

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.chuxue123.com



扫描进入淘宝店铺

主讲内容



01

编译过程

02

程序的组成、存储与运行

03

编译工具链

04

MDK工程的文件类型

05

实验：自动分配变量到外部SDRAM

06

**实验：优先使用内部SRAM并
分配堆到SDRAM**

MDK的编译过程及文件类型全解



编译工具链

在前面编译过程中，MDK调用了各种编译工具，平时我们直接配置MDK，不需要学习如何使用它们，但了解它们是非常有好处的。

例如，若希望使用MDK编译生成bin文件的，需要在MDK中输入指令控制fromelf工具；在本章后面讲解AXF及O文件的时候，需要利用fromelf工具查看其文件信息，这都是无法直接通过MDK做到的。

关于这些工具链的说明，在MDK的帮助手册《ARM Development Tools》都有详细讲解，点击MDK界面的“help->uVision Help”菜单可打开该文件。

MDK的编译过程及文件类型全解



设置环境变量

调用这些编译工具，需要用到Windows的“命令行提示符工具”，为了让命令行方便地找到这些工具，我们先把工具链的目录添加到系统的环境中。

查看本机工具链所在的具体目录可根据上一小节讲解的工程编译提示输出信息中找到，如本机的路径为“D:\work\keil5\ARM\ARMCC\bin”。

1. 添加路径到PATH环境变量

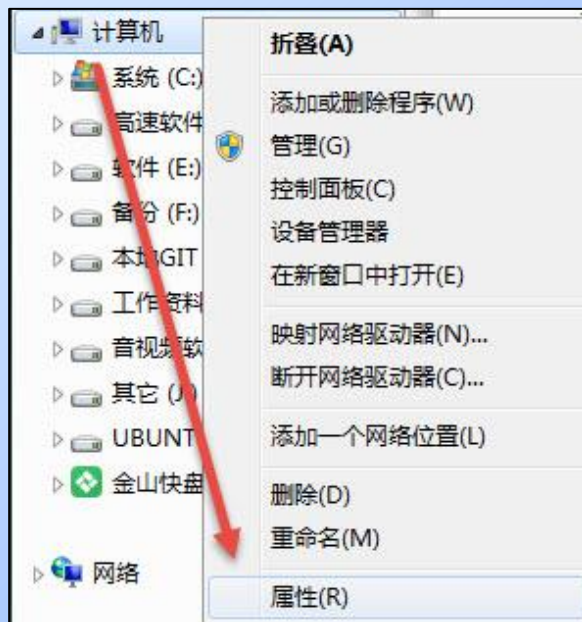
本章以Win7系统为例添加工具链的路径到PATH环境变量，其它系统是类似的。

MDK的编译过程及文件类型全解



1. 添加路径到PATH环境变量

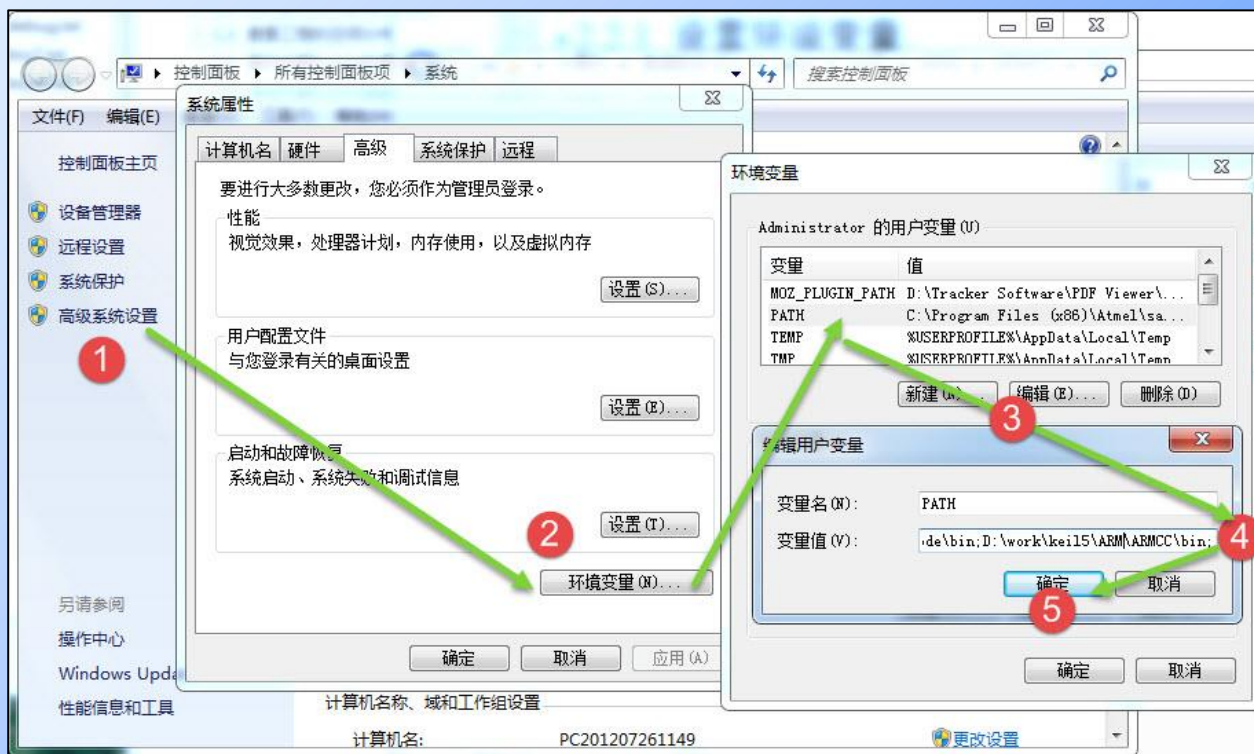
- 右键电脑系统的“计算机图标”，在弹出的菜单中选择“属性”。



MDK的编译过程及文件类型全解

1. 添加路径到PATH环境变量

- 在弹出的属性页面依次点击“高级系统设置”->“环境变量”，在用户变量一栏中找到名为“PATH”的变量，若没有该变量，则新建一个。编辑“PATH”变量，在它的变量值中输入工具链的路径，如本机的是“;D:\work\keil5\ARM\ARMCC\bin”，注意要使用“分号;”让它与其它路径分隔开，输入完毕后依次点确定。

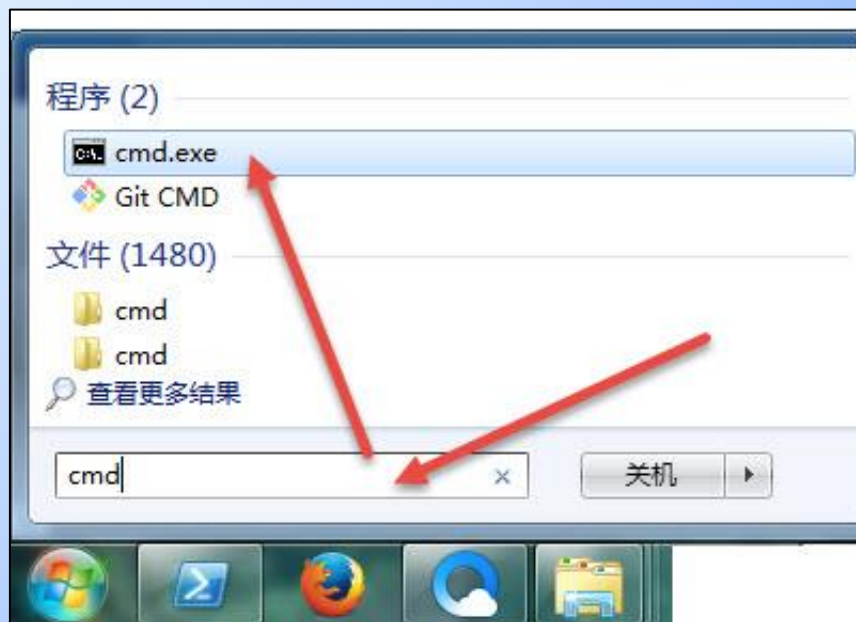


MDK的编译过程及文件类型全解



1. 添加路径到PATH环境变量

- 打开Windows的命令行，点击系统的“开始菜单”，在搜索框输入“cmd”，在搜索结果中点击“cmd.exe”即可打开命令行。



MDK的编译过程及文件类型全解



1. 添加路径到PATH环境变量

- 在弹出的命令行窗口中输入“`fromelf`”回车，若窗口打印出`formelf`的帮助说明，那么路径正常，就可以开始后面的工作了；若提示“不是内部名外部命令，也不是可运行的程序...”信息，说明路径不对，请重新配置环境变量，并确认该工作目录下有编译工具链。

这个添加环境变量的过程本质就是让命令行通过“**PATH**”路径找到“`fromelf.exe`”程序运行，默认运行“`fromelf.exe`”时它会输出自己的帮助信息，这就是工具链的调用过程，MDK本质上也是如此调用工具链的，只是它集成为**GUI**，相对于命令行对用户更友好，毕竟上述配置环境变量的过程已经让新手烦躁了。

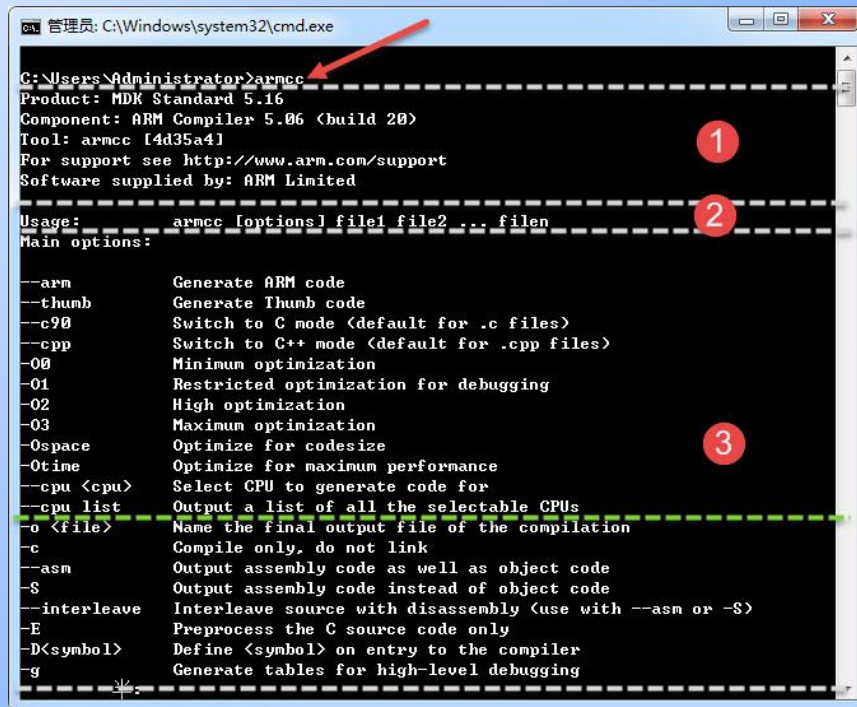
MDK的编译过程及文件类型全解

armcc、armasm及armlink

接下来我们看看各个工具链的具体用法，主要以armcc为例。

1. armcc

armcc用于把c/c++文件编译成ARM指令代码，编译后会输出ELF格式的O文件(对象、目标文件)，在命令行中输入“armcc”回车可调用该工具，它会打印帮助说明：



```
管理员: C:\Windows\system32\cmd.exe

C:\Users\Administrator>armcc
Product: MDK Standard 5.16
Component: ARM Compiler 5.06 (build 20)
Tool: armcc [4d35a4]
For support see http://www.arm.com/support
Software supplied by: ARM Limited

-----
Usage:      armcc [options] file1 file2 ... fileN
Main options:
--arm       Generate ARM code
--thumb     Generate Thumb code
--c90       Switch to C mode (default for .c files)
--cpp       Switch to C++ mode (default for .cpp files)
-O0         Minimum optimization
-O1         Restricted optimization for debugging
-O2         High optimization
-O3         Maximum optimization
-Ospace     Optimize for codesize
-Otime     Optimize for maximum performance
--cpu <cpu> Select CPU to generate code for
--cpu list  Output a list of all the selectable CPUs
-o <file>   Name the final output file of the compilation
-c          Compile only, do not link
--asm       Output assembly code as well as object code
-S          Output assembly code instead of object code
--interleave Interleave source with disassembly (use with --asm or -S)
-E          Preprocess the C source code only
-D<symbol>  Define <symbol> on entry to the compiler
-g          Generate tables for high-level debugging
-----
```

MDK的编译过程及文件类型全解



1. armcc

帮助提示中分三部分，第一部分是armcc版本信息，第二部分是命令的用法，第三部分是主要命令选项。

根据命令用法：`armcc [options] file1 file2 ... filen`，在[option]位置可输入下面的“`--arm`”、“`--cpu list`”选项，若选项带文件输入，则把文件名填充在file1 file2...的位置，这些文件一般是c/c++文件。

MDK的编译过程及文件类型全解



1. armcc

例如根据它的帮助说明，“--cpu list”可列出编译器支持的所有cpu，我们在命令行中输入“armcc --cpu list”，可查看cpu列表。

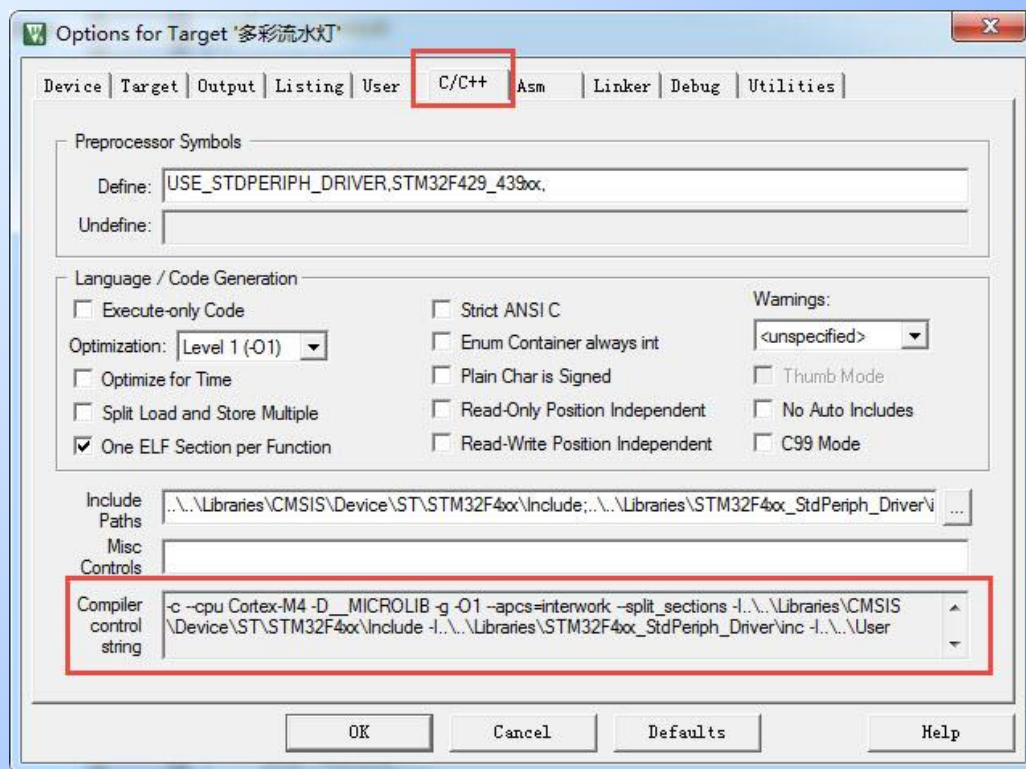
```
管理员: C:\Windows\system32\cmd.exe
C:\Users\Administrator>armcc --cpu list
The following arguments to option 'cpu' can be selected:
--cpu=ARM7EJ-S
--cpu=ARM7TDMI
--cpu=ARM720T
--cpu=ARM7TDMI-S
--cpu=ARM9TDMI
--cpu=ARM920T
--cpu=ARM922T
--cpu=ARM9E-S
--cpu=ARM926EJ-S
--cpu=ARM946E-S
--cpu=ARM966E-S
--cpu=Cortex-M0
--cpu=Cortex-M0plus
--cpu=SC000
--cpu=Cortex-M1
--cpu=Cortex-M1.os_extension
--cpu=Cortex-M1.no_os_extension
--cpu=Cortex-M3
--cpu=Cortex-M3-rev0
--cpu=SC300
--cpu=Cortex-M4
--cpu=Cortex-M4.fp.sp
--cpu=Cortex-M7
--cpu=Cortex-M7.fp.sp
--cpu=Cortex-M7.fp.dp
--cpu=Cortex-R4
--cpu=Cortex-R4F
```

MDK的编译过程及文件类型全解



1. armcc

打开MDK的Options for Target->c/c++菜单，可看到MDK对编译器的控制命令，



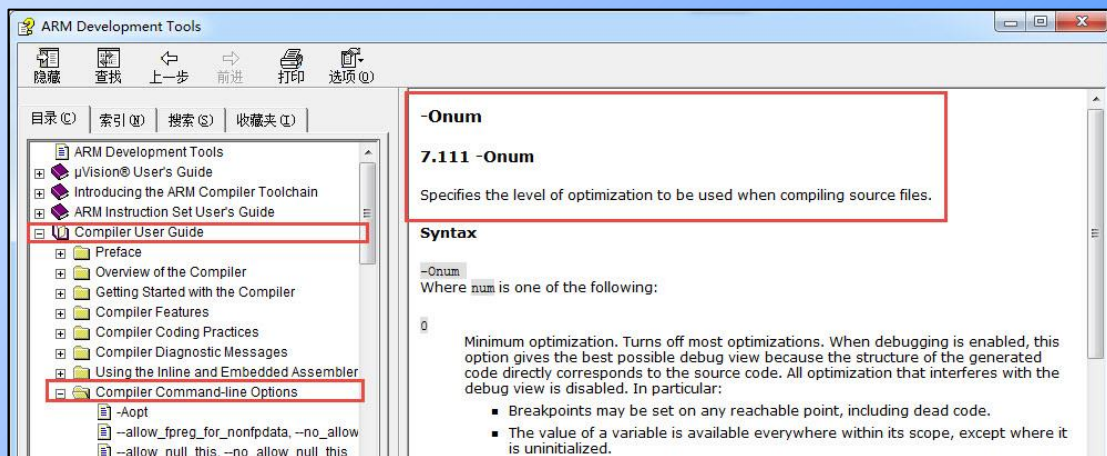
MDK的编译过程及文件类型全解



1. armcc

从该图中的命令可看到，它调用了-c、-cpu -D -g -O1等编译选项，当我们修改MDK的编译配置时，可看到该控制命令也会有相应的变化。然而我们无法在该编译选项框中输入命令，只能通过MDK提供的选项修改。

了解这些，我们就可以查询具体的MDK编译选项的具体信息了，如c/c++选项中的“Optimization: Leve 1 (-O1)”是什么功能呢？首先可了解到它是“-O”命令，命令后还带个数字，查看MDK的帮助手册，在armcc编译器说明章节，可详细了解。



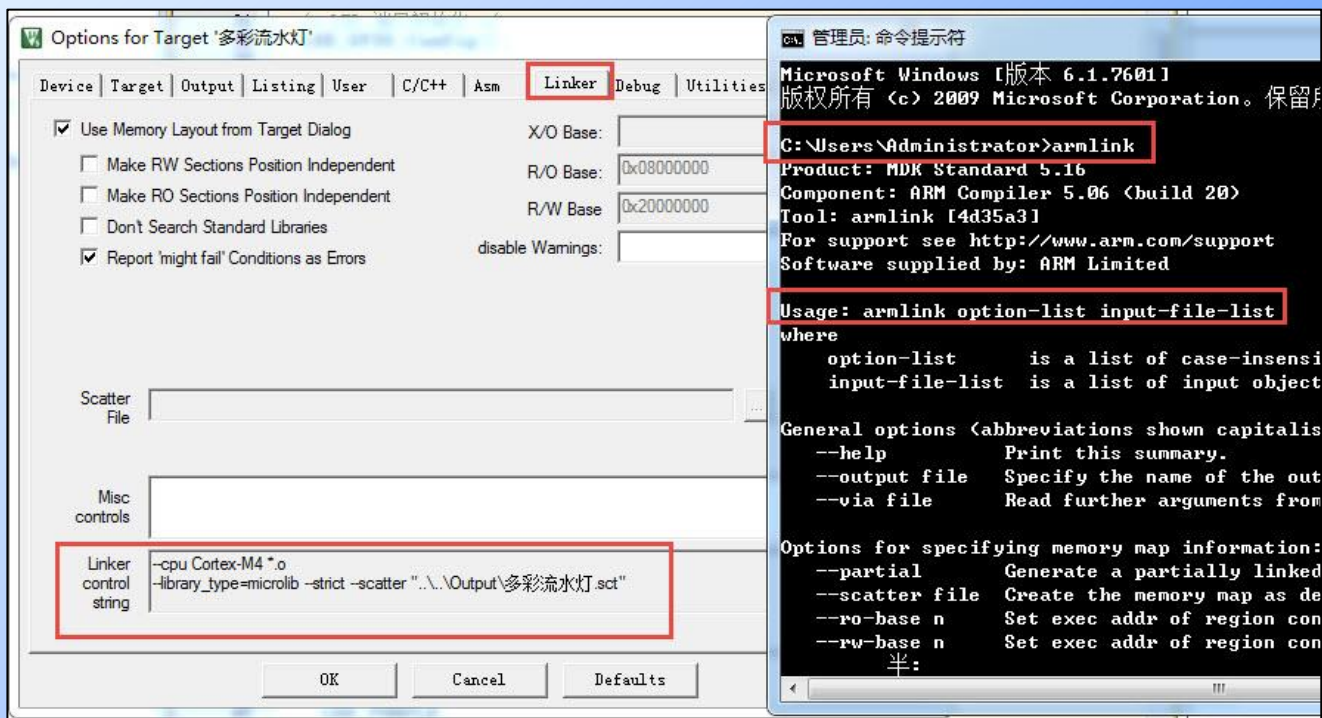
利用MDK，我们一般不需要自己调用armcc工具，但经过这样的过程就会对MDK有更深入的认识，面对它的各种编译选项，就不会那么头疼了。

MDK的编译过程及文件类型全解



3. armlink

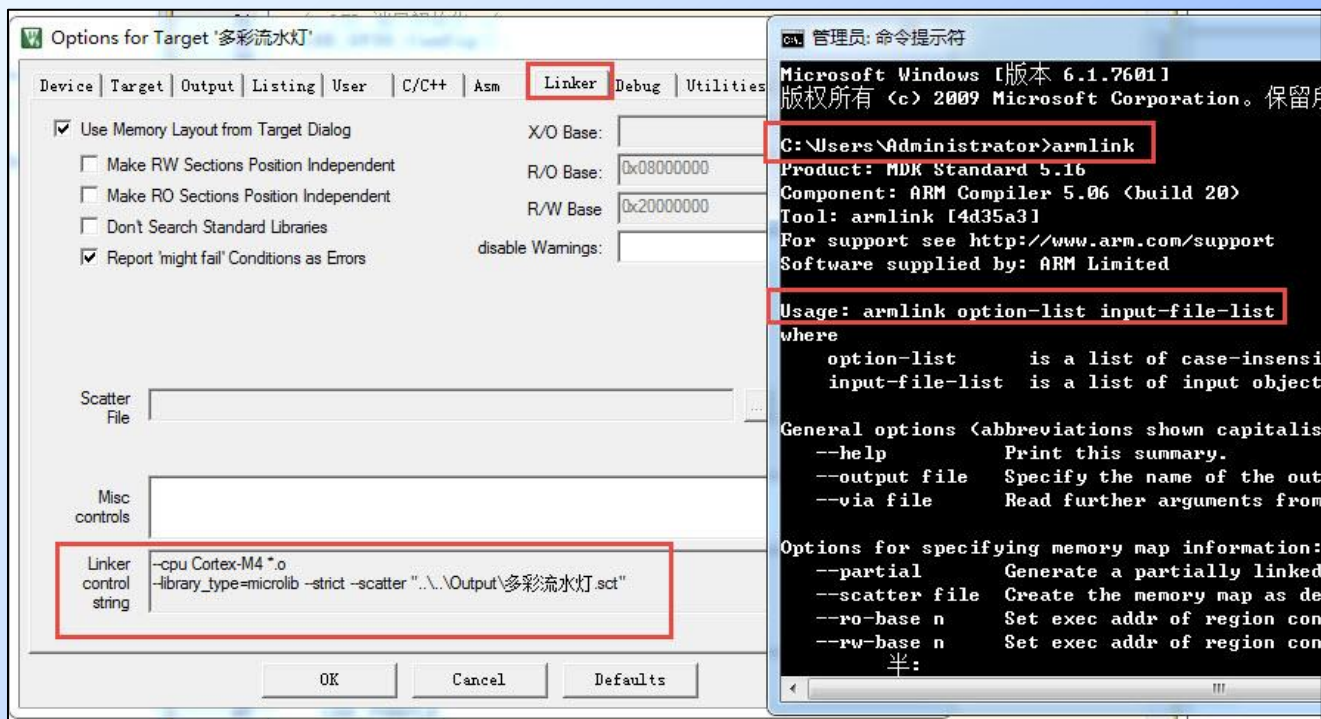
armlink是链接器，它把各个O文件链接组合在一起生成ELF格式的AXF文件，AXF文件是可执行的，下载器把该文件中的指令代码下载到芯片后，该芯片就能运行程序了；利用armlink还可以控制程序存储到指定的ROM或RAM地址。在MDK中可在“Option for Target->Linker”页面配置armlink选项：



MDK的编译过程及文件类型全解



3. armlink



链接器默认是根据芯片类型的存储器分布来生成程序的，该存储器分布被记录在工程里的sct后缀的文件中，有特殊需要的话可自行编辑该文件，改变链接器的链接方式。

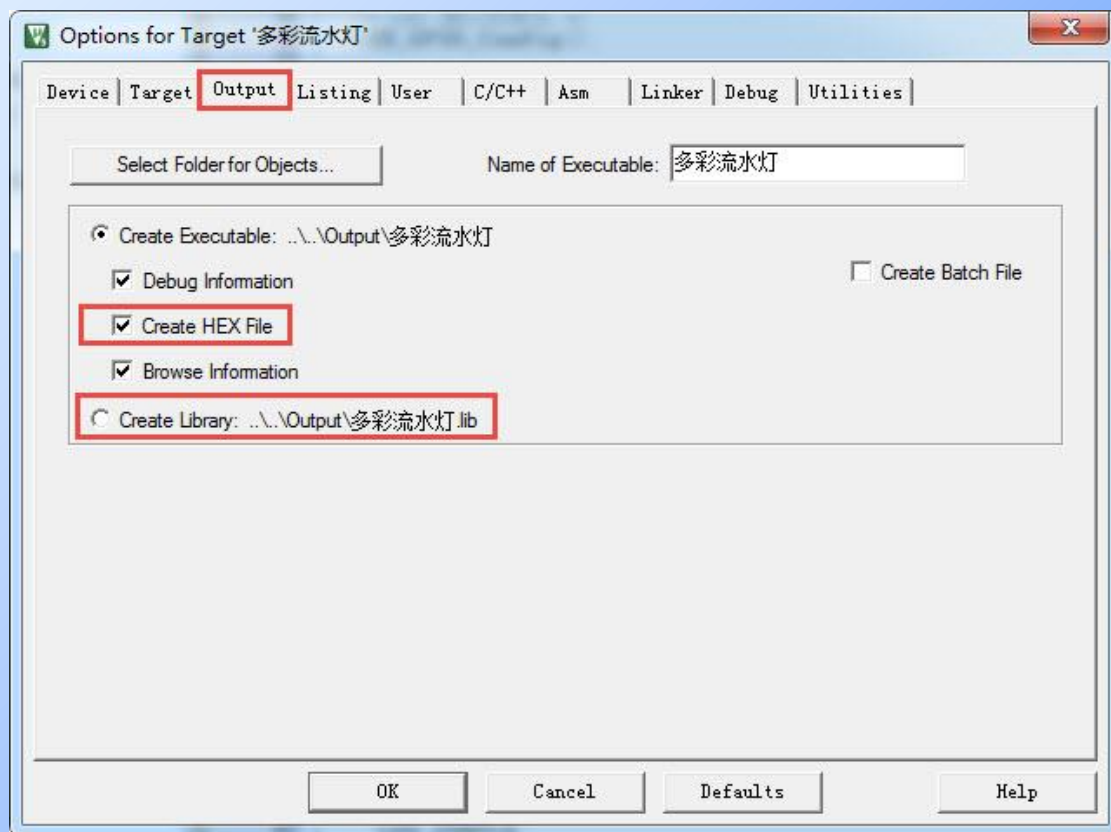
MDK的编译过程及文件类型全解



armar、fromelf及用户指令

armar工具用于把工程打包成库文件，fromelf可根据axf文件生成hex、bin文件，hex和bin文件是大多数下载器支持的下载文件格式。

在MDK中，针对armar和fromelf工具的选项几乎没有，仅集成了生成HEX或Lib的选项。

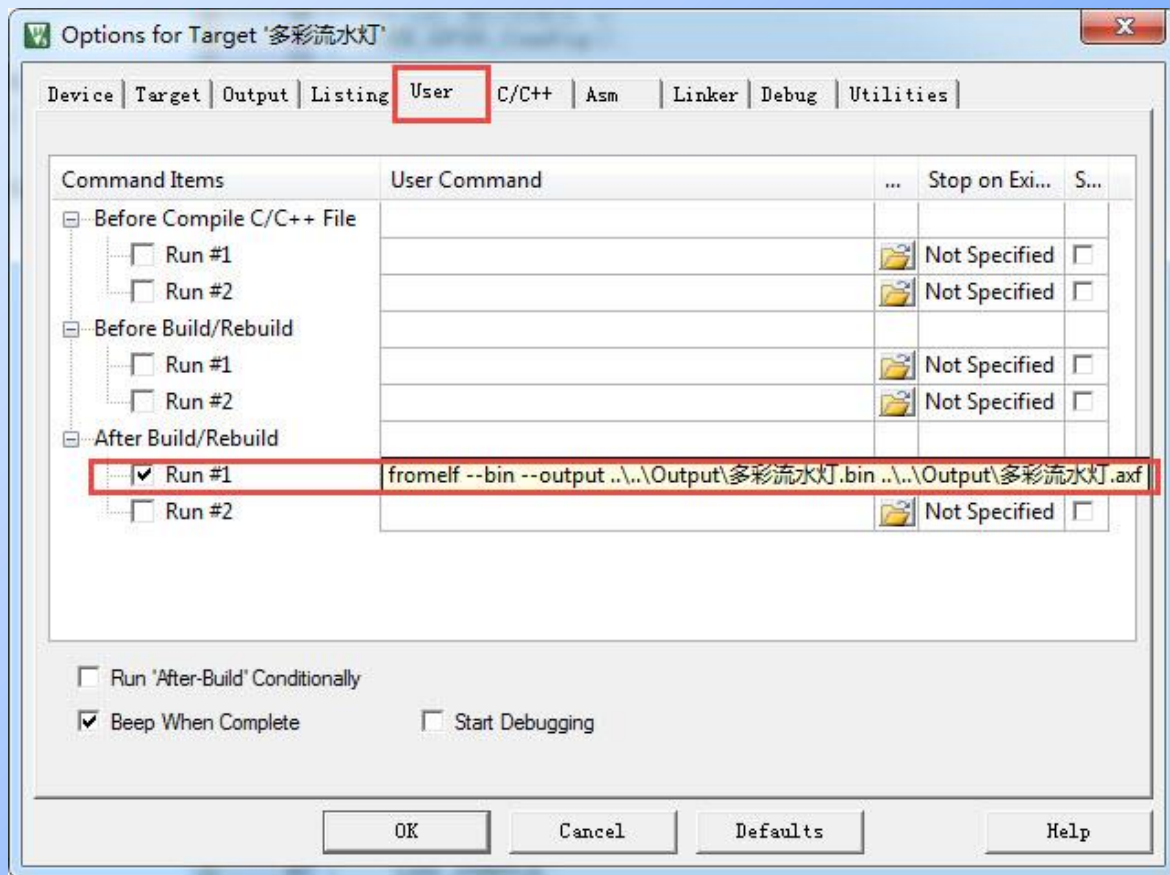


MDK的编译过程及文件类型全解



armar、fromelf及用户指令

例如如果想利用fromelf生成bin文件，可以在MDK的“Option for Target->User”页中添加调用fromelf的指令：



MDK的编译过程及文件类型全解



armar、fromelf及用户指令

在User配置页面中，提供了三种类型的用户指令输入框，在不同组的框输入指令，可控制指令的执行时间，分别是编译前(Before Compile c/c++ file)、构建前(Before Build/Rebuild)及构建后(After Build/Rebuild)执行。这些指令并没有限制必须是arm的编译工具链，例如如果您自己编写了python脚本，也可以在这里输入用户指令执行该脚本。

图中的生成bin文件指令调用了fromelf工具，紧跟后面的是工具的选项及输出文件名、输入文件名。由于fromelf是根据axf文件生成bin的，而axf文件又是构建(build)工程后才生成，所以我们把该指令放到“After Build/Rebuild”一栏。

零死角玩转STM32



THANKS

论坛：www.chuxue123.com

淘宝：firestm32.taobao.com



扫描进入淘宝店铺