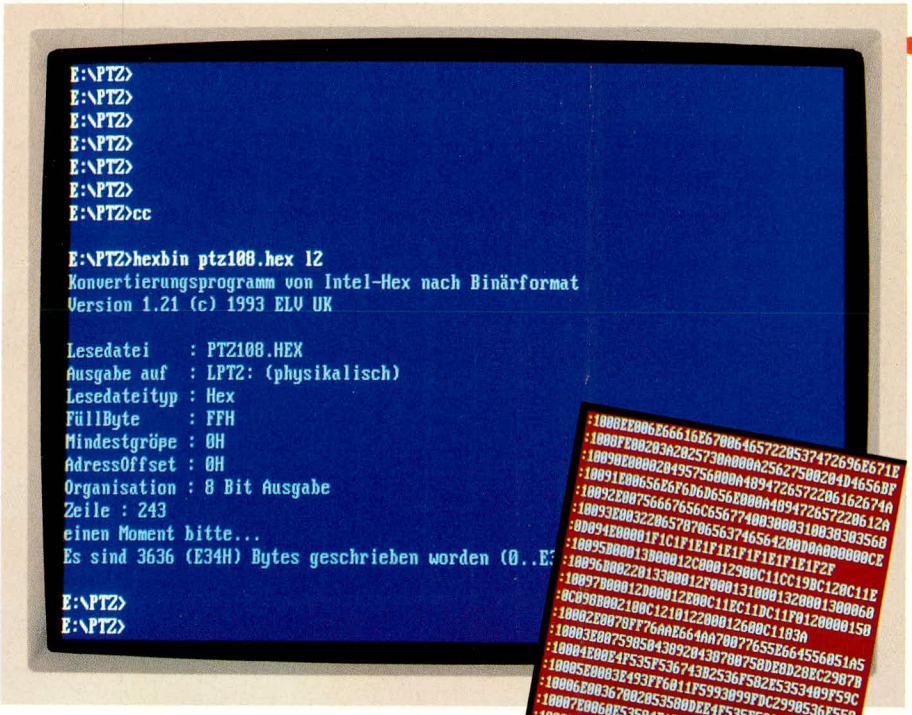

Hex-Binär-Konverter

Unterschiedliche Objektformate wie Intel-Hex, Motorola S 1 bis S 3, Tektronix-Hex und MOS-Hex lassen sich auf einfache Weise mit dieser Konvertierungssoftware in ein Binärformat umwandeln. Außerdem kann die Software den im „ELVjournal" 3 und 4/93 vorgestellten EPROM-Simulator direkt mit den konvertierten Daten beschreiben.

Allgemeines

Viele Cross-Assembler und Compiler für Mikrocontroller liefern nach der Übersetzung eine Hex- oder auch Objekt-codierte Ausgabedatei. Hierbei handelt es sich um ASCII-Dateien, in denen der eigentliche Programmcode verschlüsselt vorliegt. Allen Objektdateien gemeinsam ist eine Startkennung mit der jede Zeile beginnt sowie eine Prüfsumme mit der ein verarbeitendes Programm feststellen kann, ob die Daten in der gerade bearbeiteten Zeile korrekt empfangen wurden. Mikrocontroller benötigen zum Betrieb aber ein Binärformat, in dem der Code direkt hintereinander abgelegt ist. Das erste Byte wird der Adresse 0, das zweite Byte der Adresse 1 usw. zugeordnet. Der Inhalt einer solchen Datei kann direkt in den Speicher eines EPROM-Simulators oder auch über ein Programmiergerät in einem EPROM gespeichert werden.

Darüber hinaus läßt die vorgestellte Konvertierungssoftware eine Offset-Verschiebung der Ausgabedaten, die Festlegung einer Mindestgröße bzw. die für einen EPROM-Typ genau passende Dateigröße oder auch die Einstellung des Auffüllwertes für nicht benutzte Speicherbereiche zu.

Ausgabeseitig kann noch eine 16-Bit-Trennung vorgenommen werden, entsprechend einer Zusammenfassung aller geraden bzw. ungeraden Adressen zu einer Datei. Diese 16-Bit-Trennung wird zum Betrieb der meisten 16-Bit-Prozessoren benötigt.

Die Ausgabe der Daten kann nun wahlweise auf eine Datei, auf eine der logischen Schnittstellen oder auch direkt auf eine der angegebenen Parallelschnittstellen erfolgen.

Zum Laden des EPROM-Simulators EPS 1000 ist die letztgenannte Möglichkeit der schnellste Weg, um die Daten in das Simulator-RAM zu schreiben. Gleichzeitig prüft die Software das Vorhandensein des EPS 1000 und meldet dies in einer Ausgabezeile.

Tabelle 1: Das Intel-Intellec-8-(86) MDS-Format (Intel-Hex)

Aufbau jeder Zeile:
:CCAAAATTDDDD...DDPP<CR><LF>

:	=	Startcode jeder Zeile
CC	=	Anzahl der Datenbytes
AAAA	=	Adresse für das erste Daten-Byte
TT	=	Recordtyp:
		„00" für Datenrecord
		„01" für Endrecord
		„02" für Extended Adreß Record
		„03" für Start Adreß Record
DD	=	ein Datenbyte
PP	=	Prüfsumme (8 Bit)
<CR>	=	Carriage Return (Zeilenanfang)
<LF>	=	Line Feed (neue Zeile)

Tabelle 2: Das Motorola-Exorciser- (S1-S3) Format

Struktur jeder Zeile :
STCCAAAADDDD...DDDDPP<CR><LF>

S	=	Startcode jeder Zeile
T	=	Recordtyp:
		„0" = Komentarrecord
		„1" = Datenrecord
		„2" = 24-Bit-Adreßrecord
		„3" = 32-Bit-Adreßrecord
		„7" = Endrecord
		„8" = Endrecord
		„9" = Endrecord
CC	=	Anzahl der Datenbytes
AAAA	=	16,24, oder 32 Bit-Adresse für das erste Daten-Byte
DD	=	ein Datenbyte
PP	=	Prüfsumme (8 Bit)
<CR>	=	Carriage Return (Zeilenanfang)
<LF>	=	Line Feed (neue Zeile)

Dateiformate

Tabelle 1 zeigt in übersichtlicher Form die Codierung des Intellec-8(86)/MDS-Formates (Intel-Hex). Dieses Standard-Format läßt einen Adreßbereich von 64 kByte zu, wurde aber später auf die Bedürfnisse der 16-Bit-Prozessorfamilie hin erweitert.

In Tabelle 2 ist die Codierung des Motorola Exorciser S 1 bis S 3-Formates dargestellt. Das S 1-Format ist für einen 16 Bit-Adreßbereich ausgelegt, während das S 2-Adreßformat mit einem 24 Bit und das

Tabelle 3: Das Tektronix-Hexadezimal-Format

Struktur jeder Zeile:
/AAAACCP1DDDD...DDDDP2<CR><LF>

/	=	Startcode jeder Zeile
AAAA	=	Adresse für das erste Daten-Byte
CC	=	Anzahl der Datenbytes
P1	=	erste Prüfsumme über Adresse und Anzahl (8 Bit)
DD	=	ein Datenbyte
P2	=	Prüfsumme über die Datenbytes (8 Bit)
<CR>	=	Carriage Return (Zeilenanfang)
<LF>	=	Line Feed (neue Zeile)

Tabelle 4: MOS-Technology-Hex-Format

Struktur jeder Zeile:
; CCAAAADDDD..DDDDPPPP<CR><LF>

;	= Startcode
CC	= Anzahl der Datenbytes
AAAA	= Adresse für das erste Datenbyte
DD	= ein Datenbyte
PPPP	= 16 Bit Prüfsumme
<CR>	= Carriage Return (Zeilenanfang)
<LF>	= Line Feed (neue Zeile)

S 3-Format mit einem 32 Bit-Adreßbereich arbeitet.

Tabelle 3 zeigt die Codierung des Tektronix Hexadezimal-Formats. Auch hier wird der Adreßbereich von maximal 64 kB unterstützt, wobei auffällt, daß pro Zeile 2 Prüfsummen Verwendung finden.

Das weniger gebräuchliche MOS-Technology-Hex-Format (Tabelle 4) unterstützt ebenfalls einen 16-Bit-Adreßraum, wobei anstatt einer 8-Bit- eine 16-Bit-Prüfsumme verwendet wird.

Programmaufruf

Das Hex-Binär-Konvertierungsprogramm ist ausschließlich als Kommandozeilenversion konzipiert und läuft auf allen IBM-PC, XT, AT und Kompatiblen. Das Programm muß mit mindestens einem Parameter, nämlich dem Namen der einzulesenden Datei aufgerufen werden.

Der Aufruf sieht z. B. wie folgt aus: HEXBIN <Dateiname>. Vor dem Dateinamen kann auch eine Laufwerks- und/oder Pfad-Angabe stehen.

Tabelle 5 zeigt eine Kurzübersicht der verfügbaren Parameter.

Defaultmäßig sucht das Programm nach einem Dateinamen mit der Endung <.HEX>, <.MS1>, <.MS2>, <.MS3> oder <.TEK>. Abweichend davon kann diese natürlich auch mit angegeben werden.

Weiterhin lassen sich zusätzliche Parameter beim Programmaufruf anfügen. Die Ausgabe erfolgt defaultmäßig auf die erste physikalische Schnittstelle LPT 1. Wird hier der EPROM-Simulator EPS 1000 an der physikalischen Schnittstelle nicht gefunden, so erfolgt die Ausgabe auf den angegebenen Dateinamen mit der Extension <.BIN>. Durch die Angabe von **L1**, **L2** oder **L3** ist ein Umlenken der Ausgabe auch explizit auf eine der 3 physikalischen Schnittstellen LPT 1 bis LPT 3 möglich.

Die Ausgabe kann auch auf die logischen Schnittstellen **LPT 1**, **LPT 2** oder **LPT 3** umgeleitet werden. In diesem Fall wird direkt auf die entsprechende DOS-Schnittstelle geschrieben, so daß z. B. in Netzwerkanwendungen die Ausgabe auch über den Netzwerk-Druckertreiber erfolgen kann.

Durch die Angabe des Parameters **<D>**, gefolgt von einem Dateinamen kann die Ausgabe auch auf eine beliebige Datei optional mit Laufwerk- und/oder Pfad-Angaben erfolgen.

Die Parameter **<HEX>** oder **<BIN>** teilen dem Programm explizit mit, daß es sich bei der vorhandenen Datei unabhängig von der Endung um eine Intel/Motorola/Tektronix-Hex-Datei oder um ein Binär-Format handelt. Letztgenannte Dateiform wird nicht mehr konvertiert, sondern unverändert übernommen.

Die Angabe des Parameters **<B>**, gefolgt von einer zweistelligen hexadezimalen Zahl 00 bis FF bestimmt, mit welchem Byte nicht genutzte Speicherbereiche aufzufüllen sind. Defaultmäßig wird programmseitig hier FFH eingesetzt.

Der Parameter **<T>**, gefolgt von einer EPROM-Typenbezeichnung 2716 bis 27512 legt für die Ausgabedatei eine Mindestgröße fest. Diese ist dann je nach EPROM-Typenbezeichnung 2 kByte bis 64 kByte groß. Nicht benutzte Speicherbereiche werden, wie bereits beschrieben, aufgefüllt.

Alternativ zu vorstehenden Angaben kann auch der Parameter **<G>**, gefolgt von einer bis zu 4stelligen Hexadezimalzahl 0 bis FFFF für die Mindestgröße der Ausgabedatei genutzt werden oder aber, wie beschrieben, der Parameter <T> Einsatz finden.

Durch die Angabe eines Parameters **<O>**, gefolgt von einem <+> oder <-> und einer bis zu 8stelligen Hexadezimalzahl läßt sich ein Offset zu jeder Adresse addieren bzw. davon subtrahieren. Hierdurch ist es ohne weiteres möglich, eine entsprechende Adreßverschiebung vorzunehmen.

Die Angabe eines negativen Offsetwertes kann erforderlich sein, wenn das Programm z. B. für den Adreßbereich 2000H bis 4000H assembliert wurde, und das EPROM ab der Adresse 2000H im Adreßbereich des Zielprozessors liegt. In diesem Fall muß der Programmcode, der später ab Adresse 2000H erreichbar sein soll, in das EPROM unter der Adresse 0 geschrieben werden. Hier müßte also die Angabe zu dem Offset-Parameter <-2000> erfolgen.

Im umgekehrten Fall kann es erforderlich sein, einen positiven Offsetwert anzugeben, um z. B. eine Binär-Datei im EPROM ab der Adresse 8000H abzulegen. In dem Fall ist der Parameter <+8000> anzugeben.

Die Angabe des Parameters **<SG>**, **<SU>**, **<SE>** oder **<SO>** ermöglicht eine 16-Bit-Trennung der Daten. Mit <SG> bzw. <SE> werden alle geraden (Even) Adressen und über <SU> oder <SO> werden alle ungeraden (odd) berücksichtigt. Hierdurch erfolgt eine 16-Bit-Trennung, wie sie für Controller oder Prozessoren mit einem 16-Bit-Datenbus erforderlich ist. Hierzu wird dann das Programm zweimal nacheinander mit unterschiedlichen Parametern aufgerufen.

Sollten in der Hex-Datei Prüfsummenfehler vorhanden sein, so kann der Abbruch des Programmes durch den Parameter **<NC>** verhindert werden. Es folgt eine Addition der Prüfsummenfehler und die Ausgabe nach dem Programmende.

Der Aufruf des Programms kann auch komplett über die automatische Batch-Verarbeitung von DOS erfolgen (Beispiel: Tabelle 5 unten).

Die Hex-Binär-Konvertierungssoftware stellt in idealer Weise das Bindeglied zwischen einem Cross-Assembler bzw. Compiler und dem ELV-EPROM-Simulator oder auch einem EPROM-Programmiergerät dar und erleichtert somit erheblich die Entwicklung von Mikroprozessor- bzw. Mikrocontroller-Schaltungen. ELV

Tabelle 5 zeigt den Programmaufruf und die möglichen Parameter

Programmaufruf:
HEXBIN [Laufwerk] [Pfad] Dateiname [.EXT]

Parameter:

L1	L2	L3	:	Ausgabe auf die physikalische Schnittstelle LPT1..3
LPT 1	LPT2	LPT3	:	Ausgabe auf die logische Schnittstelle LPT1..3
D [Dateiname]			:	Ausgabe auf eine Datei mit optionaler Namensangabe
HEX			:	Interpretation als Intel-Motorola- oder Tektronixs-Hex-Datei
BIN			:	Interpretation als Binär-Datei (keine Konvertierung)
B00..FF			:	nicht genutzten Speicherbereich mit 00..FFH auffüllen
T2716..27512			:	Mindestgröße der Ausgabedatei je nach EPROM-Typ
G0..FFFF			:	Mindestgröße der Ausgabedatei in Bytes (Hex)
O+-0..7FFFFFFF			:	Offsetangabe für den Adreßbereich
SG, SE			:	nur alle geraden (even) Adressen berücksichtigen
SU, SO			:	nur alle ungeraden (odd) Adressen berücksichtigen
NC			:	Prüfsummenfehler zählen und bei Programmende ausgeben

Beispiel:
HEXBIN TEST
HEXBIN C:\TEST.HEX LPT1 B00 O-1000