IAP升级文档

目录

[IAP简介 2](#_Toc10034)

[IAP定义 2](#_Toc15610)

[如何实现IAP 2](#_Toc3460)

[普通程序执行流程 2](#_Toc23537)

[IAP程序执行流程 3](#_Toc3963)

[IAP升级过程 5](#_Toc15264)

[STM32F103R8T6 Flash结构 5](#_Toc14679)

[BootLoader代码说明 6](#_Toc30317)

[功能固件代码说明 6](#_Toc9837)

[串口升级相关介绍 8](#_Toc726)

[串口设置 8](#_Toc467)

[通信模式 8](#_Toc31713)

[通讯协议 8](#_Toc19864)

[CRC校验算法 8](#_Toc27132)

[上位机使用说明 10](#_Toc11430)

[STM32F103R8T6-串口-IAP 13](#_Toc30676)

[测试环境 13](#_Toc3258)

[接线方式 13](#_Toc6892)

[STM32F407-串口-IAP 14](#_Toc14615)

[测试环境 14](#_Toc13829)

[接线方式 14](#_Toc25569)

**IAP简介**

**IAP定义**

IAP（In Application Programming）即在应用编程， IAP 是用户自己的程序在运行过程中对User Flash 的部分区域进行烧写，目的是为了在产品发布后可以方便地通过预留的通信口对产品汇总的固件程序进行更新升级。

如何实现IAP

通常实现IAP功能，是用户程序可以在运行中跟新自身Flash代码。需要在设计固件程序时编写两个项目代码， 第一个项目程序不执行正常的功能操作，而只是通过某种通信方式（如 USB，USART）接收程序或数据，执行对第二部分代码的更新；第二个项目代码才是真正的功能代码。 这两部分项目代码都同时烧录在 User Flash 中，当芯片上电后，首先是第一个项目代码开始运行，它作如下操作：

1）检查是否需要对第二部分代码进行更新

2）如果不需要更新则转到4）

3）执行更新操作

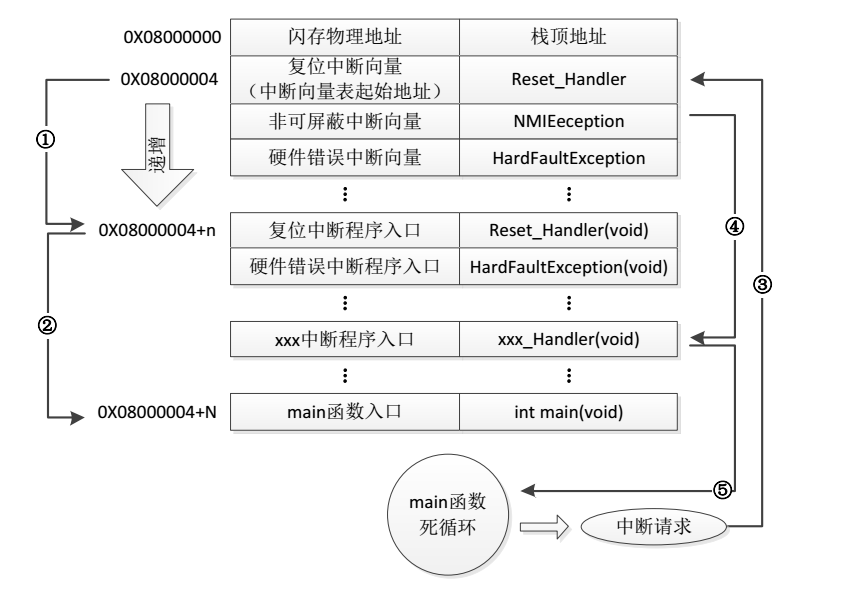
4）跳转到第二部分代码执行

第一部分代码必须通过其他手段，如JTAG或ISP烧入；第二部分代码可以使用第一部分代码IAP功能烧入，也可以和第一部分代码一起烧入，以后需要程序更新时再通过第一部分IAP代码更新。

我们将第一个项目代码称之为Bootloader程序，第二个项目代码称之为APP程序，他们存放在 FLASH 的不同地址范围，一般从最低地址区开始存放 Bootloader，紧跟其后的就是APP程序（注意，如果FLASH容量足够，是可以设计很多APP程序的，下文中我们只讨论一个APP程序的情况）。这样我们就需要实现2个程序Bootloader 和 APP。

普通程序执行流程

我们先来看看 STM32 正常的程序运行流程，如下图：

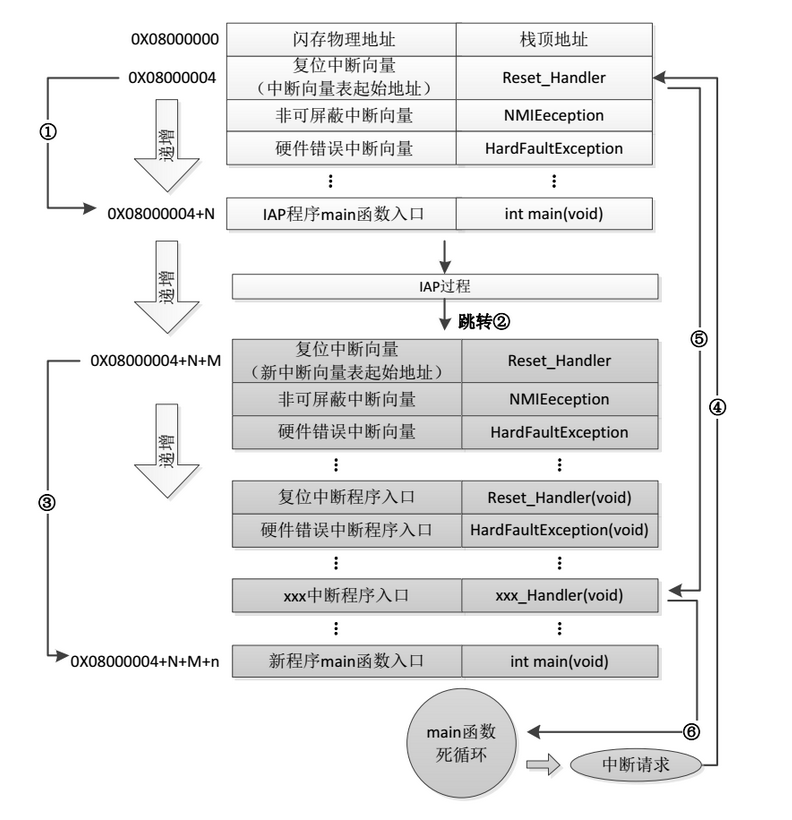


STM32 的内部闪存（FLASH）地址起始于0x08000000，一般情况下，程序文件就从此地址开始写入。 此外 STM32 是基于 Cortex-M3 内核的微控制器，其内部通过一张“中断向量表”来响应中断，程序启动后，将首先从“中断向量表”取出复位中断向量执行复位中断程序完成启动，而这张 “中断向量表”的起始地址是0x08000004，当中断来临，STM32 的内部硬件机制亦会自动将PC指针定位到“中断向量表”处，并根据中断源取出对应的中断向量执行中断服务程序。

在上图中，STM32 在复位后，先从0x08000004地址取出复位中断向量的地址，并跳转到复位中断服务程序，如图标号①所示；在复位中断服务程序执行完之后，会跳转到我们的main函数，如图标号②所示；而我们的main函数一般都是一个死循环，在main函数执行过程中，如果收到中断请求（发生重中断），此时STM32强制将PC指针指回中断向量表处，如图标号③所示；然后，根据中断源进入响应的中断服务程序，如图标号④所示；在执行完中断服务程序以后，程序再次返回main函数执行，如图标号⑤所示。

IAP程序执行流程

当加入 IAP 程序之后，程序运行流程如下图所示：



在上图所示流程中，STM32复位后，还是从0x08000004地址取出复位中断向量的地址，并跳转到复位中断服务程序，在运行复位中断服务程序之后跳转到IAP的main函数，如图标号①所示；在执行完IAP以后（即将新的APP代码写入STM32的FLASH，灰底部分。新程序的复位中断向量起始地址为0x08000004+N+M），跳转至新写入程序的复位向量表，取出新程序的复位中断向量的地址，并跳转执行新程序的复位中断服务程序，随后跳转至新程序的main函数，如图标号②和③所示，同样main函数为一个死循环，并且注意到此时STM32的FLASH，在不同位置上，共有两个中断向量表。

在main函数执行过程中，如果CPU得到一个中断请求，PC指针仍强制跳转到地址0x08000004中断向量表处，而不是新程序的中断向量表，如图标号④所示；程序再根据我们设置的中断向量表偏移量，跳转到对应中断源新的中断服务程序中，如图标号⑤所示；在执行完中断服务程序后，程序返回main函数继续运行，如图标号⑥所示。

通过以上两个过程的分析，我们知道IAP程序必须满足两个要求：

1）新程序必须在IAP程序之后的某个偏移量为x 的地址开始；

2）必须将新程序的中断向量表相应的移动，移动的偏移量为x；

**IAP升级过程**

CPU上电或者复位以后，首先执行bootload程序，bootload程序启动后，LED快速闪烁，每秒4次，检查串口是否有升级数据，等待6S无升级数据，则跳转到功能固件程序开始运行，如果收到升级指令，则进入升级模式，直到接收完成全部升级数据，则将新固件，升级到对应的Flash区域，启动运行新固件程序，升级过程结束，如果升级中断，超时时间固定为3S，直接运行之前的功能固件，升级失败！

**STM32F103R8T6 Flash结构**

STM32F103R8T6 Flash地址范围：0x0800 0000 -- 0x0801 0000 （总大小64KB）

BootLoader代码区地址范围 ：0x0800 0000 -- 0x0800 2000（占用大小8KB）

功能固件代码区地址范围 ：0x0800 2000 -- 0x0800 9000（占用大小28KB）

升级代码缓存区地址范围 ：0x0800 8000 -- 0x0801 0000（占用大小28KB）

STM32F103C8T6 一个扇区大小为1KB，数据擦除最小单位为1个扇区

可升级固件最大尺寸为28KB，超出此范围的固件升级将发生异常



**BootLoader代码说明**

BootLoader主要处理两件事情：

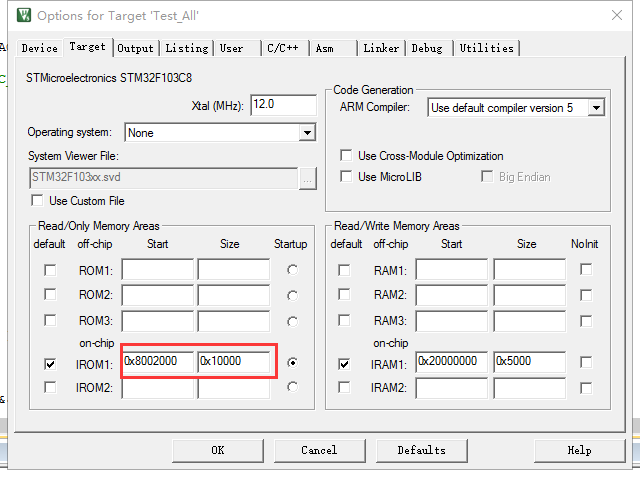
1.接收串口升级数据，并将数据写入到CPU Flash指定区域

2.跳转到功能固件去运行，至此Bootloader的功能就全部完成了，CPU全部交给功能固件去支配

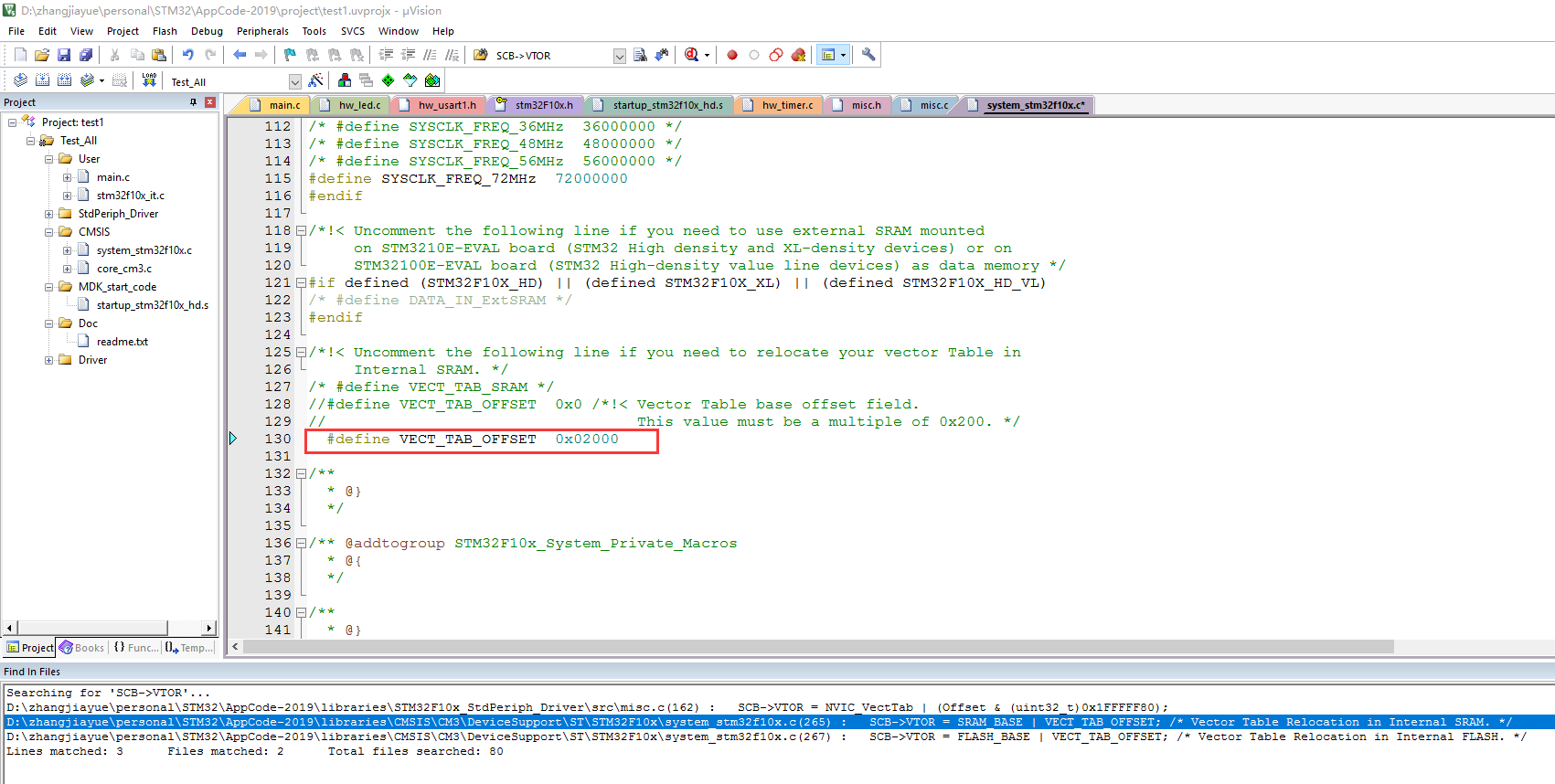
**App 代码说明**

功能固件的代码功能，完全由程序员去管理，这里需要注意两点：

1. 在keil开发环境中设置程序的Flash地址，因为编译器是按照绝对地址编译的，因为我们需要跳转到0x0800 2000运行，这里设置为0x0800 2000



1. 修改程序的中断向量偏移地址，如下图所示，默认偏移地址为0



**注意这两点以后，功能固件与普通STM32程序就没有其它不同了**

**串口升级相关介绍**

**串口设置**

升级通讯速率：115200 bps

串口其它通信参数： 8 N 1

**通信模式**

采用主从通讯模式，PC-- 主机 CPU -- 从机

**通讯协议**

PC-CPU（数据帧长度140）

帧头（固定3字节）：0xAA 0xAA 0xAA

功能码 （1字节）：0x00-0xFF（升级功能码，采用0xFF）

数据长度（2字节）：128+6=134

升级文件总数据帧（2字节）： N

升级文件当前帧 （2字节）： 1-N

升级数据1：

升级数据2：

...

升级数据128：

校验字节（固定2字节）：采用CRC16校验

CPU--PC（数据帧总长度：12）

帧头（固定3字节）：0xAA 0xAA 0xAA

功能码 （1字节）：0x00-0xFF（升级功能码，采用0xFF）

数据长度（1字节）：6

升级文件总数据帧（2字节）： N

升级文件当前帧 （2字节）： 1-N

校验字节（固定2字节）：采用CRC16校验

（注意多字节传输时候，均采用高字节存储在高地址，低字节存储低地址原则）

**CRC校验算法**

· unsigned **short** CRC16\_CCITT(unsigned **char** \*puchMsg, unsigned **int** usDataLen)

· {

·   unsigned **short** wCRCin = 0x0000;

·   unsigned **short** wCPoly = 0x1021;

·   unsigned **char** wChar = 0;

·

·   **while** (usDataLen--)

·   {

·         wChar = \*(puchMsg++);

·         InvertUint8(&wChar,&wChar);

·         wCRCin ^= (wChar << 8);

·         **for**(**int** i = 0;i < 8;i++)

·         {

·           **if**(wCRCin & 0x8000)

·             wCRCin = (wCRCin << 1) ^ wCPoly;

·           **else**

·             wCRCin = wCRCin << 1;

·         }

·   }

·   InvertUint16(&wCRCin,&wCRCin);

·   **return** (wCRCin) ;

· }

**CAN升级相关介绍**

**内容正在更新中...**

**USB升级相关介绍**

**内容正在更新中...**

**U盘升级相关介绍**

**内容正在更新中...**

**SD卡升级相关介绍**

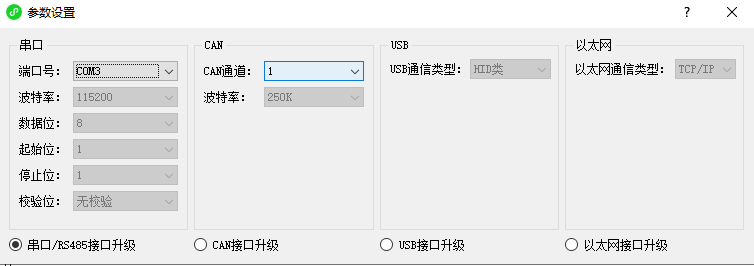
**内容正在更新中...**

**上位机使用说明**

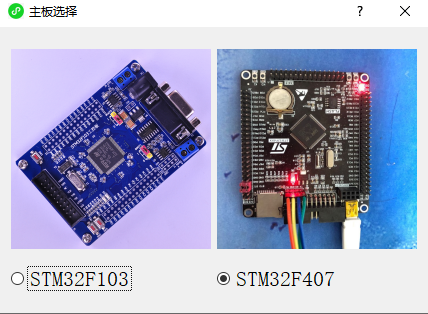
1. 根据选择的升级方式，将电路板与电脑连接
2. 打开上位机软件界面如下图所示：



1. 点击“参数设置”按钮，选择升级方式和对应的参数，如下图所示



1. 点击“主板选择”选择使用的主板



1. 点击“连接”，连接状态指示灯变成绿色，并且信息栏，提示连接成功，则表示电路板与电脑通信连接成功



1. 点击“选择固件”，选择需要升级的固件，可以支持hex和bin两种格式
2. 点击“升级固件”，上位机开始发送升级请求，此时复位或者重启电路板,即可启动升级，进度条显示升级进度



1. 进度条走完，且提示框提示升级成功，则升级过程完成！

**STM32F103R8T6-串口-IAP**

**测试环境**

硬件测试环境：STM32F103R8T6 最小系统板

CPU源码编译环境 ：keil5 MDK

上位机源码编译环境 ：QT5.12（MinWG7.4-32bit）

CPU端硬件串口: UART1（PA9,PA10）也可以定制修改为其它端口

LED端口：GPIO\_PC13

**接线方式**

USB转串口模块

STM32F103R8T6

电路板

使用串口1

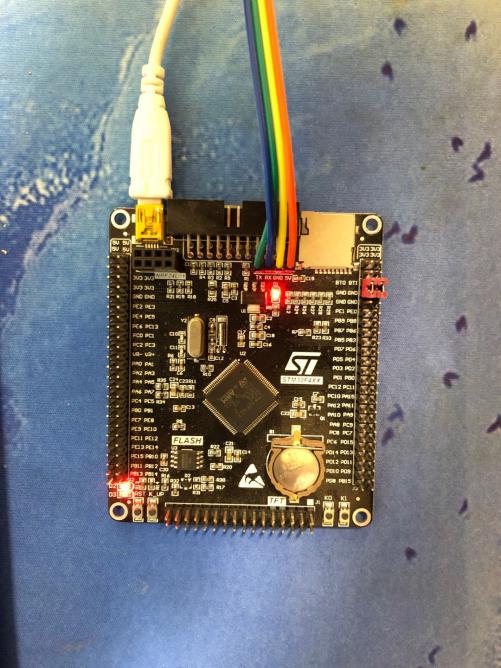
（PA9，PA10）

计算机

**STM32F407-串口-IAP**

**测试环境**

STM32F407开发板，实物图片如下：

****

CPU源码编译环境：Keil MDK4/5

上位机源码编译环境：QT5.12（5.0以上版本理论上都可以）

**接线方式**

计算机

USB转串口模块

STM32F103R8T6

电路板

使用串口1

（PA9，PA10）