STM32 的 IAP 方案

几乎所有的同类书籍都介绍综合性的应用示例如“万年历 + 温度显示 + 闹钟响铃 + 计时表”这样的一个实时时钟范例或“STM32 + 音频解码 + 大容量存储方案”这样的 MP3 播放器范例。这些综合性实例的目的在于引领读者进行综合性实验，达到把单片机的基础模块整合运用的目的。这些实例普遍存在一种共同点，即“练手”意义要大于“实用”的意义。本文将讲述一个 STM32 的综合性应用示例，该示例将涉及到 STM32 微控制器的时钟系统、GPIO、定时器、中断系统、异步串口以及内置可编程 flash 等设备的应用，作为一个综合性实验的同时还具有很强的“实用”意义。这个示例就是 STM32 的 IAP 方案。

IAP，全称是“In-Application Programming”，中文解释为“在程序中编程”。IAP 是一种对通过微控制器的对外接口（如 USART，IIC，CAN，USB，以太网接口甚至是无线射频通道）对正在运行程序的微控制器进行内部程序的更新的技术（注意这完全有别于 ICP 或者 ISP 技术）。ICP（In-Circuit Programming）技术即通过在线仿真器对单片机进行程序烧写，而 ISP 技术则是通过单片机内置的 bootloader 程序引导的烧写技术。无论是 ICP 技术还是 ISP 技术，都需要有机械性的操作如连接下载线，设置跳线帽等。若产品的电路板已经层层密封在外壳中，要对其进行程序更新无疑困难重重，若产品安装于狭窄空间等难以触及的地方，更是一场灾难。但若进引入了 IAP 技术，则完全可以避免上述尴尬情况，而且若使用远距离或无线的数据传输方案，甚至可以实现远程编程和无线编程。这绝对是 ICP 或 ISP 技术无法做到的。某种微控制器支持 IAP 技术的首要前提是其必须是基于可重复编程闪存的微控制器。STM32 微控制器带有可编程的内置闪存，同时 STM32 拥有在数量上和种类上都非常丰富的外设通信接口，因此在 STM32 上实现 IAP 技术是完全可行的。

实现 IAP 技术的核心是一段预先烧写在单片机内部的 IAP 程序。这段程序主要负责与外部的上位机软件进行握手同步，然后将通过外设通信接口将来自于上位机软件的程序数据接收后写入单片机内部指定的闪存区域，然后再跳转执行新写入的程序，最终就达到了程序更新的目的。

在 STM32 微控制器上实现 IAP 程序之前首先要回顾一下 STM32 的内部闪存组织架构和其启动过程。STM32 的内部闪存地址起始于 0x8000000，一般情况下，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 0x8000004，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。最后还需要知道关键的一点，通过修改 STM32 工程的链接脚本可以修改程序文件写入闪存的起始地址。

在 STM32 微控制器上实现 IAP 方案，除了常规的串口接收数据以及闪存数据写入等常规操作外，还需注意 STM32 的启动过程和中断响应方式。图 1 显示了 STM32 常规的运行流程。

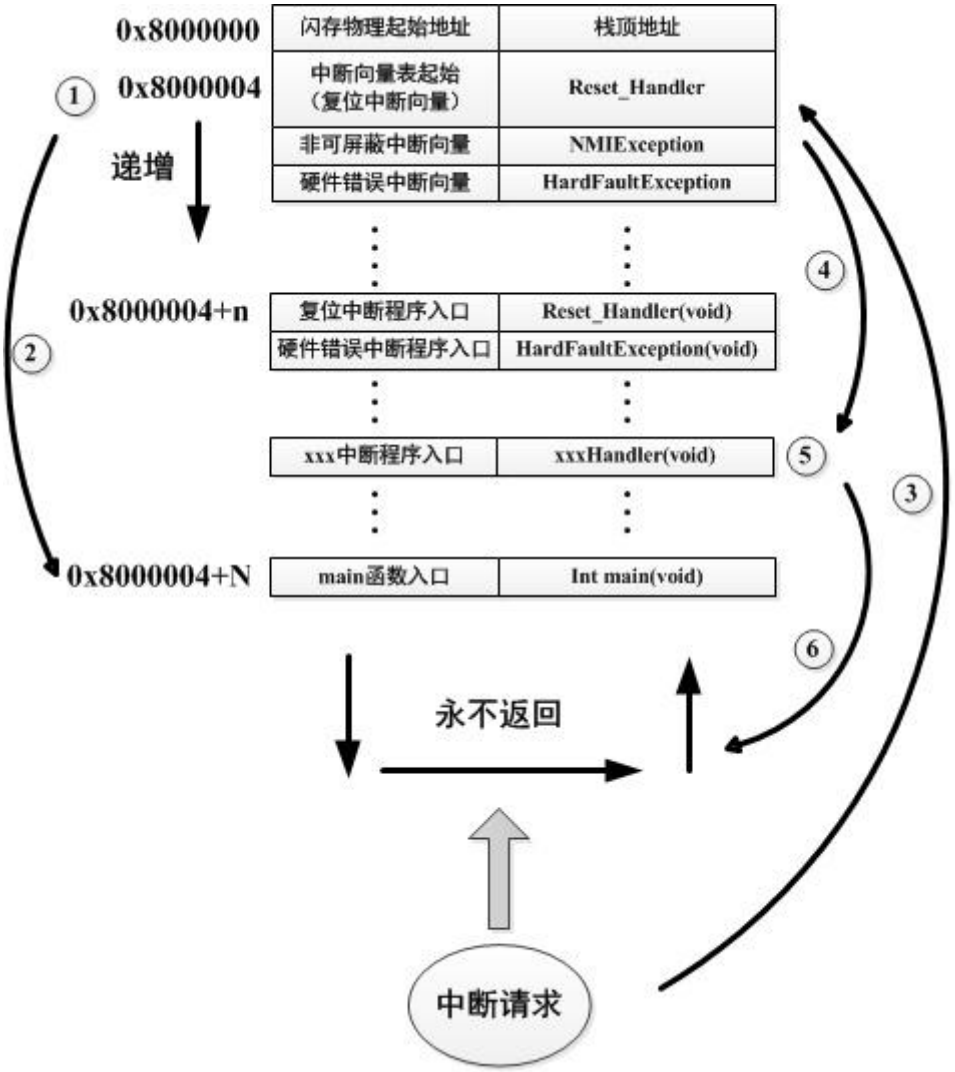

图 1

对图 1 解读如下：

1、 STM32 复位后，会从地址为 0x8000004 处取出复位中断向量的地址，并跳转执行复位中断服务程序，如图 1 中标号①所示。

2、 复位中断服务程序执行的最终结果是跳转至 C 程序的 main 函数，如图 1 中标号②所示，而 main 函数应该是一个死循环，是一个永不返回的函数。

3、 在 main 函数执行的过程中，发生了一个中断请求，此时 STM32 的硬件机制会将 PC 指针强制指回中断向量表处，如图 1 中标号③所示。

4、 根据中断源进入相应的中断服务程序，如图 1 中标号⑤所示。

5、 中断服务程序执行完毕后，程序再度返回至 main 函数中执行，如图 1 中标号⑥所示。

若在 STM32 中加入了 IAP 程序，则情况会如图 2 所示。

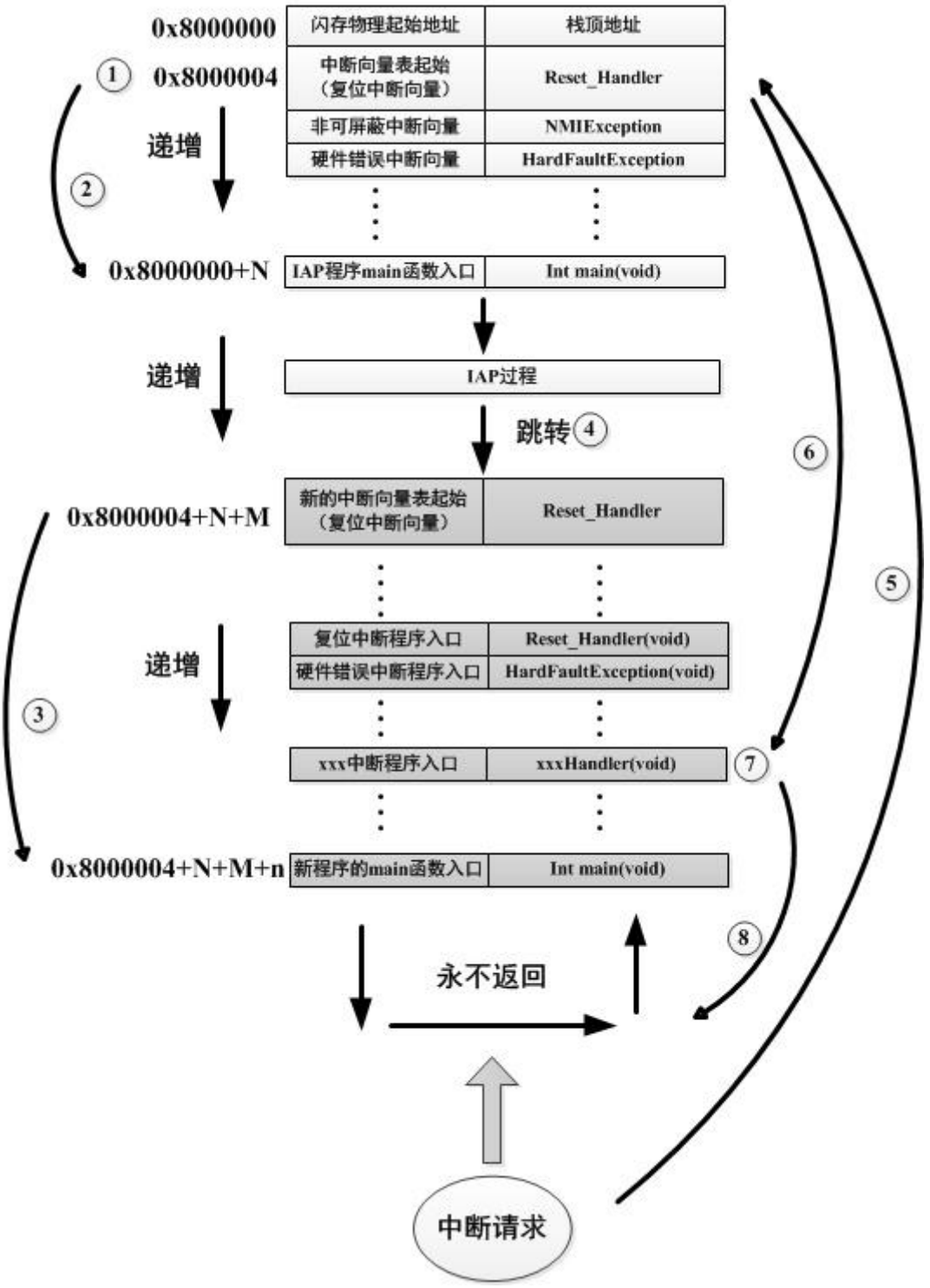

图 2

对图 2 的解读如下:

1、 STM32 复位后，从地址为 0x8000004 处取出复位中断向量的地址，并跳转执行复位中断服务程序，随后跳转至 IAP 程序的 main 函数，如图 2 中标号①、②所示。这个过程和图 1 相应部分是一致的。

2、 执行完 IAP 过程后（STM32 内部多出了新写入的程序，图 2 中以灰色底纹方格表示，地址始于 0x8000004+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，其过程如图 2 的标号③所示。新程序的 main 函数应该也具有永不返回的特性。同时应该注意在 STM32 的内部存储空间在不同的位置上出现了 2 个中断向量表。

3、 在新程序 main 函数执行的过程中，一个中断请求来临，PC 指针仍会回转至地址为 0x8000004 中断向量表处，而并不是新程序的中断向量表，如图 2 中标号⑤所示。**注意到这是由 STM32 的硬件机制决定的。**

4、 根据中断源跳转至对应的中断服务，如图 2 中标号⑥所示。**注意此时是跳转至了新程序的中断服务程序中。**

5、 中断服务执行完毕后，返回 main 函数。如图 2 中标号⑧所示。

从上述两个过程的分析可以得知，对将使用 IAP 过程写入的程序要满足 2 个要求：

- 新程序必须从 IAP 程序之后的某个偏移量为 x 的地址开始；
- 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；

而设置程序起始位置的方法是（keil uvision4 集成开发环境）在工程的“Option for Target….”界面中的“Target”页里将“IROM”的“Start”列改为欲使程序起始的地方，如图 3 中将程序起始位置设为 0x8002000。

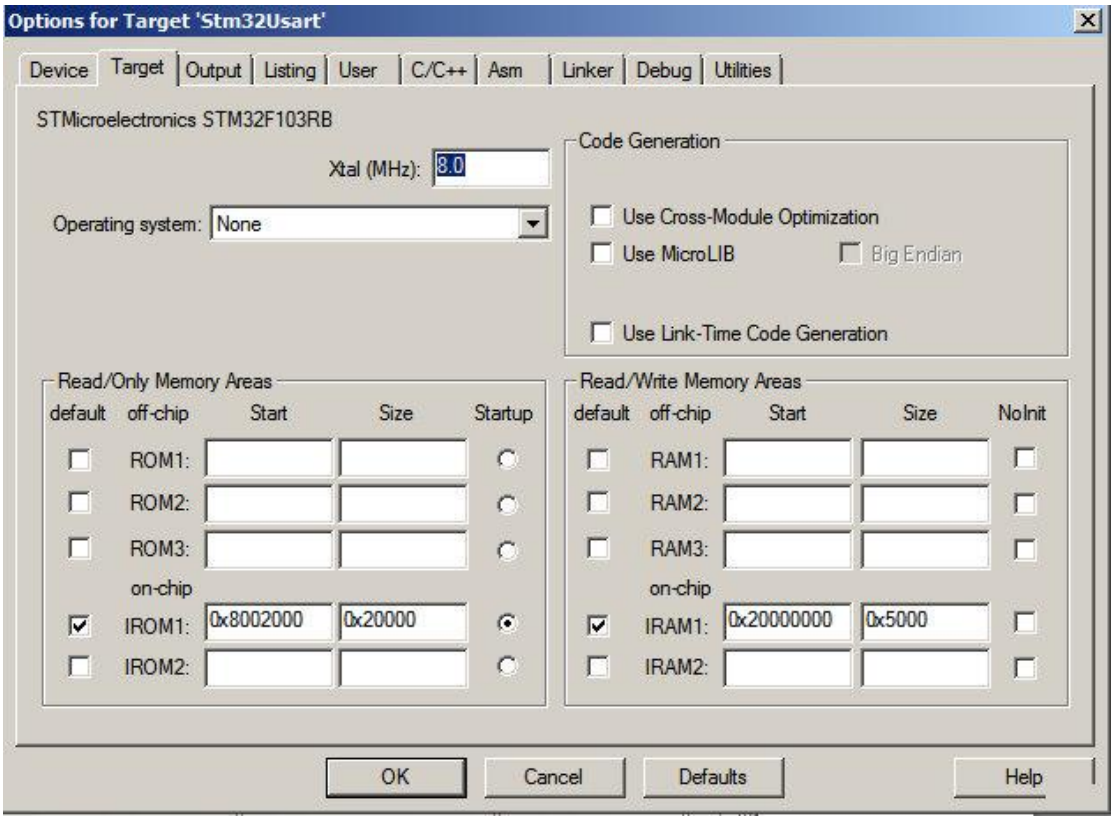

图 3

将中断向量表移动的方法是在程序中加入函数：

void NVIC_SetVectorTable(u32 NVIC_VectTab, u32 Offset)；

其中参数 NVIC_VectTab 为中断向量表起始位置，而参数 Offset 则为地址偏移量，如将中断向量表移至 0x8002000 处，则应调用该函数如下：

void NVIC_SetVectorTable(0x8000000, 0x2000)；

同时有必要提醒读者注意的是，此函数只会修改 STM32 程序中用于存储中断向量的结构体变量，而不会实质地改变中断向量表在闪存中的物理位置，详情请研究该程序原型。

有了以上准备后就可以着手设计一个 IAP 方案了，如下：

1、STM32 复位后，利用一个按键的状态进行同步，当按键按下时表示将要进行 IAP 过程；
2、IAP 过程中，通过上位机软件向 STM32 的 USART1 设备发送所要更新的程序文件，STM32 接收到数据后转而从 0x8002000 地址开始写入收到的数据；
3、STM32 借助定时器来判断数据是否完全接收，完全接收后 IAP 过程结束；
4、再次复位后，跳转 0x8002004 地址开始运行新写入的程序；

最后提出几点注意事项：

- 具体实现的工程见附书光盘；
- 利用 IAP 写入的程序文件最好是.bin 格式的文件，但不能是.hex 格式的文件；
- 向 STM32 发送程序文件时尽量慢一些，因为 STM32 对 FLASH 的写入速度往往跟不上通讯外设接口的速度；
- 建议在 STM32 和上位机之间设计一套握手机制和出错管理机制，这样可以大幅提高 IAP 的成功率；
- 附书光盘中的 IAP 工程具体运行现象为，按着连接于 GPIOA.0 引脚上的按键后对 STM32 进行复位操作，若连接于 GPIOA.4 引脚上的 LED 被点亮则表示进入了 IAP 程序，等待从 USART1 接口传入欲更新的程序文件。程序文件更新完毕后，LED 被熄灭。此时再度对 STM32 进行复位，就开始运行新写入的程序了。