ESCO-Texteditor und -Assembler

Kurzbeschreibung

ESCO
EUROPA SINGLE-BOARD COMPUTER

Vorläufig

ESCO-Texteditor und -Assembler

Inhaltsverzeichnis

ESCO-Texteditor und -Assembler

1. Einführung

Der ESCO-Texteditor und -Assembler ermöglicht die Erstellung und Änderung von Alphanumerischen Daten-Texten sowie die Assemblierung von beliebig langen Texten.

Die zu editierenden Texte können sich sowohl im Hauptspeicher als auch auf der Kassette befinden.

Die wesentlichen Eigenschaften sind:

- Zeilenorientierte Text-Eingabe und -Änderung,
- Automatische Schnittstelle zum ESCO-Assembler,
- Assemblierung von Laufspeicher-residenten sowie beliebig langen, in Segmente aufgeteilten Quelltexten, die sich auf der Kassette befinden.

1.1 Texteditor und Assembler-Kommandos

Übersicht

I	Eingabe des Textes von der Kassette
O	Ausgabe des Textes auf die Kassette
E	Exit zum ESCO-Monitor, Edit-Modus verlassen
?	Setze ? als Promting-Zeichen
? x	Setze x als Promting-Zeichen (x-beliebig)
F xyz	Auffinden und Ausgeben aller Zeilen, die den String xyz enthalten
S	Editor gibt Status des Textes aus: - Anfangs- und Endadresse (hexadezimal) - Anzahl der Zeilen (dezimal)
R	Renumerierung der Zeilen mit dem Schritt 5
Q	Zurück zur TEXT: Abfrage zur Angabe der neuen Adresse für den Arbeitsbereich

P A	Ausgabe aller Zeilen auf Printer
T A	Ausgabe aller Zeilen auf Terminal
P n	Ausgabe ab Zeile n auf Printer
T n	Ausgabe ab Zeile n auf Terminal
A P	Assembler 1. Pass mit Ausgabe auf Printer
A T	Assembler 1. Pass mit Ausgabe auf Terminal
L	Assembler 2. Pass mit Ausgabe auf Printer
D	Assembler 2. Pass mit Ausgabe auf Terminal
G	Fortsetzung des Assemblierens mit dem nächsten Textsegment. Das Segment befindet sich im Arbeitsbereich des Editors (z.B. das Fortsetzungssegment wurde vor der Kassette eingelesen). G wird als Fortsetzung in allen Assembleraufrufen benutzt - im 1. und 2. Pass und unabhängig vom Ausgabegerät.
I	Eingabe des Textes von der Kassette
O	Ausgabe des Textes auf die Kassette

2. ESCO-Texteditor

2.1 Start und Arbeitsweise

Der ESCO-Texteditor arbeitet zeilenorientiert. Die Zeilennummern müssen bei der Eingabe vorgegeben werden. Sie liegen im Bereich von 1 bis 9999.

Der Editor wird gestartet mit F100 G. Daraufhin gibt er

TEXT:

aus und wartet auf die Angabe der Anfangsadresse des Arbeitsbereichs. Die Angabe kann auf zwei Arten angegeben werden:

- hexadezimale Adresse mit anschließendem Carriage Return (CR),
- nur CR: in diesem Fall wird diejenige Adresse genommen, die zuletzt auf TEXT: Anfrage angegeben wurde. Editor arbeitet dann weiter im gleichen Bereich, auch wenn der Editormodus mit E (Exit) zwischendurch verlassen wurde.

Daraufhin gibt der Editor

N OR O ?

aus und erwartet Angabe N oder O.

- N: im Arbeitsbereich soll ein neuer Text erstellt werden,
- O: der Text befindet sich bereits im Editor-Bereich, entweder von der Kassette eingelesen oder im Hauptspeicher, mit Hilfe des Editors aufgebaut. Zur Information über den vorhandenen Text gibt der Editor den Status aus:

- Anfangs- und Endeadresse des Textes (hexadezimal),
- Anzahl der Zeilen (dezimal).

Man befindet sich danach im Editormodus. Im Editormodus werden folgende Funktionen zur Verfügung gestellt:

- Textaufbau und -Änderung,
- Start der kommandogesteuerten Funktionen (siehe Übersicht). Alle Kommandos werden mit CR abgeschlossen. Bei falschen Kommandoaufrufen gibt der Editor

  BAD COM

  aus und wartet auf die nächste Eingabe.

2.1.1 Zeilenerstellung und Zeilenänderung

Die Funktionen zur Zeilenerstellung und -änderung sind:

- Zeile erstellen: Zeilennummer Zwischenraum neuer Text CR
- Zeile überschreiben: Zeile neu erstellen
- Zeile löschen: Zeilennummer CR
- Zeichen zurücksetzen: Backspace

2.1.2 Text-Protokoll

Das Text-Protokoll wird mit den Kommandos P A, T A, P n und T n erstellt (siehe Übersicht). Am Ende des Textes gibt der Editor *ET aus. Mit der BREAK-Taste kann die Protokollierung abgebrochen werden.

2.1.3 Kassetten-Ein- und -Ausgabe

Texttransfer zwischen dem Editor-Arbeitsbereich und der Kassette wird mit Hilfe von Kommandos

- I für die Eingabe,
- O für die Ausgabe

gestartet.

Der Editor fordert zusätzliche Informationen über die Transferrate und über das Blockkennzeichen mit

R/I:

an.

Sie können auf zwei Arten angegeben werden:

- vierstellige hexadezimale Zahl:
  - o die ersten zwei Ziffern geben Auskunft über die Transferrate: 01 für die einfache und 03 für die dreifache Geschwindigkeit.
  - o die nächsten zwei Ziffern geben Auskunft über das Blockkennzeichen. Das Blockkennzeichen kann zwischen 01 und FE liegen.
- nur CR:
  Editor setzt Standard-Werte ein:
  - o dreifache Geschwindigkeit und 01 als Blockkennzeichen

Nach dem Transfer werden folgende Meldungen ausgegeben:

- *ET beim erfolgreichen Transfer,
- ERRORS = *ET beim Fehler.

3. ESCO-Assembler

3.1 Start und Arbeitsweise

Der ESCO-Assembler ist ein Zwei-Pass-Assembler, der zur Programmerstellung für alle 6502-Mikroprozessor-Systeme benutzt werden kann.

Der Aufruf des Assemblers kann im Editor-Modus erfolgen. Als Quelle wird der von Editor erstellte Text benutzt.

Der erste Pass wird gestartet mit A P, wenn die Ausgabe auf Printer bzw. A T, wenn die Ausgabe auf Terminal erfolgen soll. Nach dem Start meldet sich der Assembler mit einem Dialog und fordert Adreß-Angaben für die Symboltabelle und den Objektcode an.

SYMS: Anfang der Symboltabelle - Standard 0300
SYME: Ende der Symboltabelle - Standard 04FF
CODE: Page-Anfangsadresse für den Objektcode.

Jede Angabe wird mit CR abgeschlossen.
Sollen Standardwerte eingesetzt werden, so genügt CR. Der Assembler nimmt 8 Bytes für ein Symbol in Anspruch. Standardmäßig wird in 200 Bytes (300-4FF) Platz für 64 Symbole in der Symboltabelle zur Verfügung gestellt.

Der Assembler bietet die Möglichkeit der Erstellung des lauffähigen Codes für eine Ablaufadresse, die sich von der zur Verfügung stehenden Ablageadresse unterscheidet. Dies ist besonders zur Assemblierung von PROM-Programmen geeignet. In diesem Fall wird der Assembler-Ausdruck der Ablaufadresse angepaßt und ein auf diese Adresse ablauffähiger Objektcode aufgebaut.

Die folgende Tabelle gibt Auskunft über die vorhandenen Möglichkeiten je nach CODE: und ORG (* = Adresse) Anweisungen.

Angaben CODE:	ORG	Ablageadresse	Ablaufadresse
-	-	0500	0500
-	ADR_O	ADR_O	ADR_O
ADR_C	-	ADR_C	ADR_C
ADR_C	ADR_O	ADR_C	ADR_O

- bedeutet, daß bei CODE: - nur CR eingegeben wurde,
  bei ORG - im betreffenden Abschnitt keine ORG-Anweisung vorkommt.

- ADR_C - bei CODE: spezifiziert
- ADR_O - in der ORG-Anweisung spezifiziert.

Nach Ende des Dialogs läuft der 1. Pass an. Es empfiehlt sich, den ersten Pass so oft zu wiederholen, bis alle dort gemeldeten Fehler beseitigt sind. Den Fehler # 30 - undefiniertes Symbol - kann erst im zweiten Pass beseitigt werden.

Der zweite Pass wird mit L bzw. D erst gestartet, wenn sich der Assembler nach dem ersten Pass mit END OF ASSEMBLY gemeldet hat. Dazu sind keine weiteren Adreßangaben erforderlich.

3.1.1 Instruktionsformat

Das Instruktionsformat hat den folgenden Aufbau:

<MARKE> OPCODE <OPERANDEN> <KOMMENTAR>.

Marke ist ein 6-stelliger alphanumerischer String, der mit einem Alphazeichen anfangen muß und darf nicht wie einer der 55 Op-Codes und nicht A, S, P, X oder Y heißen. Die Marke kann in jeder Spalte anfangen.

Der Operandenteil kann aus einem Ausdruck von Symbolen und Werten bestehen. Er stellt entweder den Wert (mit Prefix #) oder Adresse dar.

Jede Zeichenkette nach dem Operandenfeld wird als Kommentar ausgewertet.
Steht der Kommentar am Zeilenanfang, so muß /* davor gesetzt werden.

Der Quelltext endet mit der Assembler-Anweisung .END.

3.1.2 Konstante

Die Konstante wird mit fünf Darstellungsarten vorgegeben:

dezimal	kein Prefix
hexadezimal	$
octal	@
binär	%
ASCII Character	' '

3.1.2 Wert-, Programmzähler- und Speicherzuweisung

. BYTE reserviert ein Speicherbyte und weist ihm den spezifizierten Wert zu. Sind mehrere Operanden in einer Anweisung vorhanden, so werden sie in die aufeinanderfolgenden Bytes abgelegt. Zeichenketten in ESCCI Code werden in Apostrophen eingeklammert.

. WORT reserviert zwei Bytes und weist den spezifizierten Wert zu. ASCII-Zeichen können mit dieser Anweisung nicht geladen werden.

= EQUATE-Anweisung reserviert Speicherplatz.

* = * + n	reserviert n-Speicherplätze
* = Adresse	ORG oder Programmzähleranweisung setzt den Programmzähler auf Adresse

Assembler-Fehlerliste

ERROR	# 2	- Marke mehrmals definiert
	# 3	- Ilegaler oder fehlender OP-Code
	# 4	- Adresse nicht gültig
	# 5	- Akkumulator-Modus nicht erlaubt
	# 6	- Vorwärts-Referenz in Byte oder Word
	# 7	- Unvollständige Zeile
	# 8	- Marke beginnt nicht mit Buchstaben
	# 9	- Marke länger als sechs Zeichen
	# 10	- Marke enthält Sonderzeichen
	# 11	- Vorwärts-Referenz in der Adreßzuweisung oder ORG
	# 12	- Ungültiger Text - soll x oder y sein
	# 13	- Ungültiger Ausdruck im Operand
	# 15	- Ungültiger Operand für Null-Page-Modus
	# 16	- Ungültiger Operand für absolute Modus
	# 17	- Branch außerhalb des Bereichs (0 - 128)
	# 18	- Ilegaler Operandentyp für diese Anweisung
	# 19	- Außerhalb der Grenzen der indirekten Adressierung
	# 20	- Marke ist eines der Zeichen A, X, Y, S und P
	# 21	- Programmzähler negativ
	# 20	- Ungültiges Zeichen - mit Ausnahme "=" für ORG
	# 30	- Undefiniertes Symbol