

hex文件格式

hex文件格式:

(1)以行为单位，每行以冒号开头，内容全部为16进制码（以ASCII码形式显示）

(2)在HEX文件里面，每一行代表一个记录。记录的基本格式为：

冒号	本行数据长度	本行数据起始地址	数据类型	数据	校验码
	1 byte	2 bytes	1 byte	n byte	1 byte

第一个字节 表示本行数据的长度:

第二、三字节表示本行数据的起始地址;

第四字节表示数据类型，数据类型有：0x00、0x01、0x02、0x03、0x04、0x05。

'00' Data Rrecord: 用来记录数据，HEX文件的大部分记录都是数据记录

'01' End of File Record:用来标识文件结束，放在文件的最后，标识HEX文件的结尾

'02' Extended Segment Address Record:用来标识扩展段地址的记录

'03' Start Segment Address Record:开始段地址记录

'04' Extended Linear Address Record:用来标识扩展线性地址的记录

'05' Start Linear Address Record:开始线性地址记录

然后是数据，最后一个字节为校验和。

校验和的算法为：计算校验和前所有16进制码的累加和(不计进位)，校验和 = 0x100 - 累加和

打开.hex内容如下：（中间部分数据略去）

[\[plain\]](#) [view plaincopy](#)

```

1. <strong>:020000040800F2
2. :10000000B80B00207D250008850300088703000841</strong>
3. :100010009B300089F030008A30300080000000E2
4. :10002000000000000000000000000000A70300081E
5. :10003000A903000800000000AB030008AD0300089E
6. 。
7. 。
8. 。
9. :102B4000040000000000000000000000000000000081
10. :102B5000000000000000000000000000000000000075
11. :102B6000010203040102030406070809020406081F
12. :102B700000366E01000000000000000001020304A6
13. :042B80000607080933
14. <strong>:0400000508000121CD
15. :00000001FF</strong>

```

先分析第一条语句---“: 02 0000 04 0800 F2”

冒号	本行数据长度	本行数据起始地址 (偏移地址)	数据类型	数据	校验码
	1 byte	2 bytes	1 byte	n byte	1 byte
[plain]view plaincopy 1.		[plain]view plaincopy			
:	02	0000	04	0800	F2

在上面的数据类型后2个记录(04, 05)都是用来提供地址信息的。每次碰到这2个记录的时候，都可以根据记录计算出一个“基”地址。对于后面的数据记录，计算地址的时候，都是以这些“基”地址为基础的。以我们的语句为例：

第1条记录的长度为02, LOAD OFFSET为0000, RECTYPE为04, 说明该记录为扩展段地址记录。数据为0800, 校验和为F2。从这个记录的长度和数据, 我们可以计算出一个基地址, 这个地址为 $(0x0800 \ll 16) = 0x0800\ 0000$ 。后面的数据记录都以这个地址为基地址。

第二条语句---“:10000000B80B00207D250008850300088703000841”

冒号	本行数据长度	本行数据起始地址(偏移地址)	数据类型	数据	校验码
	1 byte	2 bytes	1 byte	n byte	1 byte
[plain]view plain copy 1.	[plain]view plain copy 1.				
:	10	0000	00	B80B00207D2500088503000887030008	41

第2条记录的长度为10 (0x10=16字节)，LOAD OFFSET为0000，RECTYPE为00 ('00' Data Record: 用来记录数据，HEX文件的大部分记录都是数据记录)，数据为B80B00207D2500088503000887030008 校验码为41；此时基地址为：0x0800 0000 加上偏移地址：0x0000 这条记录的16个字节的数据的起始地址为：0x0800000 + 0x0000 =0x0800 0000

第3条语句----“:0400000508000121CD”

冒号	本行数据长度	本行数据起始地址(偏移地址)	数据类型	数据	校验码
	1 byte	2 bytes	1 byte	n byte	1 byte
[plain]view plain copy 1.	[plain]view plain copy 1.				
:	04	0000	05	08000121	CD

记录的长度为04，LOAD OFFSET为0000，RECTYPE为05，此时，EIP寄存器里存放的地址：0x0800 0121;即IP指向下一个要执行的指令所在地址，我们来看一下IAP工程list目录下的.map文件，其中第393行处如图：（看到没？0x0800 0121值 main函数的入口地址）🤔

```

C:\Documents and Settings\Administrator\桌面\STM32F10x IAP源码和测试代码\STM32 IAP源码和测试代码\IAP\RVMDK\Listing...
文件(F) 编辑(E) 搜索(S) 视图(V) 格式(O) 语言(L) 设置(T) 宏(O) 运行(R) 插件(P) 窗口(W) 2
Project.hex Project.map
391  __Vectors                                0x08000000  Data      4
    startup_stm32f10x_md_vl.q(RESET)
392  __Vectors_End                          0x08000120  Data      0
    startup_stm32f10x_md_vl.q(RESET)
393  __main                                  0x08000121  Thumb Code 8  __main.q(!!!main)
394  __scatterload                          0x08000129  Thumb Code 0  __scatter.q(!!!scatter)
395  __scatterload_rt2                      0x08000129  Thumb Code 44 __scatter.q(!!!scatter)
396  __scatterload_rt2_thumb_only          0x08000129  Thumb Code 0  __scatter.q(!!!scatter)
397  __scatterload_null                    0x08000137  Thumb Code 0  __scatter.q(!!!scatter)
398  __scatterload_copy                    0x0800015d  Thumb Code 26
    __scatter_copy.q(!!!handler_copy)
399  __scatterload_zeroinit                 0x08000179  Thumb Code 28
    __scatter_zi.q(!!!handler_zi)
400  __rt_lib_init                          0x08000195  Thumb Code 0
    libinit.q(.ARM.Collect$$libinit$$00000000)
401  __rt_lib_init_alloca_1                  0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$0000002C)
402  __rt_lib_init_argv_1                    0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$0000002A)
403  __rt_lib_init_atexit_1                  0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$00000019)
404  __rt_lib_init_clock_1                   0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$0000001F)
405  __rt_lib_init_cpp_1                     0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$00000030)
406  __rt_lib_init_exceptions_1              0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$0000002E)
407  __rt_lib_init_fp_1                      0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$00000002)
408  __rt_lib_init_fp_trap_1                 0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$0000001D)
409  __rt_lib_init_getenv_1                  0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$00000021)
410  __rt_lib_init_heap_1                    0x08000197  Thumb Code 0
    libinit2.q(.ARM.Collect$$libinit$$00000008)
Normal text file      length: 78893  lines: 857  Ln: 393  Col: 4  Sel: 90 | 0  Dos\Windows  ANSI as UTF-8  INS

```

EIP是32位机的指令寄存器，IP是指令寄存器，存放当前指令的下一条指令的地址。CPU该执行哪条指令就是通过IP来指示的

Start Linear Address Record (32-bit format only)

RECORD MARK ' : '	RECLLEN '04'	LOAD OFFSET '0000'	RECTYP '05'	EIP	CHKSUM
1-byte	1-byte	2-bytes	1-byte	4-bytes	1-byte

The Start Linear Address Record is used to specify the execution start address for the object file. The value given is the 32-bit linear address for the EIP register. Note that this record only specifies the code address within the 32-bit linear address space of the 80386. If the code is to start execution in the real mode of the 80386, then the Start Segment Address Record should be used instead, since that record specifies both the CS and IP register contents necessary for real mode.

The Start Linear Address Record can appear anywhere in a 32-bit hexadecimal object file. If such a record is not present in a hexadecimal object file, a loader is free to assign a default start address.

The contents of the individual field within the record are:

RECORD MARK

This field contains 03AH, the hexadecimal encoding of the ASCII colon (':') character.

RECLLEN

The field contains 03034H, the hexadecimal encoding of the ASCII characters '04', which is the length, in bytes, of the EIP register content within this record.

LOAD OFFSET

This field contains 030303030H, the hexadecimal encoding of the ASCII characters '0000', since this field is not used for this record.

RECTYP

This field contains 03035H, the hexadecimal encoding of the ASCII character '05', which specifies the record type to be a Start Linear Address Record.

EIP

This field contains eight ASCII hexadecimal digits that specify the 32-bit EIP register contents. The high-order byte is the 10th/11th character pair.

CHKSUM

This field contains the check sum on the RECLLEN, LOAD OFFSET, RECTYP, and EIP fields.

上图参考hex数据文档: <http://pages.interlog.com/~speff/usefulinfo/Hexfmt.pdf> 或 <http://microsym.com/editor/assets/intelhex.pdf>

第4条语句——“:00000001FF” (每一个.hex文件的最后一行都是固定为这个内容)

冒号	本行数据长度	本行数据起始地址 (偏移地址)	数据类型	数据	校验码
	1 byte	2 bytes	1 byte	n byte	1 byte
[plain]view plaincopy 1.					
:	00	0000	01		FF

(每一个.hex文件的最后一行都是固定为这个内容)

记录的长度为00, LOAD OFFSET为0000, RECTYPE为01 (01' End of File Record:用来标识文件结束, 放在文件的最后, 标识HEX文件的结尾)