Intel HEX

Intel HEX is a file format that conveys binary information in ASCII text form. It is commonly used for programming microcontrollers, EPROMs, and other types of programmable logic devices. In a typical application, a compiler or assembler converts a program's source code (such as in C or assembly language) to machine code and outputs it into a HEX file. The HEX file is then imported by a programmer to “burn” the machine code into a ROM, or is transferred to the target system for loading and execution.[1]

1 Format

Intel HEX consists of lines of ASCII text that are separated by line feed or carriage return characters or both. Each text line contains hexadecimal characters that encode multiple binary numbers. The binary numbers may represent data, memory addresses, or other values, depending on their position in the line and the type and length of the line. Each text line is called a *record*.

1.1 Record structure

A record (line of text) consists of six fields (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 values.
4. **Record type** (see record types 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 $2n$ hex digits. Some records omit this field (n equals zero). The meaning and interpretation of data bytes depends on the application.
6. **Checksum**, two hex digits, a computed value that can be used to verify the record has no errors.

1.1.1 Color legend

As a visual aid, the fields of Intel HEX records are colored throughout this article as follows:

Start code Byte count Address Record type Data Checksum

1.1.2 Checksum calculation

A record’s checksum byte is the two’s complement (negative) of the least significant byte (LSB) of the sum of all decoded byte values in the record preceding the checksum. It is computed by summing the decoded byte values and extracting the LSB of the sum (*i.e.*, the data checksum), and then calculating the two’s complement of the LSB (*e.g.*, by inverting its bits and adding one).

For example, in the case of the record :0300300002337A1E, the sum of the decoded byte values is 03 + 00 + 30 + 00 + 02 + 33 + 7A = E2. The two’s complement of E2 is 1E, which is the checksum byte appearing at the end of the record.

The validity of a record can be checked by computing its checksum and verifying that the computed checksum equals the checksum appearing in the record; an error is indicated if the checksums differ. Since the record’s checksum byte is the negative of the data checksum, this process can be reduced to summing all decoded byte values — including the record’s checksum — and verifying that the LSB of the sum is zero.

1.2 Text line terminators

Intel HEX records are separated by one or more ASCII line termination characters so that each record appears alone on a text line. This enhances legibility by visually delimiting the records and it also provides padding between records that can be used to improve machine parsing efficiency.

Programs that create HEX records typically use line termination characters that conform to the conventions of their operating systems. For example, Linux programs use a single LF (line feed, hex value 0A) character to terminate lines, whereas Windows programs use a CR (carriage return, hex value 0D) followed by a LF.

1.3 Record types

Intel HEX has six standard record types:

1.4 Named formats

Special names are sometimes used to denote the formats of HEX files that employ specific subsets of record types. For example:

- **I8HEX** files use only record types 00 and 01 (16 bit addresses)
- **I16HEX** files use only record types 00 through 03 (20 bit addresses)
- **I32HEX** files use only record types 00, 01, 04, and 05 (32 bit addresses)

2 File example

This example shows a file that has four data records followed by an end-of-file record:

```
:10010000214601360121470136007EFE09D2190140
:100110002146017E17C20001FF5F16002148011928
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
```

3 See also

Articles

- Binary-to-text encoding, a survey and comparison of encoding algorithms
- SREC, hex file format by Motorola
- TekHex, hex file format by Tektronix

Other

- Template:Intel HEX, useful for displaying Intel Hex records in Wikipedia articles

4 References

[1] *Hexadecimal Object File Format Specification* (PDF) (Revision A ed.). Intel. 1988-01-06. Archived (PDF) from the original on 2016-06-07. Retrieved 2016-06-07.

5 External links

Documentation

- Intel HEX format description
- Intel HEX reference with examples

Software Programs

- binex - a converter between Intel HEX and binary for Windows.
- SRecord, a converter between Intel HEX and binary for Linux (usage), C++ source code.

Software Source Code

- libgis, open source C library that converts Intel HEX, Motorola S-Record, Atmel Generic files.

6 Text and image sources, contributors, and licenses

6.1 Text

6.2 Images

6.3 Content license