Kinetis脱机编程工具代码简介

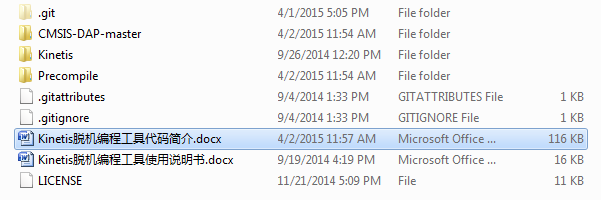
(QQ：17389711 左物右码)

下载地址：

<https://github.com/hudieka/OpenSourceOfflineProgrammerTools/>

目录结构：

该工具主要由三个工程组成：(CMSIS-DAP-master中有两个，Kinetis中有一个)



1. CMSIS-DAP-master:

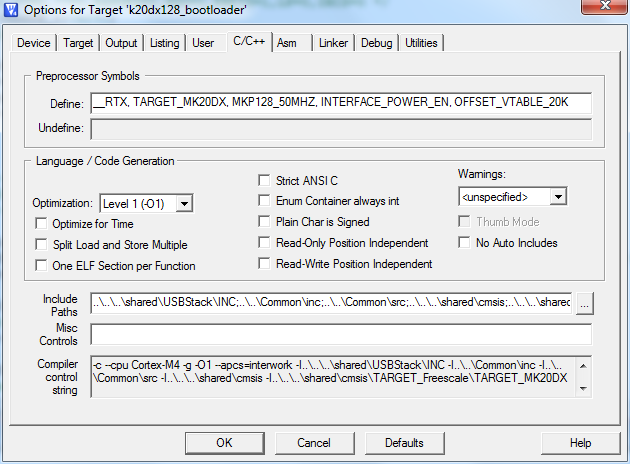
该目录下包含两个工程：bootloader以及prog\_app



bootloader是MK20上电后运行的第一段程序，它从0地址开始运行的。他的工作是可以加载应用程序，应用程序的运行地址由flash.h文件中的START\_APP\_ADDRESS宏来确定。如果您手里有FRDM评估板，这个代码是不需要使用的，因为FRDM板芯片内部同样有一个bootloader，但是目前的FRDM板有两个加载地址，一个是从0x8000(OPENSDA, FRDM-KL26，KE02等)，另一个是从0x5000(MBED, FRDM-K64,K22)，区分方法就是插上USB后，看枚举成的U盘是Mbed还是opensda，这不同的加载地址也是Jlink\_opensda有2.0和2.1两个版本的原因。该代码主要用于自己设计的板子，MK20芯片内部无任何程序。

prog\_app:该工程是脱机编程工具的源代码，其编译出来的firmware运行在MK20的0x5000(或0x8000)地址上，具体地址由bootloader工程中的START\_APP\_ADDRESS以及本工程在预编译中的设置：OFFSET\_VTABLE\_20K或OFFSET\_VTABLE\_32K

如果在FRDM板上做验证，可以不烧写Bootloader，只修改prog\_app中的OFFSET\_VTABLE\_xxK定义来适应FRDM默认的bootloader，这样可以不需要通过仿真器对MK20进行编程，而通过按下复位按钮进入bootloader模式后，将prog\_app生成的bin文件拖至bootloader枚举成的U盘中进行烧写。



prog\_app上电运行后，如果通过USB连接到PC，会枚举成一个PROGLOADER的U盘，用户将需要烧写的目标文件(仅支持bin文件)拖至该U盘中，prog\_app会自动将该bin文件放置MK20芯片内部0x10000(flash.h的START\_APP\_ADDRESS确定)这个地址。

本代码中需要注意以下几个宏：

1. flash.h中

#define NB\_SECTOR (127)

这里需要写入实际的sector -1，因为算法程序放在芯片的最后一个sector，如果这里填写的是全部数量的sector，在更新目标文件时会将除bootloader, prog\_app以外的所有数据擦除，这样会丢失编程算法程序。

1. Debug\_cm.h中

#define FILE\_SIZE 0x1F800

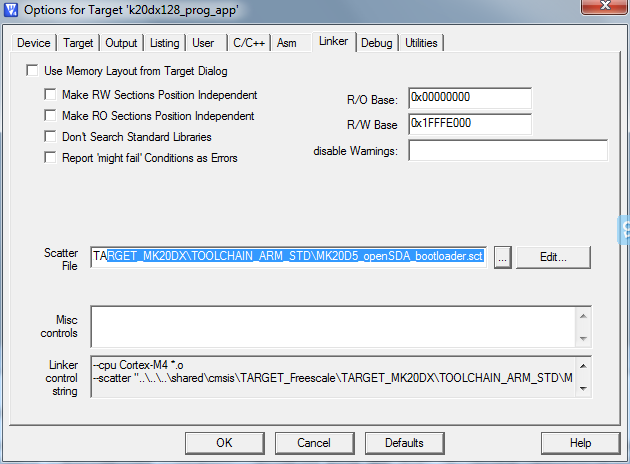
该地址定义了目标程序的大小。

#define KINETIS\_FLASHPRG 0x1FC00

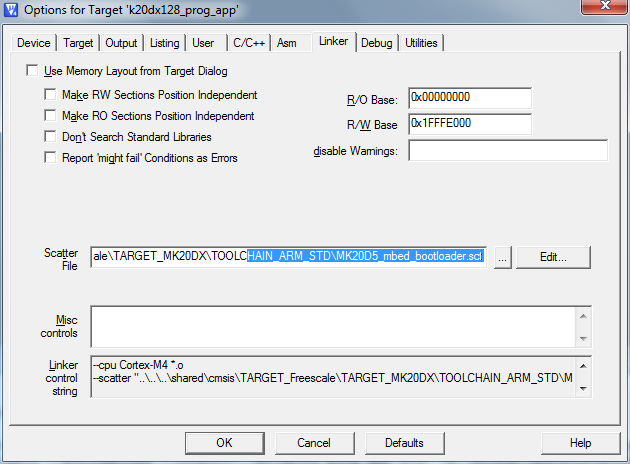
该地址定义了算法程序的存储地址

1. 链接文件

由于有mbed和opensda两个不同的启动地址，所以需要两个不同的链接文件，如果用户使用opensda(0x8000)地址为其实地址，那么sct必须选择正确：



如果使用mbed地址那么需要选择mbed对应文件：



文件的地址在：

\shared\cmsis\TARGET\_Freescale\TARGET\_MK20DX\TOOLCHAIN\_ARM\_STD

1. 目标文件的大小不能超过61K，prog\_app是从0x10000这个地址开始存放目标文件的，可以用到0x1F7FF。
2. Kinetis:

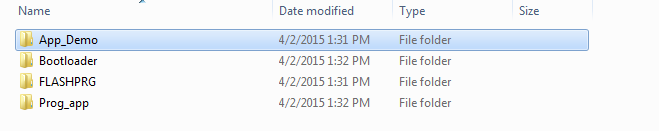
该目录下存放了编程算法，由于不同的目标芯片有不同的flash IP，所以为了适配所有的Kinetis系列，这里提供了常用的编程算法，如果Kinetis有新的flash IP，仅仅修改两个函数即可添加新的目标芯片。

该工程编译后生成.CRD文件名的文件，表示是编程算法，更新编程算法需要通过USB连接到PC，将生成的. CRD文件拖入PROGLOADER这个U盘中，prog\_app会根据扩展名.CRD来判断此次更新的是算法文件，而非目标程序。算法文件放在prog\_app代码的宏 KINETIS\_FLASHPRG(默认0x1FC00，128Kflash最后一个扇区的起始地址)对应地址。

注意：由于本工程使用IAR 7.3进行调试开发，如果用户IAR版本过低，会出现无法编译的现象。

1. Precompile:

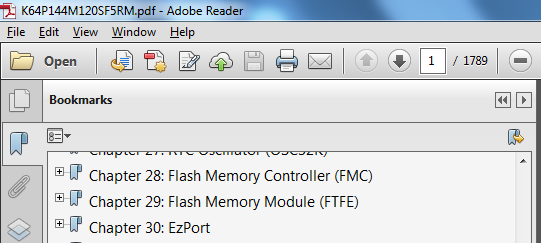
该目录下存放的预先编译好的固件。



App\_Demo：用于测试的目标二进制文件，例如：FRDM\_K64\_BLUE.bin

Bootloader：预编译的bootloader二进制文件

FlashPRG：预编译的编程算法二进制文件。用户需要根据目标MCU来选择其对应的算法程序，目标MCU的RM手册会说明其Flash的种类，以K64为例，其Flash就为FTFE。



|  |  |
| --- | --- |
| 文件名 | 支持的MCU |
| Kinetis\_FTFL.CRD | K60 |
| Kinetis\_FTFA.CRD | KL26 |
| Kinetis\_FTFE.CRD | K64 |

Prog\_app：预编译的二进制文件

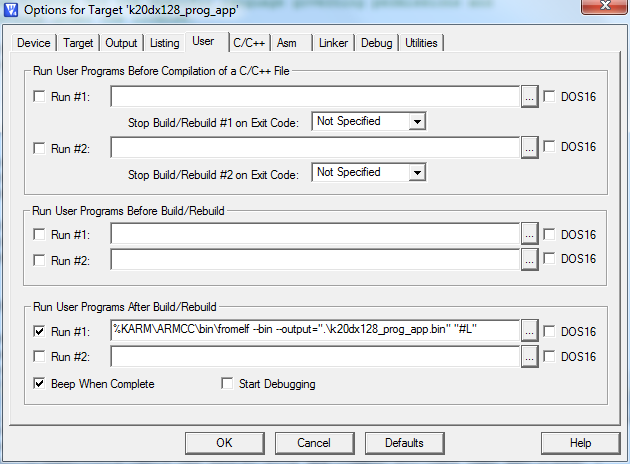
动手实践：

1. 断开USB线与FRDM-K64板的连接，按下SW1连接FRDM-K64进入bootloader模式。
2. 将\Precompile\Prog\_app\k20dx128\_prog\_app\_20K.bin文件拷贝至bootloader中。
3. 重新插拔USB，FRDM会重新枚举成PROGLOADE的U盘。
4. 将Precompile\FLASHPR\Kinetis\_FTFE.CRD拷贝至bootloader中，此时USB会断开连接。至此，工具基本功能以完成，如果需要烧写其他类型的芯片，可以通过拷贝不同的CRD文件来实现，芯片的类型需要根据Kinetis内部flash的类型来决定，Kinetis\_FTFE.CRD对应的FTFE. 下面我们将提供一个烧写红灯闪烁文件的例子
5. 重新插拔USB，让设备重新枚举为PROGLOADE的U盘，然后将\Precompile\App\_Demo\FRDM\_K64\_RED拷贝至U盘中。
6. 长按SW1键，当绿灯亮起后松开。此时绿灯将开始闪烁，闪烁完毕后表示编程完成。
7. 重新插拔USB，可以看到FRDM-K64的板子红灯开始闪烁。
8. 您可以重复5~7步，完成对蓝灯闪烁文件的烧写。

FRDM\_KL26/43等用openSDA的FRDM板，只需要将步骤2的k20dx128\_prog\_app\_20K更换为k20dx128\_prog\_app\_32K，并根据目标MCU的Flash更换Kinetis\_FTFX.CRD即可。

附录

通过Keil生成S19或bin文件的方法



S19文件

%KARM\ARMCC\bin\fromelf --m32combined --output=".\k20dx128\_ prog\_app.S19" "#L"

Bin文件

%KARM\ARMCC\bin\fromelf --bin --output=".\k20dx128\_prog\_app.bin" "#L"