Intel HEX

From Wikipedia, the free encyclopedia

Intel HEX is a file format for conveying binary information for applications like programming microcontrollers, EPROMs, and other kinds of chips. It is one of the oldest file formats available for this purpose, having been in use since the 1970s.[*citation needed*]

In a typical application, a compiler converts a program (such as in C or assembly language) to machine code and outputs it into a HEX file; that file is then imported by a programmer to "burn" the machine code into a chip.

Contents

Format

The format is a text file, with each line containing hexadecimal values encoding a sequence of data and their starting offset or absolute address.

There are three types of Intel HEX: 8-bit, 16-bit, and 32-bit. They are distinguished by their byte order.

Each line of Intel HEX file consists of six parts:

1. **Start code**, one character, an ASCII colon ':'.
2. **Byte count**, two hex digits, a number of bytes (hex digit pairs) in the data field. 16 (0x10) or 32 (0x20) bytes of data are the usual compromise values between line length and address overhead.
3. **Address**, four hex digits, a 16-bit address of the beginning of the memory position for the data. Limited to 64 kilobytes, the limit is worked around by specifying higher bits via additional record types. This address is big endian.
4. **Record type**, two hex digits, *00* to *05*, defining the type of the data field.
5. **Data**, a sequence of *n* bytes of the data themselves, represented by 2*n* hex digits.
6. **Checksum**, two hex digits - the least significant byte of the two's complement of the sum of the values of all fields except fields 1 and 6 (Start code ":" byte and two hex digits of the Checksum). It is calculated by adding together the hex-encoded bytes (hex digit pairs), then leaving only the least significant byte of the result, and making a 2's complement (either by subtracting the byte from 0x100, or inverting it by XOR-ing with 0xFF and adding 0x01). If you are not working with 8-bit variables, you must suppress the overflow by AND-ing the result with 0xFF. The overflow may occur since both 0x100-0 and (0x00 XOR 0xFF)+1 equal 0x100. If the checksum is correctly calculated, adding all the bytes (the Byte count, both bytes in Address, the Record type, each Data byte and the Checksum) together will always result in a value wherein the least significant byte is zero (0x00).
   For example, on :0300300002337A1E
   03 + 00 + 30 + 00 + 02 + 33 + 7A = E2, 2's complement is 1E

There are six record types:

- **00**, *data record*, contains data and 16-bit address. The format described above.
- **01**, *End Of File record*, a file termination record. No data. Has to be the last line of the file, only one per file permitted. Usually every field except the record type (=01) and obviously the checksum field

is zero, which gives ':00000001FF'. Originally the *End Of File record* could contain a start address for the program being loaded, e.g. :00AB2F0125 would make a jump to address AB2F. This was convenient when programs were loaded from punched paper tape.

- **02**, *Extended Segment Address Record*, segment-base address (two hex digit pairs in big endian order). Used when 16 bits are not enough, identical to 80x86 real mode addressing. The address specified by the data field of the most recent 02 record is multiplied by 16 (shifted 4 bits left) and added to the subsequent 00 record addresses. This allows addressing of up to a megabyte of address space. The address field of this record has to be 0000, the byte count is 02 (the segment is 16-bit). The least significant hex digit of the segment address is always 0.
- **03**, *Start Segment Address Record*. For 80x86 processors, it specifies the initial content of the CS:IP registers. The address field is 0000, the byte count is 04, the first two bytes are the CS value, the latter two are the IP value.
- **04**, *Extended Linear Address Record*, allowing for fully 32 bit addressing (up to 4GiB). The address field is 0000, the byte count is 02. The two data bytes (two hex digit pairs in big endian order) represent the upper 16 bits of the 32 bit address for all subsequent 00 type records until the next 04 type record comes. If there is not a 04 type record, the upper 16 bits default to 0000. To get the absolute address for subsequent 00 type records, the address specified by the data field of the most recent 04 record is added to the 00 record addresses.
- **05**, *Start Linear Address Record*. The address field is 0000, the byte count is 04. The 4 data bytes represent the 32-bit value loaded into the EIP register of the 80386 and higher CPU.

There are various format subtypes:

- **I8HEX** or **INTEL 8**, 8-bit format. Allows usage of 00 and 01 records.
- **I16HEX** or **INTEL 16**, 16-bit format (supertype of 8-bit format). Allows usage of 00, 01, 02 and 03 records. The data field endianness may be byte-swapped.
- **I32HEX** or **INTEL 32**, 32-bit format (supertype of 16-bit format). Allows usage of 00, 01, 02, 03, 04 and 05 records. The data field endianness may be byte-swapped.

Beware! Byte-swapped data might be more confusing. It is possible to misinterpret the byte order in case of I16HEX and I32HEX.

A similar encoding, with slightly different ASCII formatting, termed SREC is used with Motorola processors.

Example

```
:10010000214601360121470136007EFE09D2190140
:100110002146017EB7C20001FF5F16002148011988
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
```

☐ Start code
☐ Byte count
☐ Address
☐ Record type
☐ Data
☐ Checksum

See also

- SREC - File format

External links

- Intel Hexadecimal Object File Format Specification 1988 (PDF) (http://microsym.com/editor/assets/intelhex.pdf) , Revision A, January 6, 1988
- Format description at PIC List (http://www.piclist.com/techref/fileext/hex/intel.htm)
- Format description (http://www.lucidtechnologies.info/intel.htm)
- SRecord (http://sf.net/projects/srecord) , multi-platform GPL'ed tool for manipulating EPROM load files.
- Binex (http://home.kpn.nl/newlife-software/Binex/binex.htm) , a converter between Intel HEX and binary.
- libgis (http://dev.frozeneskimo.com/software_projects/libgis) , open source library to handle Intel HEX (and more).
- SB-Projects: fileformats: intel hex (http://www.sbprojects.com/knowledge/fileformats/intelhex.php) , clear and well structured reference with various examples

Retrieved from "http://en.wikipedia.org/wiki/Intel_HEX"
Categories: Binary-to-text encoding formats | Embedded systems

- This page was last modified on 5 August 2011 at 23:54.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of use for details.
  Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.