# 基于KE02的SD卡升级Demo

作者：Jianhui Tong

邮箱：Jianhui.tong@nxp.com

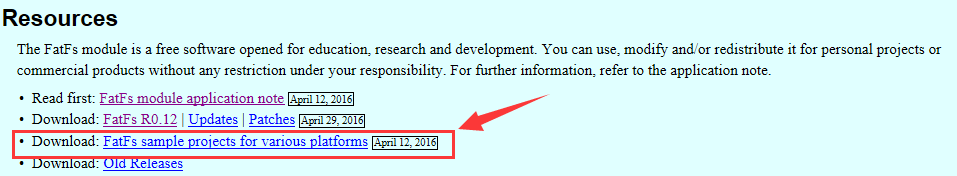
# 简介

本实验基于FRDM-KE02Z40M，完成了使用SD卡升级用户程序的功能。使用的SD卡文件格式为FAT32格式，用户程序为二进制的bin文件。主要硬件使用了FRDM-KE02Z40M开发板和一个SPI通信的SD卡卡座模块，软件部分的升级程序占用FLASH约14KB，RAM约1KB。

# 第一部分 FatFS的移植

1. 下载FatFS的移植sample：

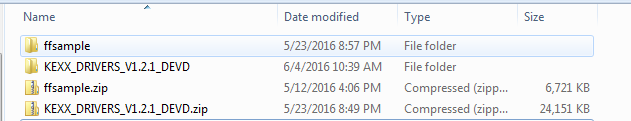
<http://elm-chan.org/fsw/ff/00index_e.html>



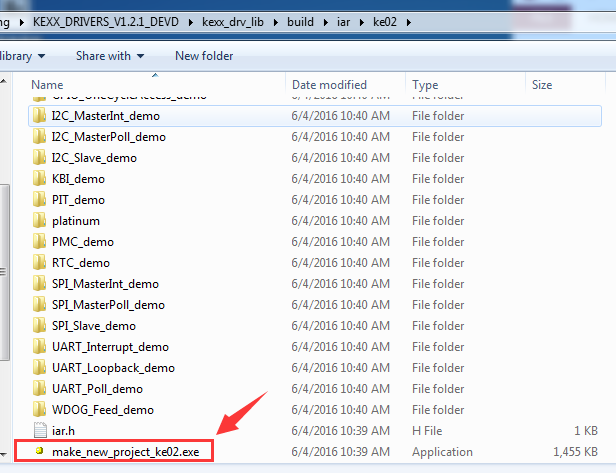
1. 下载NXP官方KE驱动库：

<http://www.nxp.com/webapp/sps/download/license.jsp?colCode=KEXX_DRIVERS_V1.2.1_DEVD&location=null&fsrch=1&sr=6&pageNum=1&Parent_nodeId=&Parent_pageType=>

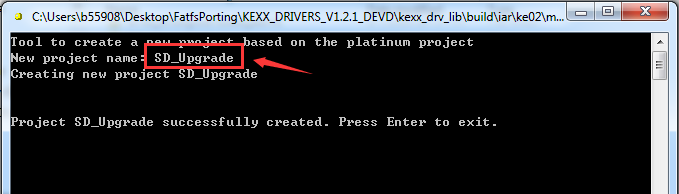
1. 下载完成后解压压缩文件



1. 使用make\_new\_project\_ke02.exe工具新建工程

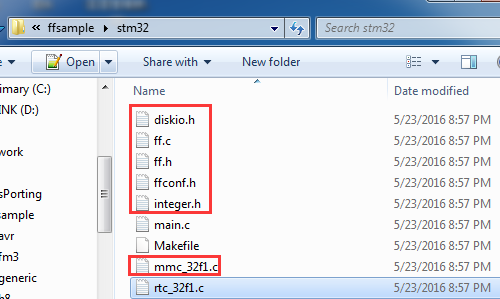


1. 输入工程名字，如“SD\_Upgrade”，按回车确认，再按回车退出。

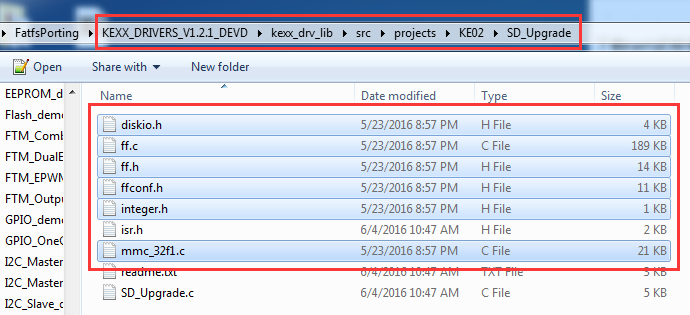


1. 回到ffsample文件夹，我们以stm32的移植范例为例，我们只需要拷贝其中的

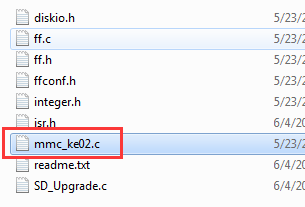
mmc\_32f1.c, integer.h, ffconf.h, ff.h, ff.c, diskio.h 六个文件。



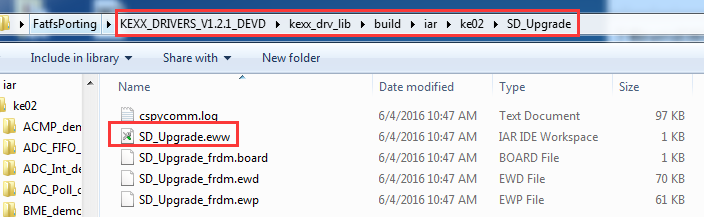
1. 回到SD\_Upgrade工程存放源文件的目录，并将复制的文件粘贴进来。



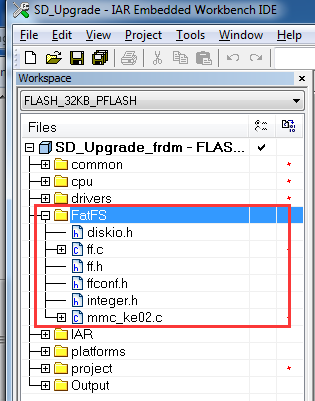
1. 将mmc\_32f1.c重命名为mmc\_ke02.c



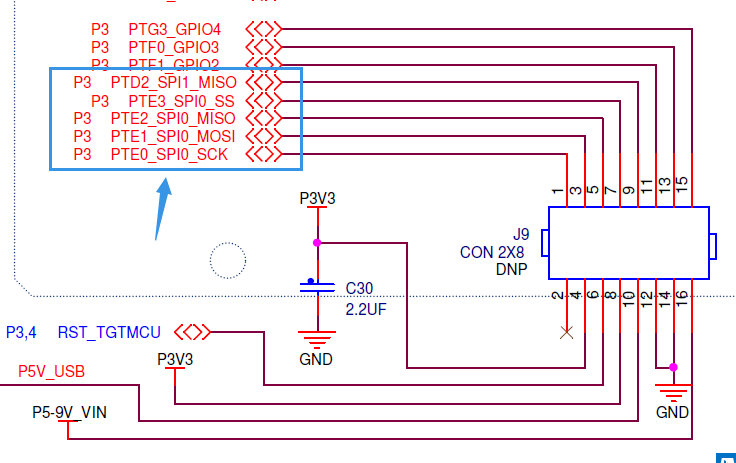
1. 回到SD\_Upgrade工程所在目录，并打开IAR工程文件。



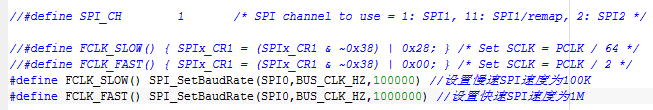
1. 在IAR工程窗口中，新建FatFS文件夹，并将刚才的6个文件添加进来。



1. 本Demo使用FRDM-KE02Z40M开发板，使用SPI0与SD卡通信，PTD2用于检测SD卡的CD信号(Card Detection)。参照开发板的原理图，实际需要配置的引脚如下图所示。



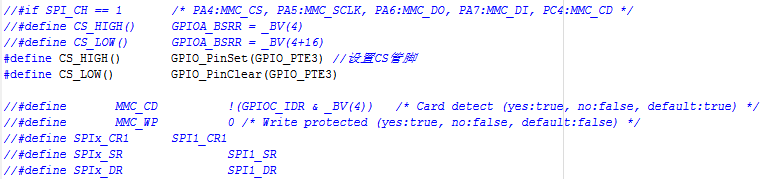
1. 打开mmc\_ke02.c文件，开始修改。



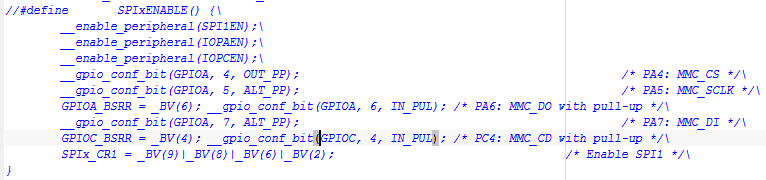
SPI\_CH用于选择SPI通道，在本Demo中使用固定的SPI，因此可以去掉此定义。

FCLK\_SLOW和FCLK\_FAST按照上图所示进行修改。

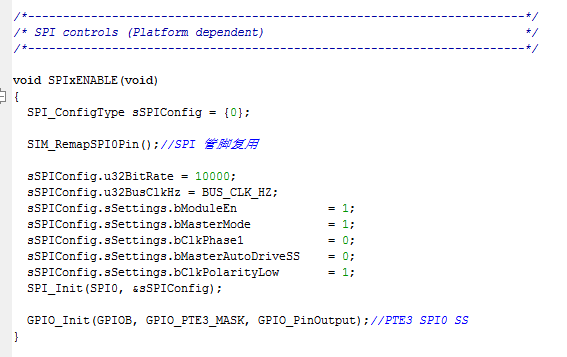
1. 将CS\_HIGH和CS\_LOW进行重新定义，其他未使用的部分可以注释掉。



1. SPIxENABLE()在这里是以宏定义的方式出现，我们注释掉，并在文件后面使用函数进行替代。



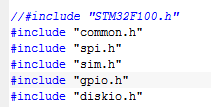
在函数定义部分前，加上SPIxENABLE函数的定义。



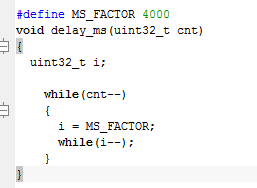
1. 文件中还有一些使用SPI其他通道时相应的宏定义，都可以注释掉。



1. 修改头文件部分，注释掉STM32F100头文件，添加common,spi,sim,gpio的头文件。

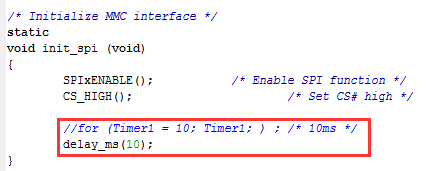


1. 我们直接使用指令进行延时替代原有的定时器延时，因此需要将程序中延时部分进行修改。

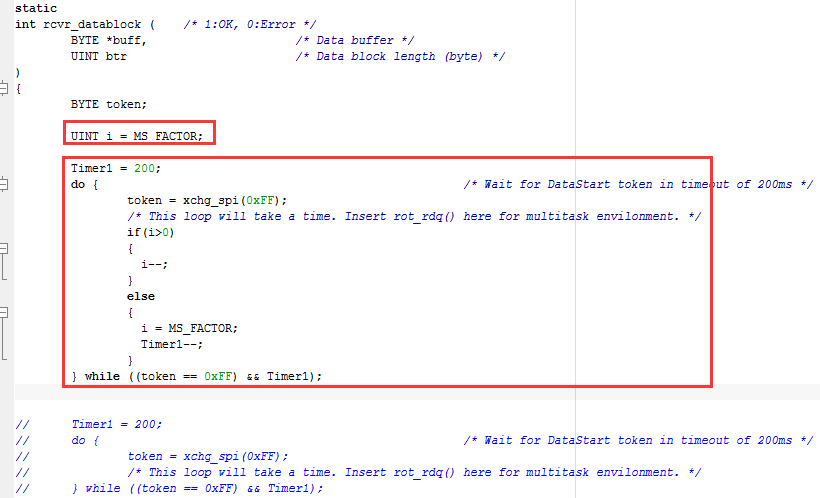


添加MS\_FACTOR宏定义，和delay\_ms延时函数。MS\_FACTOR可以根据实际CPU工作频率调整，使得每1ms，delay\_ms函数中的cnt变量减1。

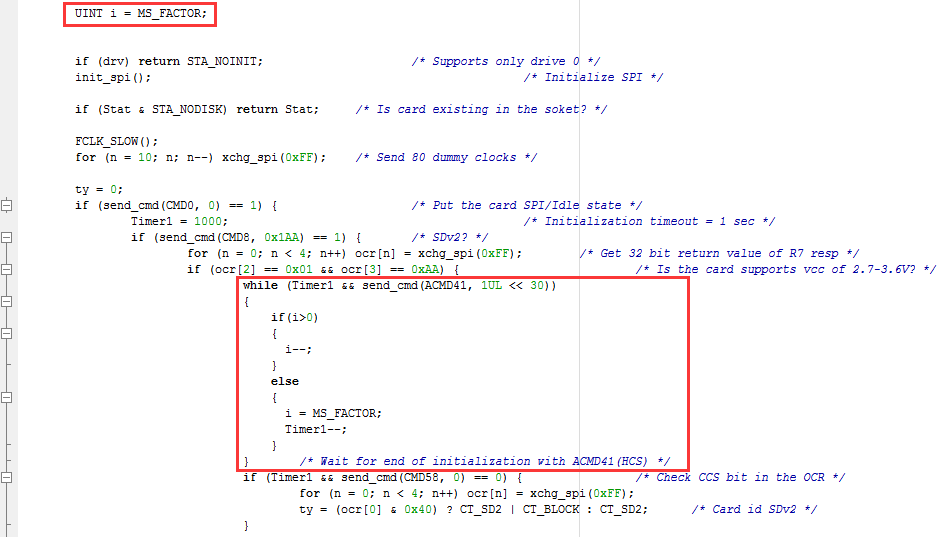
对原来使用Timer1和Timer2变量进行延时的地方进行修改。

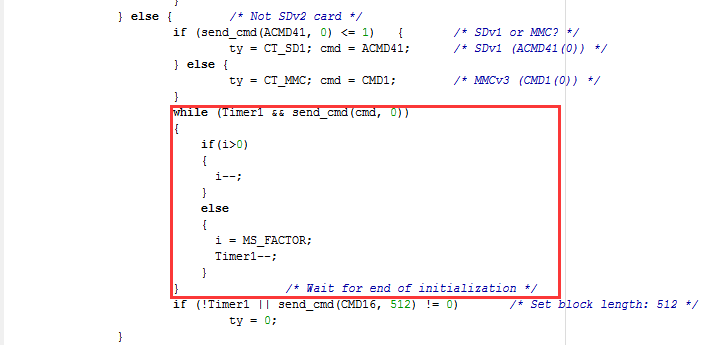


修改init\_spi函数。

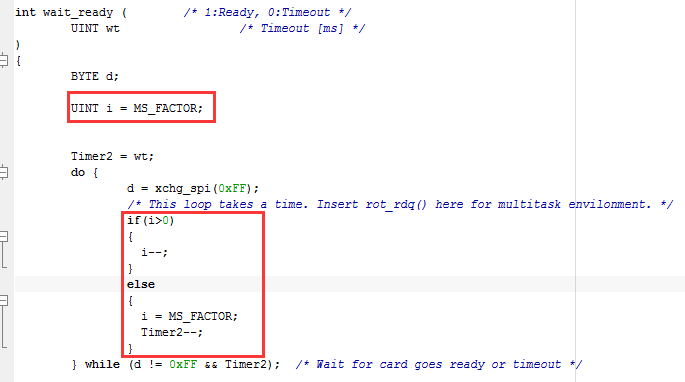


修改rcvr\_datablock函数。





修改disk\_initialize函数

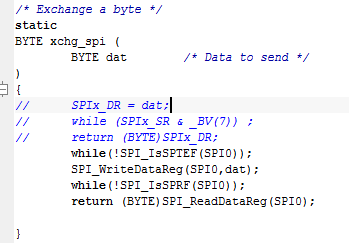


修改wait\_ready函数

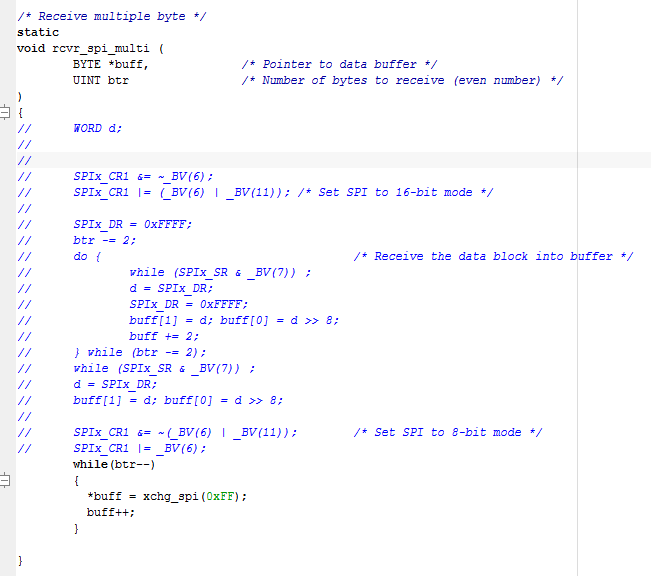


注释disk\_timeproc函数。

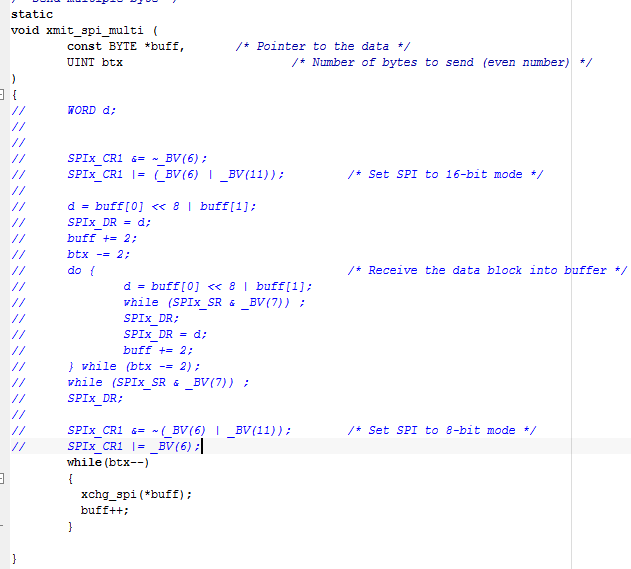
1. 修改SPI底层相关函数。



修改xchg\_spi函数。

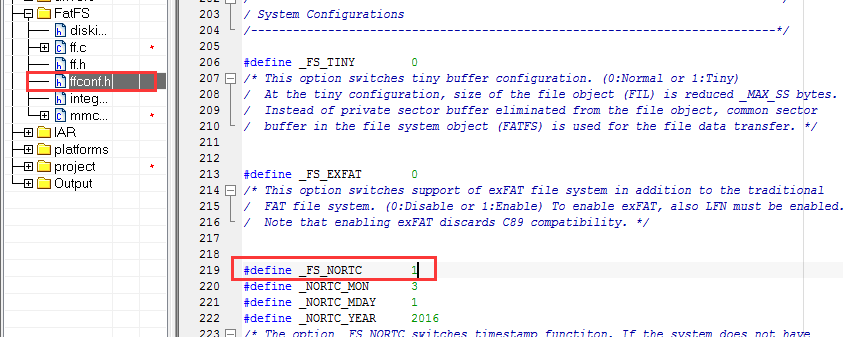


修改rcvr\_spi\_multi函数。



修改xmit\_spi\_multi函数。

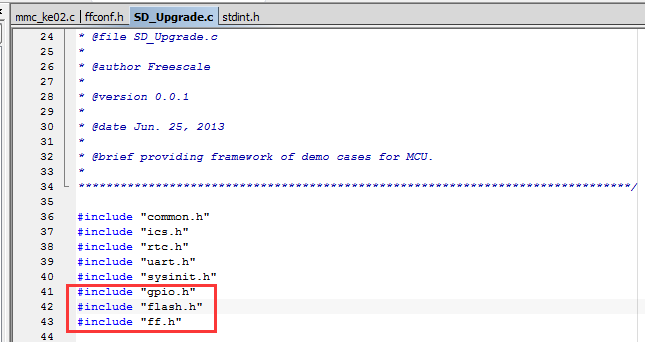
1. 修改ffconf.h中的NORTC定义为1



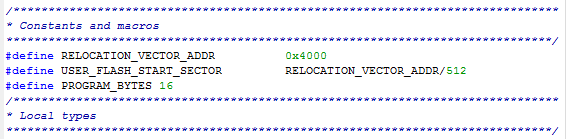
1. 此时编译就能通过了，FatFS也就移植完成了，使用FatFS只用包含ff.h文件，就能调用其中的函数啦。

# 第二部分 SD升级程序

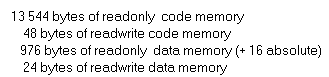
1. 对主函数所在源文件进行修改。
2. 添加使用到的头文件 gpio.h, flash.h, ff.h。



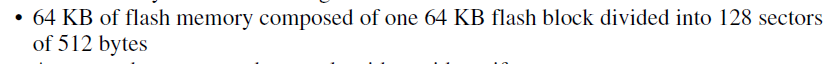
1. 添加三个宏定义:



RELOCATION\_VECTOR\_ADDR，定义了用户程序向量表的起始地址，这个必须超出SD卡升级Demo所占空间的尾地址，此处定义为0x4000，即留出16KB给SD升级Demo程序，实际上用不了这么多。下图为程序编译后，实际占用大小情况。

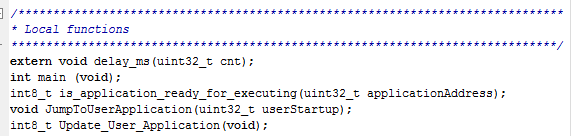


USER\_FLASH\_START\_SECTOR，定义了用户程序所在的起始Sector，由于使用了KE02Z64VQH4芯片，可以在参考手册中查到，每个sector为512 Bytes，如下图所示。

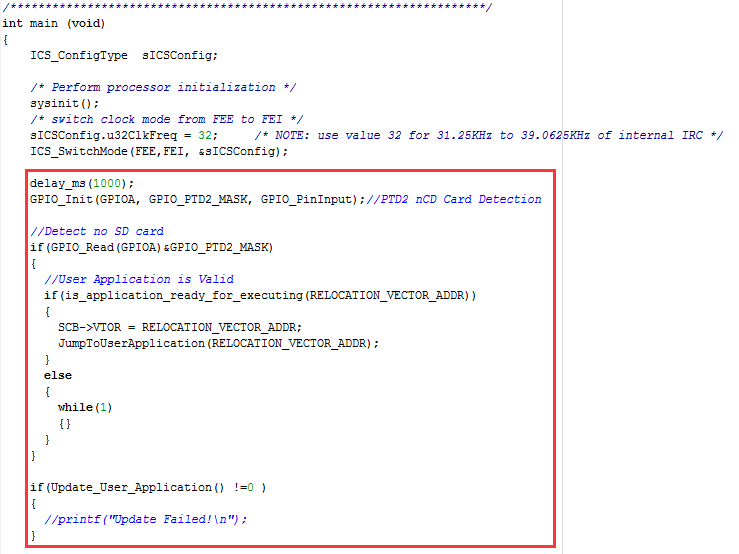


PROGRAM\_BYTES，定义了每次读取bin文件的大小，同时也是烧写用户程序每次的大小。此数值越大，则每次读取烧写时所需要的buffer也就越大。

1. 添加函数声明。



1. 修改main函数

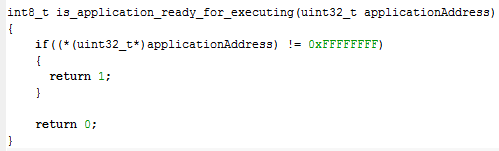


上电时会延时1s，等待CD信号稳定，然后通过PTD2管脚检测SD卡座的CD信号来判断是否有SD卡插入。

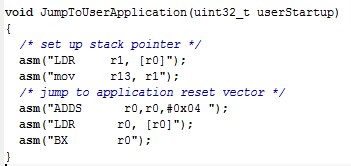
如果没有SD卡插入，则首先通过is\_application\_ready\_for\_executing函数判断应用程序地址是否存在有效程序，有的话就直接跳转过去运行。否则进入while(1)死循环。

如果有SD卡插入，则通过Update\_User\_Application函数进行用户程序升级。

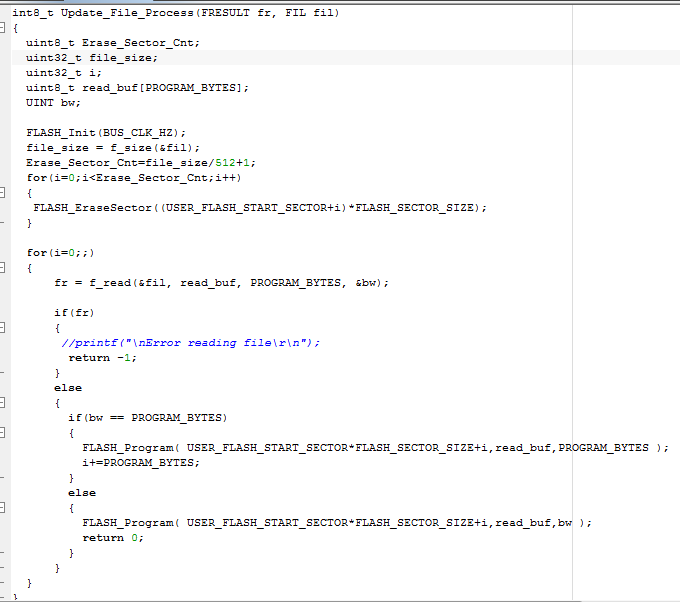
1. Is\_application\_ready\_for\_executing函数实现。这里只对用户程序首地址是否为全0xFF进行了简单的判断，实际使用中，可根据需要添加代码。



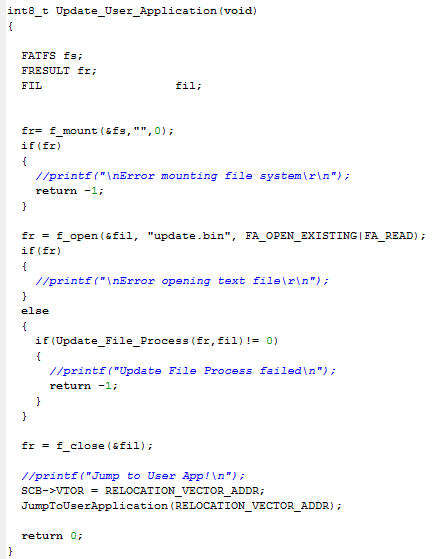
1. JumpToUserApplication跳转函数的实现



1. Bin文件烧写函数Update\_File\_Process实现。函数会首先根据bin文件大小来擦除占用的Sectors。之后在for循环里，根据PROGRAM\_BYTES指定的字节数，来读取文件，并烧写到Flash中。



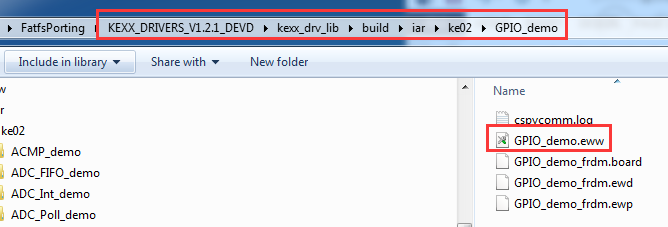
1. Update\_User\_Application函数实现。此函数会打开SD卡中的update.bin文件，如果文件存在，则调用Update\_File\_Process函数进行文件烧写。并在烧写完成后，通过JumpToUserApplication函数跳转到用户程序运行。



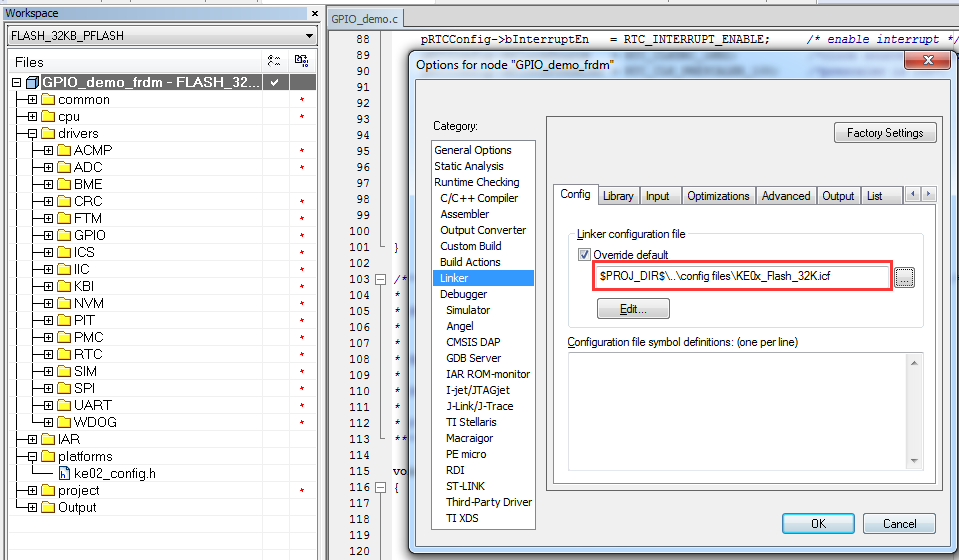
1. 修改完成后，编译整个工程，并将程序下载到单片机中。

# 第三部分 用户程序

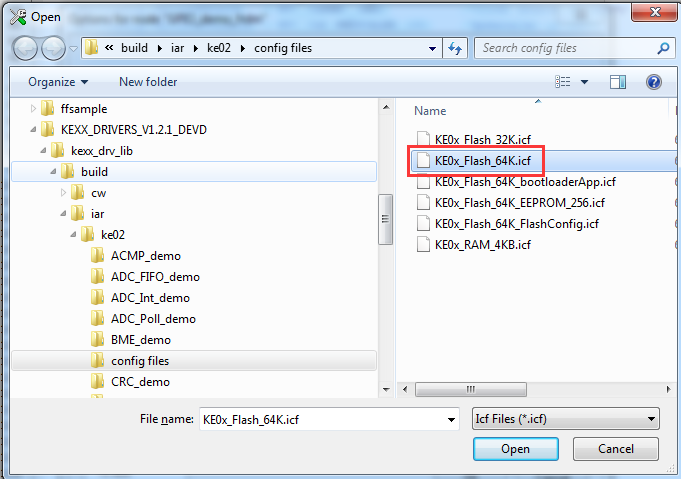
1. 用户程序我们直接使用KE驱动库里的GPIO example 为例。



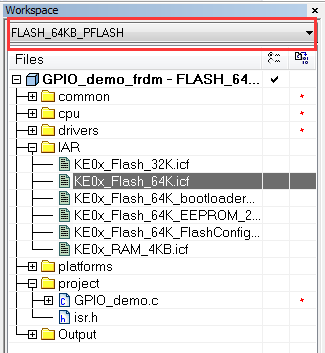
1. 打开GPIO工程，并在Option中找到链接文件，我们需要对链接文件进行修改。



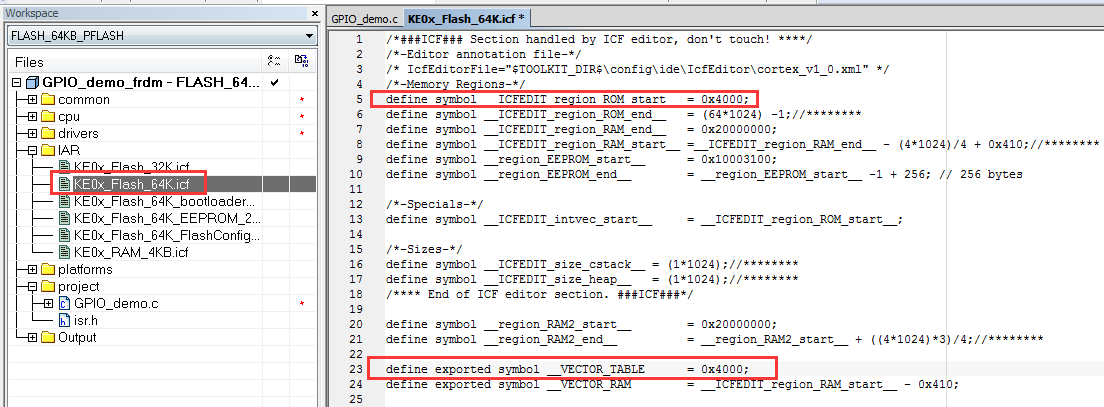
1. 选择64K的icf文件。



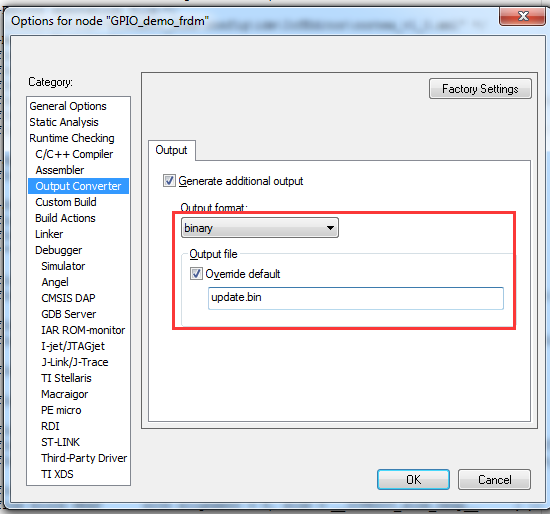
1. Workspace中选择64KB的版本。



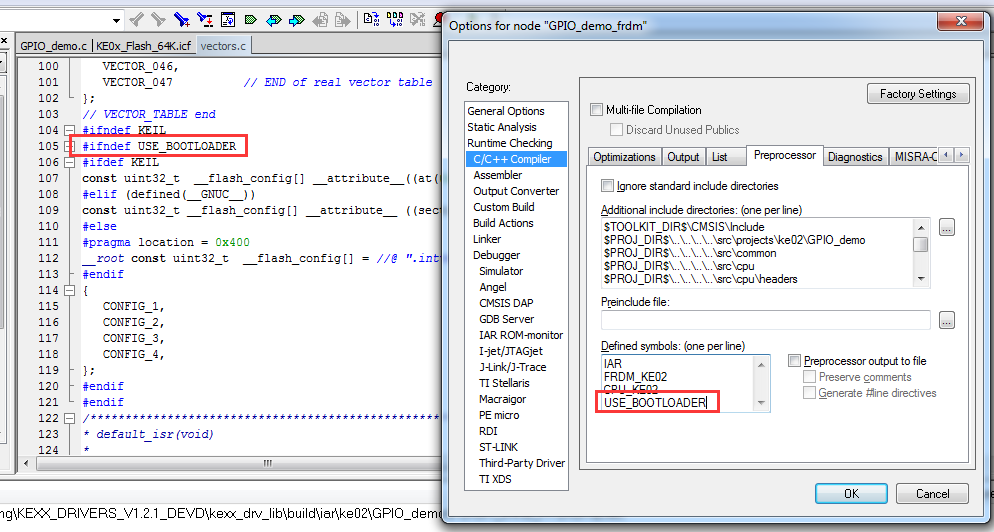
1. 选择64K的链接文件，并将ROM起始地址和向量表地址修改为0x4000，和SD卡升级程序定义的宏RELOCATION\_VECTOR\_ADDR保持一致。



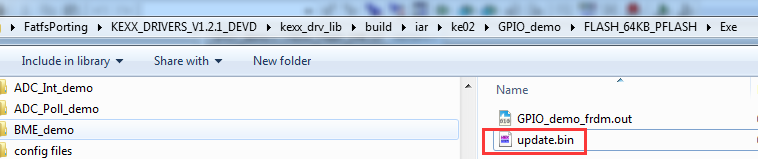
1. 在工程option中将输出格式改为bin文件，并命名为update.bin



1. 在预处理中，添加USE\_BOOTLOADER的定义，从而去掉flash\_config指定地址的定义部分。

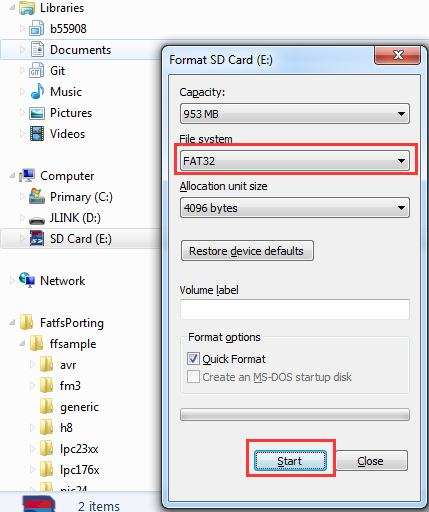


1. 重新编译工程，就能在如下目录中找到生成的用户程序的update.bin文件啦。

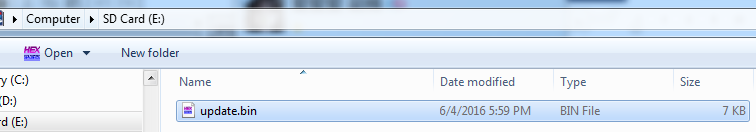


# 第四部分 SD卡

1. 将SD卡插入电脑，首次使用需要将SD卡格式化为FAT32格式。



1. 完成后，将上一部分生成的update.bin文件，拷贝到SD卡中。



1. 接下来就可以使用SD卡，将GPIO的点灯程序下载到单片机里啦！