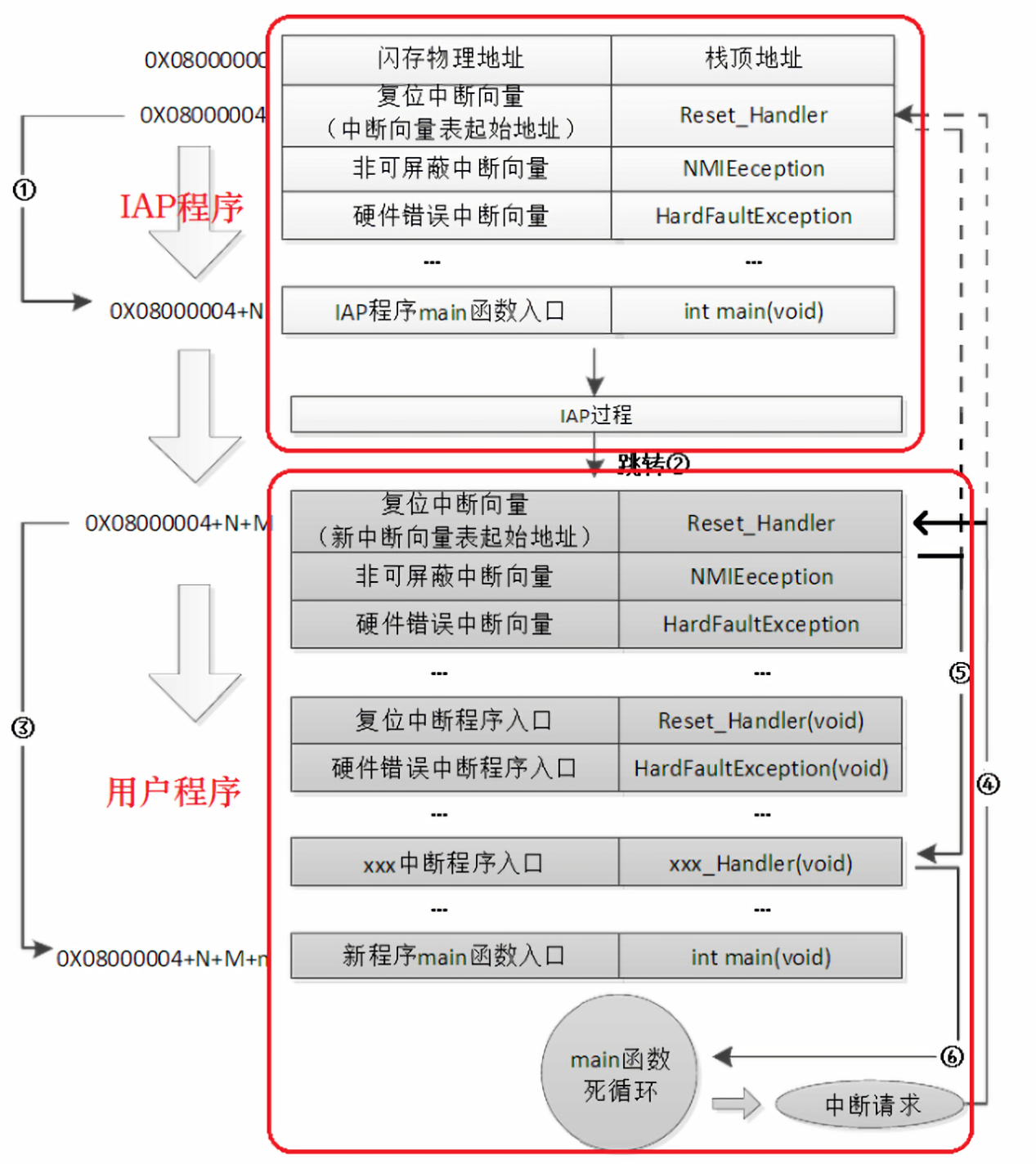
执行完中断函数后再返回到main函数中来(标号⑤所示)。

## 使用IAP方案的内部FLASH分配

若在STM32F103x中使用IAP方案，则内置的Flash分配情况大致如下图。



在内置的Flash里面添加一个BootLoader程序，BootLoader程序和user application各有一个中断向量表，假设BootLoader程序占用的空间为N+M字节，则程序的走向应该如下图所示。



上电初始程序依然从0x08000004处取出复位中断向量地址，执行复位中断函数后跳转到IAP的main(标号①所示)，

在IAP的main函数执行完成后强制跳转到0x08000004+N+M处(标号②所示)，

最后跳转到新的main函数中来(标号③所示)，

当发生中断请求后，程序跳转到新的中断向量表中取出新的中断函数入口地址，再跳转到新的中断服务函数中执行(标号④⑤所示)，

执行完中断函数后再返回到main函数中来(标号⑥所示)。

对于步骤④⑤，在main函数的执行过程中，如果CPU得到一个中断请求，PC指针本来应该跳转到0x08000004处的中断向量表，由于我们设置了中断向量表偏移量为N+M，因此PC指针被强制跳转到0x08000004+N+M处的中断向量表中得到相应的中断函数地址，再跳转到相应新的中断服务函数，执行结束后返回到main函数中来。

## OTA技术

空中编程（英语：Over-the-air programming，缩写OTA）是一种为设备分发新软件、配置，乃至更新加密密钥（为例如移动电话、数字视频转换盒或安全语音通信设备——加密的双向无线电）的方法。OTA的一项重要特征是，一个中心位置可以向所有用户发送更新，其不能拒绝、破坏或改变该更新，并且该更新为立即应用到频道上的每个人。

OTA：Over-the-Air Technology，即空中下载技术。

OTA升级：通过OTA方式实现固件或软件的升级。

只要是通过无线通信方式实现升级的，都可以叫OTA升级，比如网络/蓝牙---OTA升级

通过有线方式进行升级，叫本地升级，比如通过UART，USB或者SPI通信接口来升级设备固件--IAP升级

IAP升级---通过有线的方式进行升级

## 双区OTA

双区模式中老固件和新固件在flash中各占一块bank（存储区）。假设老固件放在bank0（运行区）中，新固件放在bank1（下载区）中，升级的时候，应用程序先把新固件下载到bank1中，只有当新固件下载完成并校验成功后，系统才会跳入BootLoader程序，然后擦除老固件所在的bank0区，并把bank1的新固件拷贝到bank0中。

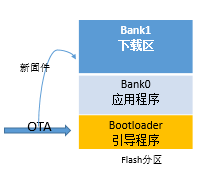
后台式下载必须采用双区模式进行升级。

优点：

升级过程中出现问题或者新固件有问题，它还可以选择之前的老固件老系统继续执行而不受其影响。

缺点：

多占用flash空间的一个存储区，在系统资源比较紧张的时候较为困难。



## 单区OTA

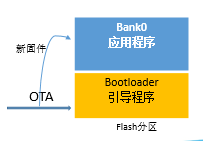
单区模式的非后台式下载只有一个bank0（运行区），老固件和新固件共享这一个bank0。升级的时候，进入bootloader程序后先擦除老固件，然后直接把新固件下载到同一个bank中，下载完成后校验新固件的有效性，新固件有效升级完成，否则要求重来。

优点：

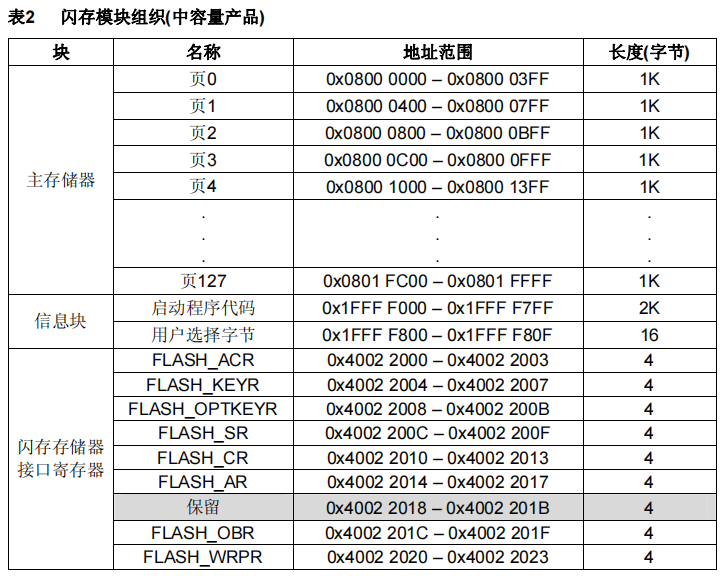
跟双区模式相比，单区模式节省了Flash空间的一个bank，在系统资源比较紧张的时候，单区模式是一个不错的选择。

缺点：

如果升级过程中出现问题或者新固件有问题，单区模式碰到这种情况就只能一直待在bootloader中，然后等待再次升级尝试，此时设备的正常功能已无法使用，从用户使用这个角度来说，可以说此时设备已经“变砖”了。



## 第一步：分配代码存储区大小

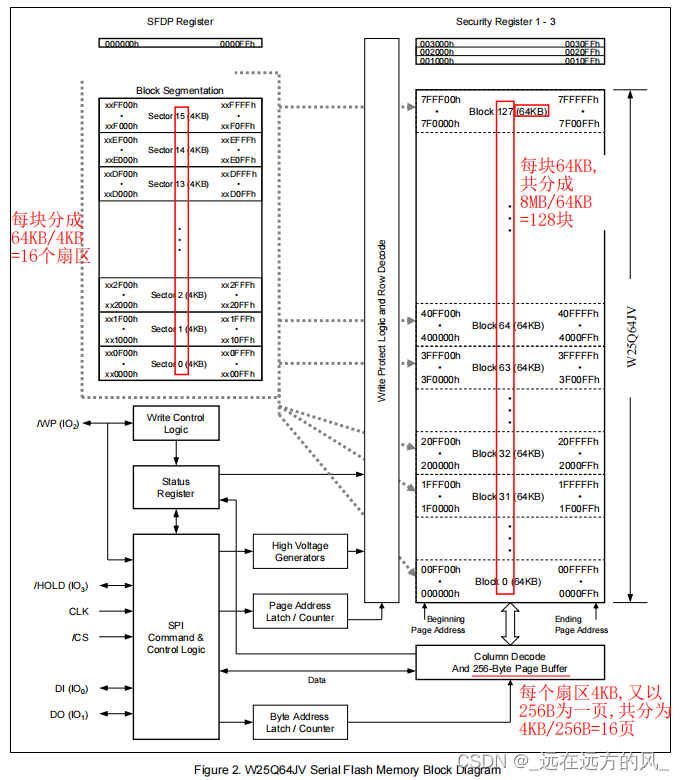


STM32F103C8T6内部FLASH起始地址0800 0000，容量64KB，每页1KB，地址字长32bit，数据宽度8bit，页内地址是低位10bit，页号是页内地址左侧6bit。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 第一页 | | 最后一页 | |
| 分区 | 容量 | 地址 | 页号 | 地址 | 页号 |
| BootLoader | 20KB | 0800 0000 | 0 | 0800 4C00 | 19 |
| UserApplication | 43KB | 0800 5000 | 20 | 0800 F800 | 62 |
| UserData | 1KB | 0800 FC00 | 63 |  |  |

## 启用W25Q64外部FLASH：实现双区OTA

STM32F103C8T6的RAM有20KB，起始地址0x2000 0000，不足以存储接受到的bin文件，如智慧大棚的bin文件有23KB。因此，在接收来自外部模块的bin文件时，可选择直接写入内部FLASH或者外部FLASH存储，这里我们选择写入外部FLASH中。



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 第一扇区 | | 最后扇区 | |
| 分区 | 容量 | 地址 | 扇区号 | 地址 | 扇区号 |
| Repository | 48KB | 000000 | 0 | 00B000 | 11 |

完成分配后，在代码中以宏定义的形式记录各个区域的起始地址



## 各模块接线方法

【USB转串口(CH340)】

串口RX接STM32的PA9，串口TX接STM32的PA10，串口波特率115200，8-0-1协议，无校验

【W25Q64】

模块给3.3V供电，模块CS引脚接STM32的PA4（推挽输出），模块CLK接STM32的PA5（复用推挽输出SPI1\_SCK），模块DO接STM32的PA6（上拉输入SPI1\_MISO），模块DI接STM32的PA7（复用推挽输出SPI1\_MOSI）

【WH-LTE-7S0(4G LTE CAT-1)】

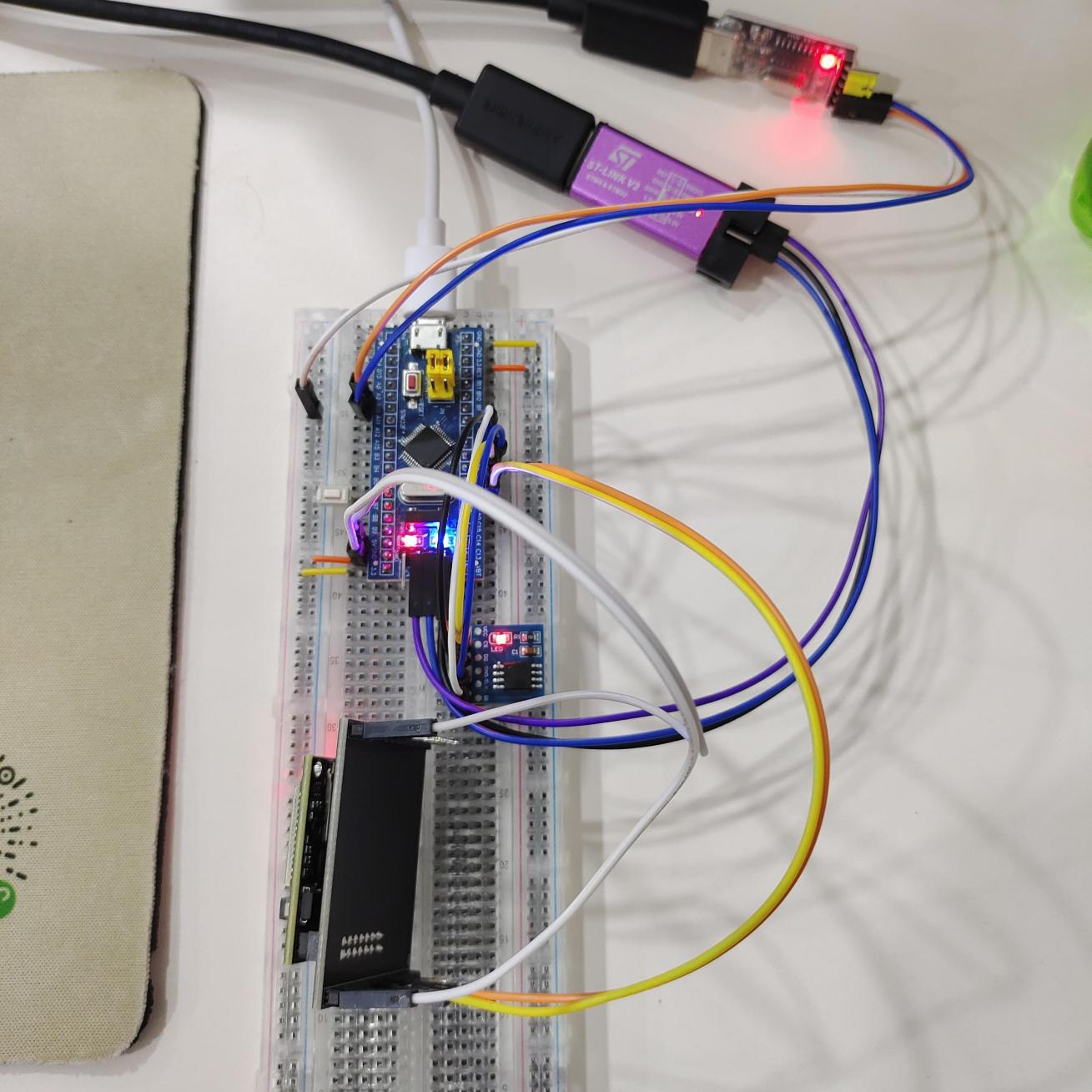
模块给12V独立供电，模块URX01接STM32的PA2，模块UTX01接STM32的PA3，串口波特率115200，8-0-1协议，无校验

【按键】

一端接PB6，一端接GND

【LED灯】

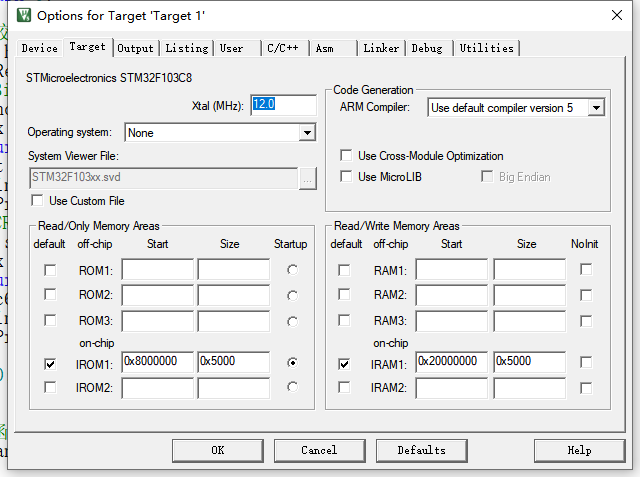
正极接VCC，负极接STM32的PC13



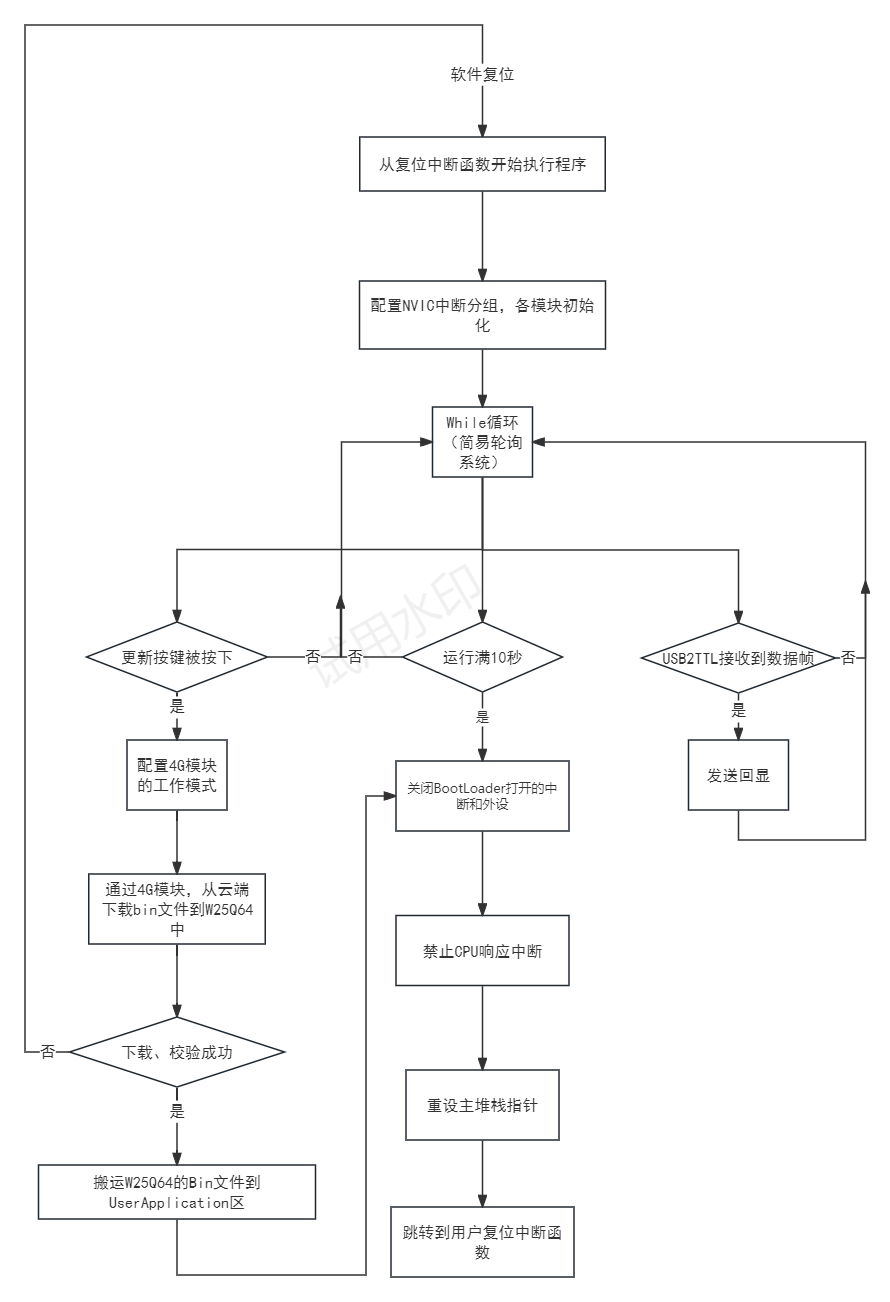
# 首先实现程序跳转

## 设置BootLoader起始地址

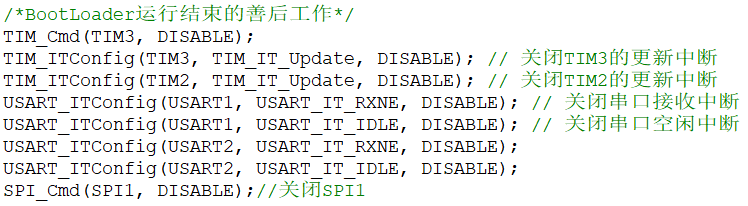
0x0800 0000



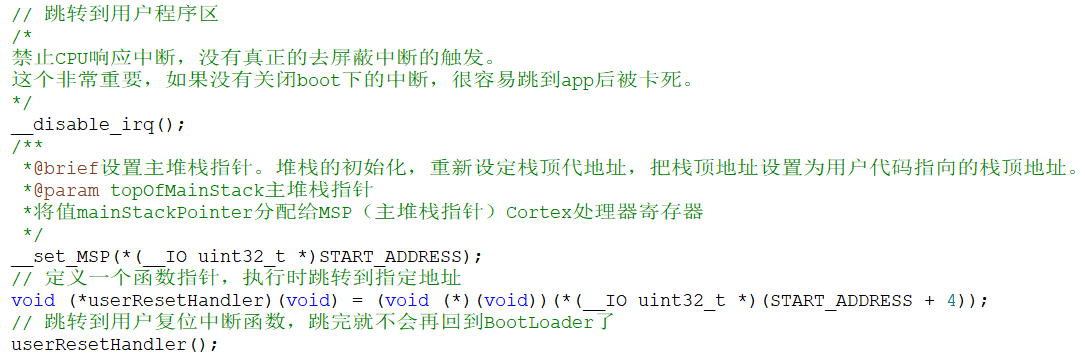
## BootLoader执行流程



跳转之前要把BootLoader打开的中断或外设都关上，以免影响UserApplication的运行。

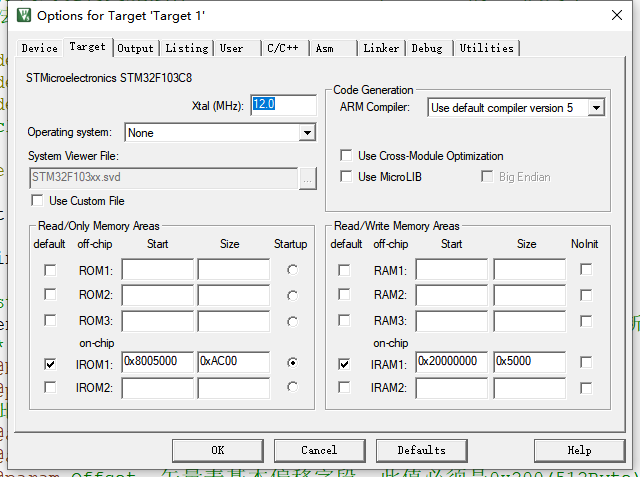


跳转部分关键代码：



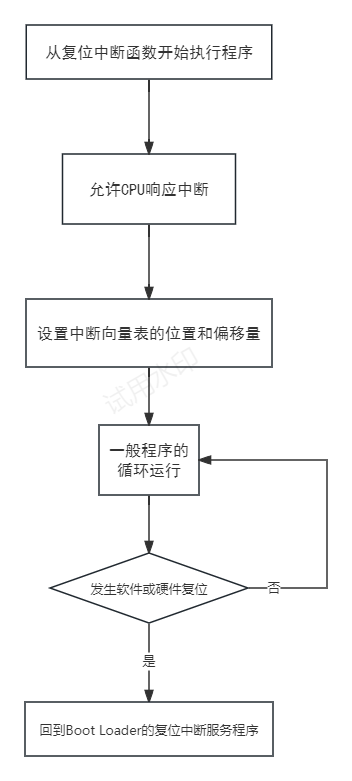
## 设置UserApplication起始地址

0x0800 5000

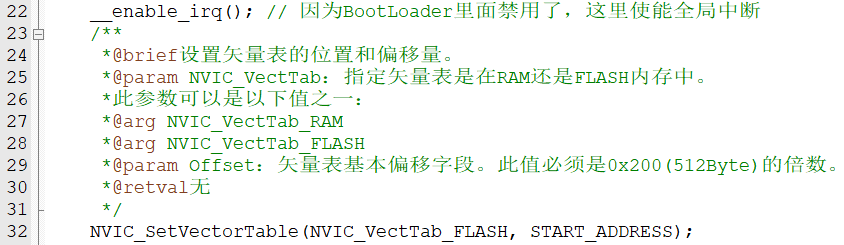


## UserApplicationr执行流程

IAP方案的用户程序只比普通程序多2行代码。



main函数起始部分关键代码：

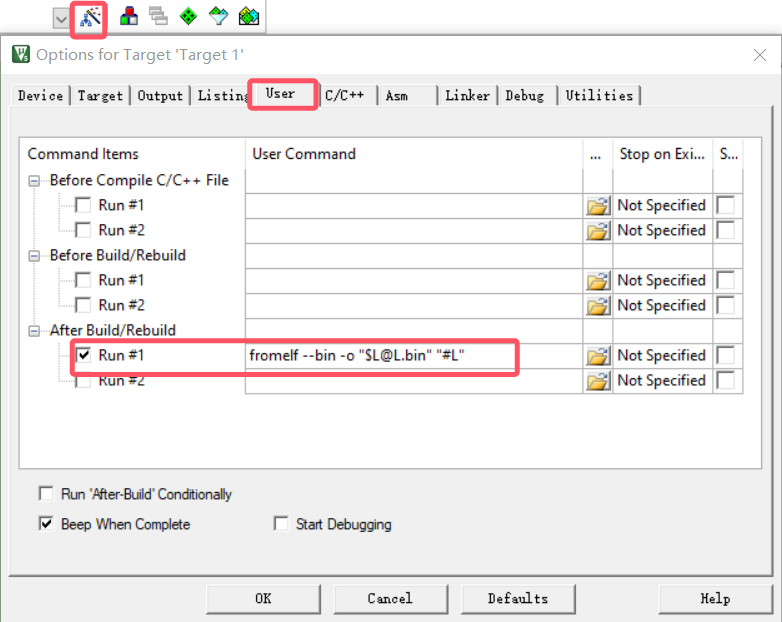


# 实现代码的远程下载及搬运

## 生成UserApplication的bin文件

bin文件大小 = Code机器代码指令 + RO-data常量 + RW-data初始非0全局变量

运行时占用RAM大小≥RW-data初始非0全局变量 + ZI-data初始为0全局变量



表示在编译后执行指令，bin文件位于项目的Objects文件夹下

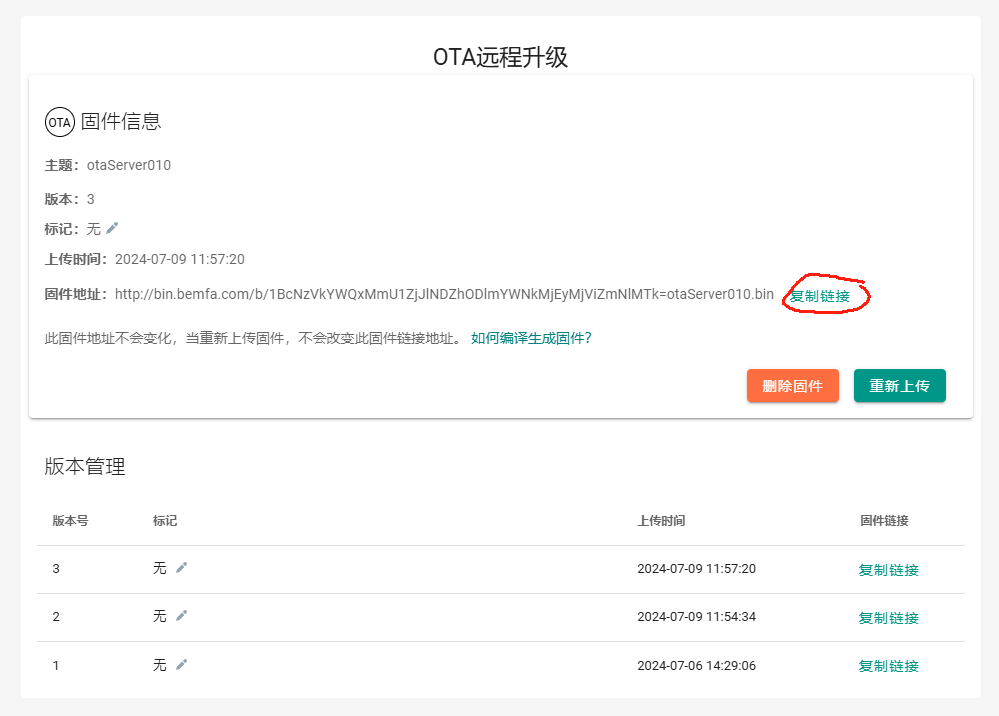
fromelf --bin -o "$L@L.bin" "#L"

## 巴法云上传bin文件

新建一个主题，点击OTA



上传用户代码的bin文件，复制链接



浏览器地址栏输入链接就能下载文件

<http://bin.bemfa.com:80/b/1BcNzVkYWQxMmU1ZjJlNDZhODlmYWNkMjEyMjViZmNlMTk=otaServer010.bin>

## 配置4G模块

模块采用串口通信，使用AT指令集配置为HTTPD模式

>[Tx->][09:42:26][asc]

AT+WKMOD=HTTPD // 设置工作模式为 HTTPD 模式

>[Rx<-][09:42:26][asc]

AT+WKMOD=HTTPD

OK

>[Tx->][09:42:26][asc]

AT+HTPTP=GET // 设置 HTTPD 的请求方式

>[Rx<-][09:42:26][asc]

AT+HTPTP=GET

OK

>[Tx->][09:42:26][asc]

AT+HTPURL=/ // 设置 HTTP 的请求 URL

>[Rx<-][09:42:26][asc]

AT+HTPURL=/

OK

>[Tx->][09:42:26][asc]

AT+HTPSV=bin.bemfa.com,80 // 设置 HTTP 的请求服务器

>[Rx<-][09:42:26][asc]

AT+HTPSV=bin.bemfa.com,80

OK

>[Tx->][09:42:26][asc]

AT+HTPTIM=10 // 设置 HTTP 的请求超时时间

>[Rx<-][09:42:26][asc]

AT+HTPTIM=10

OK

>[Tx->][09:42:26][asc]

AT+HTPHD=[0D][0A] // 设置 HTTP 的请求头信息

>[Rx<-][09:42:27][asc]

AT+HTPHD=[0D][0A]

OK

>[Tx->][09:42:27][asc]

AT+HTPPK=ON // 设置是否过滤回复信息包头

>[Rx<-][09:42:27][asc]

AT+HTPPK=ON

OK

>[Tx->][09:42:27][asc]

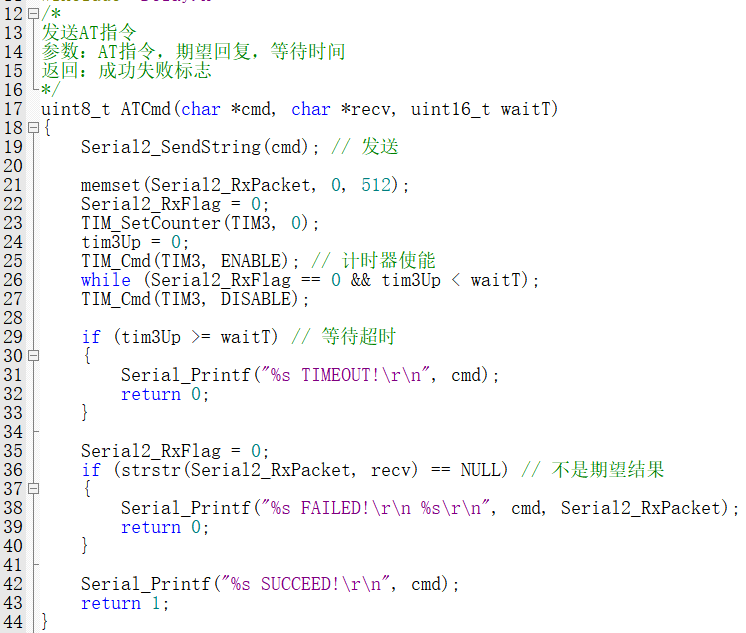
AT+S // 发送保存指令，发送之后模块会自动保存和重启

>[Rx<-][14:24:00][asc]

OK

## 思考：将手动发送AT指令转换为程序代码自动化发送

封装成一个函数



## 获取bin文件

>[Tx->][14:24:40][asc]

b/1BcNzVkYWQxMmU1ZjJlNDZhODlmYWNkMjEyMjViZmNlMTk=otaServer010.bin

【4G模块封装的HTTP请求帧】

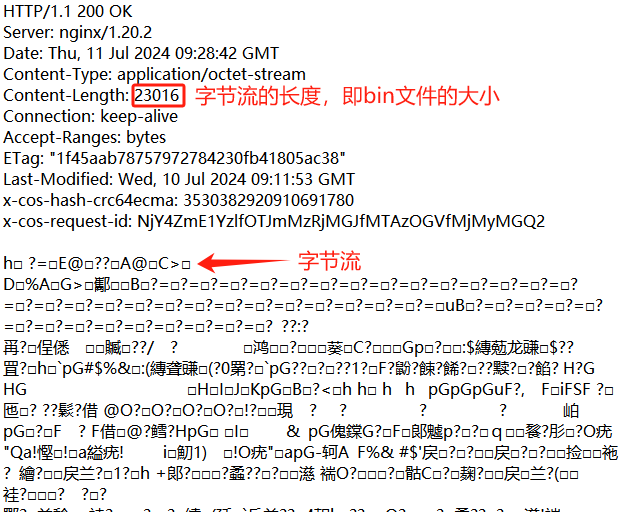
GET /b/1BcNzVkYWQxMmU1ZjJlNDZhODlmYWNkMjEyMjViZmNlMTk=otaServer010.bin HTTP/1.1

Host: bin.bemfa.com

Connection: keep-alive

## HTTP响应

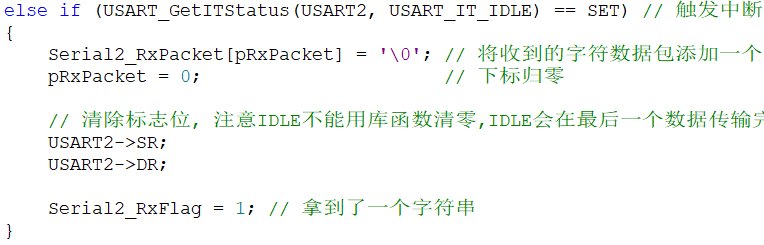
>[Rx<-][14:24:41][asc]



## 思考：下载bin文件还能用发送AT指令集的函数吗？

不能。

①因为一般的串口判定一个数据帧的完结，使用的是IDLE空闲中断，而一次大量数据的传输过程中难免产生空闲中断，使得程序误认为传输结束了。串口时序图（RX线），低位先行，RX上拉输入。



②缓冲区太小，能不能增加缓冲区？内存四区：常量/全局区，代码区，栈区（空间较小），堆区（空间较大）。STM32F103C8T6的内存（RAM）只有20KB，智慧大棚的bin文件23KB，不行。



## 切换串口接收模式，一口气完成接收数据和写入W25Q64

类似RS485的方案，以100ms的空闲周期为传输结束的标志。

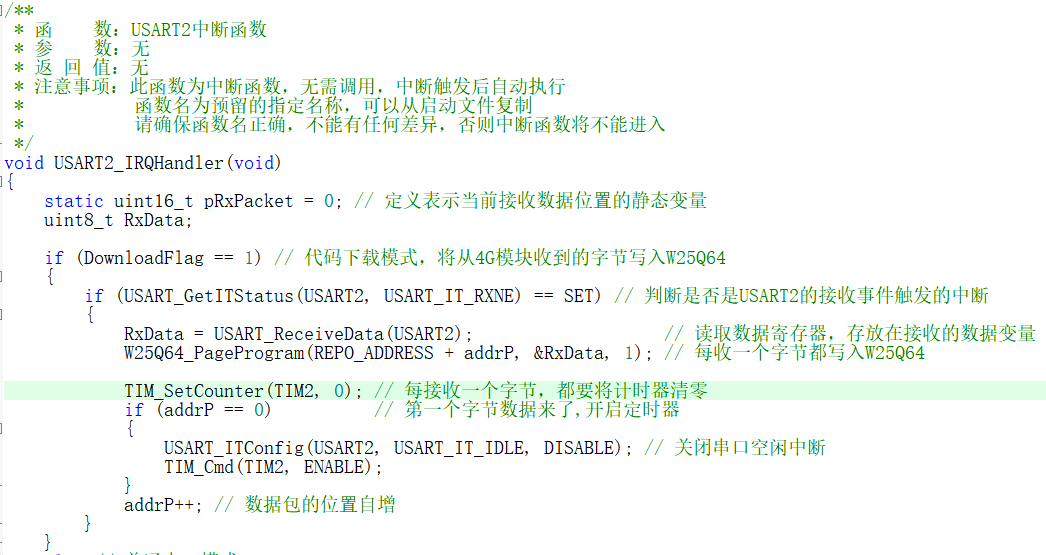
先使用定时器实现一个100ms进入一次的更新中断

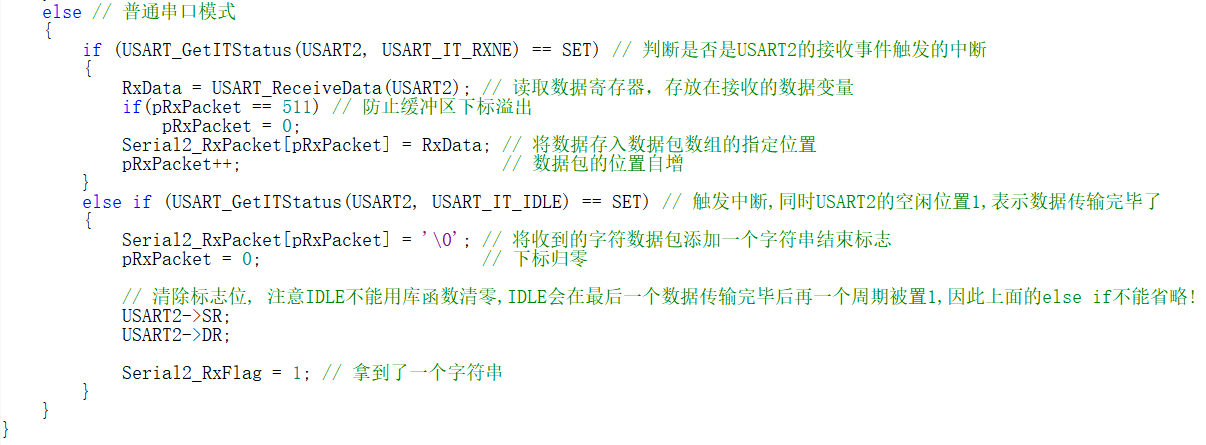


定义切换模式的标志和数据下标

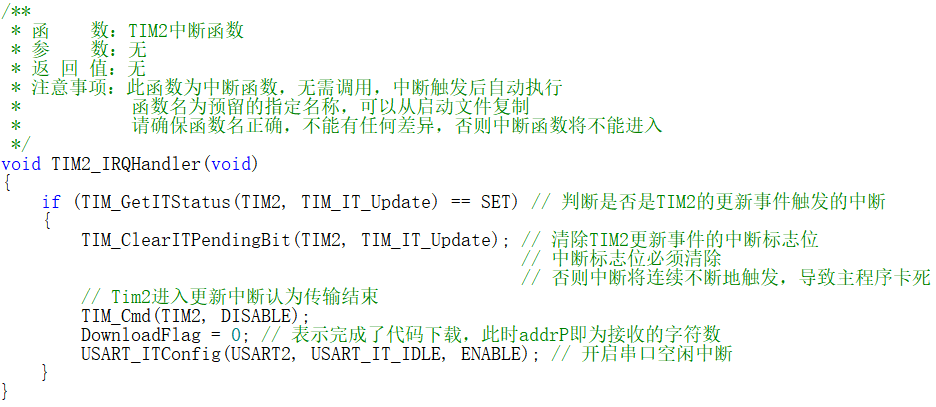


串口的下载模式

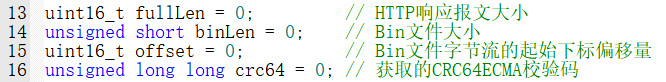




HTTP完整数据帧结束的判断



OTA.c文件中的全局变量

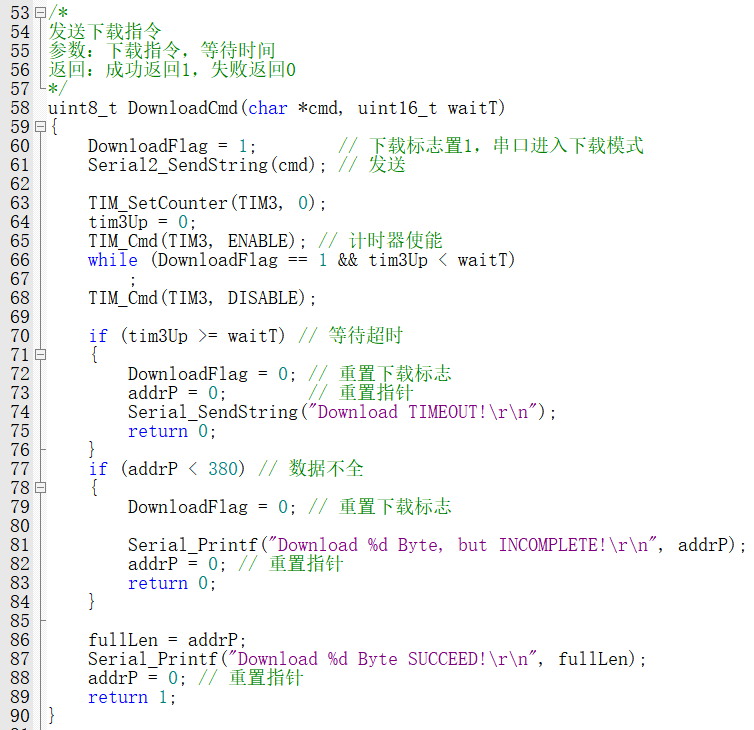


封装的代码下载函数





发送下载指令的函数



## 思考：串口每接收一个字节就写入W25Q64，跟得上吗？

串口波特率115200，RXNE接收中断间隔86.8微秒。

W25Q64的SPI时序图（指定地址写）

1. SPI起始，交换发送写使能的指令0x06，SPI终止；
2. SPI起始，交换发送页编程的指令0x02，交换发送地址24位高中低，写入数据，SPI终止；
3. 等待忙：SPI起始，交换发送读状态寄存器1的指令0x05，循环读取等待忙标志位即最低位直到其变成0，SPI终止；

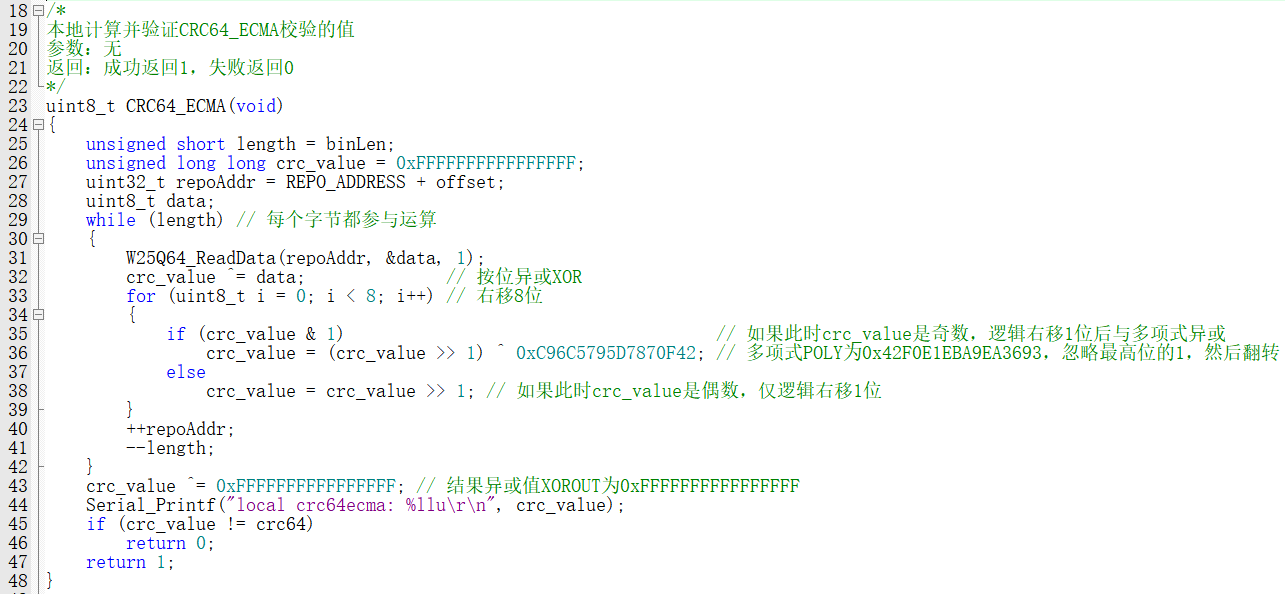
波特率72M / 32分频，忽略起始、终止条件、循环多次等待忙的耗时，至少交换8\*8个bit，用时28.4微秒。



在连接4G模块的串口的接收中断中：



## 本地计算并验证CRC64\_ECMA校验的值



## 将W25Q64的代码搬运到用户程序区

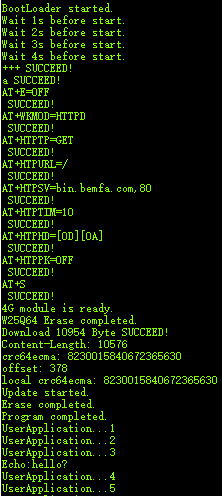
按半字为单位写入内部FLASH



## 完整测试

1. 按照要求接好线。
2. 对单片机进行全片擦除，将BootLoader直接通过ST-LINK或者串口FlyMcu下载到单片机中。
3. 电脑上打开对应串口，波特率115200，8-0-1协议，按下单片机复位键后10秒内按下PB6按键，启动OTA
4. 默认链接下载的程序是UserApplication\_v1：按一下PB6按键切换PC13的LED灯的开关，串口持续打印字符串，串口发送任意数据的回显。

BootLoader在Debug串口输出的内容



## 拓展

1. 版本控制：
   1. 用户主导：服务器提供多种版本，用户可选下载特定的版本，常见于面向个人用户的产品，如无线鼠标固件，蓝牙耳机，电脑主板BIOS。
   2. 服务器主导：UserApplication定时查询服务器有没有新版本，有就在UserData设置更新标志位，然后软件复位到BootLoader下载新版本，常见于面向公司用户的大规模部署产品，人工更新成本太高。
2. 使用WIFI模块获取Bin文件，同样也是AT指令集

AT\r\n

AT+CWMODE=1\r\n

AT+CIPSTATUS\r\n

AT+CWJAP="jikecheng","41124112"\r\n

AT+CIPSTART="TCP","bin.bemfa.com",80\r\n

AT+CIPMODE=1\r\n

AT+CIPSEND\r\n

GET /b/1BcNzVkYWQxMmU1ZjJlNDZhODlmYWNkMjEyMjViZmNlMTk=otaServer010.bin HTTP/1.1

Host: bin.bemfa.com

Connection: keep-alive

+++

\r\n

AT+CIPCLOSE