# STM32-USB-DFU 升级 APP 程序中利用到同样中断时的处理办法

相信大家在实现了 STM32 的 USB_DFU 升级 APP 程序，因为 Bootloader 程序中用到了 USB 的相关中断，在 APP 程序中同样也用到了相同的中断，这时是不是就乱套了呢？的确是乱套了，不过还是有解决办法的。下面为大家演示一下：

**使用资源：**

**1、ST 官方库函数 V3.5**

**2、两个 LED 灯，一个红色 LED0，一个蓝色 LED1：**

**红色 LED 指示的是进入 DFU 模式的 USB 中断唤醒点亮，**

**蓝色 LED 指示的是进入 USB 读卡器模式的 USB 唤醒点亮**

**3、前面的 STM32_DFU_Bootloader 程序，STM32 读卡器程序**

第一、在正常的 DFU 移植的时候做法在 APP 程序中的 main 函数开头加入修改中断向量表地址的语句

SCB->VTOR = FLASH_BASE | 0X30000;

第二、假设 APP 程序中没有用到和 Bootloader 中的中断服务函数的时候，一切运行正常，当用上的时候呢？？？

第三、通过前面的方法，在 USB 读卡器实验程序中的 main 函数开头增加 SCB->VTOR = FLASH_BASE | 0X30000;语句，编译后 DFU 下载到板子上复位运行，这时可以进入板子可以进入 USB 读卡器模式。

但是，当你插上 USB 线连接电脑的时候，问题来了，电脑弹出提示框提示无法识别的 USB 设备。是不是意味着就不能这样用了呢？？


第四、这时我们就看看 STM32 程序执行的流程怎么走的（参考原子的《STM32 开发指南-库函数版本 V1.2 .pdf》中的 APP 程序执行图）

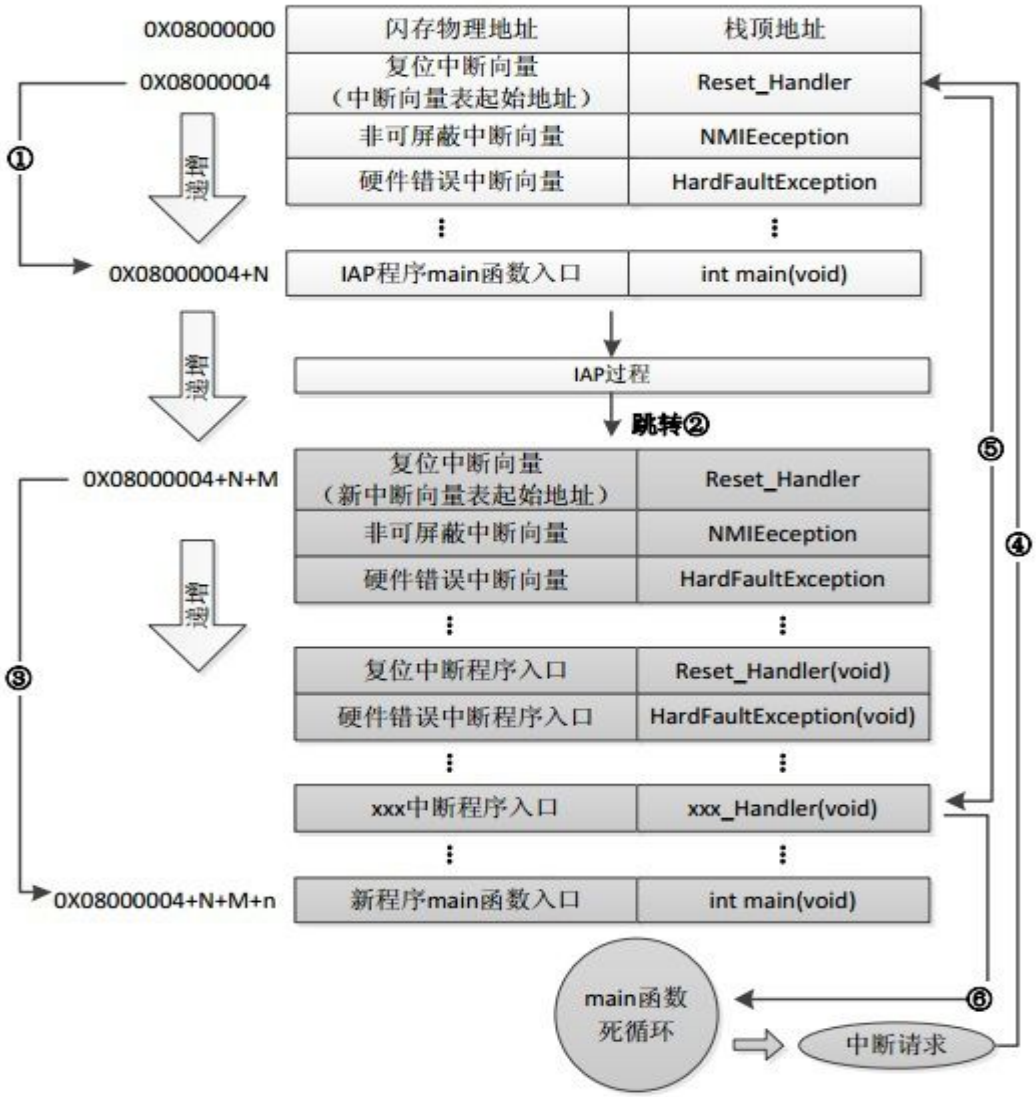



图 53.1.2 加入 IAP 之后程序运行流程图

具体解说这个流程图请参考原子《STM32 开发指南-库函数版本 V1.2 .pdf》第 700 页开头的解说。

图中知道这么一个消息，在执行 APP 程序中发生了中断请求，程序仍然跳转到了 0x0800 0004 这个中断向量表地址去提取相关的中断服务函数执行，问题就来了，在这个地址存放的是 Bootloader 的 USB 中断服务函数，读取了，USB 枚举的时候就来了，因为 DFU 的 USB 枚举参数和 USB 读卡器枚举的参数不一样，所以就发生了电脑无法识别的 USB 设备的错误提示框。是不是蒙了呢？？？【具体可以对照代码 usb_desc.c 中的字符串描述内容等】

第五、不用怕，看到图 53.1.2 中的第⑤步跳转，下面编辑代码验证一下。

A、USB 读卡器中也用到了 USB 的相关中断，有两个还是和 Bootloader 中的 USB 中断服务函数一样的调用。

B、在 Bootloader 程序中，USB 唤醒中断服务函数增加 LED0 红色 LED 指示（试过想在 LCD 上显示字符来的，加了 LCD 显示字符语句电脑识别设备失败，具体原因可能显示字符时间太长了）

```
063 /*************************************************************
064  * 函数功能 ---> USB唤醒中断处理
065  * 入口参数 ---> none
066  * 返回数值 ---> none
067  * 功能说明 ---> none
068  *************************************************************/
069 void USBWakeUp_IRQHandler(void)
070 {
071     EXTI_ClearITPendingBit(EXTI_Line18);     //清除USB中断标志
072     LED0 = 0;
073 }
```

C、编译成功后下载到板子上，按住 Bootloader 程序中设定的按键后

复位开发板，显示 DFU 模式界面时插上 USB 连接电脑。


图中**红色框框**中显示的是进入了 DFU 模式

图中**蓝色框框**的**红色 LED** 点亮，说明进入的是 Bootloader 中的 USB 唤醒中断服务函数。

第六、在 USB 读卡器程序中，USB 唤醒中断服务函数增加 LED1 蓝色 LED 指示

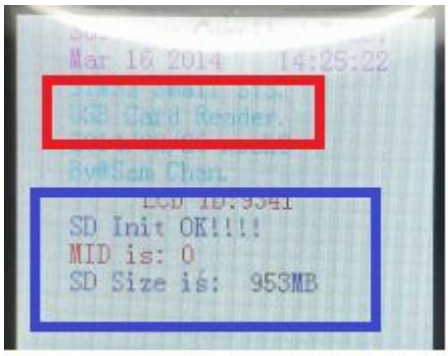
```
/*******************************************************
* 函数功能 ---> USB唤醒中断处理
* 入口参数 ---> none
* 返回数值 ---> none
* 功能说明 ---> none
*******************************************************/
void USBWakeUp_IRQHandler(void)
{
    EXTI_ClearITPendingBit(EXTI_Line18);     //清除USB中断标志
    LED1 = 0;
}
```

第七、这时我不在 main 函数开头处加写中断向量表地址重设代码，我直接修改“system_stm32f10x.c”文件中“void SystemInit (void)”

函数末尾的代码，修改如下

```
63
64  #ifdef VECT_TAB_SRAM
65    SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM. */
66  #else
67  //  SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH. */
68    SCB->VTOR = FLASH_BASE | 0X30000; /* Vector Table Relocation in Internal FLASH. */
69  #endif
70  }
```

第八、编译成功后通过 DFU 下载到板子上，复位进入如下界面




图中红色框框显示了 USB 读卡器实验

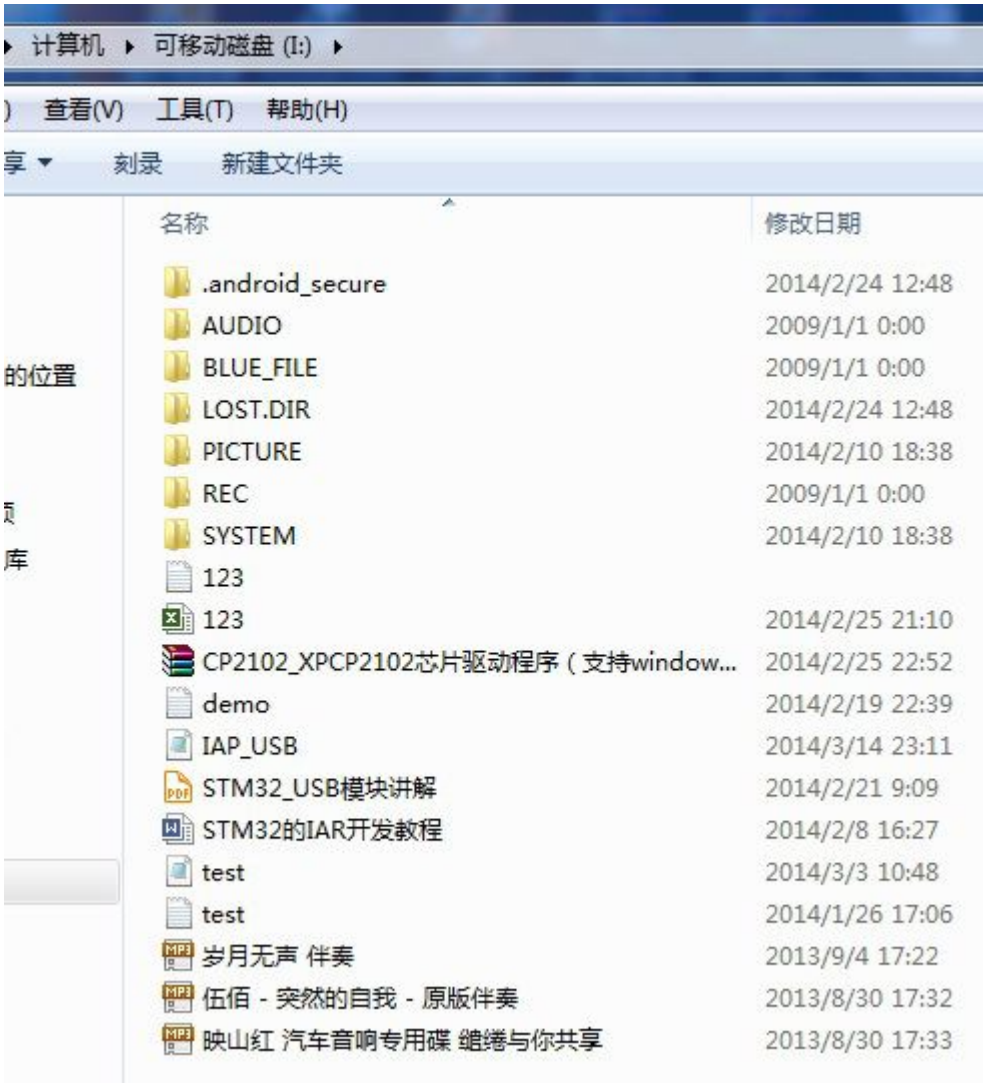
图中蓝色框框是读取到了 TF 卡（我用的）的容量信息

第九、这时插上 USB 线连接电脑，板子上显示如下界面

图中红色框框显示 USB 读卡器模式，已经连接上电脑

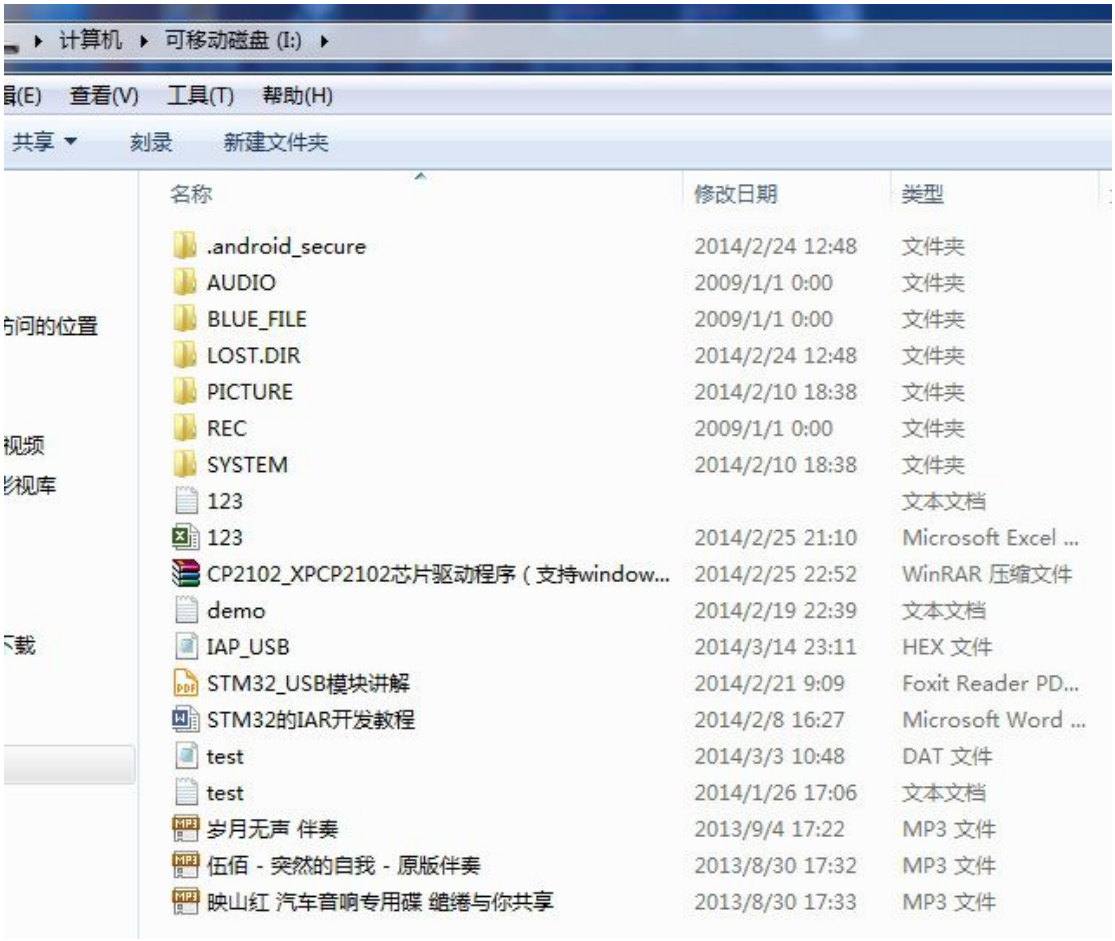
图中蓝色框框显示的蓝色 LED 点亮，说明已经进入 USB 读卡器的 USB 唤醒中断服务函数了。（前面我的 Bootloader 程序中的 USB 唤醒中断也点了灯的，红色不亮）

第十、电脑上已经正常识别到了 STM32 开发板上的 USB 读卡器了，电脑进去，双击打开可移动磁盘，里面内容




第十一、看到 TF 的内容了，说明已经成功了，可以进行读、写、删除等操作尝试。

第十二、不相信的话可以用读卡器将你开发板上的 TF 卡在电脑上看下里面的内容




最后，至于为什么这样改就可以了呢？打开我们程序中的启动文件代码看下，有这么一段

```
055
056 ; Vector Table Mapped to Address 0 at Reset
057                 AREA    RESET, DATA, READONLY
058                 EXPORT  __Vectors
059                 EXPORT  __Vectors_End
060                 EXPORT  __Vectors_Size
061
062 __Vectors       DCD     __initial_sp              ; Top of Stack
063                 DCD     Reset_Handler             ; Reset Handler
064                 DCD     NMI_Handler               ; NMI Handler
065                 DCD     HardFault_Handler         ; Hard Fault Handler
066                 DCD     MemManage_Handler         ; MPU Fault Handler
067                 DCD     BusFault_Handler          ; Bus Fault Handler
068                 DCD     UsageFault_Handler        ; Usage Fault Handler
069                 DCD     0                         ; Reserved
```

然后再看下面的代码

```
145
146 ; Reset handler
147 Reset_Handler   PROC
148                 EXPORT  Reset_Handler             [WEAK]
149                 IMPORT  __main
150                 IMPORT  SystemInit
151                 LDR     R0, =SystemInit
152                 BLX     R0
153                 LDR     R0, =__main
154                 BX      R0
155                 ENDP
156
```

完全汇编，是不是看不懂呢？？？结合前面的图 53.1.2 来看下就知道什么意思了，呵呵，自己琢磨一下吧。

至于寄存器操作的没试过，大家自己依葫芦画瓢吧。。。