Intel Hex-Format

Das **Intel HEX**-Format ist ein Datenformat zur Speicherung und Übertragung von binären Daten. Es wird hauptsächlich verwendet, um Programmierdaten für Mikrocontroller bzw. Mikroprozessoren, EPROMs und ähnliche Bausteine zu speichern. Das HEX-Format ist das älteste Datenformat seiner Art und seit den 1970er Jahren in Gebrauch. Es ist deutlich erkennbar auf die Erfordernisse der Intel-80x86-Prozessoren zugeschnitten.

Ein **Intel HEX**-File liegt im ASCII-Format vor. Die Bytes der kodierten Binärdaten werden jeweils als Hexadezimalzahl aus zwei ASCII-Zeichen (0...9 und A...F) dargestellt. HEX-Dateien können mit einem Texteditor geöffnet und mit etwas Erfahrung auch verstanden und modifiziert werden. Die HEX-Datei ist also deutlich größer als das enthaltene Binärprogramm. Die Datensätze sind mit einer Prüfsumme versehen, so dass Übertragungsfehler erkannt werden können.

Aufbau eines Datensatzes

Ein Datensatz (*Record*) ist eine Textzeile in der Datei. Er besteht aus sechs Teilen:

1. **Startcode** : Das ASCII-Zeichen ":" (Doppelpunkt, ASCII-Kodierung $3A_{HEX}$)
2. **Anzahl der Bytes** (*byte count*): Zwei Hexadezimalziffern, sie zeigen die Anzahl der Bytes im Datenfeld an. Hier wird üblicherweise 16 (10_{HEX}) oder 32 (20_{HEX}) gewählt, es sind aber alle anderen Datenfeldlängen möglich.
3. **Adresse**: Vier Hexadezimalziffern repräsentieren eine 16bit-Adresse (in Big-Endian), also den Ort im Speicher des Mikrocontrollers, an dem die Daten abgelegt werden sollen. Um die Begrenzung auf 64kByte durch diesen Adressumfang aufzuheben, gibt es Datensatztypen, in denen ein Adressoffset übertragen wird.
4. **Datensatztyp** (*Record type*): Zwei Hexadezimalziffern. Es gibt sechs Datensatztypen (*00* bis *05*), welche die Art des Datenfeldes festlegen.
5. **Datenfeld**: *n* aufeinanderfoldende Bytes werden als *2n* Hexadezimalziffern dargestellt.
6. **Prüfsumme**: Sie ist das niederwertige Byte des Zweierkomplements der Summe aus den Bytes aus allen Teilen des Datensatzes, ausschließlich des Startcodes und der Prüfsumme selbst (siehe Berechnung der Prüfsumme).

Die Bytereihenfolge ist *big endian*, d.h. das höchstwertige Byte wird als erstes, also auf der kleineren Adresse gespeichert.

Datensatztypen

Übersicht

Es gibt sechs Datensatztypen (*record types*):

Typ	Bezeichnung	Verwendung
00	Data Record	Binärdaten
01	End of File Record	Dateiende
02	Extended Segment Address Record	Real Mode-Adressierung

03	Start Segment Address Record	CS:IP Register
04	Extended Linear Address Record	32Bit-Adressierung
05	Start Linear Address Record	EIP-Register

Data Record

Typ **00**: Er enthält die 16bit-Adresse und ein Datenfeld mit Binärdaten.

	Startcode	Anzahl der Bytes	Adresse	Typ	Datenfeld	Prüfsumme
Länge	1 Zeichen	2 Ziffern	4 Ziffern	2 Ziffern	2*n* Ziffern	2 Ziffern
Inhalt	:	*n*	*Adresse*	00	*Daten*	*Prüfsumme*

n: Anzahl der Bytes im Datenfeld
Adresse: 16bit-Adresse für die Speicherung des Datensatzes
Daten: Datenfeld, n Bytes
Prüfsumme: Siehe Berechnung der Prüfsumme

Extended Linear Address Record

Typ **04**: Dieser Datensatztyp unterstützt die volle 32bit-Adressierung. Das Datenfeld enthält die oberen 16 Bit einer 32bit-Adresse. Dieser Wert wird den Adressen der darauf folgenden Typ *00* Datensätze vorangestellt. Das Adressfeld ist immer *0000*, der *byte count* ist immer *02*.

	Startcode	Anzahl der Bytes	Adresse	Typ	Datenfeld	Prüfsumme
Länge	1 Zeichen	2 Ziffern	4 Ziffern	2 Ziffern	4 Ziffern	2 Ziffern
Inhalt	:	02	0000	04	*Adresse (high word)*	*Prüfsumme*

Adresse (high word): Die 16 höchstwertigen Bit der 32bit-Adresse
Prüfsumme: Siehe Berechnung der Prüfsumme

Start Linear Address Record

Typ **05**: Der Datensatz enthält den Wert (4 Byte) des EIP-Registers der Prozessoren ab dem 80386. Das Adressfeld ist immer *0000*, der *byte count* ist immer *04*

	Start code	Anzahl der Bytes	Adresse	Typ	Datenfeld	Prüfsumme
Länge	1 Zeichen	2 Ziffern	4 Ziffern	2 Ziffern	8 Ziffern	2 Ziffern
Inhalt	:	04	0000	05	*EIP*	*Prüfsumme*

EIP: Inhalt des EIP-Registers (32 bit)
Prüfsumme: Siehe Berechnung der Prüfsumme

Berechnung der Prüfsumme

Die Prüfsumme wird aus dem gesamten Datensatz ausschließlich des Startcodes und der Prüfsumme selbst berechnet. Der Datensatz wird byteweise summiert, von der Summe wird das niederwertige Byte genommen und davon wiederum das Zweierkomplement gebildet.

Das Zweierkomplement wird gebildet, indem man die Bits des niederwertigen Bytes invertiert und dann 1 addiert. Dies kann man z.B. durch die Exklusiv-Oder-Verknüpfung mit FF_{HEX} und Addition von 01_{HEX} erreichen. So bleibt 00_{HEX} unverändert, aus 01_{HEX} wird FF_{HEX} u.s.w.

Das Zweierkomplement drückt im Binärsystem eine negative Zahl aus. Da die Prüfsumme damit die negative Summe der restlichen Bytes darstellt, gestaltet sich die Überprüfung eines Datensatzes auf Fehler sehr einfach. Man summiert einfach die einzelnen Bytes eines Datensatzes *inklusive* der Prüfsumme und erhält als niederwertiges Byte 00_{HEX}, falls der Datensatz korrekt ist.

Beispiel

```
:020000021000EC
:10010000214601360121470136007EFE09D2190140
:100110002146017EB7C20001FF5F16002148011988
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
```

Startcode
Byte count
Adresse
Typ
Datenfeld
Prüfsumme