ANLEITUNG JOGI MON 2.Ø (C) 1985 BY JOGI SOFT

Dieses ist ein Maschinensprache-Monitor fuer den ernsthaften Programmierer
Er besteht aus 2 Eprom's die relokativ sind, und somit ueberall laufen.
Benoetigt wird das Grundprogramm ab Version 4.Ø
Mit dem Monitor ist es moeglich Hex-Dump's von Programmen und Daten in hexadezimaler Form ein- und auszugeben, zu verschieben, zu suchen, zu fuellen, zu disassemblieren, zu editieren, zu assemblieren, zu laden, zu speichern und vieles andere mehr. Der Monitor ist somit fuer den Profi genauso gut wie fuer den Anfaenger, da man ueberhaupt nicht mehr im Grundprogramm hin und her muss.

STARTEN DES MONITOR:
Als erstes wird die Bibliothek angewaehlt. Wenn dort steht 'JOGI MON', dann druecken Sie 'J' fuer ja und der Monitor wird gestartet.
Zunaechst erscheinen die Register DØ-D7/AØ-A7/STATUS.

Der Monitor ist ein Zeilenmonitor, das heisst, dass die Befehle nicht ueber ein Menue, sondern direkt auf dem Bildschirm eingegeben werden muessen. Dieses hat den Vorteil, dass man auf einem Blick sieht, was man vorher getan hat, man kann so Funktionen gleich noch einmal wiederholen und somit genau sehen was sich geaendert hat.

==========

XXXX = Startadresse YYYY = Endaresse ZZZZ = Zieladresse
ZZ = Operant, Symbol oder 'Text'
(XXXX) = Startadresse (nicht angegeben wird der momentane Textstart genommen)

Der Monitor besitzt 28 Funktionen:

Befehl:	Beschreibung der Befehle:
A (XXXX)	Assemblieren des Texteditors
B	Bibliotheks Funktion (J = Start, CR = weiter, M - Mon.)
C XXXX,YYYY,ZZZZ	Convert. verschieben von Speicherbereichen von,bis,nach
D XXXX,YYYY	Disassemblieren von Maschinenprogrammen
E (XXXX)	Editieren von Texten (normaler Editor)
F XXXX,YYYY,ZZ	Suchen von HEX- oder ASCII-Folgen
G (XXXX)	Starten von Programmen
H	HELP anzeigen aller Funktionen
I XXXX,YYYY	ASCII-Dump ausgabe von ASCII-Folgen
J	Speicherbereiche
K 'Text := XXXX'	Zuweisung von Symbolen
L (XXXX)	Laden von Kassette, egal ob Text oder Daten
M XXXX,YYYY	HEX-Dump hexadezimale.Ausgabe des Speichers
NN XXXX	Textstart neu (loescht den Textspeicher)
NA XXXX	Textstart alt
O XXXX,YYYY,ZZ	Fuellen von Speicherbereichen (nur BYTE-Laenge)
P CRT	Ausgabe auf den Bildschirm
P LST	Ausgabe auf Drucker
P LST1	Ausgabe auf Drucker
P LST2	Ausgabe auf Drucker ohne LF
P NIL	Ausgabe beim assemblieren nur mit Fehler
Q	Ausgabe der Symboltabelle
R	Register anzeigen
SD XXXX,YYYY,'Name'	Speichern Daten auf Kassette
ST XXXX,'Name'	Speichern Text auf Kassette
T (XXXX)	Texteditor ausdrucken
U XXXX,YYYY,V1,V2	Grundprogramme auf andere Versionen anpassen Ø=3.Ø 1=3.1 2=4.2 3=4.3 4=Trap Siehe Sonderanleitung
V (XXXX)	Verify Kassette pruefen
W	Symboltabelle loeschen
X	<< EXIT >> Monitor beenden
YT XXXX,YYYY	Eprom testen ob leer
YR XXXX,YYYY,ZZZZ	Eprom lesen
YW XXXX,YYYY,ZZZZ	Eprom programmieren
Z	Starten des JOGI DOS (wenn vorhanden)

Symboltabelle:

Es wird die Symboltabelle ausgegeben.

Register:

Es werden die Register angezeigt wie sie nach einem Programm stehen.

Save auf Kassette:

SD #A000,#B000,'Name'

Speichern von Daten.

ST #10000,'Name'

Speichern von Texten.

Text drucken:

#A000

Text ab #A000 drucken.

Momentanen Text drucken.

Versionsanpassung:

#A000,#B000,1,3

Anpassen eines Programmes von #A000-#B000
das Orginal laeuft auf Version 3.1
das Angepasste laeuft nun auf Version 4.3

Es werden die Einsprungadressen <JSR @> umgeschrieben.
Die Eingabe sieht so aus:

Startadresse,Endadresse,alte Version,neue Version

Alte Version:
Hier wird die Nummer des Grundprogramms eingegeben, unter der das anzupassende Programm geschrieben wurde.
0 fuer 3.0 , 1 fuer 3.1 , 2 fuer 4.2 , oder 3 fuer 4.3

Neue Version:
Hier wird die Nummer des Grundprogramms eingegeben, unter der das anzupassende Programm laufen soll.
0 fuer 3.0 , 1 fuer 3.1 , 2 fuer 4.2 , 3 fuer 4.3 oder 4 um auf Trap #1 umzuschreiben.

(Nach der Aenderung auf Trap #1 ist das uebersetzte Programm nicht immer 100% lauffaehig, da Trap #1 einige Register zerstoert)

Achtung:
Mit dem Grundprogramm auf $E0000 funktioniert das Anpassen nur auf Trap #1

Verify von Kassette:

#A000

Vergleichen mit Speicher ab #A000 ob richtig Abgespeichert wurde.

Vergleichen mit Kopfadresse.

Symboltabelle loeschen:

Es wird die Symboltabelle geloescht.

Exit:

```
X
```

Ruecksprung ins Grundprogramm.

Eprom:

```
YT $0,$1FFF
```

Testen ob Eprom leer.

```
YR $0,$1FFF,$A000
```

Lesen des Eprom nach $A000

```
YW $A000,$BFFF,$0
```

Programmieren des Eprom ab $A000-$BFFF nach $0
danach grosses 'B' fuer bereit druecken.

Starten des JOGI DOS:

```
Z
```

Es wird (falls vorhanden) das JOGI DOS gestartet.

Der Monitor ersetzt fast vollstaendig das Grundprogramm von Rolf-Dieter Klein, nur der Debugger konnte aus Platzmangel nicht mit uebernommen werden.

BEISPIELE ZU DEN BEFEHLEN:

Assemblieren:

```
A $10000
```

Es wird der Text ab $10000 assembliert.

```
A
```

Es wird der Text an dem momentanen Textstart assembliert.

Bibiothek:

```
B
```

Es wird die Bibliothek ausgegeben.
z.B

```
JOGI MON
00C020
004000
START? J=JA CR=WEITER
```

'J' Programmstart
'CR' Ausgabe naechste Bibliothek
'M' Monitor

Verschieben:

```
C $A000,$B000,$C000
```

Es werden die Daten von $A000-$B000 nach $C000-$D000 verschoben.
Es kann auch von $A000-$B000 nach $A100-$B100 verschoben werden.

Disassemblieren:

```
D $A000,$B000
```

Es werden die Daten von $A000-$B000 als Assemblercode ausgegeben.
Das Listen kann mit jeder Taste gestoppt und weitergefuert werden.
Mit <ESC> wird abgebrochen.

Editor:

```
E $10000
```

Editieren des Textes ab Adresse $10000.

```
E
```

Editieren des momentanen Textes.

Find:

```
F $A000,$B000,$12
F $A000,$B000,$1234
F $A000,$B000,$12345678
F $A000,$B000,''SUCH-TEXT'
```

Es wird nach BYTE,WORT,LANGWORT oder Texten im Bereich $A000-$B000 gesucht. Dann werden die gefundenen Adressen und die Gesamtzahl der gefundenen Werte ausgegeben.

Starten:

```
G $18000
```

Starten des Programms ab Adresse $18000

Hilfe:

```
H
```

Gibt die Befehlstabelle aus.

ASCII Ausgabe:

```
I $A000,$B000
```

Es wird der Speicher von $A000-$B000 als Zeichen ausgegeben.

Speicherbereiche:

```
J
```

Es werden die Ram-Bereiche ausgegeben.

Zuweisung von Symbolen:

```
K 'Text:=$10000'
```

Es wird dem Symbol 'Text' die Zahl $10000 zugewiesen.

Laden von Kassette:

```
L $9000
```

Es wird, egal ob Text oder Daten, nach der Adresse $9000 geladen.

```
L
```

Es wird nach der im Kopf stehenden Adresse geladen.

Memorie HEX Dump:

```
M $A000,$B000
```

Es wird der Speicher von $A000-$B000 Hexadezimal ausgegeben.
z.B.

00A000 00F1 57F9 7FF9 CFFD BFF9 56F1 EFF9 CCF1 <00047CB4>

Adresse 8 Worte Pruefsumme

Man kann nun mit dem Cursor die Daten aendern, wird dann <CR> gedrueckt, so werden die Daten im Speicher umgeschrieben und die neue Pruefsumme errechnet.

Man kann die Daten auch eingeben ohne vorher ein Memorie zu machen. Es muss nur aufgepasst werden, dass die Adresse mit einer 0 anfaengt, z.B. 010000. Es werden maximal 8 Woerter verarbeitet, weniger duerfen es sein.

Textstart:

```
NN $10000
```

Textstart neu $10000 (der Textspeicher wird geloescht).

```
NA $9000
```

Textstart alt $9000 (der Text im Speicher bleibt erhalten).

Fill:

```
O $A000,$B000,$FF
```

Fuellen des Speichers von $A000-$B000 mit Bytewerten $FF

CRT Ausgabe:

```
P CRT
```

Bildschirmausgabe.

```
P LST
P LST1
```

Druckerausgabe.

```
P LST2
```

Druckerausgabe ohne LF

```
P NIL
```

Befehlstabelle: xxxx = Startadr. yyyy = Endadr. zzzz = nach
(xxxx) kann angegeben werden. zz = wert (auch ASCII)

A (xxxx)	Assembler
B	Bibliothek
C xxxx,yyyy,zzzz	Convert
D xxxx,yyyy	Disassembler
E (xxxx)	Editor
F xxxx,yyyy,zz	Find
G xxxx	Go
H	Help
I xxxx,yyyy	ASCII Dump
J	Speicherbereiche
K text:=xxxx	Zuweisung von Symbolen
L (xxxx)	Laden von Cassette
M xxxx,yyyy	Memorie hex Dump
NA xxxx	Textstart alt
NN xxxx	Textstart neu
O xxxx,yyyy,zz	Fill
P crt/lst/lst1/lst2	Ausgabeart
Q	Symboltabelle
R	Register
SD xxxx,yyyy, name	Daten speichern
ST xxxx, name	Text speichern
T (xxxx)	Text drucken
U xxxx,yyyy,z,z	Version-anpassung
V (xxxx)	Verify
W	Symboltabelle loeschen
X	<EXIT>
YT xxxx,yyyy	EPROM test ob leer
YR xxxx,yyyy,zzzz	EPROM read
YW xxxx,yyyy,zzzz	EPROM write
Z	Jogi Dos

A N L E I T U N G

TS / TSAVE @name@,drive,seite,startadresse
TL / TLOAD @name@,drive,seite,startadresse
DS / DSAVE @name@,drive,seite,startadresse,endadresse
DL / DLOAD @name@,drive,seite,startadresse
GS / GSAVE @name@,drive,seite,bildseite
GL / GLOAD @name@,drive,seite,bildseite
D / DIR drive,seite
VE / VERIFY @name@,drive,seite,startadresse
FO / FORMAT @name@,drive,seite
G / GET drive,seite,track,sector,anzahl,startadresse
P / PUT drive,seite,track,sector,anzahl,startadresse
SC / SCRATCH @name@,drive,seite
DISK drive,seite
H / HELP
Q / QUIT