

# 零死角玩转STM32



## MDK的编译过程及文件类型全解

淘宝：[firestm32.taobao.com](http://firestm32.taobao.com)

论坛：[www.firebbs.cn](http://www.firebbs.cn)



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SRAM**

---

06

**实验：优先使用内部SRAM并  
分配堆到外部SRAM**

---

# MDK的编译过程及文件类型全解



## 5.hex文件及bin文件

若编译过程无误，即可把工程生成前面对应的\*.axf文件，而在MDK中使用下载器(DAP/JLINK/ULINK等)下载程序或仿真的时候，MDK调用的就是\*.axf文件，它解释该文件，然后控制下载器把\*.axf中的代码内容下载到STM32芯片对应的存储空间，然后复位后芯片就开始执行代码了。

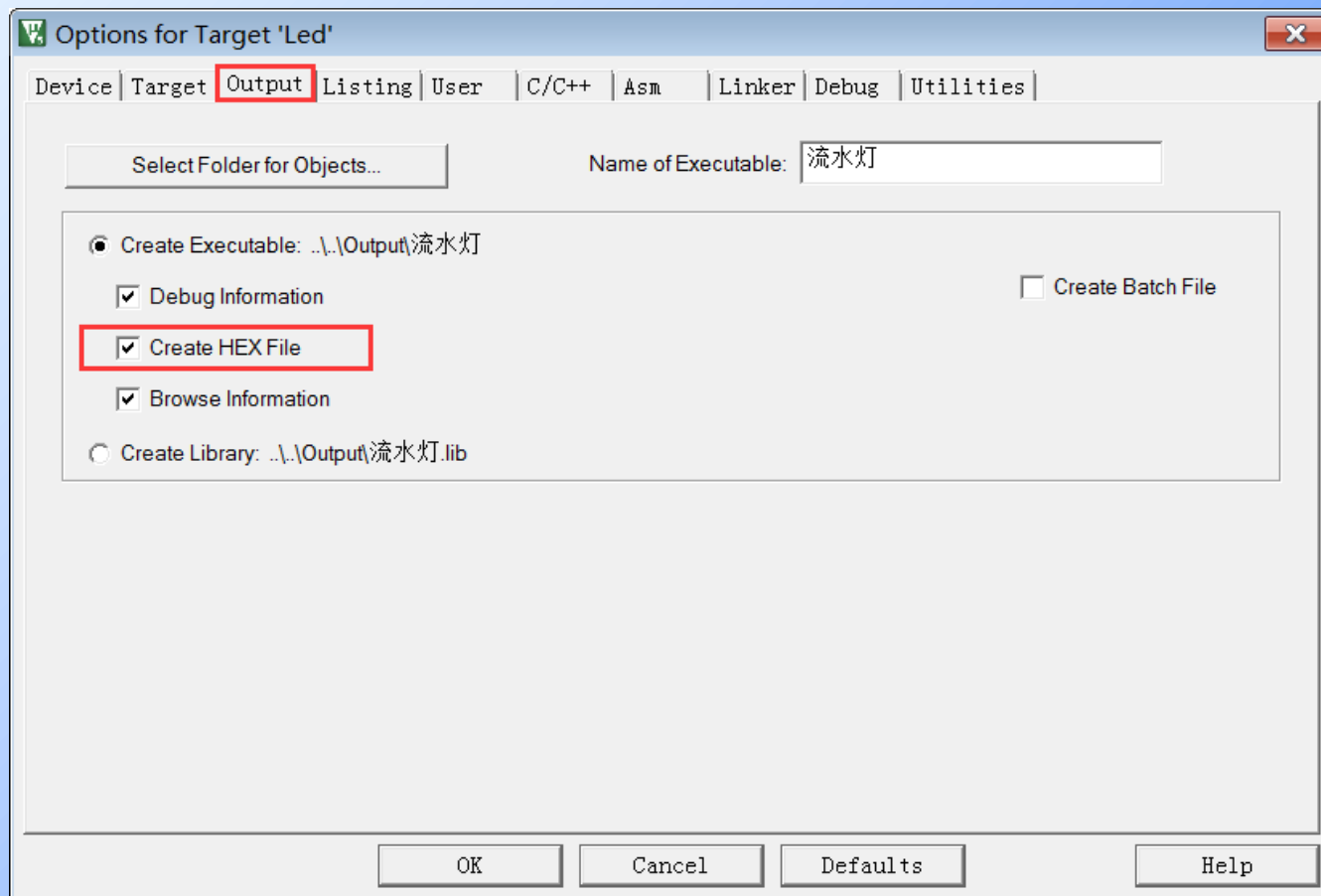
然而，脱离了MDK或IAR等工具，下载器就无法直接使用\*.axf文件下载代码了，它们一般仅支持hex和bin格式的代码数据文件。默认情况下MDK都不会生成hex及bin文件，需要配置工程选项或使用fromelf命令。

# MDK的编译过程及文件类型全解



## 生成hex文件

生成hex文件的配置比较简单，在“Options for Target->Output->Create Hex File”中勾选该选项，然后编译工程即可：



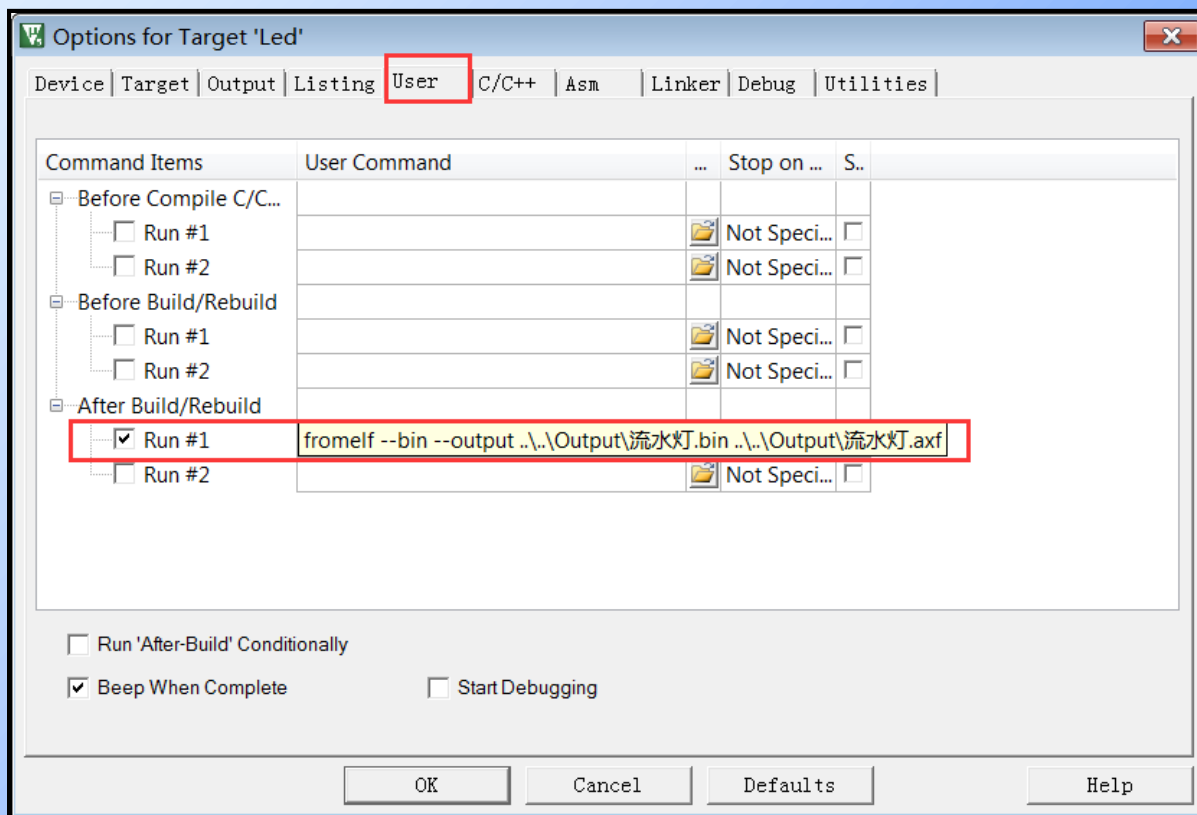
# MDK的编译过程及文件类型全解



## 生成bin文件

使用MDK生成bin文件需要使用fromelf命令，在MDK的“Options For Target->Users”中加入命令：

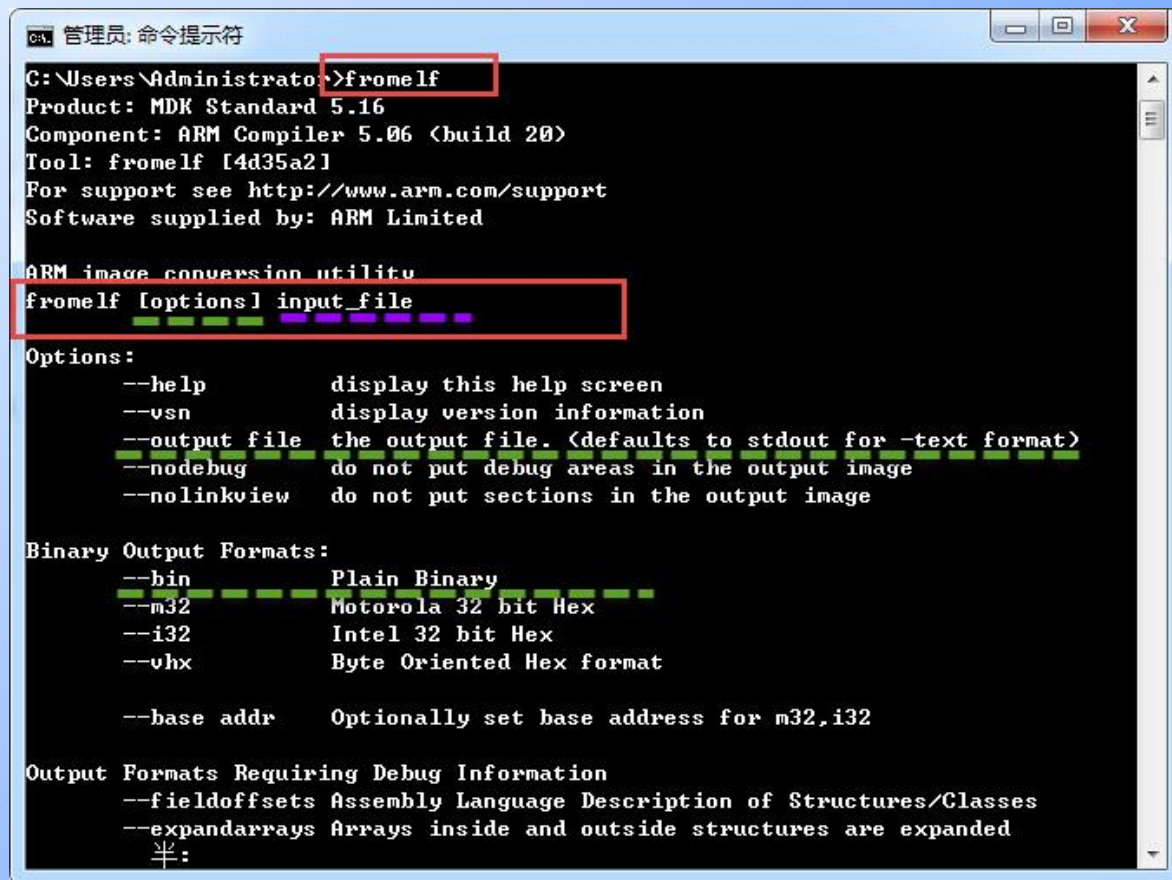
“fromelf --bin --output ..\..\Output\多彩流水灯.bin ..\..\Output\多彩流水灯.axf”



# MDK的编译过程及文件类型全解

## 生成bin文件

该指令是根据本机及工程的配置而写的，在不同的系统环境或不同的工程中，指令内容都不一样，需要理解它，才能为自己的工程定制指令，首先看看fromelf的帮助：



```
管理员: 命令提示符
C:\Users\Administrator>fromelf
Product: MDK Standard 5.16
Component: ARM Compiler 5.06 (build 20)
Tool: fromelf [4d35a21]
For support see http://www.arm.com/support
Software supplied by: ARM Limited

ARM image conversion utility
fromelf [options] input_file
-----

Options:
--help          display this help screen
--vsn           display version information
--output file   the output file. (defaults to stdout for -text format)
--nodebug       do not put debug areas in the output image
--nolinkview    do not put sections in the output image

Binary Output Formats:
--bin           Plain Binary
--m32           Motorola 32 bit Hex
--i32           Intel 32 bit Hex
--vhx          Byte Oriented Hex format

--base addr     Optionally set base address for m32,i32

Output Formats Requiring Debug Information
--fieldoffsets  Assembly Language Description of Structures/Classes
--expandarrays  Arrays inside and outside structures are expanded
半:
```



# MDK的编译过程及文件类型全解



## 生成bin文件

fromelf需要根据工程的\*.axf文件输入来转换得到bin文件，所以在命令的输入文件参数中要选择本工程对应的\*.axf文件，在MDK命令输入栏中，我们把fromelf指令放置在“After Build/Rebuild” (工程构建完成后执行)一栏也是基于这个考虑，这样设置后，工程构建完成生成了最新的\*.axf文件，MDK再执行fromelf指令，从而得到最新的bin文件。

置完成生成hex的选项或添加了生成bin的用户指令后，点击工程的编译(build)按钮，重新编译工程，成功后可看到如下输出，打开相应的目录即可找到文件。

### Build Output

```
*** Using Compiler 'V5.05 update 2 (build 169)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Build target 'Led'
After Build - User command #1: fromelf --bin --output ..\..\Output\流水灯.bin ..\..\Output\流水灯.axf
"..\..\Output\流水灯.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00
```

# MDK的编译过程及文件类型全解



## hex文件格式

hex是Intel公司制定的一种使用ASCII文本记录机器码或常量数据的文件格式，这种文件常常用来记录将要存储到ROM中的数据，绝大多数下载器支持该格式。

一个hex文件由多条记录组成，而每条记录由五个部分组成，格式形如“:llaaaatt[dd...]cc”，例如本“多彩流水灯”工程生成的hex文件前几条记录：

```
1  :020000040800F2
2  :10000000000400204501000829030008BF02000881
3  :10001000250300088D0100089D0400080000000071
4  :1000200000000000000000000000000004D03000878
5  :10003000910100080000000002B03000839040008AB
6  :100040005F0100085F0100085F0100085F01000810
```



# MDK的编译过程及文件类型全解



## hex文件格式

记录的各个部分介绍如下：

- “:”：每条记录的开头都使用冒号来表示一条记录的开始；
- **ll**：以16进制数表示这条记录的主体数据区的长度(即后面[**dd...**]的长度)；
- **aaaa**:表示这条记录中的内容应存放到FLASH中的起始地址；
- **tt**: 表示这条记录的类型，它包含中的各种类型；

tt的值	代表的类型
00	数据记录
01	本文件结束记录
02	扩展地址记录
04	扩展线性地址记录(表示后面的记录按个这地址递增)
05	表示一个线性地址记录的起始(只适用于ARM)

# MDK的编译过程及文件类型全解



## hex文件格式

- **dd:** 表示一个字节的数据，一条记录中可以有多字节数据，**ll**区表示了它有多少个字节的数据；
- **cc:** 表示本条记录的校验和，它是前面所有**16**进制数据 (除冒号外，两个为一组)的和对**256**取模运算的结果的补码。

# MDK的编译过程及文件类型全解



## hex文件格式

:020000040800F2

- 02: 表示这条记录数据区的长度为2字节;
- 0000: 表示这条记录要存储到的地址;
- 04: 表示这是一条扩展线性地址记录;
- 0800: 由于这是一条扩展线性地址记录, 所以这部分表示地址的高16位, 与前面的“0000”结合在一起, 表示要扩展的线性地址为“0x0800 0000”, 这正好是STM32内部FLASH的首地址;
- F2: 表示校验和, 它的值为 $(0x02+0x00+0x00+0x04+0x08+0x00)\%256$ 的值再取补码。

再来看第二条记录:

:10000000000400204501000829030008BF02000881

- 10: 表示这条记录数据区的长度为16字节;
- 0000: 表示这条记录所在的地址, 与前面的扩展记录结合, 表示这条记录要存储的FLASH首地址为 $(0x0800\ 0000+0x0000)$ ;
- 00: 表示这是一条数据记录, 数据区的是地址;
- 000400204501000829030008BF020008: 这是要按地址存储的数据;
- 81: 校验和

# MDK的编译过程及文件类型全解



## hex、bin及axf文件的区别与联系

bin、hex及axf文件都包含了指令代码，但它们的信息丰富程度是不一样的。

- bin文件是最直接的代码映像，它记录的内容就是要存储到FLASH的二进制数据(机器码本质上就是二进制数据)，在FLASH中是什么形式它就是什么形式，没有任何辅助信息，包括大小端格式也没有，因此下载器需要有针对芯片FLASH平台的辅助文件才能正常下载(一般下载器程序会有匹配的这些信息);
- hex文件是一种使用十六进制符号表示的代码记录，记录了代码应该存储到FLASH的哪个地址，下载器可以根据这些信息辅助下载;
- axf文件在前文已经解释，它不仅包含代码数据，还包含了工程的各种信息，因此它也是三个文件中最大的。



# MDK的编译过程及文件类型全解



## hex、bin及axf文件的区别与联系

流水灯.bin	流水灯.hex
1 0004 0020 4501 0008 2903 0008 bf02 0008	1 :020000040800F2
2 2503 0008 8d01 0008 9d04 0008 0000 0000	2 :10000000000400204501000829030008BF02000881
3 0000 0000 0000 0000 0000 0000 4d03 0008	3 :10001000250300088D0100089D0400080000000071
4 9101 0008 0000 0000 2b03 0008 3904 0008	4 :100020000000000000000000000000000004D03000878
5 5f01 0008 5f01 0008 5f01 0008 5f01 0008	5 :100030009101000800000000002B03000839040008AB
6 5f01 0008 5f01 0008 5f01 0008 5f01 0008	6 :100040005F0100085F0100085F0100085F01000810
7 5f01 0008 5f01 0008 5f01 0008 5f01 0008	7 :100050005F0100085F0100085F0100085F01000800
8 5f01 0008 5f01 0008 5f01 0008 5f01 0008	8 :100060005F0100085F0100085F0100085F010008F0
9 5f01 0008 5f01 0008 5f01 0008 5f01 0008	9 :100070005F0100085F0100085F0100085F010008E0

流水灯_axf_elfInfo_c.txt
43 =====
44
45
46 ** Section #1 'ER_IROM1' (SHT_PROGBITS) [SHF_ALLOC + SHF_EXEC
47 Size : 1492 bytes (alignment 4)
48 Address: 0x08000000
49
50 \$d.realdata
51 RESET
52 _Vectors
53 0x08000000: 20000400 ... DCD 536871936
54 0x08000004: 08000145 E... DCD 134218053
55 0x08000008: 08000329 )... DCD 134218537
56 0x0800000c: 080002bf .... DCD 134218431
57 0x08000010: 08000325 %... DCD 134218533
58 0x08000014: 0800018d .... DCD 134218125
59 0x08000018: 0800049d .... DCD 134218909

在“多彩流水灯\_axf\_elfInfo\_c.txt”文件中不仅可以看到代码数据，还有具体的标号、地址以及反汇编得到的代码，虽然它不是\*.axf文件的原始内容，但因为它是通过\*.axf文件fromelf工具生成的，我们可认为\*.axf文件本身记录了大量这些信息，它的内容非常丰富，熟悉汇编语言的人可轻松阅读。

在hex文件中包含了地址信息以及地址中的内容，而在bin文件中仅包含了内容，连存储的地址信息都没有。观察可知，bin、hex及axf文件中的数据内容都是相同的，它们存储的都是机器码。这就是它们三都之间的区别与联系。



流水灯.bin				流水灯.hex			
15	5f01 0008 5f01 0008 5f01 0008 5f01 0008			22	:10014000000400200648804/0648804/FEE/FEE/1/		
16	5f01 0008 5f01 0008 5f01 0008 5f01 0008			23	:10015000FEE7FEE7FEE7FEE7FEE7FEE7FEE777		
17	5f01 0008 5f01 0008 5f01 0008 5f01 0008			24	:100160003D04000831010008064C074D06E0E06838		
18	5f01 0008 5f01 0008 5f01 0008 5f01 0008			25	:1001700040F0010394E8070098471034AC42F6D3EE		
19	5f01 0008 5f01 0008 5f01 0008 5f01 0008			26	:10018000FFF7DAFFC4050008D405000800BFFEE7A4		
20	dff8 0cd0 00f0 18f8 0048 0047 c104 0008			27	:100190007047018502E00998401E009000980028CA		
21	0004 0020 0648 8047 0648 0047 fee7 fee7			28	:1001A000F9D108BDD2E9F0410246002500260020C6		
22	fee7 fee7 fee7 fee7 fee7 fee7 fee7 fee7			29	:1001B00000230024002791F803C00CF00F0591F8EC		
23	3d04 0008 3101 0008 064c 074d 06e0 e068			30	:1001C00003C00CF0100CBFC1000F03D091F802C07A		
24	40f0 0103 94e8 0700 9847 1034 ac42 f6d3			31	:1001D0004CEA050591F800C0BCF1000F31D014685D		
25	fff7 daff c405 0008 d405 0008 00bf fee7			32	:1001E00000202BE04FF0010C0CFA00F3B1F800C036		
26	7047 01b5 02e0 0098 401e 0090 0098 0028			33	:1001F0000CEA03069E4220D183004FF00F0C0CFA4C		
27	f9d1 08bd 2de9 f041 0246 0025 0026 0020			34	:1002000003F7BC4305FA03FC4CEA040491F803C06D		
28	0023 0024 0027 91f8 03c0 03f0 0f05 91f8			35	:10021000BCF1280F06D14FF0010C0CFA00FCC2F81B		
29	03c0 0cf0 100c bcf1 000f 03d0 91f8 02c0			36	:1002200014C00AE991F803C0BCF1480F05D14FF0AB		
30	4cea 0505 91f8 00c0 bcf1 000f 31d0 1468			37	:10023000010C0CFA00FCC2F810C0401C0828D1D3F5		

所以经验丰富的人是有可能从bin或hex文件中恢复出汇编代码的，只是成本较高，但不是不可能。



# MDK的编译过程及文件类型全解



## hex、bin及axf文件的区别与联系

如果芯片没有做任何加密措施，使用下载器可以直接从芯片读回它存储在FLASH中的数据，从而得到bin映像文件，根据芯片型号还原出部分代码即可进行修改，甚至不用修改代码，直接根据目标产品的硬件PCB，抄出一样的板子，再把bin映像下载芯片，直接山寨出目标产品，所以在实际的生产中，一定要注意做好加密措施。

由于axf文件中含有大量的信息，且直接使用fromelf即可反汇编代码，所以更不要随便泄露axf文件。

lib文件也能反使用fromelf文件反汇编代码，不过它不能还原出C代码，由于lib文件的主要目的是为了保护C源代码，也算是达到了它的要求。

# MDK的编译过程及文件类型全解



## 6.htm静态调用图文件

在Output目录下，有一个以工程文件命名的后缀为\*.bulid\_log.htm及\*.htm文件，如“多彩流水灯.bulid\_log.htm”及“多彩流水灯.htm”，它们都可以使用浏览器打开。其中\*.build\_log.htm是工程的构建过程日志，而\*.htm是链接器生成的静态调用图文件。

在静态调用图文件中包含了整个工程各种函数之间互相调用的关系图，而且它还给出了静态占用最深的栈空间数量以及它对应的调用关系链。

Static Call Graph for image ..\..\Output\流水灯.axf

#<CALLGRAPH># ARM Linker, 5050169: Last Updated: Wed Oct 19 11:55:41 2016

Maximum Stack Usage = 32 bytes + Unknown(Cycles, Untraceable Function Pointers)

Call chain for Maximum Stack Depth:

main ⇒ LED\_GPIO\_Config ⇒ GPIO\_Init

# 零死角玩转STM32



**THANKS**

论坛：[www.firebbs.cn](http://www.firebbs.cn)

淘宝：[firestm32.taobao.com](http://firestm32.taobao.com)



扫描进入淘宝店铺