Hex文件解析功能

驱动组 王彬 2016.08.13

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 时间 | 作者 | 说明 |
| V1.0 | 2016.8.13 | 王彬 | 第一版本draft |
|  |  |  |  |

# 1:hex文件概述

## 1.1：概述

Hex 全称 （Intel HEX）文件是由一行行符合Intel HEX文件格式的文本所构成的ASCII文本文件。在Intel HEX文件中，每一行包含一个HEX记录。这些记录由对应机器语言码和/或常量数据的十六进制编码数字组成。Intel HEX文件通常用于传输将被存于ROM或者EPROM中的程序和数据。大多数EPROM编程器或模拟器使用Intel HEX文件。

Intel HEX由任意数量的十六进制记录组成，每行为一个记录，每个记录由“：”（0x3a）开始，以回车键结束(0x0d,0x0a)且每个记录包含5个域，它们按以下格式排列：

:llaaaatt[dd...]cc （eg：:10000000020003787FE4F6D8FD75812B02004A02D6）

每一组字母对应一个不同的域，每一个字母对应一个十六进制编码的数字。每一个域由至少两个十六进制编码数字组成，它们构成一个字节，就像以下描述的那样：

　　1）: 每个Intel hex记录都由冒号开头.

　　2）ll （10，即16个数据）是数据长度域,它代表记录当中数据字节(dd...)的数量.

　　3）aaaa（0000） 是地址域,它代表记录当中数据的起始地址.

　　4）tt （00）是代表hex记录类型的域,它可能是以下数据当中的一个:

　　 00 – 数据记录

　　 01 – 文件结束记录

　　 02 – 扩展段地址记录

　　 04 – 扩展线性地址记录

　　5）dd 是数据域,它代表一个字节的数据。一个记录可以有许多数据字节。记录当中数据字节的数量必须和数据长度域(ll)中指定的数字相符。

　　6）cc 是校验和域（Checksum）,所谓的Checksum值是一种标准的校验码，把它加在每一行机器码的最后，可以使每一行所有的十六进制值(两个为一组)，加总后所得到最后两位十六进制码应为00H（即：通过将记录当中所有十六进制编码数字对的值相加,以256为模进行以下补足）。以本例来说，将所有的值相加所得到的值应该是：

10H + 00H + 00H + 00H + 02H + 00H + 03H + 78H + 7FH + E4H + F6H + D8H+ FDH + 75H + 81H + 2BH + 02H + 00H + 4AH + 02H + D6H = 700H

每个HEX格式的最后一行都是固定为：:00000001FF ，它所代表的意思也等于我们写程序的END差不多 。当编译器看到这一段后，就不会再继续编译下去了。

图表标示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RecordMark | RecordLength | LoadOffset | RecordType | Data | Checksum |
| 记录标志 | 记录长度 | 装载偏移 | 记录类型 | 数据 | 校验和 |

**数据记录**适用于8位、16位和32位格式，其详细格式如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 记录名 | RecordMark | RecordLength | LoadOffset | RecordType | Data | Checksum |
| 记录标志 | 记录长度 | 装载偏移 | 记录类型 | 数据 | 校验和 |
| 内容 | “:” | X | - | “00” | - | - |
| 字节数 | 1 | 1 | 2 | 1 | X | 1 |

**文件结束记录**适用于8位、16位和32位格式，其详细格式如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 记录名 | RecordMark | RecordLength | LoadOffset | RecordType | Checksum |
| 记录标志 | 记录长度 | 装载偏移 | 记录类型 | 校验和 |
| 内容 | “:” | “00” | “0000” | “01” | “FF” |
| 字节数 | 1 | 1 | 2 | 1 | 1 |

## 1.2：详细说明

**1）00-数据记录**

Intel HEX文件由任意数量以回车换行符结束的数据记录组成.数据记录外观如下:

　　:10246200464C5549442050524F46494C4500464C33

　　其中:

　　10 是这个记录当中数据字节的数量

　　2462 是数据将被下载到存储器当中的地址

　　00 是记录类型(数据记录)

　　464C…464C是数据

　　33 是这个记录的校验和

**2）01-文件结束(EOF)记录**

Intel HEX文件必须以文件结束(EOF)记录结束。这个记录的记录类型域的值必须是01.EOF记录外观总是如下:

　　:00000001FF

　　其中:

　　00 是记录当中数据字节的数量

　　0000 是数据被下载到存储器当中的地址.在文件结束记录当中地址是没有意义被忽略的.0000h是典型的地址

　　01 是记录类型 01(文件结束记录)

　　FF 是这个记录的校验和,计算方法如下:

　　01h + NOT(00h + 00h + 00h + 01h).

**3）02-扩展段地址记录(HEX86)**

扩展段地址记录也叫HEX86记录,它包括4-19位数据地址段.扩展段地址记录总是有两个数据字节,外观如下:

　　:020000021200EA

　　其中:

　　02 是记录当中数据字节的数量

　　0000 是地址域.对于扩展段地址记录,这个域总是0000

　　02 是记录类型 02(扩展段地址记录)

　　1200 是地址段

　　EA 是这个记录的校验和,计算方法如下:

　　01h + NOT(02h + 00h + 00h + 02h + 12h + 00h)

　　当一个扩展段地址记录被读取,存储于数据域的扩展段地址被保存,它被应用于从Intel HEX文件读取来的随后的记录。段地址保持有效,直到它被另外一个扩展地址记录所改变。

通过把记录当中的地址域与被移位的来自扩展段地址记录的地址数据相加获得数据记录的绝对存储器地址。

　　以下的例子演示了这个过程..

　　来自数据记录地址域的地址 2462

　　扩展段地址记录数据域 + 1200

　　绝对存储器地址 00014462

**4）04-扩展线性地址记录(HEX386)**

扩展线性地址记录也叫作32位地址记录或HEX386记录.这些记录包含数据地址的高16位.扩展线性地址记录总是有两个数据字节,外观如下:

　　:02000004FFFFFC

　　其中:

　　02 是这个记录当中数据字节的数量.

　　0000 是地址域,对于扩展线性地址记录,这个域总是0000.

　　04 是记录类型 04(扩展线性地址记录)

　　FFFF 是地址的高16位.

　　FC 是这个记录的校验和,计算方法如下:

　　01h + NOT(02h + 00h + 00h + 04h + FFh + FFh).

　　当一个扩展线性地址记录被读取,存储于数据域的扩展线性地址被保存,它被应用于从Intel HEX文件读取来的随后的记录.线性地址保持有效,直到它被另外一个扩展地址记录所改变.

通过把记录当中的地址域与被移位的来自扩展线性地址记录的地址数据相加获得数据记录的绝对存储器地址.

　　示例

　　以下的例子演示了这个过程..

　　来自数据记录地址域的地址 2462

　　扩展线性地址记录的数据域 + FFFF

　　绝对存储器地址 FFFF2462

## 1.3：hex与bin区别

下面总结一下hex文件和bin文件的区别：

hex文件是用ASCII来表示二进制的数值，例如一个8bit的二进制数值0x3F，用ASCII来表示就需要分别表示为字符‘3’和字符‘F’，每个字符需要一个Byte，所以hex文件需要>2倍的空间。对一个bin文件而言，你查看文件的大小就可以知道文件包括的数据的实际大小。而对hex文件而言，你看到的文件大小并不是实际的数据大小。一是因为hex文件是用ASCII来表示数据，二是因为hex文件本身还包括其他的附加信息。

## 1.4：hex记录分类

关于hex文件的记录类型还有下面的一种分类：

'00' Data Record

'01' End of File Record

'02' Extended Segment Address Record

'03' Start Segment Address Record

'04' Extended Linear Address Record

'05' Start Linear Address Record