

USB DFU IAP 例程移植的两个话题

前言

在 STM32 的系列产品中，很多型号都带有 USB 接口，为使用 USB 来进行代码升级提供了便利。这些型号中又有很大一部分可以通过内部 System Memory 中的 Bootloader 直接进行 USB DFU 升级，具体哪些型号支持 USB DFU，可参考应用笔记 AN2606《STM32 微控制器系统存储器自举模式》。有些型号虽然有 USB，但是 System Memory 中的 Bootloader 并没有支持 USB DFU，比如 STM32F102 / STM32F103、或者 Bootloader V2.x 的 STM32F2xxx、STM32F303，等等，或者用户希望通过不同的触发方式进入 bootloader 来进行 USB 下载，比如接收一串编制好的数据来触发。那么，就要使用 USB DFU IAP 了。关于如何使用 USB DFU IAP 的简要说明，可参考另一份文档《利用 USB DFU 实现 IAP 功能》。在这里，主要要谈的是在 USB DFU IAP 例程进行移植时，需要注意的两个地方。

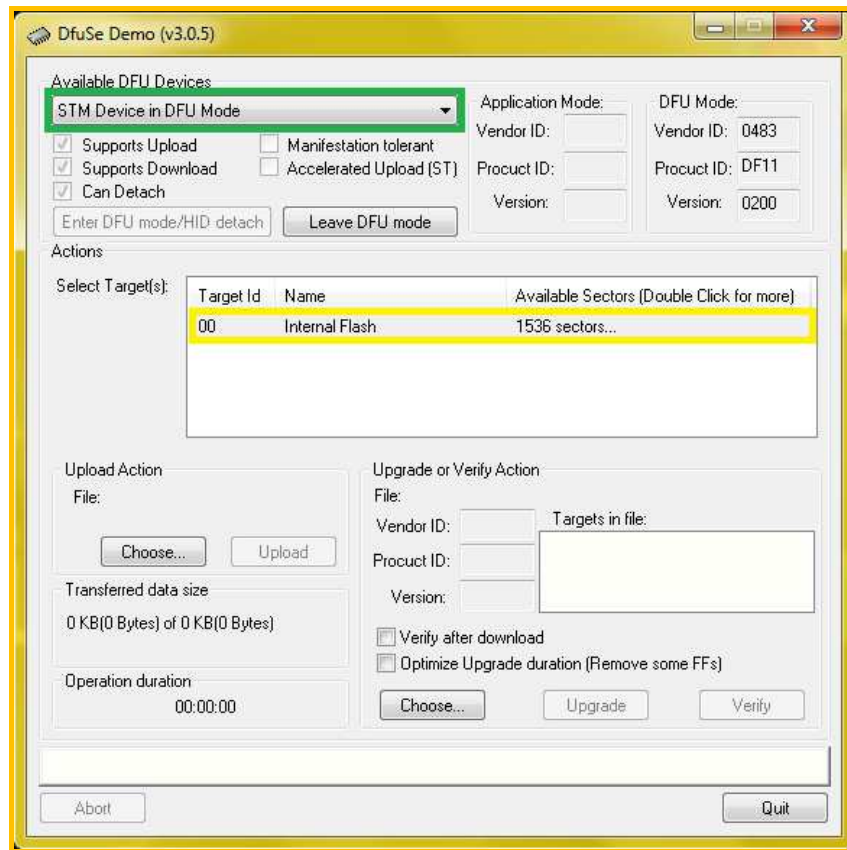
问题一

某客户在其产品的设计中，使用了 STM32L073RBT6。客户在开发过程中，使用 STM32L0Cube 库中的 STM32L073Z_EVAL 的 DFU_Standalone 进行代码移植，完成后在使用 Dfuse Demo 软件烧写用户代码时发生了错误。

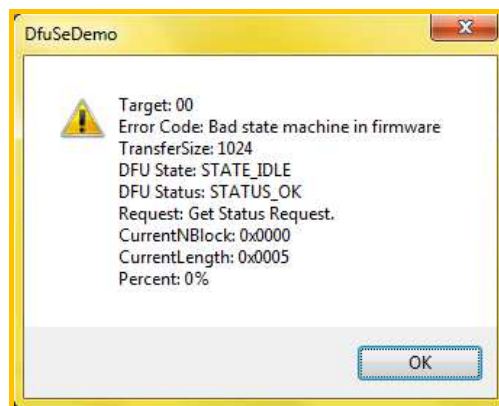
调研

1. 了解问题

客户在开发中使用了 STM32L0Cube 库 STM32Cube_FW_L0_V1.7.0，对里边的 \Projects\STM32L073Z_EVAL\Applications\USB_Device\DFU_Standalone 例程进行修改，以应用于用户板。客户已经根据硬件上的区别，对 LED 灯和按键的 I/O 口配置做了相应的修改，并在 main.h 中使能了 USE_USB_CLKSOURCE_CRSHSI48，因为其使用 STM32L073 内部的 48MHz 振荡作为 USB 时钟源。客户编译通过后，使用 ST-Link 将其下载到 STM32L073RBT6 中。然后断开 ST-Link，使用 USB 进行连接，PC 可以认到“STM Device in DFU Mode”。打开 Dfuse Demo 软件，也可发现已经识别到 STM32L073 处于 DFU Mode。



但是，当用户选择了“Verify after download”，并点击“Choose”按钮选择用户代码.dfu文件后，并点击“Upgrade”进行烧写，发现弹出了提示发生错误的对话框，如下：



2. 问题分析

STM32L073Z_EVAL 开发板使用的芯片型号为 STM32L073VZT6，其 Flash 容量为 192KB，地址从 0x08000000 到 0x0802FFFF。而客户所使用的 STM32L073RBT6，其 Flash 容量为 128KB，地址从 0x08000000 到 0x0801FFFF。检查项目中的 usbd_conf.h 文件中的代码，客户并未作任何修改，也就是说，以下两个定义没有根据实际的型号进行修改：

```
#define USBDFU_APP_DEFAULT_ADD 0x08003C00 /* Start user code address:
ADDR_FLASH_PAGE_120 */
```

```
#define USBDFU_APP_END_ADD          0x0802FF80 /* Start address of latest
flash_page: ADDR_FLASH_PAGE_1535 */
```

USBDFU_APP_DEFAULT_ADD 和 USBDFU_APP_END_ADD 定义了用户代码空间的开始页和结束页。从这可以看出，用户代码是从 0x08003C00 开始的，也就是第 120 页，而结束于第 1535 页。STM32L073VZT6 从第 0 页到第 1535 页共 1536 页，每页 128 Bytes。客户使用的是 STM32L073RBT6，总共才 1024 页。显然，这里对 USBDFU_APP_END_ADD 的定义并不对，需要修改为第 1023 页的地址。

3. 问题解决

将 usbd_conf.h 中的 USBDFU_APP_END_ADD 修改为第 1023 页的地址：

```
#define USBDFU_APP_END_ADD          0x0801FF80 /* Start address of latest
flash_page: ADDR_FLASH_PAGE_1023 */
```

问题解决，USB DFU 可以下载代码了。可是别急，这样就已经修改好了吗？再来看第二个话题。

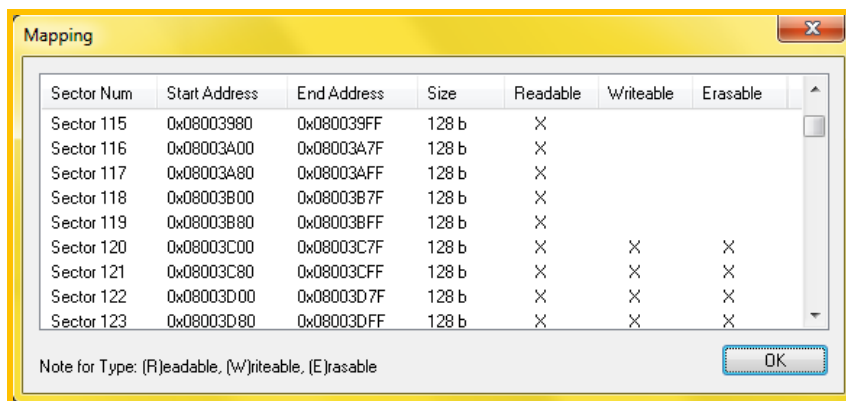
问题二

在问题的解决过程中，有没有注意到 Dfuse Demo 界面中显示 “1536 sectors”？这明显不对，来看看怎么修改。

调研

1. 了解问题

在 Dfuse Demo 界面中，双击 “1536 sectors”，可以看到 Internal Flash 的详细信息，如下：



Sector Num	Start Address	End Address	Size	Readable	Writeable	Erasable
Sector 115	0x08003980	0x080039FF	128 b	X		
Sector 116	0x08003A00	0x08003A7F	128 b	X		
Sector 117	0x08003A80	0x08003AFF	128 b	X		
Sector 118	0x08003B00	0x08003B7F	128 b	X		
Sector 119	0x08003B80	0x08003BFF	128 b	X		
Sector 120	0x08003C00	0x08003C7F	128 b	X	X	X
Sector 121	0x08003C80	0x08003CFF	128 b	X	X	X
Sector 122	0x08003D00	0x08003D7F	128 b	X	X	X
Sector 123	0x08003D80	0x08003DFF	128 b	X	X	X

Note for T type: (R)eadable, (W)riteable, (E)rasable

2. 问题分析

从上图可以了解到，实际上这里所定义的 Sector 的大小为 128Bytes，也就是 STM32L073 的 Page，所以这里的 Sector 定义与 STM32L073 的参考手册定义的 Sector 是不一样的，不要造成误解。在 RM0367 中，每 128Bytes 为 1 个 Page，每 32 个 Page 才是 1 个 Sector。所以不要误会就行了。在这个 Mapping 窗口中，也可以看到地址 0x08003C00 之前的空间为 Read-only，也就是 Bootloader 所处的空间为只读，以避免对这部分代码的重写。而后面的空间，也就是用户代码所处的空间为 Read/Write/Erase。

这些信息是从哪里来的呢？其实它来自于 usbd_dfu_flash.c 里边定义的描述符 FLASH_DESC_STR，如下：

```
#define FLASH_DESC_STR              "@Internal Flash      /0x08000000/120*128 a,1416*128 g"
```

来解释一下这个描述符的内容：

0x08000000 为起始地址。“a”代表的是 Read-only, “g”代表 Read/Write/Erase。也就是说, “a”所指明的区域为 Bootloader 的空间, “g”所指明的区别为用户代码空间。大小由前面的数字决定, 乘号 “*” 前面的为 Sector 的个数, 后面的为 Sector 的大小, 这里的意思就是从 0x08000000 开始, 前面 120 个 Sector (每个 Sector 为 128 字节) 为 Read-only, 后面 1416 个 Sector (每个 Sector 为 128 字节) 为 Read/Write/Erase。

举另外一个例子, 在\Projects\STM32L053C8-Discovery\Applications\USB_Device\DFU_Standalone\Src 下的 usbd_dfu_flash.c 是这样定义的:

```
#define FLASH_DESC_STR      "@Internal Flash      /0x08000000/28*01Ka,36*01Kg"
```

它的意思就是前面 28 个 Sector (每个 Sector 为 1KB) 为 Read-only, 后面 36 个 Sector (每个 Sector 为 1KB) 为 Read/Write/Erase。因为在这个例子中, 用户代码起始地址为 0x08007000。在 Dfuse Demo 的界面中, 你也将看到只有 64 个 Sector, 双击打开后能看到每个 Sector 为 1KB。

搞明白这个事, 就知道如何去修改这个描述符 FLASH_DESC_STR, 让它符合 STM32L073RBT6 的大小了。

3. 问题解决

STM32L073RBT6 有 1024 页, 每页 128 字节, 所以需要修改描述符 FLASH_DESC_STR 定义如下:

```
#define FLASH_DESC_STR      "@Internal Flash      /0x08000000/120*128Ba,904*128Bg"
```

附加话题

如果用户代码空间的定义还是这样的:

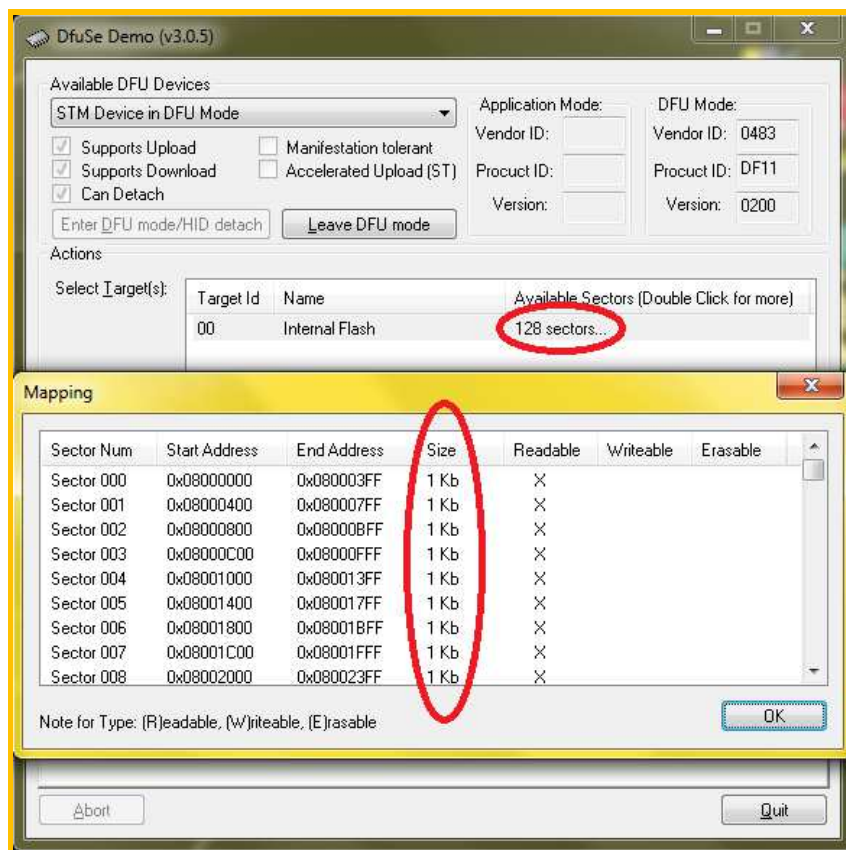
```
#define USBDFU_APP_DEFAULT_ADD      0x08003C00 /* Start user code address:
ADDR_FLASH_PAGE_120 */
#define USBDFU_APP_END_ADD          0x0801FF80 /* Start address of latest
flash page: ADDR_FLASH_PAGE_1023 */
```

但是描述符 FLASH_DESC_STR 的定义修改为:

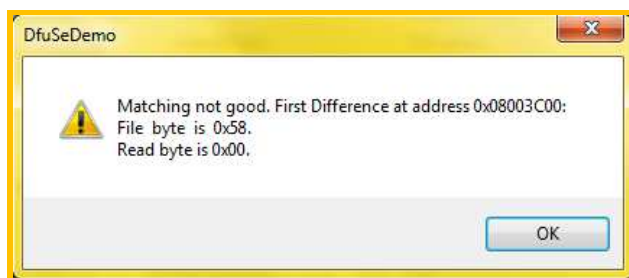
```
#define FLASH_DESC_STR      "@Internal Flash      /0x08000000/28*01Ka,100*01Kg"
```

那会发生什么情况呢?

将 Bootloader 程序编译后烧写到 STM32L073 中, 然后使用 USB 接口进行连接, 打开 Dfuse Demo。首先, 可以看到界面中显示的就是 128 Sectors, 双击打开, 每个 Sector 大小为 1KB。



接下来，来烧写一个用户代码，从 0x08003c00 地址开始的。在 Verify 时，就会弹出错误的对话框：



验证在 0x08003C00 的地址就已经发生了错误：烧录文件该地址的数据为 0x58，但是读回来的是 0x00。这就是因为我们把描述符 FLASH_DESC_STR 错误地定义成了前面 28KB 为 Read-only，也就是从 0x08007000 开始才是可读/可写/可擦除的。所以，在 0x08007000 之前的空间是不可擦除和写入的，也就导致了这样的情况。这个附加话题也只是为了强调这个描述符 FLASH_DESC_STR 的重要性。

结论

使用 USB DFU IAP 参考例程进行移植的时候，Bootloader 的空间以及用户代码的空间的定义全部都需要根据具体的 STM32 型号进行修改。

重要通知 – 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对ST 产品和/ 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST 产品的最新信息。ST 产品的销售依照订单确认时的相关ST 销售条款。

买方自行负责对ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST 产品如有不同于此处提供的信息的规定，将导致ST 针对该产品授予的任何保证失效。

ST 和ST 徽标是ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。