STM32 DFU 升级 APP 程序移植笔记

应用到的例程：版本 V4.0，官网下载地址：

http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1743/PF258157

1、下载的压缩包里面有 USB 的固件库和应用例程相关的代码。

2、具体工作原理什么的分析就不说了，可以参考相关的资料说明，再加上我也不是很了解 DFU 的工作原理

3、我移植所用的 STM32 固件库版本是 V3.5 的，USB 固件库用的是 V4.0 版本。

4、USB 固件库里面代码非必要情况下不需要修改即可应用。我们要做的工作就是将我们的底层操作函数涵接进去即可，修改其他相应的配置等等。

下面是移植步骤：

第一步：拷贝文件

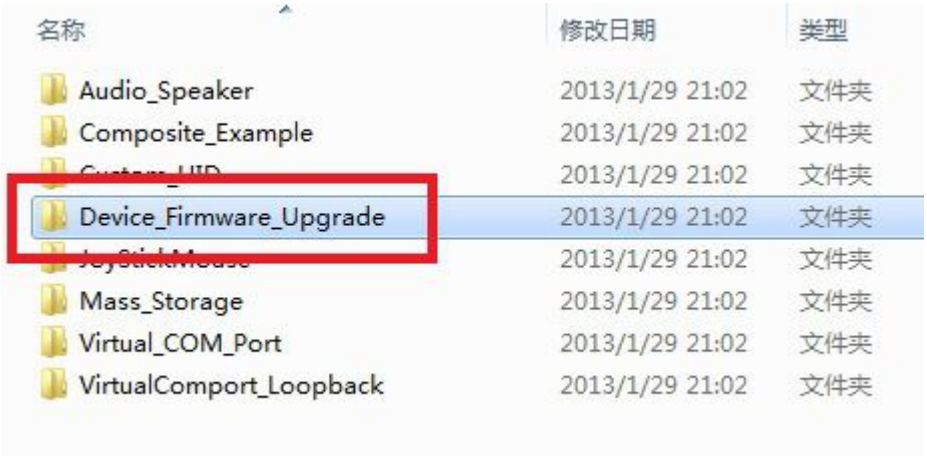
A、拷贝 USB 相关的 LIB 文件。

打开我们下载下来的压缩包，在“...\STM32_USB-FS-Device_Lib_V4.0.0\Libraries\STM32_USB-FS-Device_Driver”文件路径下的 src 和 inc 这两个文件夹拷贝到我们的工程中，如下图

红色框框的是我的工程路径文件夹；蓝色框框的文件夹是我建立的 USB 库文件存放的地方；剩下的“Mass_Storage”文件夹就是我存放的 USB 硬件相关的实现代码。这里具体大家怎么放就随便了哈。

B、拷贝相关的例程代码。

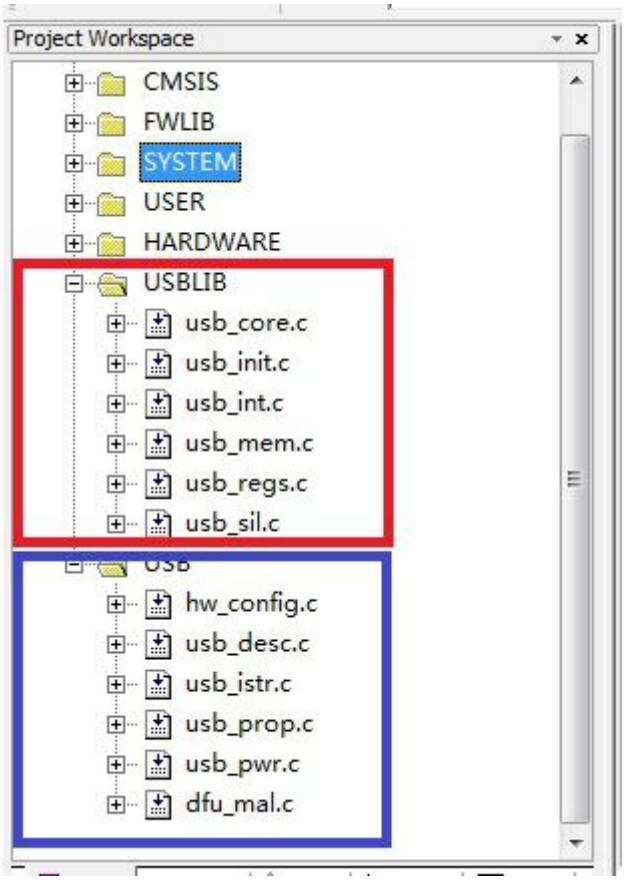
在“...\STM32_USB-FS-Device_Lib_V4.0.0\Projects”这个路径下我们会看到这么一个家伙



没错，就是《Device_Firmware_Upgrade》这个文件夹，里面存放这我们的 USB_DFU 实现的底层接口代码，进去会看到 src 和 inc 这两个文件夹

红色框框的这两个文件夹，将它拷贝到前面提到的“Mass_Storage”文件夹中。

C、打开（或者创建）工程。并在工程中添加刚才拷贝过去的 USB 相关文件和其他的一些底层函数文件等。具体要添加哪些则视情况和自己的喜好了哈，呵呵。。。最终效果如下图



红色框框的就是前面拷贝过来的 USB 库相关文件了，非必要的情况下不要修改

蓝色框框里面的就是底层实现代码文件了，就是这次我们开刀的代码文件了。

添加头文件路径什么的就自己搞定，不懂的站墙角面壁思过去，嘿嘿。

第二步、开始修改代码

一、hw_config.c 和 hw_config.h 文件代码修改。

A、添加头文件#include "led.h"，这个视你是否使用 LED 指示而添加，这里我用到了，所以添加进来。

B、void Set_System(void)这个函数代码实现，将不必要的全部删掉，保留相关有用的代码，如下图

```
void Set_System(void)
{
    FLASH_Unlock(); //解锁FLASH

    /* Init the media interface */
    MAL_Init();
    USB_Cable_Config(ENABLE);

    /* Configure the EXTI line 18 connected internally to the USB IP */
    EXTI_ClearITPendingBit(EXTI_Line18);
    EXTI_InitStructure.EXTI_Line = EXTI_Line18;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);
}
```

这里这个 FLASH_Unlock();这个函数很重要，因为我操作的是 STM32 内部的 FLASH，所以要解锁。具体可以参考战舰光盘中“8，STM32 参考资料”中的《STM32F10xxx 闪存编程参考手册.pdf》。

C、void Set_USBClock(void)这个函数就是设置 USB 时钟相关的，具体见工程代码。

D、找到 void USB_Cable_Config 这个函数，我将其修改成下面的样子，简洁大方，得体，呵呵

void USB_Cable_Config (FunctionalState NewState)

{

if(NewState != DISABLE) LED1 = 0; //USB 连接上

else LED1 = 1; //USB 断开

}

E、void DFU_Button_Config(void)这个函数呢，就是初始化你进入 DFU 升级模式的按键，用其他的方式进入也是可以的，比如短路某个管脚等等方式，这里我用到了 PA.0 管脚，也就是 WAKE_UP 管脚，具体见战舰板子原理图。修改成下面这个样子

```
void DFU_Button_Config(void)     //初始化升级按键
{
    STM32_GPIOx_Init(Button_KEYInit);   //初始化接口寄存器

    //初始化中断设置
    EXTI_InitStructure.EXTI_Line = EXTI_Line0;  //外部中断线0
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt; //中断模式
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;  //上升沿触发
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;

    EXTI_Init(&EXTI_InitStructure);

    //初始化中断优先级
    NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;     //外部中断线0
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x0F;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x0F;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;

    NVIC_Init(&NVIC_InitStructure);
}
```

F、uint8_t DFU_Button_Read (void)接下来这个函数就是获取按键的状态的，怎么实现，如下图

uint8_t DFU_Button_Read (void) //读取升级按键状态

{

return GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0);

}

G、void USB_Interrupts_Config(void)这个函数配置了 USB 的中断优先级等参数。

```
void USB_Interrupts_Config(void)    //中断优先级配置
{

  /* 2 bit for pre-emption priority, 2 bits for subpriority */
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);

  /* Enable the USB interrupt */
  NVIC_InitStructure.NVIC_IRQChannel = USB_LP_CAN1_RX0_IRQn;    //设置USB低中断优先级
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);

    /* Enable the USB Wake-up interrupt */
  NVIC_InitStructure.NVIC_IRQChannel = USBWakeUp_IRQn;  //USB唤醒中断
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

H、打开 hw_config.h 头文件，找到下面这个宏定义。

#define ApplicationAddress 0x08030000

没错，这个宏定义很重要，看说明，就是你的 APP 程序的起始地址了，关系到后面的成败的，呵呵，言重了。

二、usb_desc.c 和 usb_desc.h 文件修改

A、打开 usb_desc.c 文件，你会看到很多什么的 #ifdef USE_STM3210B_EVAL 等等这些选择编译的东东，因为我用的是战舰的开发板，主控是 STM32F103ZET6，所以只保留 USE_STM3210B_EVAL 选择编译的代码即可，具体实现见工程中文件。

B、打开 usb_desc.c 文件，同样的和上点一样做法。

C、同样的 uint8_t DFU_StringVendor[DFU_SIZ_STRING_VENDOR]这个字符串就是定义了造商的信息等，这里修改了还要修改 DFU_SIZ_STRING_VENDOR 宏定义长度的。

uint8_t DFU_StringProduct[DFU_SIZ_STRING_PRODUCT] =这个字符串就是定义设备名称的，这里修改了还要修改 DFU_SIZ_STRING_PRODUCT 宏定义长度的。

C 点上的我没变动，所以。。。

三、dfu_mal.c 和 dfu_mal.h 文件修改

A、注释掉#include "flash_if.h"这个头文件

B、删除无关的代码，具体自己看着办哈。

C、uint16_t MAL_Init(void)这个函数是初始化设备的，这里我操作的是内部的 flash，所以直接返回 OK 即可。

D、uint16_t MAL_Erase(uint32_t SectorAddress)这个函数看名字就知道干啥的了，最好还是选择擦除页，当然你页可以整个 flash 擦除操作的，具体见代码。

E、uint16_t MAL_Write()这个函数就是写入程序代码实现函数，具体修改见工程代码。原文是用指针实现的，这里还是用简单点的实现。呵呵

F、uint8_t *MAL_Read()这个函数是读取 flash 的。看函数名字，注意了返回的是一个字节宽度的数据，所以。。。

G、uint16_t MAL_GetStatus()这个函数得到 flash 状态，这里我直接返回 OK 的。

H、dfu_mal.h 文件内容可以不用动，当然，你看着不爽的话也是可以动的，呵呵

四、platform_config.h 头文件修改

1、#include "stm32_config.h"这个头文件是我定义的一个公共头文件的集合头文件。将它添加进去。

2、这里面呢，同样的将不必要的都删掉。

3、我在这里定义了这么一个宏，具体实现代码见工程中的 sys.c 中代码。

```
#define Button_KEYInit
    RCC_APB2Periph_GPIOA,GPIOA,GPIO_Pin_0,GPIO_Speed_50MHz,GPIO_Mode_IPD
```

五、stm32f10x_it.c 中断汇总文件

1、接着在这个中断汇总文件中添加头文件

```
#include "hw_config.h"   //USB 相关头文件
#include "usb_lib.h"
#include "usb_istr.h"
#include "usb_prop.h"
#include "usb_pwr.h"
```

注：我还用到了其他的指示方式，所以头文件多了点，呵呵

2、USB 低优先级中断处理函数

```
void USB_LP_CAN1_RX0_IRQHandler(void)
{
  USB_Istr();
}
```

3、USB 唤醒中断处理函数

```
void USBWakeUp_IRQHandler(void)
{
    EXTI_ClearITPendingBit(EXTI_Line18);   //清除 USB 中断标志
}
```

4、外部中断线 0 中断处理函数

```
void EXTI0_IRQHandler(void)
{
    if(EXTI_GetITStatus(EXTI_Line0) != RESET)
    {
        if(pInformation->Current_Feature & 0x20) //Remote wake-up
enabled
        {
            Resume(RESUME_INTERNAL);
        }
```

```
        /* Clear the EXTI line 9 pending bit */
        EXTI_ClearITPendingBit(EXTI_Line0);
    }
}
```

第三步、修改 main 文件

1、在 main 文件中添加 USB 相关头文件

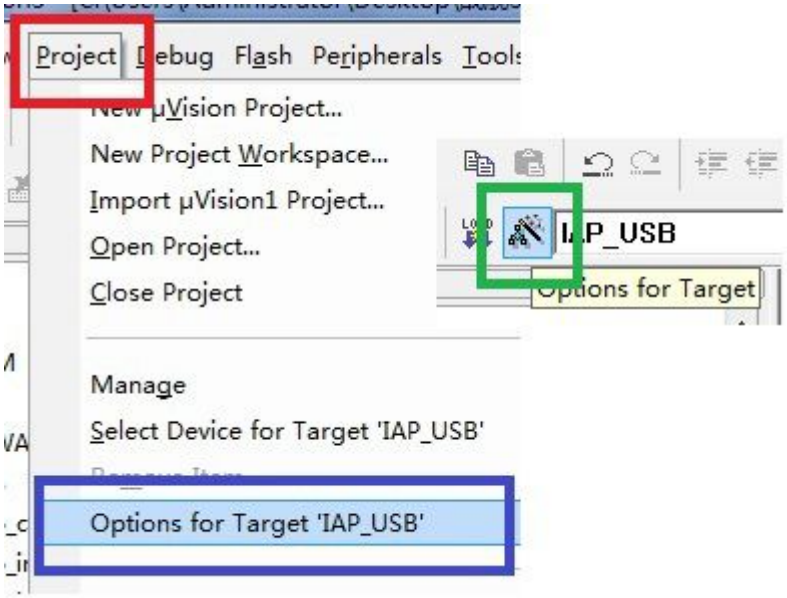
```
#include "hw_config.h"    //USB 相关头文件
#include "usb_lib.h"
#include "usb_conf.h"
#include "usb_prop.h"
#include "usb_pwr.h"
#include "dfu_mal.h"
```

2、增加相关定义，具体什么功能自己分析了哦

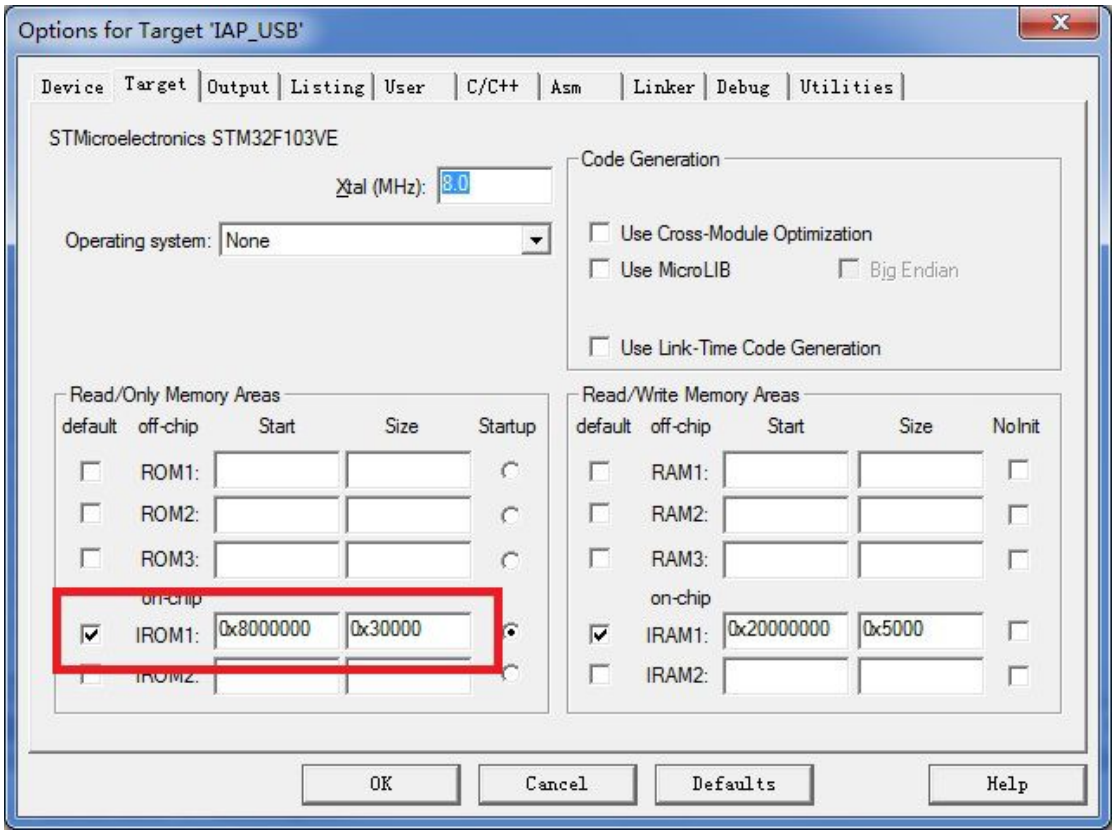
```
typedef    void (*pFunction)(void);

uint8_t DeviceState;
uint8_t DeviceStatus[6];
pFunction Jump_To_Application;
uint32_t JumpAddress;
```

3、接着就是初始化你用到的外设了，具体见工程代码。

第四步、修改 MDK 的相关地址什么的设置



点击上图中红色框按钮，再点击蓝色框选项弹出设置窗口，或者直接点击绿色的魔术棒进入。

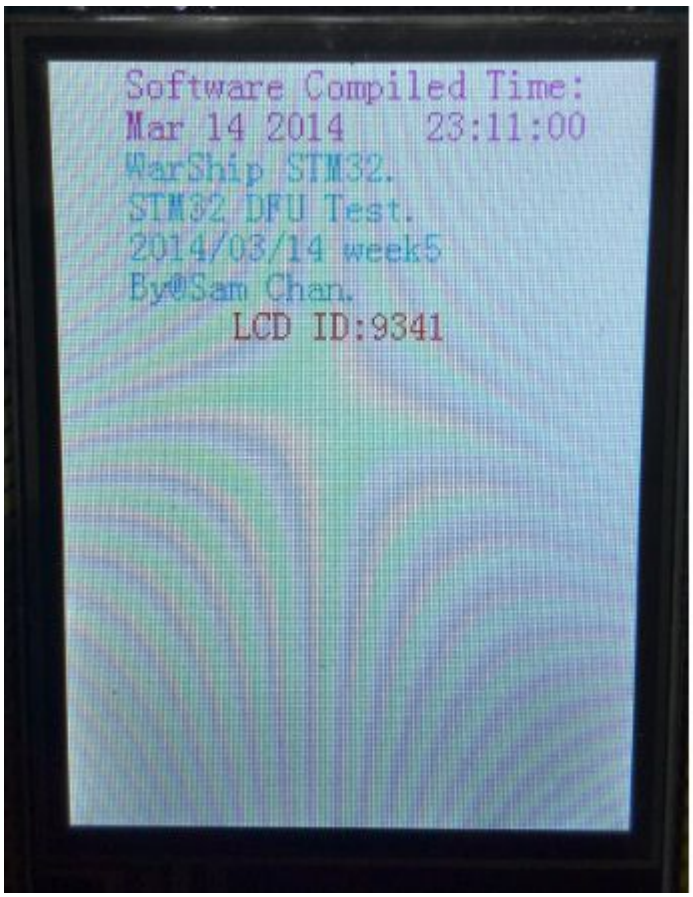


如上图，红色框框就是这次的 Bootloader 程序在 STM32 内部 flash 的

存储空间大小了，具体大小视你程序的大小而定。不一定按我途中的参数，呵呵。

剩下的什么 output 什么的自己看着设置哈。

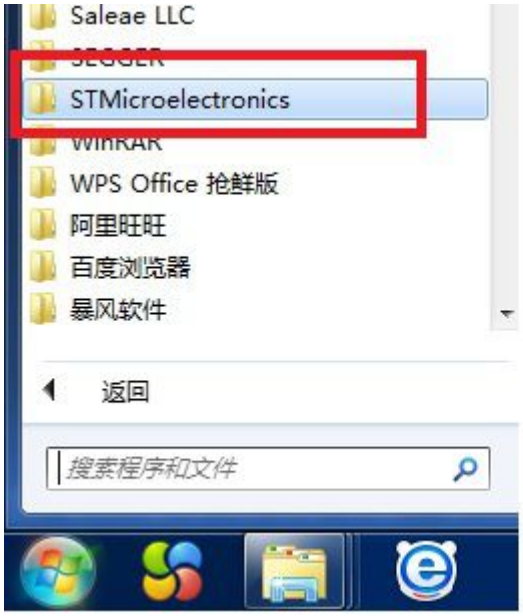
完成之后就点击编译按钮，接着就是进行改错了，完成后就用 jlink 下载到板子上，运行。别着急，前面我们设置了一个按键进入 DFU 模式的，这时按住这个按键后再复位下板子，这时你就看到如下的显示信息了（具体怎么定义自己修改）。



到这里了，我们的 Bootloader 程序就完成了。接下来就是安装电脑端的驱动什么的了，这个驱动程序在 ST 的官网上面我没有搜索到，我在百度上面下载的，具体在附件上奉上。

第五步、安装电脑端驱动软件

双击下载下来的“DfuSe_Demo_V3.0_Setup.exe”文件安装，一路点击 next 按钮下去即可，知道完成，完成后在电脑上就会有这么一个东东了。



红色框框的就是刚才安装好的驱动快捷文件夹了，依次打开将下面的三个软件图标放到桌面快捷方式去，方便操作。

电脑属性里面设备管理器可以看到相关的设备

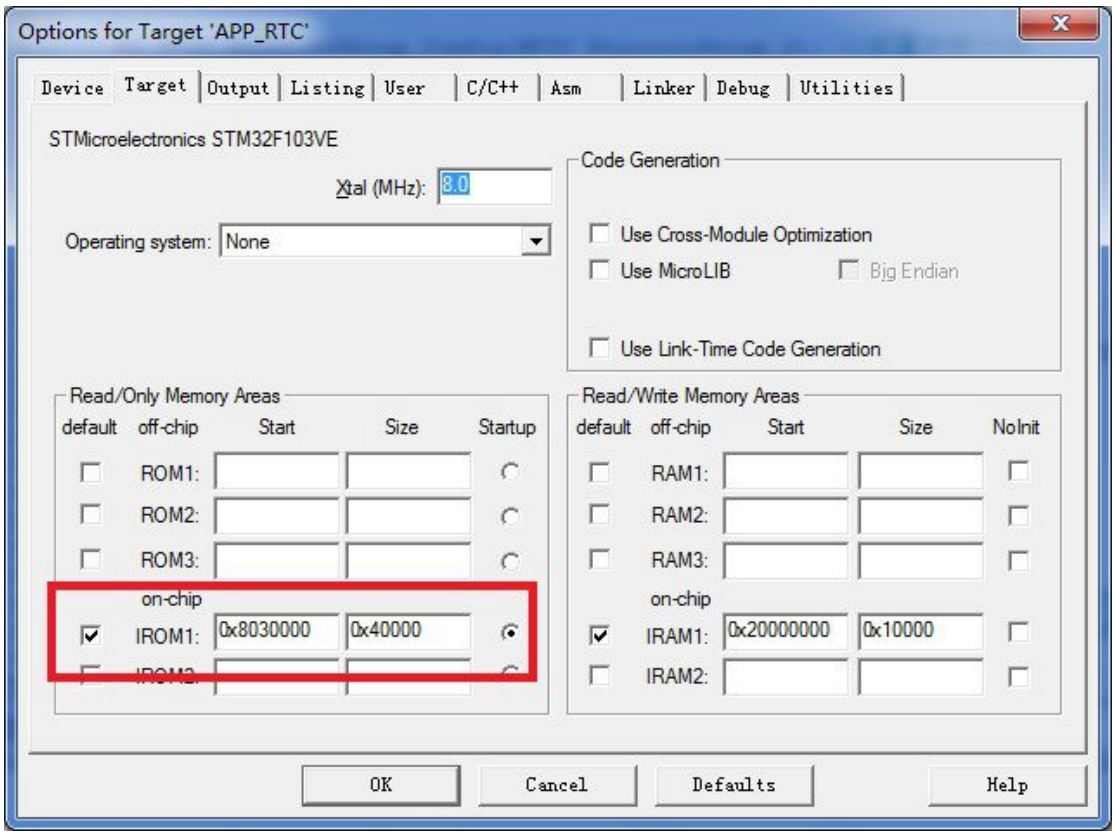
第六步、编写 APP 程序代码

1、这里的工程建立就不用细说了，大家都懂的哈。

2、这里我就建一个 RTC 的实验例程来做个说明。

3、其他的和普通工程一样，指示下面提到的地方不一样。请注意

在点击魔术棒出来的对话框中



注意图中红色框框的 ROM 地址了，因为之前的 Bootloader 程序占用了一部分的 flash 空间，所以这里的 APP 程序的存储起始地址就有变动了，具体起始地址就是你 Bootloader 程序中宏定义#define ApplicationAddress 0x08030000

这个地址了，因为战舰的主控有 512K 的 flash 程序空间，所以这个加减的运算就大家算了，呵

4、这里设置完成后，在 APP 程序的 main 函数中添加如下代码

SCB->VTOR = FLASH_BASE | 0X30000; /* Vector Table Relocation in Internal FLASH. */

这个代码修改中断向量表地址重设的操作，就是 flash 基地址 + 偏移地址，具体是你 Bootloader 程序中设定的 APP 程序起始地址有关。

具体在 system_stm32f10x.c 文件中的 void SystemInit (void)函数最后

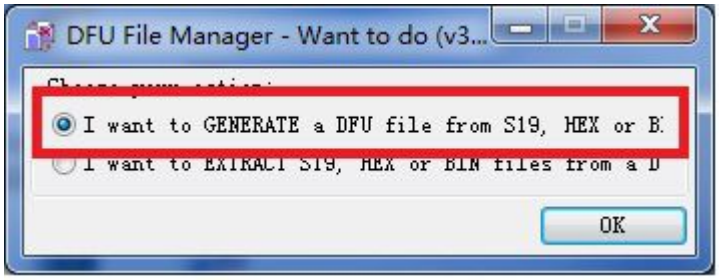
面有这个操作。

5、剩下的就和普通的程序一样编写即可。

第七步、APP 程序的下载操作

APP 程序 HEX 格式文件的转换

1、程序“DFU File Manager”就是转换 HEX 文件为 DFU 格式的程序。双击打开

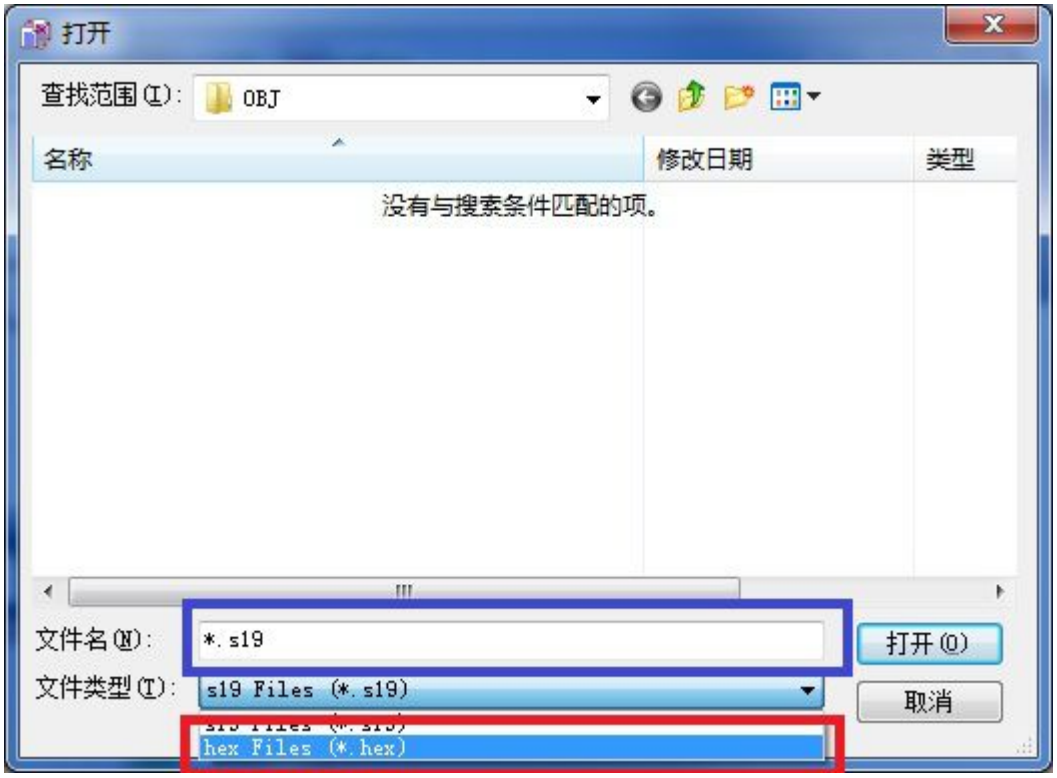


弹出的对话框中我们选择**红色框框**（也是默认值）

2、然后点击 OK 按钮，然后弹出下面的对话框

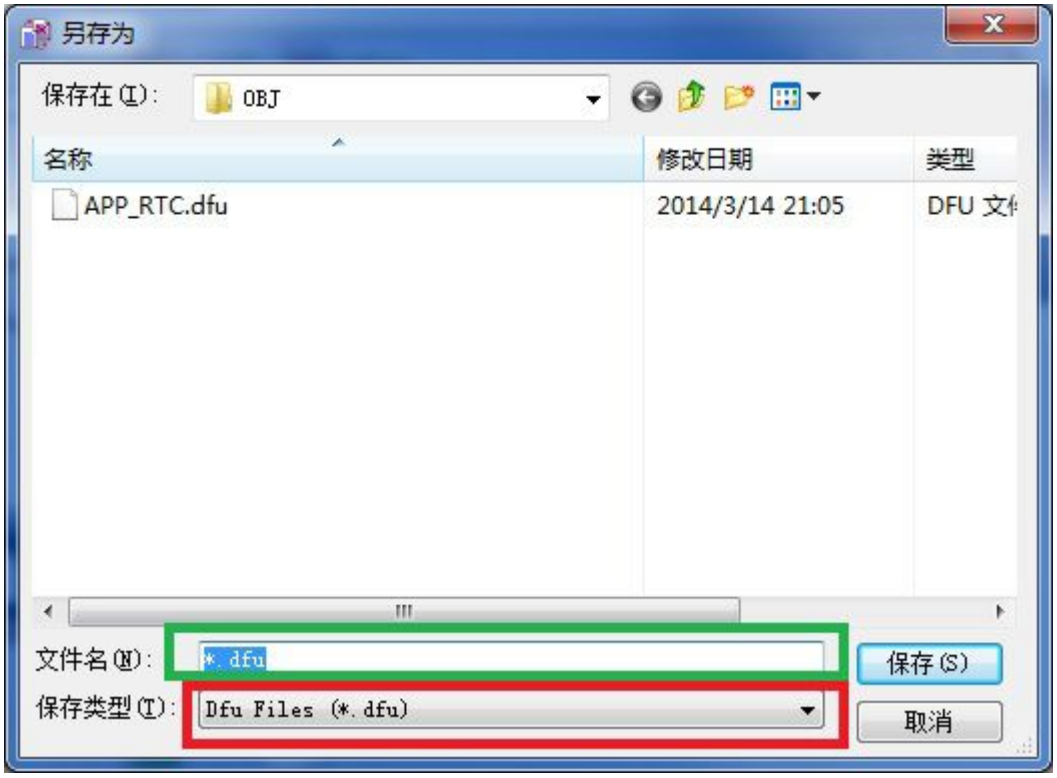
图 1

3、然后点击**红色框框**的按钮选择你的 HEX 文件了（具体哪个 APP 的 hex 格式文件就自己看着选择了哦），如下对话框



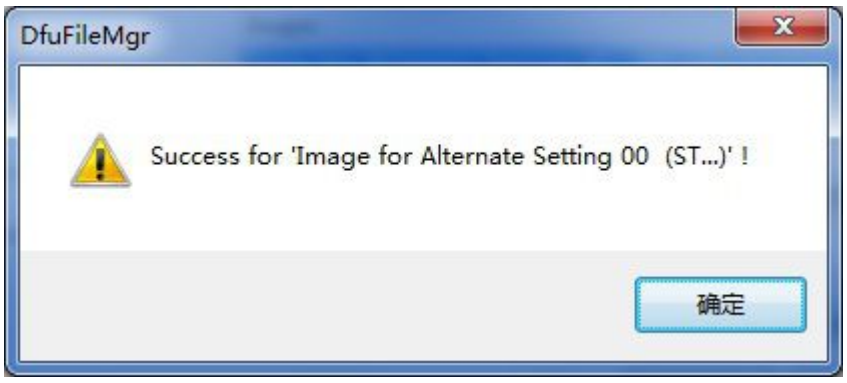
选择上图中红色框框的文件格式，后面的操作自己看着办。

4、选择好之后就点击【图 1】中绿色框框的按钮，弹出下面对话框



选择途中红色框框的格式（默认），然后在绿色框框地方填写一个保

存的 dfu 格式文件的名字，最后点击保存按钮即可，成功之后会提示这么一个对话框



说明已经将 HEX 格式文件转换成 dfu 格式文件了。

如果要转换其他的 hex 格式文件，请点击【图 1】蓝色框框的按钮清除当前的路径什么的，接着重复 2～4 步的操作即可。

第八步、转换好的 dfu 格式文件下载

1、开发板 USB 线连接上电脑，按住 Bootloader 程序中设定的 DFU 模式按键使板子进入 DFU 下载模式，然后给板子上电或者复位。等待电脑安装驱动，我的是 WIN 7 32bit 操作系统，没有提示加载什么的驱动文件，遇到有加载驱动不成功的请百度寻找解决办法，呵呵。

2、双击打开“DfuSe Demonstration”这个软件。

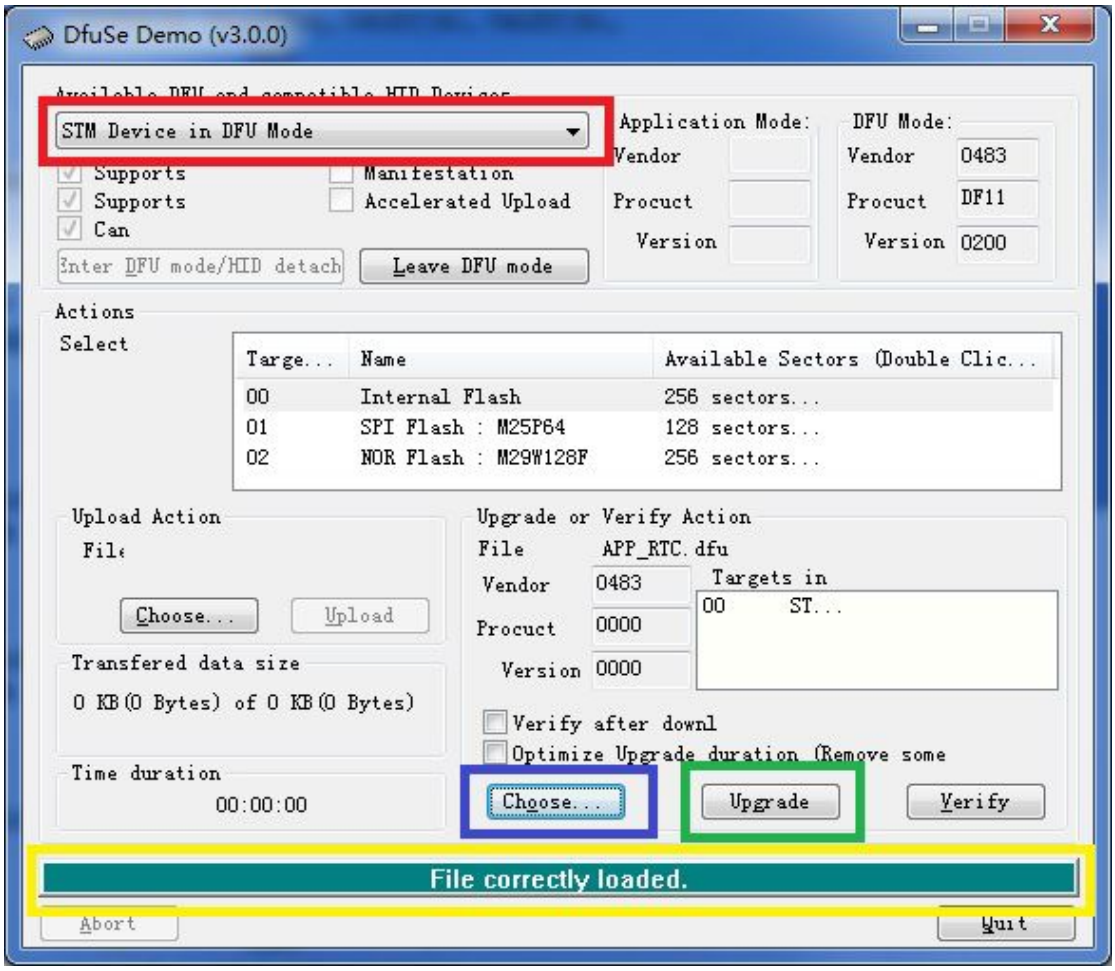
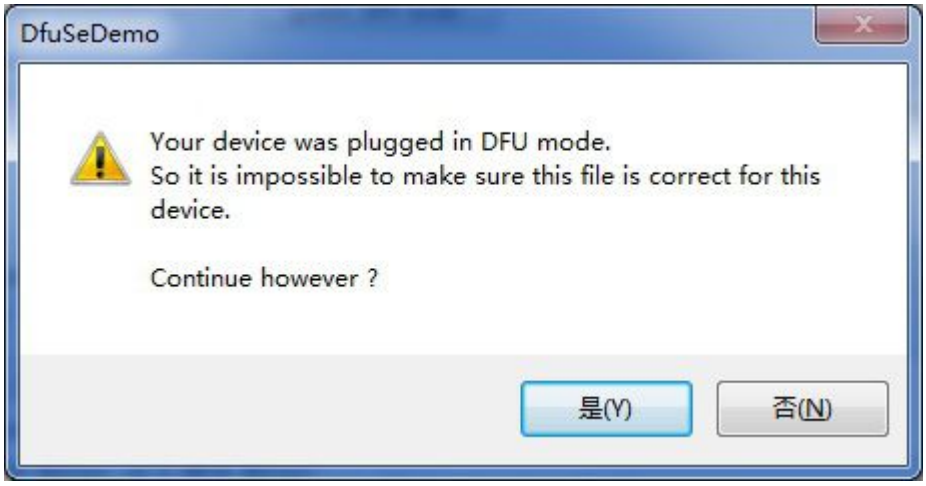


图 2

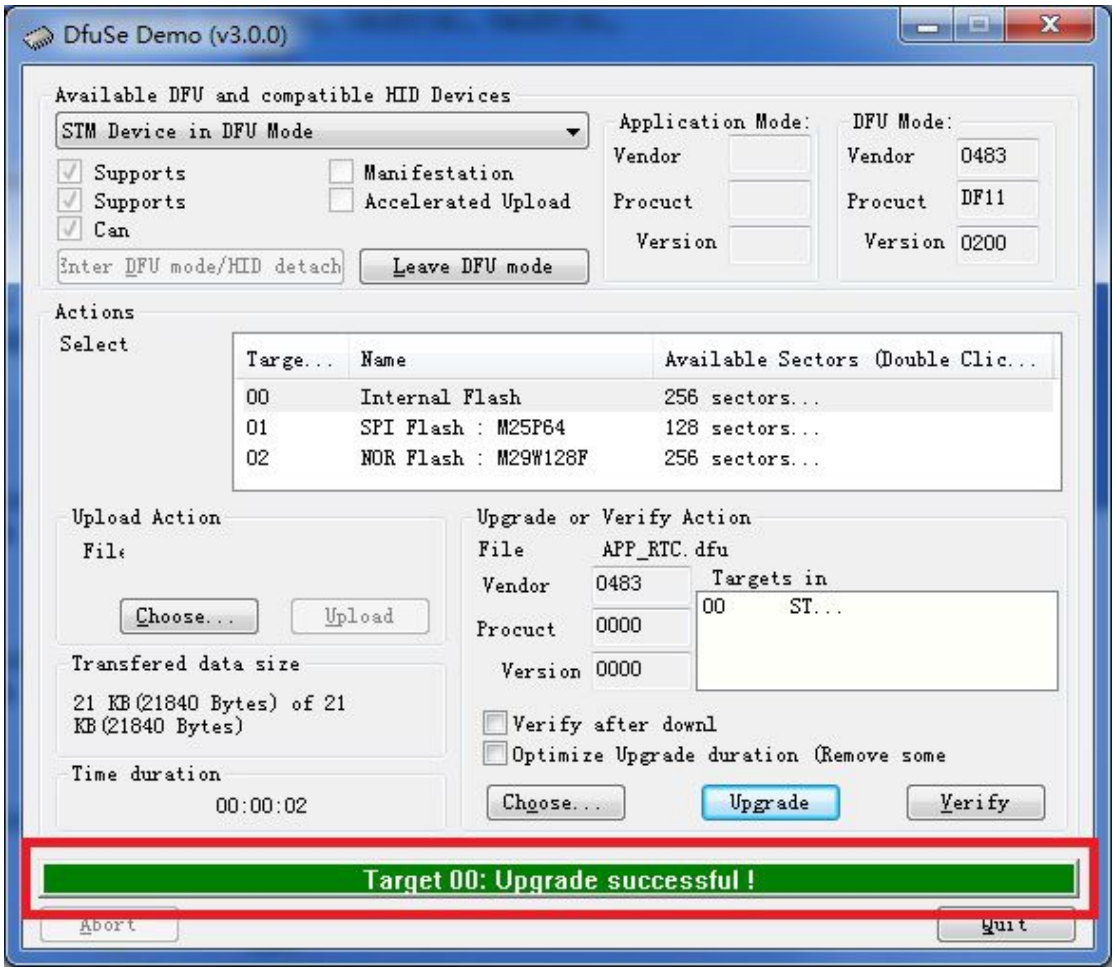
3、正确安装驱动、连接上开发板后，图 2 中红色框框就会显示这么一个东东，否则什么都没有的。

4、点击图 2 中的蓝色框框添加前面转换得到的 dfu 格式文件。

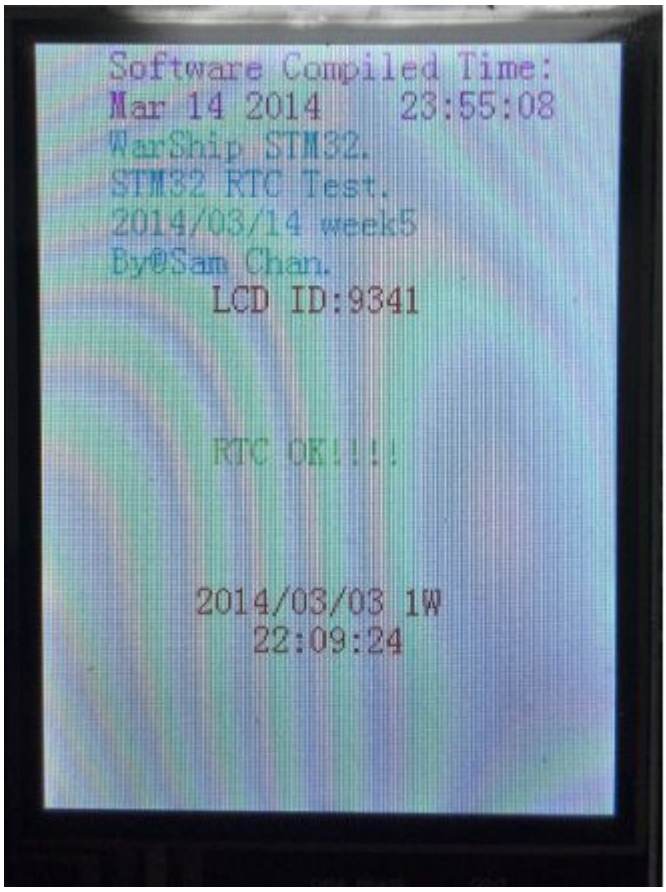
5、dfu 格式文件添加成功后，图 2 中绿色框框的按钮将激活，否则是不激活的，黄色框框地方就提示状态。这时就点击绿色框框中的按钮就可以升级 APP 程序了。弹出下面的提示框，点击 OK 即可



6、升级完成后就变成下面样子了



上图红色框框的提示信息说已经升级成功了，这时复位一下板子就可以看到你的 APP 程序运行的界面了。



最后祝大家的 DFU 升级 APP 调试成功。

制作：Sam Chan

日期：2014/03/15 00:00:10