AR# 476

PROMGen - PROM/EEPROM 文件格式描述: MCS, EXO, HEX 等

## 描述

iMPACT 软件可用于生成以下 PROM 文件格式：

* Intel MCS-86 Hexadecimal Object (.mcs)
* Motorola EXORmacs (.exo)
* Tektronix HEX (.tek)

使用 iMPACT 软件还可以生成只包含 ASCII 格式数字（无地址）的十六进制文件 ("'.hex")。

iMPACT 11.4 以后版本不支持 EXO 文件，需要用 PROMGen 将这类文件转换成 MCS 文件。

如需更多 PROM 文件及其它配置文章，敬请查看 [(Xilinx 答复 34104)](https://china.xilinx.com/support/answers/34104.htm)。

## 解决方案

PROMGen 可用于将文件格式从一种类型变为下面格式的另一种类型。

下面的语句将 ".mcs" 文件转换为 hex 文件：

promgen -p hex -r input.mcs -o output.hex

*PROM 文件格式*

**INTEL MCS-86 HEXADECIMAL OBJECT**

**File Format Code 88**

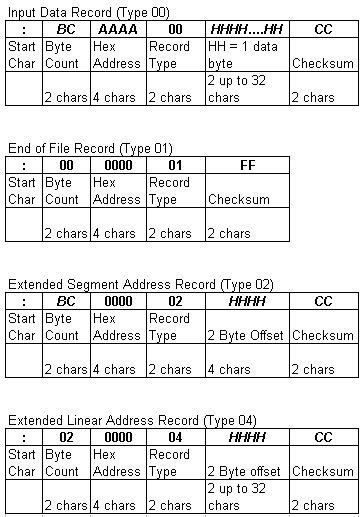
Intel Hexadecimal Object 记录格式带有一个 用于定义记录开始位置、字节数、加载地址和记录类型的9 字符（4个字段）前缀，以及一个 2 字符的校验的后缀。

16 位十六进制格式允许 20 位分段地址空间；32 位格式允许 32 位线性地址空间。

六个记录类型为：

00 = 数据记录   
01 = 文件记录结束（表示文件末尾）  
02 = 扩展段地址记录（提供用来确定绝对目标地址的偏移）  
03 = 开始段地址记录（在Data I/O 转换固件进行输入时忽略，输出时不发送）   
04 = 扩展线性地址记录 （提供用以确定绝对目的地址的偏移） \*   
05 = 起始线性地址记录（提供执行起始地址） \*

\*****注：**** 只在 32 位格式中支持扩展线性地址记录类型和起始线性地址记录类型。



（Data I/O 固件不使用起始段地址记录（第3类）。）

校验和是十六进制记录（包括字节数、地址和所有数据字节）中前面字节二进制总和的补码。

扩展段地址记录（第2类）定义20位段基地址的4至19位。该地址将被添加到后续的数据记录地址中，以提供绝对地址。（该记录随机出现在文件中，但对于这个应用而言是初始记录。）

扩展线性地址记录（第4类）定义32位线性基地址的 16 至 31 位。该地址将被添加到后续的数据记录地址中，以提供绝对地址。

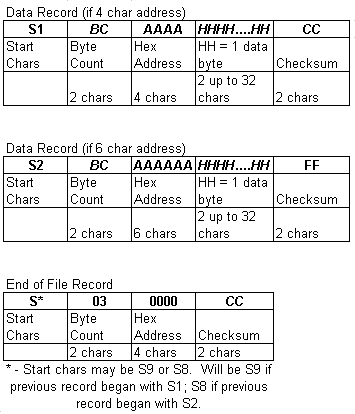
****注：**** 在使用这种格式时，即使偏移是零，也要规定地址偏移。

如果记录大小大于 16（十进制），则 Data I/O 固件会强制将记录大小设定为16（十进制）。

**MOTOROLA EXORmacs**

**File Format Code 87**

Motorola 数据文件可能以登陆记录开头，但这个应用会将其忽略。（数据记录包含一个 8 至 9 字符的前缀和一个 2 字符的校验的后缀。）



字节数量为记录中数据字节数量再加 3（四个十六进制字符的地址）或者加 4（六个十六进制字符的地址）——3 或 4 是校验和与地址的字节数。

校验和是十六进制记录（包括字节数、地址和所有数据字节）中前面字节二进制总和的补码。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tektronix maximum address:..................................................65536 (x100000)

Motorola EXORmacs maximum address:..........................16777216 (x100000)

16-bit Intel MCS-86 Hexadecimal maximum address:..........1048576 (x100000)

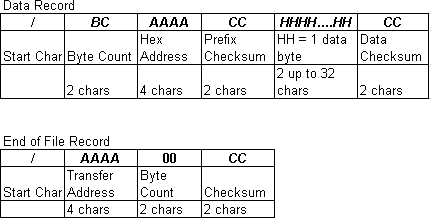
32-bit Intel MCS-86 Hexadecimal maximum address:....4294967296 (x100000)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**TEKTRONIX HEXADECIMAL**

**File Format Code 86**

Tektronix 数据文件十六进制格式包含一个 9 字符的前缀（起始字符、地址、字节数和前缀校验和），随后是数据字节，最后是一个 2 字符的数据字符校验和。文件末尾记录只包含用于标明传输结束的控制字符，以及字节数和用于验证的校验和。



前缀校验和是组成地址和字节数的六位数 4 位十六进制值的 8 位总和。数据校验和是组成数据字节的 4 位十六进制值的 8 位总和再进行 256 模运算。

链接地址：<https://china.xilinx.com/support/answers/476.html>

hex文件和bin文件区别

HEX文件和BIN文件是我们经常碰到的2种文件格式。因为自己也是新手，所以一直对这两个文件懵懵懂懂，不甚了解，最近在做STM32单片机的IAP更新，其中要考虑HEX文件和BIN文件，所以需要学习下这两种文件。下面是最近的我的了解，如有不对地方还请指正。

1. HEX文件是包括地址信息的，而BIN文件格式只包括了数据本身  
    在烧写或下载HEX文件的时候，一般都不需要用户指定地址，因为HEX文件内部的信息已经包括了地址。而烧写BIN文件的时候，用户是一定需要指定地址信息的。

2. HEX文件格式  
    HEX文件都是由记录（RECORD）组成的。在HEX文件里面，每一行代表一个记录。以下为记录（Record）的具体格式：

### Record structure

  A [record](https://en.wikipedia.org/wiki/Record_(computer_science)" \o "Record (computer science)" \t "http://blog.csdn.net/wordwarwordwar/article/details/_blank) (line of text) consists of six [fields](https://en.wikipedia.org/wiki/Field_(computer_science)" \o "Field (computer science)" \t "http://blog.csdn.net/wordwarwordwar/article/details/_blank) (parts) that appear in order from left to right:

1. ****Start code****, one character, an ASCII colon ':'.
2. ****Byte count****, two hex digits, indicating the number of bytes (hex digit pairs) in the data field. The maximum byte count is 255 (0xFF). 16 (0x10) and 32 (0x20) are commonly used byte counts.
3. ****Address****, four hex digits, representing the 16-bit beginning memory address offset of the data. The physical address of the data is computed by adding this offset to a previously established base address, thus allowing memory addressing beyond the 64 kilobyte limit of 16-bit addresses. The base address, which defaults to zero, can be changed by various types of records. Base addresses and address offsets are always expressed as [big endian](https://en.wikipedia.org/wiki/Big_endian" \o "Big endian" \t "http://blog.csdn.net/wordwarwordwar/article/details/_blank) values.
4. ****Record type**** (see [record types](https://en.wikipedia.org/wiki/Intel_HEX" \l "Record_types" \o "Intel HEX" \t "http://blog.csdn.net/wordwarwordwar/article/details/_blank) below), two hex digits, *00* to *05*, defining the meaning of the data field.
5. ****Data****, a sequence of *n* bytes of data, represented by 2*n* hex digits. Some records omit this field (*n* equals zero). The meaning and interpretation of data bytes depends on the application.
6. [Checksum](https://en.wikipedia.org/wiki/Checksum" \o "Checksum" \t "http://blog.csdn.net/wordwarwordwar/article/details/_blank), two hex digits, a computed value that can be used to verify the record has no errors.

     看个例子：  
     :020000040000FA  
     :10000400FF00A0E314209FE5001092E5011092E5A3  
     :00000001FF           
         
      对上面的HEX文件进行分析：  
      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0000，校验和为FA。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0X0000。后面的数据记录都以这个地址为基地址。  
      第2条记录的长度为10（16），LOAD OFFSET为0004，RECTYPE为00，说明该记录为数据记录。数据为FF00A0E314209FE5001092E5011092E5，共16个BYTE。这个记录的校验和为A3。此时的基地址为0X0000，加上OFFSET，这个记录里的16BYTE的数据的起始地址就是0x0000 + 0x0004 = 0x0004.  
        第3条记录的长度为00，LOAD OFFSET为0000，TYPE ＝ 01，校验和为FF。说明这个是一个END OF FILE RECORD，标识文件的结尾。  
         
        在上面这个例子里，实际的数据只有16个BYTE：FF00A0E314209FE5001092E5011092E5，其起始地址为0x4。

3. BIN文件格式  
    对二进制文件而言，其实没有”格式”。文件只是包括了纯粹的二进制数据。

4. HEX文件是用ASCII来表示二进制的数值。例如一般8-BIT的二进制数值0x3F，用ASCII来表示就需要分别表示字符'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的大小。一是因为HEX文件是用ASCII来表示数据，二是因为HEX文件本身还包括别的附加信息。

链接地址：<http://blog.csdn.net/wordwarwordwar/article/details/77151611>

论坛说明区别：

讨论1：

The MCS file is a HEX file where two ASCII chars are used to represent each byte of data. And the binary file of course just contains just the raw byte stream, in sequence.

So the MCS file seems less efficient, because it takes 2 bytes to represent 1 byte. But it has a couple of advantages (1) It has a checksum at the end of each line for integrity. (2) Each line includes the address where the line should be located in memory.

So for example, if a MCS file contains a few segments located far apart in address space, it can be small while the equivalent binary file might be huge, because it would have a lot of 0x00 or 0xFFs to fill the space between segments.

You can easily convert a MCS file to a binary file using an old command line utility called HEX2BIN.EXE You can find on the Keil webste at <http://www.keil.com/download/docs/7.asp>

讨论2：

<http://www.fpga-faq.com/archives/33100.html#33108> Is a good discussion of what is going on.

25 years ago, eeproms were programmed with prom programmers, that had Intel format, Motorola format, and various other formats. The eeproms made by Intel and Motorola are long gone, and so are the programmers, but the file formats remain, and became defacto standards.

The .mcs, .rbt, and others are ASCII representations (ASCI 1 = 31H, and ASCII 0=30H) with headers, comments, etc.

The .bin, .bit are binary files: all single bits, 1's and zeros.

All formats have evolved over the years, and are still used. New companies make programmers, and all the old formats get supported.

So a .bin is used to program the part using an Impact programming cable (JTAG or parallel), and a .rbt is the file readback when doing averify Impact (also provide a .bin readback file which is not equal to the .bin file used to program the part!).

The .mcd might get used to program a prom for the final board, to configure the part upon power on.

mcs的文件格式

**Description**

The iMPACT software can be used to generate the followingPROM file formats:

·      **IntelMCS-86 Hexadecimal Object (.mcs)**

·      Motorola EXORmacs (.exo)

·      Tektronix HEX (.tek)

Hexadecimal files ("'.hex") containing onlydata (without addressing) inASCII format can also begenerated using the iMPACT software.

EXO files will not be support past iMPACT 11.4 andPROMGen will need to be used to change these files into MCS files.

For more PROM File Articles and other ConfigurationRelated Articles, see[(Xilinx Answer34104)](http://china.xilinx.com/support/answers/34104.htm" \t "http://blog.csdn.net/wordwarwordwar/article/details/_blank).

Solution

PROMGen can be used to change file format from one typeto the another in the following manner.

The following changes an ".mcs" file into a hexfile:

promgen -p hex -r input.mcs -o output.hex

*PROM FileFormats*

**INTEL MCS-86HEXADECIMAL OBJECT**

**File FormatCode 88**

The Intel Hexadecimal Object record format has a9-character (4 field) prefix前缀 that defines thestart of the record, byte count, load address, and record type, as well as a 2-characterchecksum suffix. 开始记录、字节个数、加载地址、记录类型、2字符的校验和后缀

The 16-bit hexadecimal format allows for a 20-bitsegmented address space, and the 32-bit format allows for the 32-bit linearaddress space. 16 bit形式允许20 bit的分割的地址空间，32 bit形式允许32 bit的线性地址空间。

The six record types are:

00 = Data Record 数据记录  
01 = End of File Record (signals the end of the file) 文件结尾记录  
02 = Extended Segment Address Record (provides the offset to determine theabsolute destination address)段地址记录  
03 = Start Segment Address Record (is ignored during input and not sent duringoutput by Data I/O translator firmware)  
04 = Extended Linear Address Record (provides the offset to determine theabsolute destination address) \*线性地址记录  
05 = Start Linear Address Record (provides the execution start address) \* 起始线性地址记录

\*****NOTE:**** The Extended Linear AddressRecord type and Start Linear Address Record type are supported only in the32-bit format.

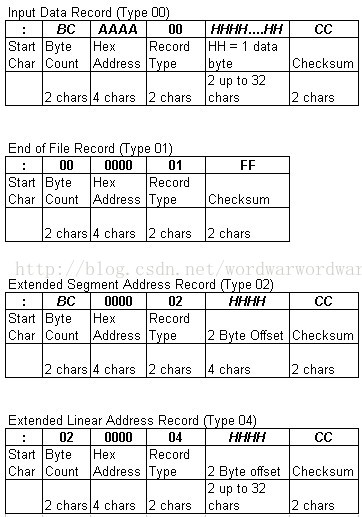
:  10   0010  00   000000E0 0400 0000 0400 0000 0C80 0480   E8

：16个字节 地址  数据

:  10   0020  00   0020 FCA7 0C80 0180 4010 01C9 0C00 0180    59

:  10   0030  00   0000 0090 0400 0000 0C00 0380 0000 6000   3D

:  10   0040  00   0C00 0580 0000 6000 0400 0000 0400 0000   B7



:02000004000BEF 020000040001F9

:020000040000FA

(The Start Segment Address record (Type 03) is not usedby Data I/O firmware.)

The checksum is the two's complement补码 of thebinary summation of the preceding bytes in the record (including thebyte count, address, and any data bytes) in hexadecimalnotation.

The extended segment address record (Type 02) definesBits4 to 19 of the 20-bit segment base address. This addresswill be added to subsequent data record addresses to provide the absoluteaddress. (This record can appear randomly in the file, but for thisapplication, it will be the initial record.)

The extended linear address record (Type 04) definesbits 16 to 31 of the 32-bit linear base address.This addresswill be added to subsequent data record addresses to provide the absoluteaddress.

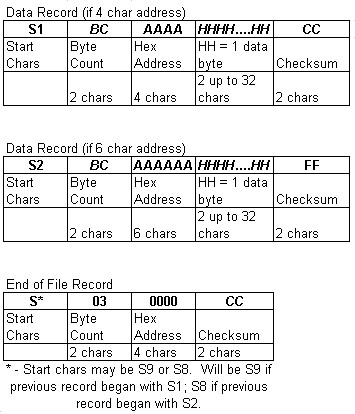
**NOTE:**Always specify the address offset when using this format, even when the offsetis zero.

The Data I/O firmware will force the record size to 16(decimal) if the record size is specified greater than 16.

**MOTOROLAEXORmacs**

**File FormatCode 87**

A Motorola data file may begin with a sign-on record, butfor this application it will be ignored. (Data records have an 8- or9-character prefix and a two- character checksum suffix.)



The Byte Count is the number of data bytes in the recordplus 3 (for a four-hexadecimal character address) or plus 4 (for asix-hexadecimal character address) for the checksum and address.

The checksum is a one's complement of binary summation ofpreceding bytes in the record (including byte count, address, and data bytes)in hexadecimal notation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tektronix maximum address:..................................................65536(x100000)

Motorola EXORmacs maximumaddress:..........................16777216 (x100000)

16-bit Intel MCS-86 Hexadecimal maximumaddress:..........1048576 (x100000)

32-bit Intel MCS-86 Hexadecimal maximumaddress:....4294967296 (x100000)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**TEKTRONIXHEXADECIMAL**

**File FormatCode 86**

The Tektronix Hexadecimal format for a data file consistsof a 9-character prefix (start character, address, byte count, and checksum ofthe prefix) followed by data bytes and ended by a 2-character checksum of thedata characters. The end-of-file record consists only of control charactersused to signal the end of the transmission and a byte count and sum-check forverification.

The prefix checksum is the 8-bit sum of the four-bithexadecimal value of the six digits that make up the address and byte count.The data checksum is the 8-bit sum, modulo 256, of the 4-bit hexadecimal valuesof the digits that make up the data bytes.

AR# 34104 Description

The Configuration Design Assistant walks you through the recommended designflow for configuration while debugging commonly encountered issues, such asiMPACT software install and error messages, Cable Support, and FPGA relatedissues.The Design Assistant not only provides useful design and troubleshootinginformation, but breaks down the scope of configuration issues through thisprocess.  
  
****NOTE:****This answer record is part ofthe Xilinx Configuration Solution Center[(Xilinx Answer34904)](http://china.xilinx.com/support/answers/34904.htm" \t "http://blog.csdn.net/wordwarwordwar/article/details/_blank) Xilinx Configuration Solution Center is available toaddress all questions related to Configuration.

<https://china.xilinx.com/support/answers/34904.html>

文章链接地址：http://blog.csdn.net/wordwarwordwar/article/details/77151619