零死角玩转STM32

MDK的编译过程及文件类型全解

淘宝：firestm32.taobao.com

论坛： www.firebbs.cn

扫描进入淘宝店铺

主讲内容

MDK的编译过程及文件类型全解

MDK工程的文件类型

除了上述编译过程生成的文件，MDK工程中还包含了各种各样的文件，下面我们统一介绍，MDK工程的常见文件类型如下表：

后缀	说明
Project目录下的工程文件	
*.uvguix	MDK5工程的窗口布局文件，在MDK4中*.UVGUI后缀的文件功能相同
*.uvprojx	MDK5的工程文件，它使用了XML格式记录了工程结构，双击它可以打开整个工程，在MDK4中*.UVPROJ后缀的文件功能相同
*.uvoptx	MDK5的工程配置选项，包含debugger、trace configuration、breakpoointsl以及当前打开的文件，在MDK4中*.UVOPT后缀的文件功能相同
*.ini	某些下载器的配置记录文件
源文件	
*.c	C语言源文件
*.cpp	C++语言源文件
*.h	C/C++的头文件
*.s	汇编语言的源文件
*.inc	汇编语言的头文件(使用“$include”来包含)

MDK的编译过程及文件类型全解

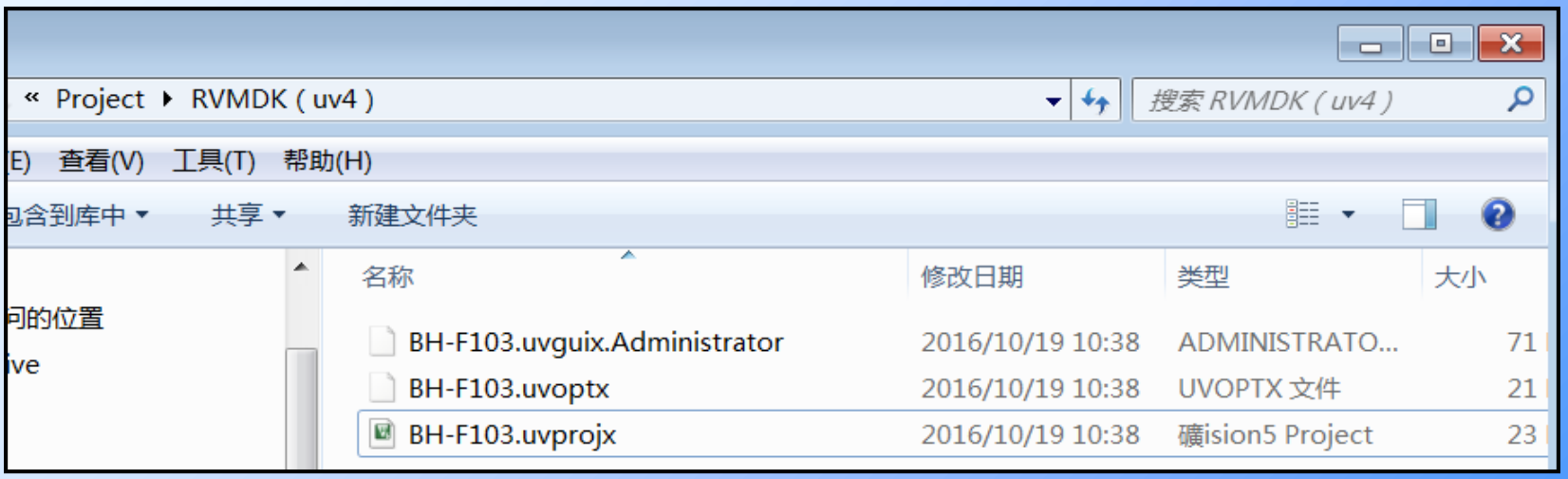
Output目录下的文件	
*.lib	库文件
*.dep	整个工程的依赖文件
*.d	描述了对应.o的依赖的文件
*.crf	交叉引用文件，包含了浏览信息(定义、引用及标识符)
*.o	可重定位的对象文件(目标文件)
*.bin	二进制格式的映像文件，是纯粹的FLASH映像，不含任何额外信息
*.hex	Intel Hex格式的映像文件，可理解为带存储地址描述格式的bin文件
*.elf	由GCC编译生成的文件，功能跟axf文件一样，该文件不可重定位
*.axf	由ARMCC编译生成的可执行对象文件，可用于调试，该文件不可重定位
*.sct	链接器控制文件(分散加载)
*.scr	链接器产生的分散加载文件
*.lnp	MDK生成的链接输入文件，用于调用链接器时的命令输入
*.htm	链接器生成的静态调用图文件
*.build_log.htm	构建工程的日志记录文件
Listing目录下的文件	
*.lst	C及汇编编译器产生的列表文件
*.map	链接器生成的列表文件，包含存储器映像分布
其它	
*.ini	仿真、下载器的脚本文件

这些文件主要分为MDK相关文件、源文件以及编译、链接器生成的文件。我们以“多彩流水灯”工程为例讲解各种文件的功能。

MDK的编译过程及文件类型全解

uvprojx、uvoptx、uvguix及ini工程文件

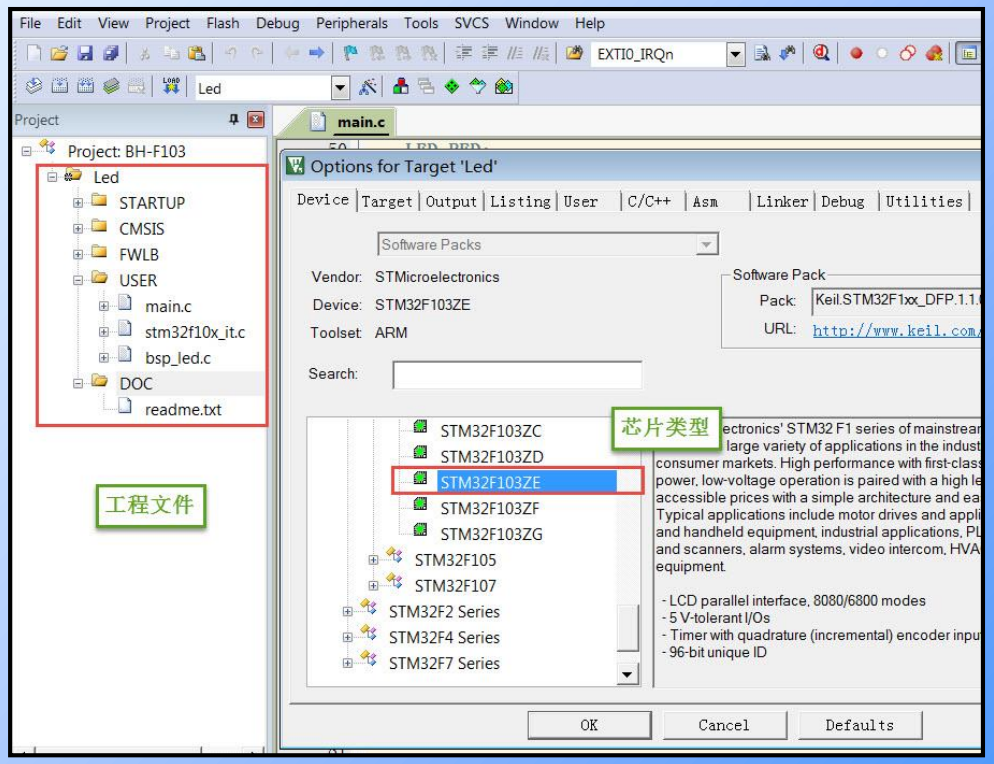
在工程的“Project”目录下主要是MDK工程相关的文件：



MDK的编译过程及文件类型全解

1.uvprojx文件

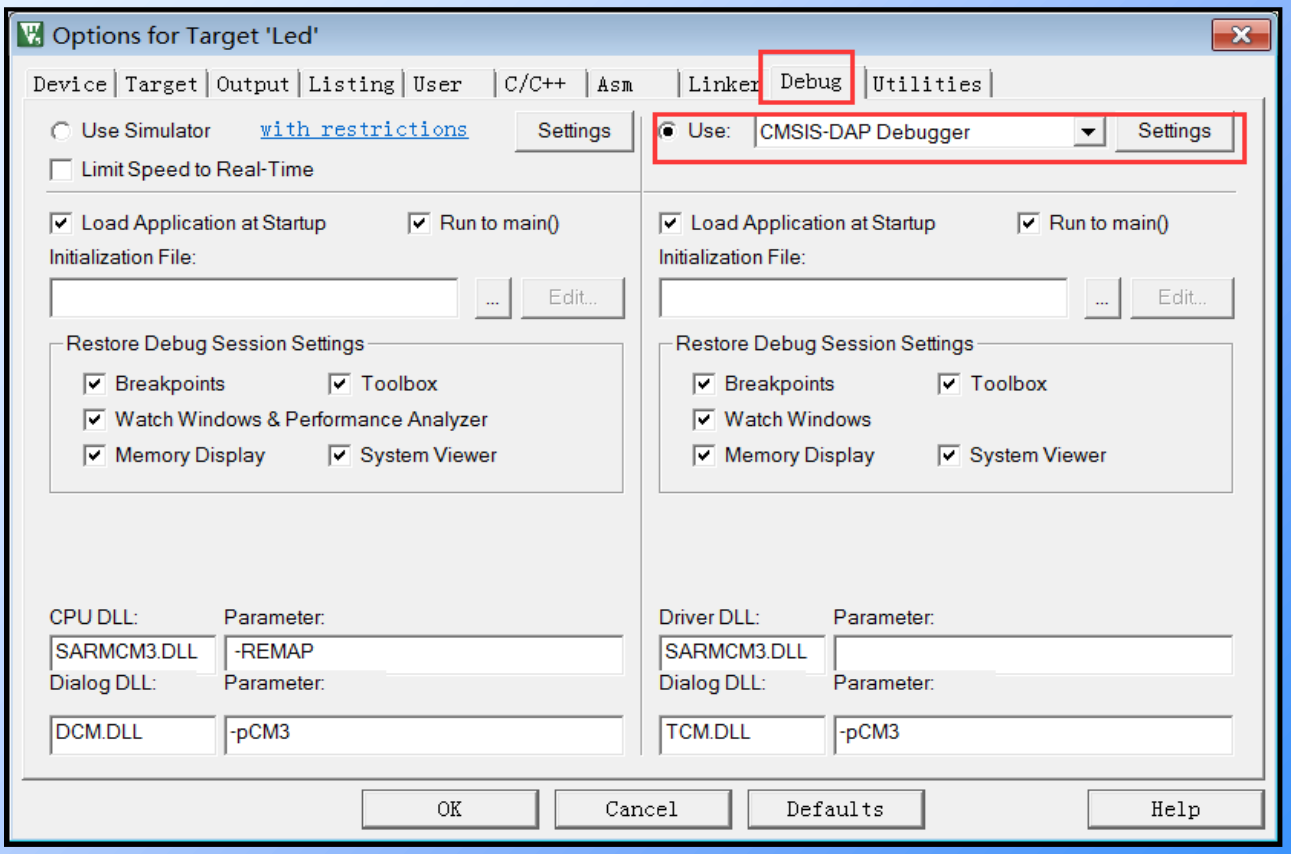
uvprojx文件就是我们平时双击打开的工程文件，它记录了整个工程的结构，如芯片类型、工程包含了哪些源文件等内容：



MDK的编译过程及文件类型全解

2.uvprojx文件

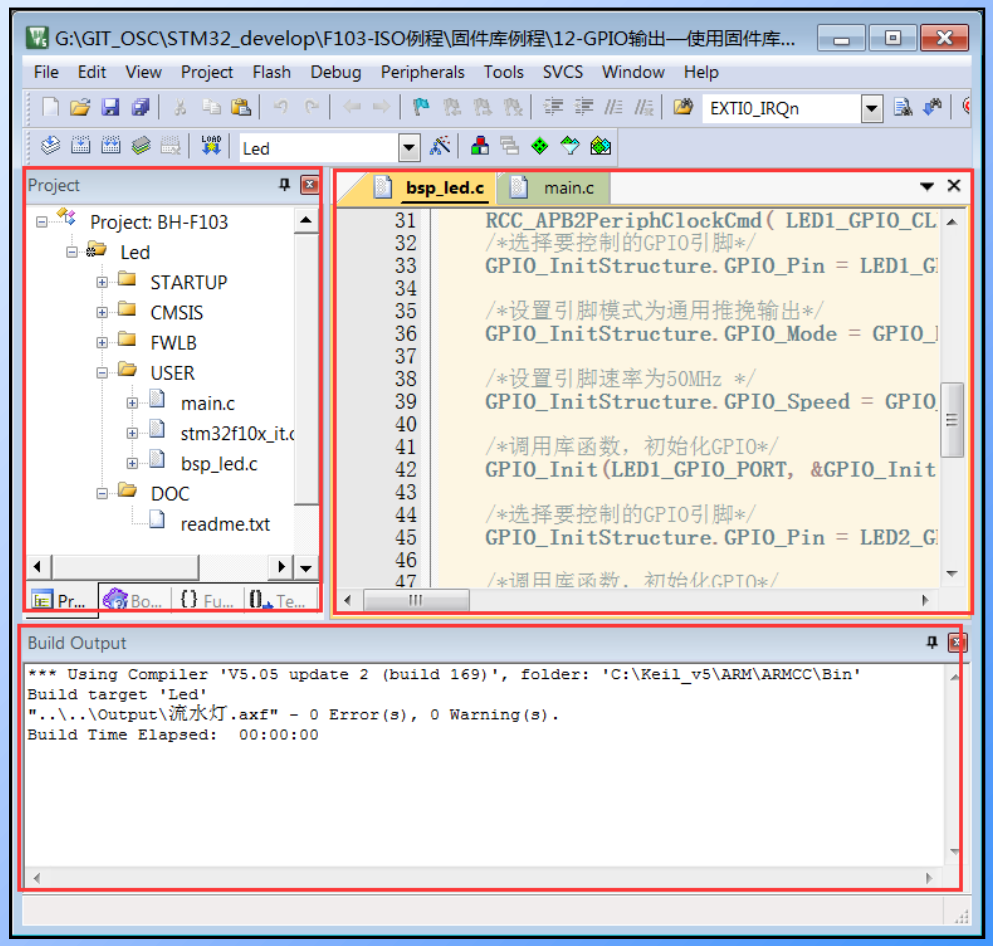
uvoptx文件记录了工程的配置选项，如下载器的类型、变量跟踪配置、断点位置以及当前已打开的文件等等：



MDK的编译过程及文件类型全解

3. uvguix文件

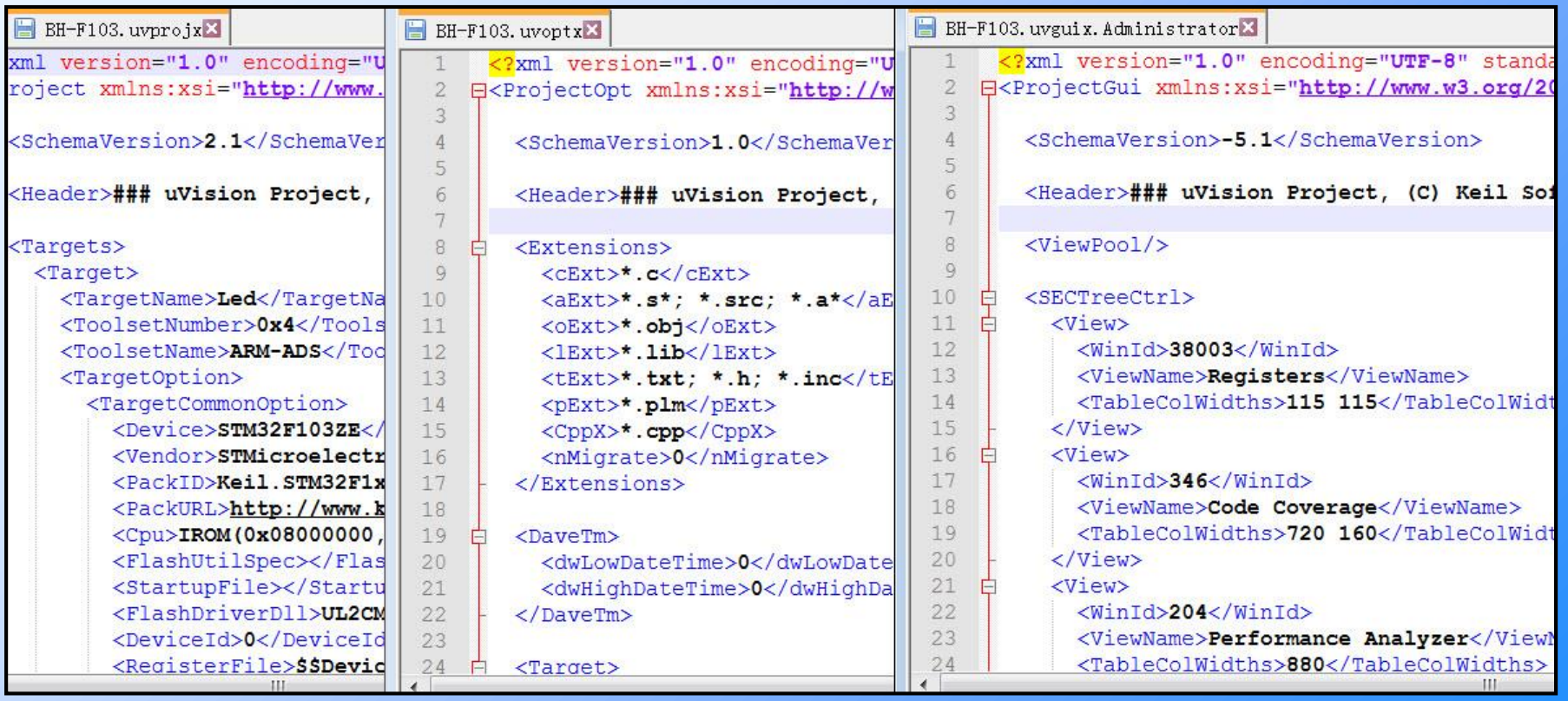
uvguix文件记录了MDK软件的GUI布局，如代码编辑区窗口的大小、编译输出提示窗口的位置等等。



MDK的编译过程及文件类型全解

uvprojx、uvoptx及uvguix都是使用XML格式记录的文件，若使用记事本打开可以看到XML代码。

而当使用MDK软件打开时，它根据这些文件的XML记录加载工程的各种参数，使得我们每次重新打开工程时，都能恢复上一次的工作环境。



MDK的编译过程及文件类型全解

这些工程参数都是当MDK正常退出时才会被写入保存，所以若MDK错误退出时(如使用Windows的任务管理器强制关闭)，工程配置参数的最新更改是不会被记录的，重新打开工程时要再次配置。

根据这几个文件的记录类型，可以知道uvprojx文件是最重要的，删掉它我们就无法再正常打开工程了，而uvoptx及uvguix文件并不是必须的，可以删除，重新使用MDK打开uvprojx工程文件后，会以默认参数重新创建uvoptx及uvguix文件。(所以当使用Git/SVN等代码管理的时候，往往只保留uvprojx文件)

MDK的编译过程及文件类型全解

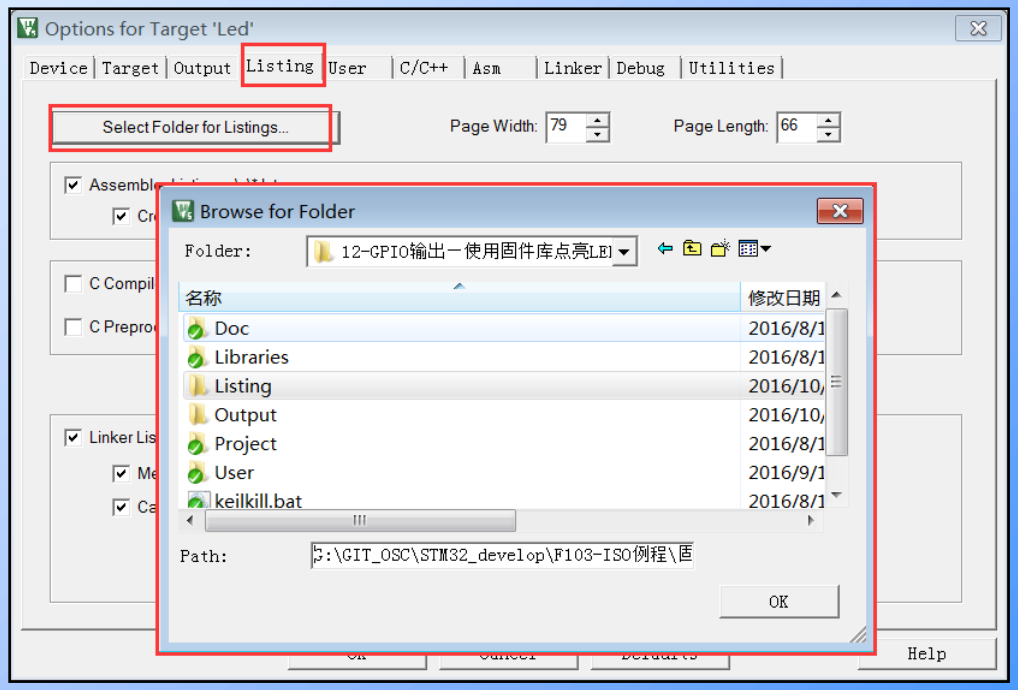
源文件

源文件是工程中我们最熟悉的内容了，它们就是我们编写的各种源代码，MDK支持c、cpp、h、s、inc类型的源代码文件，其中c、cpp分别是c/c++语言的源代码，h是它们的头文件，s是汇编文件，inc是汇编文件的头文件，可使用“$include”语法包含。编译器根据工程中的源文件最终生成机器码。

MDK的编译过程及文件类型全解

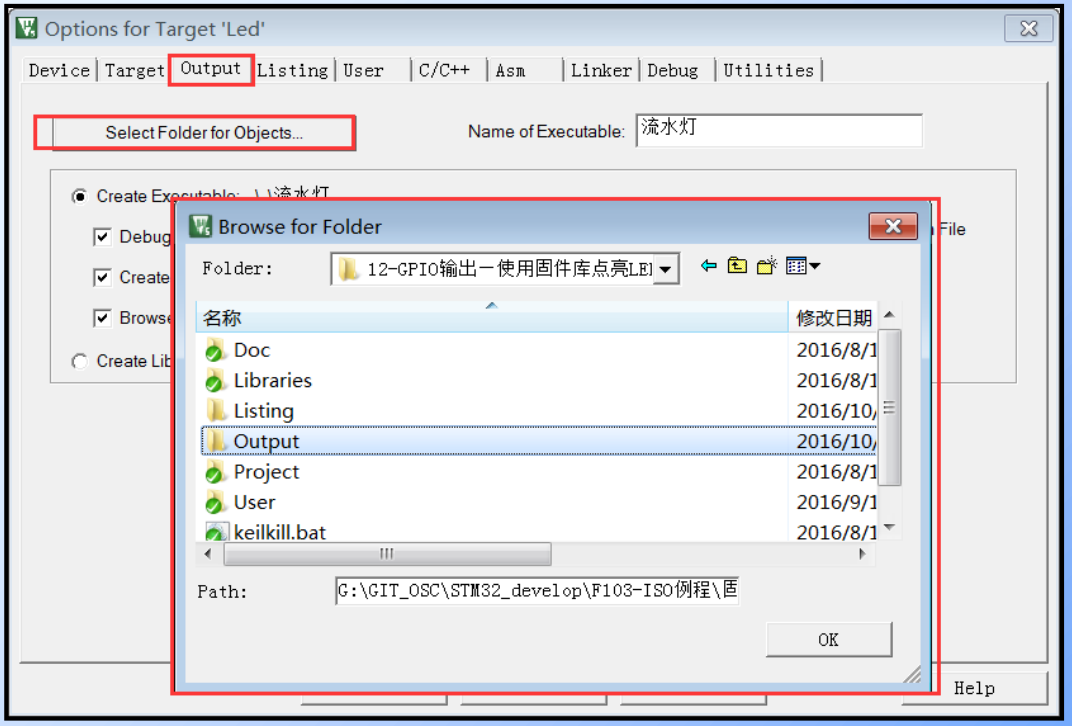
Output目录下生成的文件

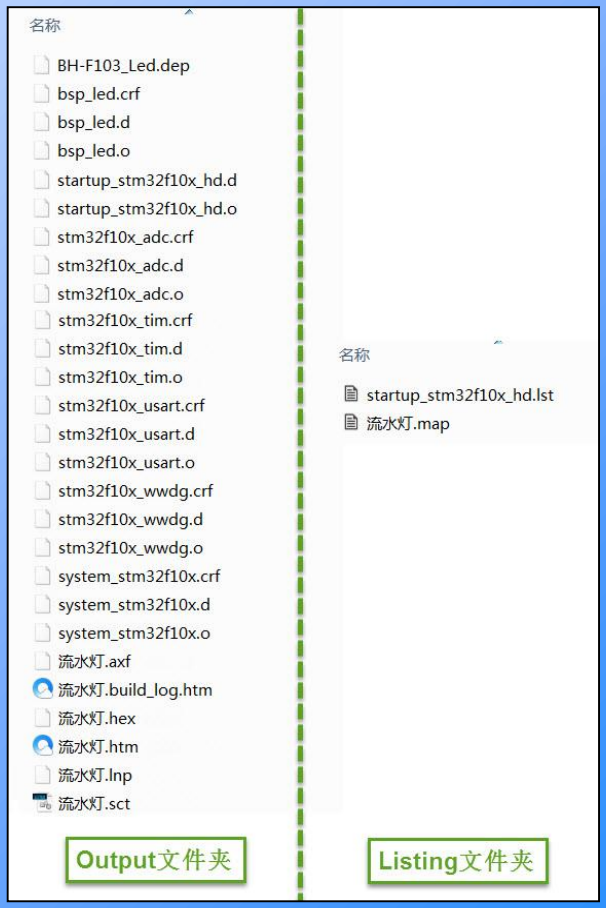
点击MDK中的编译按钮，它会根据工程的配置及工程中的源文件输出各种对象和列表文件，在工程的“Options for Targe->Output->Select Folder for Objects”和“Options for Targe->Listing->Select Folder for Listings”选项配置它们的输出路径：



MDK的编译过程及文件类型全解

Output目录下生成的文件

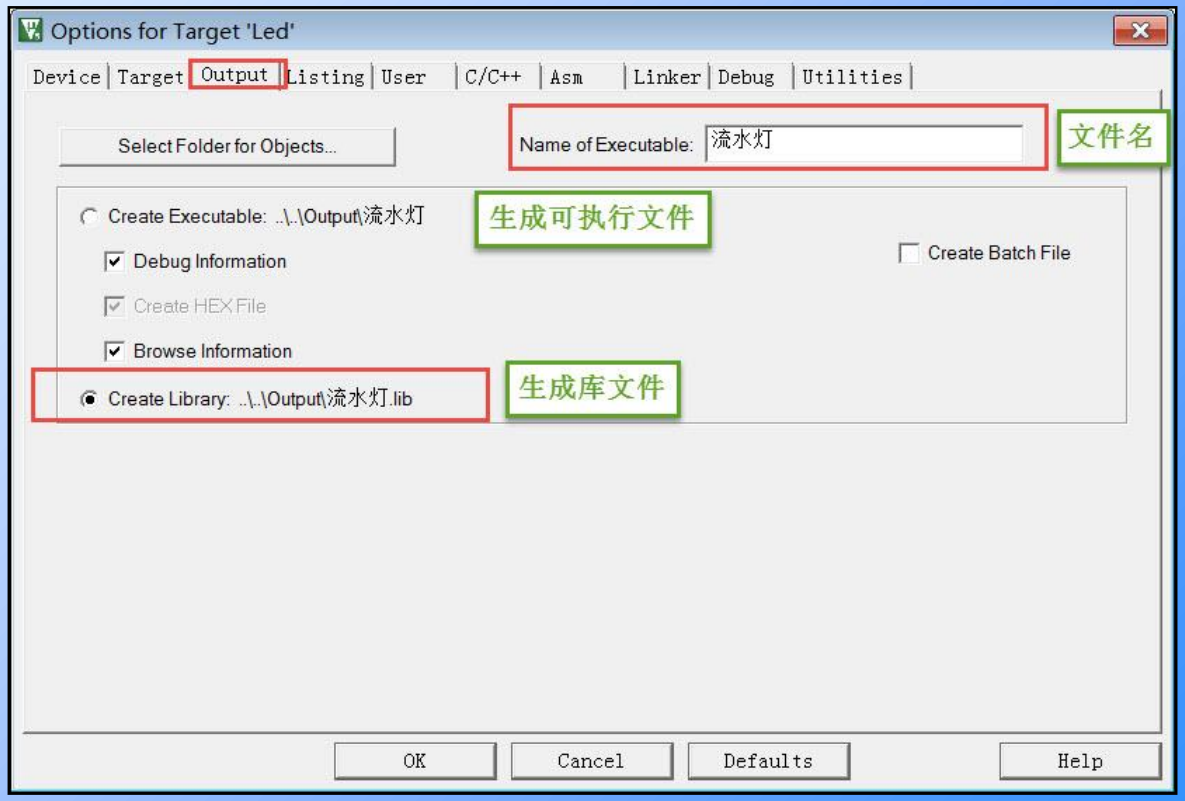




MDK的编译过程及文件类型全解

1. lib库文件

在某些场合下可能不希望提供给第三方一个可用的代码库，但不希望对方看到源码，这个时候我们就可以把工程生成lib文件(Library file)提供给对方，在MDK中可配置“Options for Target->Create Library”选项把工程编译成库文件：



MDK的编译过程及文件类型全解

1. lib库文件

工程中生成可执行文件或库文件只能二选一，默认编译是生成可执行文件的，可执行文件即我们下载到芯片上直接运行的机器码。

得到生成的*.lib文件后，可把它像C文件一样添加到其它工程中，并在该工程调用lib提供的函数接口，除了不能看到*.lib文件的源码，在应用方面它跟C源文件没有区别。

MDK的编译过程及文件类型全解

2.dep、d依赖文件

.dep和.d文件(Dependency file)记录的是工程或其它文件的依赖，主要记录了引用的头文件路径，其中*.dep是整个工程的依赖，它以工程名命名，而*.d是单个源文件的依赖，它们以对应的源文件名命名。这些记录使用文本格式存储，我们可直接使用记事本打开：

BH-F103_Led.dep | bsp_led.d

```
1  Dependencies for Project 'BH-F103', Target 'Led': (DO NOT MODIFY !)
2  F (..\..\Libraries\CMSIS\startup\startup_stm32f10x_hd.s)(0x57B3E0C6)(--cpu Cortex-M3 -g --apcs=inte
3
4  -I G:\GIT_OSC\STM32_develop\F103-ISO例程\固件库例程\MDK编译过程及文件全解\MDK文件详解-GPIO输出-使用
5
6  -I C:\Keil_v5\ARM\PACK\Keil\STM32F1xx_DFP\1.1.0\Device\Include
7
8  -I C:\Keil_v5\ARM\CMSIS\Include
9
10 --pd "__UVISION_VERSION SETA 515" --pd "STM32F10X_HD SETA 1"
11
12 --list ..\..\listing\startup_stm32f10x_hd.lst --xref -o ..\..\output\startup_stm32f10x_hd.o --deper
```

BH-F103_Led.dep | bsp_led.d

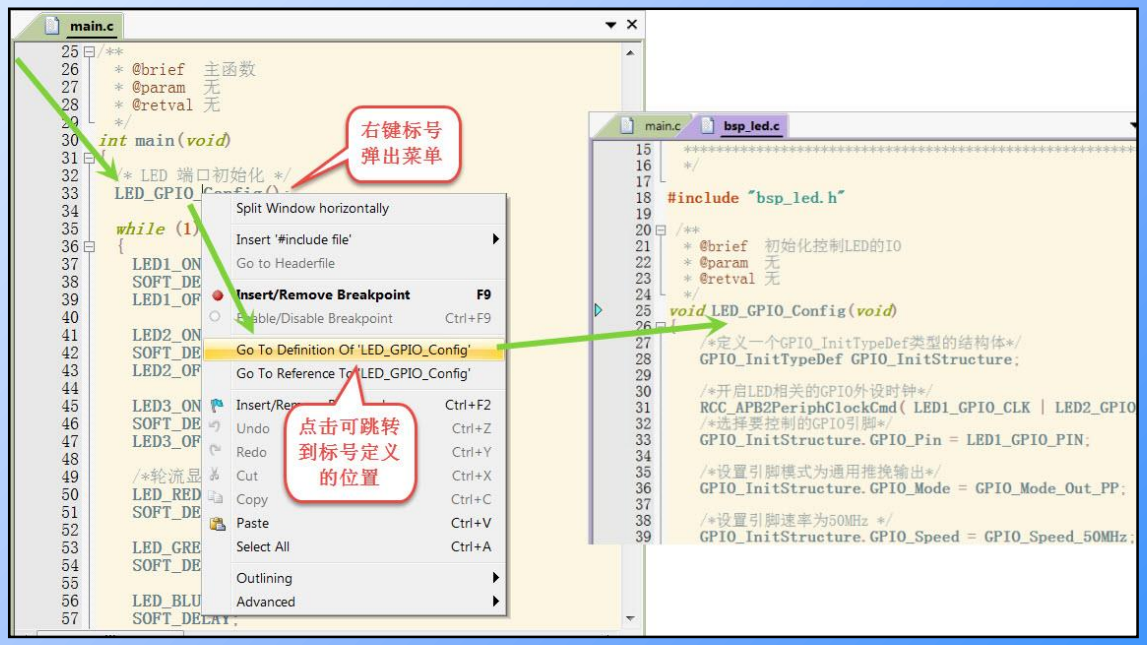
```
1  ..\..\output\bsp_led.o: ..\..\User\Led\bsp_led.c
2  ..\..\output\bsp_led.o: ..\..\User\Led\bsp_led.h
3  ..\..\output\bsp_led.o: ..\..\Libraries\CMSIS\stm32f10x.h
4  ..\..\output\bsp_led.o: ..\..\Libraries\CMSIS\core_cm3.h
5  ..\..\output\bsp_led.o: C:\Keil_v5\ARM\ARMCC\Bin\..\include\stdint.h
6  ..\..\output\bsp_led.o: ..\..\Libraries\CMSIS\system_stm32f10x.h
7  ..\..\output\bsp_led.o: ..\..\User\stm32f10x_conf.h
8  ..\..\output\bsp_led.o: ..\..\Libraries\FWlib\inc\stm32f10x_adc.h
9  ..\..\output\bsp_led.o: ..\..\Libraries\CMSIS\stm32f10x.h
10 ..\..\output\bsp_led.o: ..\..\Libraries\FWlib\inc\stm32f10x_bkp.h
11 ..\..\output\bsp_led.o: ..\..\Libraries\FWlib\inc\stm32f10x_can.h
12 ..\..\output\bsp_led.o: ..\..\Libraries\FWlib\inc\stm32f10x_cec.h
```

MDK的编译过程及文件类型全解

3.crf交叉引用文件

*.crf是交叉引用文件(Cross-Reference file)，它主要包含了浏览信息(browse information)，即源代码中的宏定义、变量及函数的定义和声明的位置。

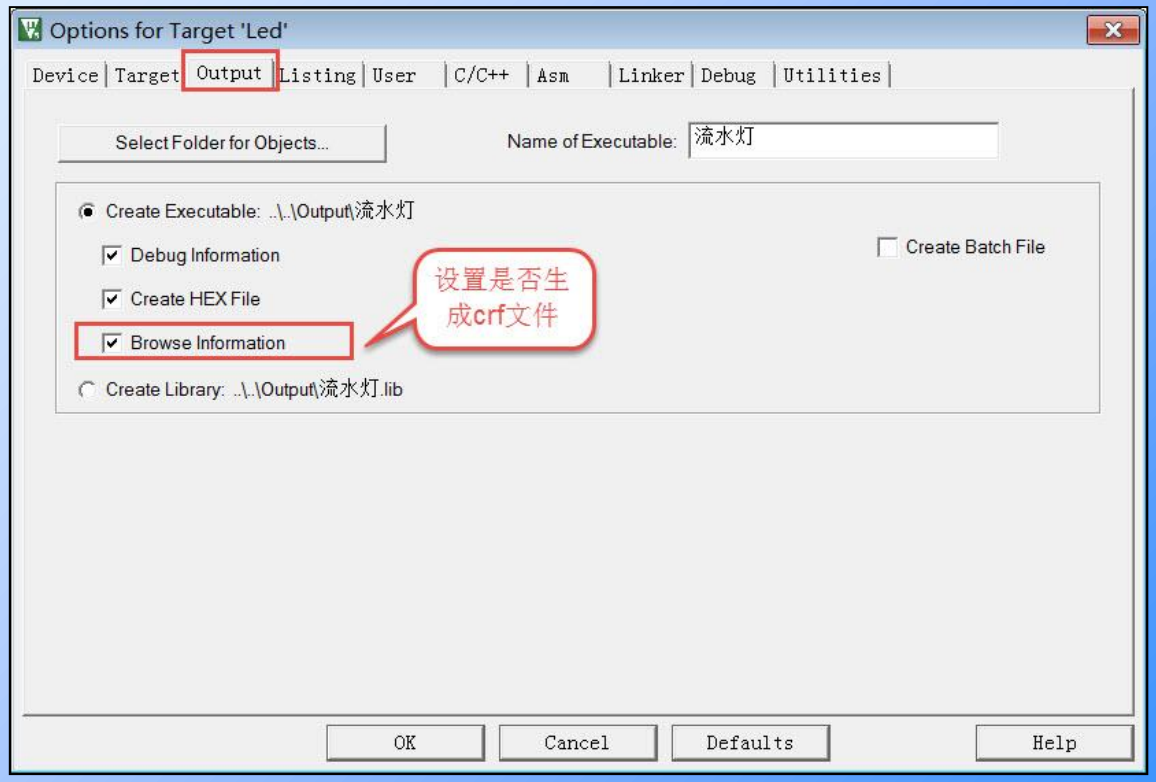
我们在代码编辑器中点击“Go To Definition Of ‘xxxx’”可实现浏览跳转，跳转的时候，MDK就是通过*.crf文件查找出跳转位置的。



MDK的编译过程及文件类型全解

3.crf交叉引用文件

通过配置MDK中的“Option for Target->Output->Browse Information”选项可以设置编译时是否生成浏览信息，只有勾选该选项并编译后，才能实现上面的浏览跳转功能。



THANKS

论坛： www.firebbs.cn

淘宝：firestm32.taobao.com

扫描进入淘宝店铺