130 XE+

Makro – Assembler
Version 2.1
© 1991/1992 by Thorsten Karwoth

+ Für ATARI © XL mit mindestens 128 KB RAM (empfohlene Ausstattung 256 KB RAM-Disk und Floppy- Speeder)

+ Nützliche Hilfsprogramme, z. B. Batch-Verarbeitung unter MyDos 4.50

+ Einfache Installation auf RON-Disk

+ Integrierter Fullscreen-Editor mit Syntaxüberwachung

+ Sehr schneller 2-Pass Assembler(1)

+ Bis zu 96 KB Quelltext im Speicher(2)

+ Läuft mit jeder 130Xe-Kompatiblen Speichererweiterung(3)

+ Mehr als 32 KB im Hauptspeicher frei für Objektcode und Daten

+ Monitor (Shell) zum Austesten der Programme (kann um eigene Kommandos erweitert werden).(4)

(1)Z.B. 48 KB in 17 Sekunden (bei abgeschaltetem Bildschirm ca. 30% weniger)
(2)Bei der Einteilung in 6 Seiten zu je 16 KB (maximal 7 Seiten möglich)
(3)Über Pagesequenz-Tabelle konfigurierbar (Konfigurationsprogramm)
(4)Zusatzdiskette in Vorbereitung

Nachtrag: Neue Fehlermeldungen (Fehlercodes) des Makro-Assemblers V 2.1

Fehlermeldungen

Neue Fehlermeldungen des File-Selektors:

Fehlercode:	Bedeutung:
FF	**Interner Fehler (Illegale Seitenanwahl). Darf nicht auftreten!**
FE	**Der einzuladende Quelltext besteht aus mehr Seiten als zur Verfügung stehen! Mit RETURN wird soviel wie möglich geladen, jede andere Taste bricht ab.**
FD	**Der zu ladende Quelltext ist nicht mit diesem Assembler erstellt worden (Inkompatibel zum internen Format)**

Fehlermeldungen der Monitor-Shell:

Wenn bei der Ausführung eine Kommandos kein Fehler auftritt (also das Kommando ordentlich abgearbeitet wurde) meldet sich der Monitor mit Okay und Command:, ansonsten fehlt das Okay!

Alle anderen Fehlermeldungen werden im Klartext in der rechten, oberen Bildschirmzeile angezeigt (in englischer Sprache).

Dieses Programm läuft nur auf Rechnern der Atari XL/XE-Serie und benötigt eine RAMDisk von mindestens 64KB. Diese Version sollte nur mit MyDOS benutzt werden (wurde mit anderen DOS-Versionen nicht "härtegetestet").

Das Copyright für den 130XE+ Makro-Assembler liegt bei mir, Torsten Karwoth. Trotzdem erlaube ich hiermit die Weitergabe dieses Programmes ausschließlich auf der Installationsdiskette, sofern folgende Bedingungen erfüllt werden, sonst darf das Programm nicht kopiert werden:

1) Alle Dateien, auch Texte, welche sich auf der Installationsdiskette befinden, dürfen nicht verändert oder entfernt werden. Dies gilt insbesondere für die Copyright-Notizen im Programm und in der Anleitung. Ausnahme: Alle Autoren, welche ein paar Module programmiert haben (für den Monitor oder Toolbox) und diese weitergeben wollen, sollten eine Funktionsbeschreibung als Textfile beilegen, eventuell in einem zusätzlich erzeugten Verzeichnis "D1:EXPANSIONS". Die einzigen änderbaren Dateien sind das Konfigurationsfile SETUP.CFG sowie PAGESEQ.DAT!

2) Das Programm trägt nicht den Status oder Titel "Public Domain", es darf jedoch im Public-Domain-Bereich weitergegeben werden ("Free Ware"). Nicht erlaubt ist der gewerbliche Verkauf. Dies gilt für die (gedruckte) Anleitung sowie für das Programm.

3) Das Copyright der mit dem 130XE+ Assembler erstellten Programme oder Quelltexte liegt bei den jeweiligen Autoren. Dies gilt auch für zusätzliche Programme in der Toolbox bzw. für Overlay-Kommandodateien, welche nicht zu diesem Assembler gehören. Hierbei ist das Copyright der Autoren zu berücksichtigen und von Diesen eine schriftliche Genehmigung zur Weitergabe einzuholen.

Braunschweig, im Januar 1992

Wenn ein anderes DOS als MyDOS 4.50 benutzt wird müssen folgende Punkte beachtet werden:

- MEMLO muss kleiner als $1FF0 sein!
- Das RAM unter dem OS-ROM darf nicht benutzt werden!
- Die Toolbox-Funktion benutzt den XIO 39/40 von MyDOS zum Laden von Binärfiles. Diese Funktion fällt also bei anderen DOS-Versionen weg!
- Ebenso kann die "Quickloadfunktion" des Monitors nicht genutzt werden, in der ein File durch Angabe des Namens ("D1:Name.Ext") geladen werden kann.
- Die auf dieser Systemdiskette vorhandenen Unterverzeichnisse lassen sich "höchstwahrscheinlich" nicht ansprechen.

Eine Arbeitsdiskette (auf welcher sich nur der Assembler befindet) kann einfach hergestellt werden (haben Sie schon eine Kopie der Systemdiskette gemacht?). Alle Leute, welche nach dem Konfigurieren keinen Speicher mehr haben (für die RAMDisk) müssen eine Arbeitsdiskette erstellen, da die RAMDisk des MyDOS auf der Systemdiskette NICHT abgemeldet werden darf! Diesen Satz sollte sich jeder an die Wand nageln!

1 - Die Kopie der Systemdisk booten und das Konfigurationsprogramm starten.

2 - Nachdem das Konfigurationsfile erzeugt wurde, gehen Sie in das DUP.SYS ("DOS"). Konfigurieren Sie das DOS wie im Konfigurationsprogramm be schrieben. Schreiben Sie das geänderte DOS allerdings NICHT auf die Systemdiskette, wenn Sie keine RAMDisk MEHR zur Verfügung haben (130XE- User). Dann läuft das Installationsprogramm nämlich nicht mehr!

3 - Formatieren Sie eine FREIE Diskette und schreiben Sie das DOS auf diese Diskette (mit "H")!

4 - Booten Sie die Systemdiskette neu, und wählen Sie "Arbeitsdiskette einrichten".

5 - Befolgen Sie die weiteren Programmanweisungen.

Speicherbelegung:

Eigene Programme können den Speicher von $3000 bis $BBFF nutzen, in der ZeroPage von $A0 bis $FF !

Der Assembler belegt folgende Speicherbereiche (alle Angaben Hexadezimal) :

für Zeiger und Zwischenspeicherung

1FF0 – 2FFF	für System-Unterprogramme, Adresstabellen, Puffer usw.
4000 – 4FFF	(RAMDisk) Editor
5000 – 5FFF	(RAMDisk) Macro-Assembler
6000 – 6FFF	(RAMDisk) Verwaltung der Menü-Leiste (FileSelect, Load, Save usw.)
7000 – 77FF	(RAMDisk) Editor - Teil 2
7800 – 7FFF	(RAMDisk) Monitor-Shell
C000 – CBFF	unter ROM weitere Unterprogramme, nachträglich Programmierte Extras
CC00 – CFFF	unter ROM Internationaler Zeichensatz / Directory Puffer
D800 – FFXX	unter ROM Symbolverwaltung & Hashingroutine, Symboltabelle (oder Wertetabelle wenn eine 16K-Page als Symboltabelle benutzt wird)

Achtung:

Es sollte vermieden werden, Interruptroutienen in den Bereich von $4000 bis $7FFF zu legen, da bei einem Programmabbruch (z.B. BRK oder BREAK/SHIFT) die Interrupts nicht ausgeschaltet werden, dort jedoch die RAM-Disk eingeschaltet wird weil dort u.A. der Monitor liegt. Dann ist ein Zugriff auf den "normalen" Speicher nicht möglich und der Interruptvektor zeigt auf alles Andere als auf das eigene Programm!

Bei einigen ATARI XL`s mit selbsgebauter 256 kB-Erweiterung kam es beim Test dieses Assemblers zu Fehlfunktionen und Programmabstürzen. Dies lag aber nicht an diesem Programm, sondern an Timing-Fehlern in der Speichererweiterung! So kann es beim benutzen dieses Programmes passieren, daß irgendwelche Störungen durch Timing-Probleme auftreten, die beim normalen Benutzen nicht auftreten oder nicht bemerkt werden (z.B. RAM-Disk Betrieb, Puffer bei Kopierprogrammen o.Ä.). Timing-Probleme können bei diesem Assembler auftreten, weil sehr häufig zwischen der Speichererweiterung und dem "normalen" Speicher hin- und hergeschaltet wird. Bei Störungen ist daher erstmal (durch einen Elektroniker) das Timingverhalten der RAM`s zu überprüfen, bei mir (und einigen Freunden) läuft das Programm nämlich störungsfrei.

Programmautor und Copyright:

Torsten Karwoth
Hermann-von-Vechelde-Straße 19
3300 Braunschweig
Tel.:(0531) - 69 66 89

Konfigurieren des Makroassemblers:

Vor dem Systemstart sollten Sie sich eine Sicherheitskopie bzw. Arbeitskopie der doppelseitigen Systemdiskette anfertigen und Die Original-Systemdiskette an einem sicheren Ort aufbewahren. Dies sage bzw. schreibe ich garantiert nicht umsonst! Alle folgenden Änderungen sollten nur auf dieser Arbeitskopie erfolgen!

Das Programm benötigt beim Systemstart eine RAM-Disk! Das MyDOS auf der Systemdiskette ist auf eine 130XE-Kompatible 64kB-RAMDisk konfiguriert. Wenn Sie im Verlauf der Installation zum Abmelden der RAMDisk im MyDOS-Menü aufgefordert werden, dürfen Sie in keinem Fall das DOS zurück auf die Installationsdiskette schreiben! Sollten Sie so wenig Speicher haben, daß keine RAMDisk mitbenutzt werden kann, so müssen Sie sich eine Systemdiskette einrichten und das MyDOS (mit der abgemeldeten RAMDisk) NUR AUF DIESE DISKETTE speichern!

Damit das Programm korrekt arbeiten kann, muß es an das vorhandene System angepaßt werden. Dies betrifft vor allem die RAMDisk-Ansteuerung. Dazu dient das Basicprogramm CONFIG.BAS im Verzeichnis SYSTEM auf der Systemdiskette. Es muß als allererstes gestartet werden. Wählen Sie also nach dem booten den Menüpunkt "E - Konfigurieren" an. Das Konfigurationsprogramm wird gestartet.

BITTE BEACHTEN: DAS PROGRAMM LÄUFT NUR MIT EINER 130XE-KOMPATIBLEN SPEICHERERWEITERUNG, D.H. SIE MUß ÜBER "PORTB" ($D301) ANGESTEUERT WERDEN!

Zuerst werden Sie nach dem RAMDisk-Typ gefragt, der in Ihren Computer eingebaut ist. Wenn Sie sich im unklaren über Ihren RAMDisk-Typ sind, starten Sie das Programm PAGEFIND, ebenso wenn sie mehr als eine 256 K-Byte RAMDisk haben! Das Programm findet JEDE mögliche Kombination um eine RAMDisk anzusprechen. Zusätzlich wird die RAMDisk auch noch getestet. Weitere Informationen zu diesem Programm finden Sie in dem Abschnitt "PAGEFIND.BAS"!

Wenn Sie im Besitz der ABBUC-Erweiterung (192 KB) sind, drücken Sie die Taste <3> und <RETURN>. Das Programm kennt intern die Pagesequenz für diese RAMDisk und beginnt mit dem Test. Wenn Ihre RAMDisk nicht im Menü auftaucht, können Sie die Pagesequenz auch selber eingeben (wenn Sie sie kennen). Waehlen Sie dazu Punkt 0 - Manual/file input. Sollten Sie vorher PAGEFIND gestartet haben, wird die von PAGEFIND erzeugte Pagesequenz verwendet. Wenn Sie dieses nicht wollen, müssen Sie die Datei "D1:PAGESEQ.DAT" löschen! Dies geschieht automatisch wenn PAGEFIND gestartet wird; beenden Sie PAGEFIND sofort ohne die Pagesequenz abzuspeichern. Sie können auch vom BASIC aus mit dem Befehl

```
XIO 33,#1,0,0,"D1:PAGESEQ.DAT"
```

die Datei löschen und das Programm mit RUN starten. Wenn Sie jetzt "manual/ file input" wählen, müssen Sie die Pagesequenz in 2-Stelligen Hexadezimalzahlen eingeben.

Das Programm testet die komplette RAM-Disk, löscht also auch alle Daten in ihr! Nach dem testen meldet das Programm den gefundenen Speicher und schlägt Ihnen verschiedene Konfigurationen vor.

Erläuterungen zur "Hashing Page":

Diese Version des Assembler verwaltet die Symboltabelle mit einem Hashingalgorhytmus und erreicht dadurch eine sehr gute Geschwindigkeit beim Assemblieren. Die "normale" Symboltabelle liegt unter dem OS-ROM. Sie bietet ca. 772 freie Einträge, und ihr Inhalt wird beim Aufruf der Menüleiste zerstört! Wenn Sie also eine "Hashing Page" wählen, hat die Symboltabelle Platz für

2048 (Zweitausendundachtundvierzig) Symbole und diese Symbole bleiben natürlich (bis zum nächsten Assembliervorgang) erhalten. 772 Symbole reichen für zwei bis drei Pages Quelltext aus). Wenn Sie über genügend Speicher verfügen oder mehr als 3 Pages für den Quelltext konfigurieren, sollten Sie auf jeden Fall eine Page für die Symboltabelle nehmen!

Die Frage des Programmes, wieviele Pages für den Assembler reserviert werden sollten, bezieht sich nicht nur auf den Quelltextspeicher, sondern auf den kompletten Speicherbedarf des Assemblers! Wenn Sie also 8 Pages benutzen wollen, so belegt das Programm selber eine davon. Wenn Sie nocht eine Page für die Symboltabelle nehmen, haben Sie sechs Pages für den Quelltext. Dies ist auch ab einer 192kB RAMDisk die Standartkonfiguration. Sie können auch 7 Pages für den Quelltext reservieren, allerdings nur mit der kleinen Symboltabelle. Dies ist aber nicht empfehlenswert, es sei denn Sie benutzen sehr wenig Symbole in ihrem Quelltext! Ich habe in einen 48 kByte-Sourcecode schon über 700 Symbole benutzt, und das ist noch recht wenig - wie ich meine.

Wenn Sie nur 4 Pages (oder weniger) für den Assembler benutzen dann antworten Sie bitte auf die Frage, ob eine Page für die Symboltabelle benutzt werden soll, mit N (für Nein). Wenn Sie mehr Pages (bis zu acht) benutzen, mit J (Ja)!

Anschließend fragt das Programm, unter welcher Laufwerksnummer die RAMDisk angesprochen werden kann. Hier ist für MyDOS-Benutzer der Wert 0 (für Autokonfig) vorgegeben.

Folgender Absatz gilt nur für nicht-MyDOS-Benutzer:

Wenn keine RAMDisk mehr vorhanden sein sollte (weil der komplette Erweiterungsspeicher für das Programm benutzt wird), geben Sie eine "1" ein, sonst geben Sie bitte die Laufwerksnummer der RAMDisk ein.

Erklärung:

Das Programm kann unter MyDOS selbst feststellen, auf welche Laufwerksnummer die RAMDisk eingestellt ist. Daher auch der Wert 0 für die Autokonfiguration. Da diese Funktion bei anderen DOS-Versionen versagt, muß hier die RAMDisk-Nummer UNBEDINGT eingegeben werden wenn ein anderes DOS benutzt wird! Wird hier also ein Wert ungleich Null eingegeben, wird die interne Autokonfiguration umgangen und die eingegebene Laufwerksnummer verwendet! Die "1" für "keine RAM-Disk" entspricht ja einem Zugriff auf Laufwerk .

Option für alle Besitzer der EPROM-Disk von R. David:

Als nächstes werden Sie nach der EPROMDisk-Nummer gefragt. Wenn Sie keine EPROMDisk haben oder den Assembler nicht von der EPROMDisk booten wollen drücken Sie bitte nur <RETURN>, ansonsten geben Sie die Laufwerksnummer ein, welche die EPROMDisk nach dem vollständigen Booten erhalten soll!

Erklärung:

Nach dem Booten von der EPROMDisk verdeckt diese ja das Diskettenlaufwerk. Mit dieser Funktion wird die EPROMDisk nach dem kompletten Bootvorgang auf die angegebene Laufwerksnummer gesetzt. Dadurch ersparen Sie sich das manuelle Umschalten mit dem Affengriff. Wird der Assembler mit der Quit-Funktion verlassen, so wird die EPROMDisk auf jeden Fall auf die Nummer zurückgesetzt, welche vor dem Hochfahren des Assemblers eingestellt war!

Als nächstes legen Sie bitte die Systemdiskette in Laufwerk #1 ein um das Konfigurationsfile auf die Diskette zu schreiben. Sollte eine alte Konfiguration schon bestehen, wird sie "geupdatet", d.H. nur die Speicherkonfiguration wird überschrieben (die alten Informationen wie Farben und die anderen
Einstellungen bleiben erhalten), erkennbar an der Meldung "Updating!".

Anmerkung:

Sollte das Konfigurationsfile noch nicht existieren, erhalten Sie nach dem Laden des Assemblers die Meldung "Setup I/O failure!". Dies liegt daran, das in dem Konfigurationsfile noch keine Systemeinstellungen gespeichert sind. Gehen Sie in das Setup-Menü und wählen Sie "Save Setup" (nachdem Sie Ihre Lieblingseinstellung eingegeben haben), um ihre Einstellungen dauerhaft zu sichern.

Anschließend gibt Ihnen das Programm die neue MyDOS-Konfiguration vor. Diese sollten Sie sich UNBEDINGT FEHLERFREI NOTIEREN (Die Page-Sequenz-Tabelle)! Eine kleine Fehleingabe kann später stundenlange Arbeit zunichte machen!

Erklärung:

Das MyDOS muß an die geänderte RAMDisk angepaßt werden. Das MyDOS läßt sich ja an (fast?) jede beliebige RAMDisk anpassen. Die Ansteuerung der RAMDisk erfolgt dabei über eine "Page-Sequence-Table". Diese Tabelle ist vom Benutzer fast uneingeschränkt änderbar, und diese Änderung muß mit größter Vorsicht geschehen, da sonst Daten in der RAMDisk verloren gehen können, evtl. sogar der Quelltext zerstört wird!

Beachten Sie bitte, das die Page-Sequence des MyDOS nur 64 Byte umfaßt. Sollten Sie zu den Leuten gehören, die eine SEHR große RAMDisk haben, dürfen Sie nur max. 64 Werte eingeben. Die "0" schließt das Ende der Tabelle ab!

Nachdem das MyDOS vollständig konfiguriert ist muß es mit der Funktion "H -Write DOS Files" auf die Systemdiskette geschrieben werden. Jetzt steht einem Test nichts mehr im Weg. Booten Sie das System neu hoch. Schalten Sie dazu den Computer aus und nach ein paar Sekunden wieder ein, damit ALLE Daten in der RAMDisk gelöscht werden.

Anmerkung zum RAMDisk-Speicher:

Ich habe einmal fälschlicherweise eine Konfiguration angegeben, in welcher ich eine notdürftige Spar-RAMDisk mit 2 Pages (32 k-Bytes) vorschlug. Dies ist aber ein Irrtum meinerseits gewesen, da in einer 32kB RAMDisk kein Inhaltsverzeichnis angelegt werden kann - Dieses liegt nämlich in den Sektoren 360-368. Eine 32kB RAMDisk hat aber nur 256 Sektoren zu je 128 Byte. Erst eine 48 kB
RAMDisk bietet Platz für das Directory (384 Sektoren zu je 128 Bytes).

PAGEFIND.BAS
==========

Das Programm PAGEFIND dient zum Herausfinden der Größe der RAMDisk, dem Testen der RAMDisk und zum Erzeugen einer Pagesequenz-Tabelle für das Konfigurationsprogramm. Nach dem starten werden Sie mit einem Auswahlmenü konfrontiert.

Wählen Sie als allererstes Punkt 1 - finde Pagesequenz:

Das Programm gibt einen Hinweistext aus, anschließend beginnt es mit dem Test der RAMDisk. Dieser Vorgang dauert dank Maschinensprache nur wenige Sekunden. Dabei wird auch gleich noch das Timingverhalten überprüft. Bei diesem Test (wie auch allen anderen) darf kein Fehler auftreten! Sollten dennoch Probleme mit dem Timing auftreten, können Sie mir schreiben. Um aber helfen zu können, benötige ich eine Fehlerbeschreibung, eine genaue Angabe des RAMDisk-Types sowie einen ausreichend frankierten und adressierten Rückumschlag. Eine Garantie auf 100%-ige Fehlerbehebung kann ich nicht geben, aber Tips zum beheben von Timingproblemen (z.B. Klaus-Peters RAMDisk, hatte da selber Probleme, die ich aber mit Einbau von zwei zusätzlichen Widerständen beheben konnte. Ich habe hier z.B. in die CAS-Leitung zu den neuen und den alten RAMs jeweils einen 470-Ohm Widerstand zwischengeschaltet und seitdem keine Probleme mehr ge-
habt. Dies könnte auch bei anderen RAMDisktypen funktionieren).

Nachdem Ihnen daß Programm die Größe der RAMDisk "verraten" hat, starten Sie bitte den "Short RAM-Test":

Das Programm schreibt in alle RAMDisk-Seiten (Pages) ein bestimmtes Muster und liest es wieder aus. Sollte ein Fehler auftreten, teilt Ihnen das Programm das mit.

Wenn Sie viel Zeit haben, können Sie auch den "Deep RAM-Test" starten, welcher etwas länger dauert. Bei diesem Test wird ein bestimmtes Bitmuster im Speicher verschoben und verglichen usw. Sollte ein Fehler auftreten, bricht das Programm ab.

Mit <5> speichern Sie die Pagesequenz-Tabelle auf D1: und starten das Konfigurationsprogramm erneut. Wählen Sie dann "0 - manual/file input". Das Programm liest die Pagesequenz aus dem File.

Der Editor des 130XE+ Makro-Assemblers:

Der Editor ist kein gewöhnlicher Texteditor, sondern ein speziell für die Erstellung von gut lesbaren Assembler-Quelltexten gedachter Fullscreeneditor. Er kann aufgrund seines Konzeptes auch nicht zum Erstellen von Briefen oder Ähnlichem benutzt werden, da der komplette Quelltext in Token-form gespeichert wird und bereits bei der Eingabe ein Syntaxcheck vorgenommen wird. Wird ein

Fehler festgestellt, muß dieser korregiert bevor die Zeile in den Quelltext übernommen wird.

Der Quelltext wird in Pages eingeteilt. Diese Pages sind aufgrund der RAM-Disk-Größe jeweils 16 k-Bytes groß. Wegen fehlender Zauberkünste ist es mir leider nicht gelungen den Speicher zusammenhängend zu verwalten (also statt 6 Pages mit je 16 k-Byte lieber 1 Page mit 96 k-Byte). Entsprechende Versuche waren aufgrund der fehlenden Geschwindigkeit unzumutbar. Ich denke aber, das die Unterteilung in jeweils 16 k-Byte große Pages sogar einen Vorteil gegenüber einer einzigen, großen Page hat.

In der ersten Bildschirmzeile befindet sich die Titelzeile des Assemblers mit der Versionsnummern und der Copyright-notiz. Wenn die Menüzeile aufgerufen wird, erscheint anstatt dieser Titelzeile eine Menüleiste (Icons). In der zweiten Zeile wird die aktuelle Seite (Page) sowie der freie Speicher in dieser Seite angezeigt, außerdem noch eine eventuelle Fehlermeldung.

Der Cursor wird mit den Cursortasten gesteuert (mit CONTROL). Wenn er an den oberen oder unteren Bildrand kommt wird der Text gescrollt. Mit der Tastenkombination SHIFT/CursorUp/Down kann um jeweils 16 Zeilen gescrollt werden. Einen Insert-Modus gibt es nicht. Um Zeichen in eine Zeile einzufügen muß mit CONTROL/Insert zuerst Platz geschaffen werden. Mit ESCape kann eine versehentliche Änderung rückgängig gemacht werden. Mit SHIFT/Delete werden einzelne Zeilen gelöscht. Mit SHIFT/Insert kann man Zeilen in den Quelltext einfügen. Mit SHIFT/CONTROL/- kommt man zum Textanfang und mit SHIFT/CONTROL/= zum Textende.

Blockmarkierung:

Ein Textblock mann mit der Tastenkombination CONTROL und "," oder "." (für die Zeichen "[" und "]") markiert werden. Dieser Block kann z.B. gelöscht, kopiert, gespeichert oder gedruckt werden. Wenn der Textblock vollständig (d.H. Anfang UND Ende) markiert wurde, wird der Text invertiert dargestellt. Die Markierung wird automatisch gelöscht wenn der Text editiert oder umgeschaltet wird.

CONTROL-Funktionen:
==========

Y	Assemblierung starten
P	Sprung in den Monitor
[	Blockstart setzen
]	Blockende setzen
?	Sprung zum Blockanfang
CLEAR	Blockmarkierung löschen
INSERT	fügt ein Leerzeichen in die Zeile ein
DELETE	löscht das Zeichen unter dem Cursor
TAB	setzt den Tabulator (siehe gesonderte Erklärung unten)
ESC	Menüzeile aktivieren
SHIFT/-	Sprung zum Textanfang
SHIFT/=	Sprung zum Textende

Quelltext-Ebenen:
==========

Aus dem Quellext kann mit SHIFT/CONTROL/Zahlentaste gemäß der Konfiguration eine der maximal 7 Seiten ausgewählt werden. In der Statuszeile wird die eingestellte Seite, der freie Speicher in dieser Seite und eine eventuelle Fehlermeldung angezeigt.

Funktionen wie Laden, Speichern oder Block-Löschen werden über die Menüzeile angewählt.

Setzen des Tabulators:
==========

Mit der Tastenkombination CONTROL/Tab kann der interne Tabulator gesetzt werden. Das ist die Spalte, wo Mnemonics (z.B. "LDA") oder Kommentare anfangen. Steht nur ein Kommentar in der Zeile, erfolgt kein Einrücken desgleichen. Beispiel:

```
    +- Tabulator 1  +-- Tabulator 2
1234567890|1234567890123456|890123456789
    |       |
 ; Nur eine Kommentarzeile (ohne Mnemonic) - wird nicht eingerückt!
symbol:  LDA #1      ; Kommentar hinter dem Mnemonic
    STA (zp_base+offset),Y ; Kommentar
    CLC        ; Alles klar?
```

Dabei entscheidet die Cursorposition, welcher Tabulator gesetzt wird. Steht der Cursor in den ersten 15 Spalten, wird der 1. Tabulator gesetzt, sonst der Zweite! Genauer kann man sich an der Anzeige des freien Speichers orientieren. Steht der Cursor unter der 100er-stelle oder rechts davon so wird der 2. Tabulator beeinflußt. Steht er unter der 1000er-Stelle oder links davon so wird der 1. Tabulator gesetzt. Man kann also z.B. ganz nach rechts gehen (Spalte 39) und den Tabulator 2 setzen, dann verbannt man alle Kommentare vom Bildschirm bis auf die "nur-Kommentarzeilen"! Diese Lösung hat den Vorteil, daß ein Ausdruck sauber und geordnet erscheint.

<u>Speicherplatzverbrauch:</u>

Bedingt durch die Tokenisierung belegt eine Zeile Quelltext des Assemblers eventuell etwas mehr Speicher als es die gleiche Zeile im ASCII-Format tun würde. Andersherum spart das Tokenformat Zeit beim assemblieren. Beispiel zu einigen Konstanten und ihrem Speicherplatzbedarf im Quelltext:

Wertebereich	Normal ASCII	Tokenformat
1-9	1 Byte(s)	2 Byte(s)
10-99	2 -"-	2 -"-
100-255	3 -"-	2 -"-
256-999	3 -"-	3 -"-
1000-9999	4 -"-	3 -"-
10000-65535	5 -"-	3 -"-
$0-$F	2 -"-	2 -"-
$10-$FF	3 -"-	2 -"-

$100-$FFF	4 -"-	3 -"-
$1000-$FFFF	5 -"-	3 -"-
%xxxxxxxx	2-9 -"-	2 -"- (8-Bit-Binärzahl)

Eine Zahl im Tokenformat belegt erstmal ein Byte ("Header"), sowie 1 oder 2 weitere Bytes für den Wert. Im Header sind verschiedene Informationen gespeichert, z.B. ob die Konstante negativ ist, wie der Editor sie darstellen soll (hexadezimal, dezimal oder Binär), ob sie 8 oder 16 Bit belegt (Zahlen von 0-255 belegen 8 Bit, Zahlen ab 256 belegen 16 Bit) und so weiter. Eine negative Konstante belegt also genausoviel Speicher wie eine Positive Konstante (bei meinem Assembler). Bei ASCII-Orientierten Editoren wie z.B. dem ATMAS kann man "LDA #-1" schreiben, um den Akkumulator mit dem Wert $FF zu laden (-1 = $FFFF, der ATMAS macht sich aber nichts aus dem 16-Bit-Wert). Beim 130XE+ Makro-Assembler spart diese Technik nichts! Man sollte sich angewöhnen $FF zu schreiben wenn man $FF meint. Ausserdem ergibt ein "LDA #-1" bei meinem Assembler eine Fehlermeldung, da hier versucht wird ein 8-Bit Register mit einem 16-Bit Wert zu laden. Diese
Fehlermeldung ist kein harter" Fehler, sondern als Warnung zu deuten. Diese Warnungen können aber im Setup auch verhindert werden, so daß keine Meldung auftritt und das Assemblieren nicht abgebrochen wird.

Bedingt mit dem neuen Tokenformat wurde eine neue Old-Routine eingebaut, um einen Quelltext nach einem harten Absturz zu retten. Mit SHIFT/CAPS wird diese Routine aufgerufen, sie wirkt nur ab der Zeile, in der der Cursor steht. Diese Old-Routine sollte nur im Notfall aufgerufen werden, wenn z. B. in der Zeile lauter Control- oder Grafikzeichen erscheinen (nach einem Absturz). Diese Routine kann aber auch die aktuelle und darauffolgende Zeilen kaputtmachen. Nach ihrem Aufruf muß übrigens die "normale" Old-Funktion aus dem Menü aufgerufen werden, damit einige interne Zeiger korregiert werden. Sonst können Sie z. B. nicht mehr hochscrollen usw.!

Anmerkung:

Es wird nur die aktuelle Zeile zerstört wenn in ihr irgendwo ein Byte vorkommt indem alle Bits gesetzt sind. Dies ist z.B. bei Zahlen wie "255","$FFxx", "$(xx)FF" usw. der Fall. In diesem Fall wird die Zeile an dieser Stelle aufgeteilt und der Rest der Zeile erscheint als neue Zeile. Beispiel: Geben Sie im Editor die Zeile "LDA #222+255" ein. Diese Old-Funktion wird die Zeile zerstören, da der Wert 255 als Zeilenheader interpretiert wird (hat was mit dem internen Aufbau des Quelltextes zu tun).

Wenn Sie diese Funktion am Ende des Quelltextes aufrufen kann es vorkommen, daß der Rest eines alten Quelltextes (der vorher bearbeitet wurde) angehängt wird. Dieser muß dann als Block markiert und wieder gelöscht werden.

Der Assembler:

Der Assembler wird aus dem Editor mit CONTROL/Y gestartet und kehrt nach der Übersetzung des Quelltextes wieder zum Editor zurück. Sollte ein Fehler aufgetreten sein, so wird der Fehler im Klartext angezeigt. Nach der Assemblierung werden die Endadresse, die Anzahl der

Symbole und Makros sowie der freie Speicherplatz in der Symboltabelle angezeigt.

Makro-Fähigkeit:

Es können maximal 7 numerische Parameter an Makros übergeben werden. Makros können nicht verschachtelt werden.

Makros müssen vor ihrem ersten Aufrufen definiert werden! Soll ein Makro benutzt werden (bzw. aufgerufen), so muß das dem Editor "gesagt" werden indem dem Mako-Namen einen Punkt vorangestellt wird. Wenn das Makro also "Test" heißt, so müssen Sie ".TEST" eingeben. Der Editor schreibt Makroaufrufe grundsätzlich in Großschrift.

Definieren von Makros:

Eine Makrodefinition beginnt mit dem Namen des Makros, gefolgt von der direktive (Anweisung an den Assembler) "MAC" und der Anzahl der zu übergebenden Parameter. Letzteres kann auch entfallen wenn keine Parameter an das Makro zu übergeben sind. Mit "END" wird die Makrodefinition beendet.

```
Name:     MAC (Anzahl der Parameter)
... Assembler Quelltext ...
          END
```

Beispiel für Makros:

```
Poke      MAC  2
          LDA  #<?2  ; Lade Accumulator mit dem Low-Byte des 2. Parameters.
          STA   ?1           ; Schreibe Accu an Adresse (1. Parameter)
          END

DPoke:    MAC  2
          LDA  #<?2  ; Lade Accu mit Low-Byte
          STA  ?1            ; Schreibe Accu an Adresse
          LDA  #>?2  ; Lade Accu mit High-Byte
          STA  ?1+1          ; Schreibe an Adresse+1
          END

Loop:     MAC                ; ohne Parameter
          LDY  #0
Maclp@: DEY
          BNE Maclp@         ; @ ist der Klammeraffe für locale Symbole
          END
```

Achtung! Symbole dürfen nur 8 Zeichen lang sein! Bei lokalen Symbolen wird an das Symbol eine 2-Byte-Integerzahl drangehängt (wenn möglich). Der "Klammeraffe" wird nicht überschrieben, da er Später zu Unterscheidung in der Symboltabelle benötigt wird:

```
Maclp@              Ok, Gesamtlänge des Symboles ist 6 Zeichen (+2 Byte-Integer)
```

Testlp@	Geht auch, allerdings "nur" 256 mal möglich.
Testsym@	Falsch, da kein Platz mehr für die Nummerierung ist!

Als Local deklarierte Symbole können auch ausserhalb von Macros verwendet werden (sollte aber nicht sein). Es wird keine Fehlermeldung ausgegeben.

Einsatz von Makros im Quelltext:

```
start:  .POKE $D301,$FF
        .DPOKE 560,DLIST
        .SCHLEIFE
```

Die Anzahl der übergebenen Parameter muß mit der in der Definition angegebenen Anzahl übereinstimmen, ansonsten erhalten Sie beim assemblieren die Fehlermeldung "Macro error!".

Mathematische Operatoren:

Ein "Ausdruck" (Expression) darf aus maximal 7 Klammerebenen bestehen. Folgende Rechenoperationen sind möglich (in Reihenfolge ihrer Priorität):

#	EXKLUSIV-ODER Verknüpfung
&	Binäre UND Verknüpfung
!	Binäre ODER Verknüpfung
*	Multiplikation
/	Division
+	Addition
-	Subtraktion

Mit ">" oder "H:"kann das High-Byte eines Wertes selektiert werden, mit "<" oder "L:" das Low-Byte. Der Fehler in der alten Matheroutine konnte behoben werden, die Selektion klappt jetzt auch bei verschachtelten Ausdrücken. Beispiel:

```
Adress:  DFB >(tab1+offset),H:(tab2+offset)...
```

Konstanten:

`	ATASCII-Code des nachfolgenden Zeichens (Beispiel: CMP #`A)
10	dezimale Konstante
$F	hexadezimale Konstante
%1	binäre Konstante
^x	Bit-Konstante. ^0 entspricht 1 , ^7 entspricht 128 und ^15 entspricht $8000.

Negative Konstanten:

Konstanten und Symbolen kann auch ein Minuszeichen vorangestellt werden.

Assemblerdirektiven:

- ORG Al(,Ap)

Festlegen der Anfangsadresse eines Programmes.

Al = logische Adresse
Ap = physikalische Adresse

ORG $4000 assembliert das Programm nach Adresse $4000. Dort ist es auch lauffähig.

ORG $4000,$6000 assembliert das Programm nach $6000. Lauffähig ist es aber ab Adresse $4000 !

- EQU oder "="

Zuweisung einer Konstanten oder eines Rechenergebnisses an ein Symbol.

Achtung: Die ATMAS-Direktive EPZ (welche EQU entspricht) wurde aus Platzgründen nicht mit übernommen!

```
Rtclock:    EQU $14
```

"rtclock=$14" wird zu "rtclock: EQU $14"

- DFB, DFW

Definieren von Konstanten im Objektcode.

```
Bittab:     DFB 1,2,4,8,16,32,64,128 ; Daten in Bytebreite (8-Bit)
AdressTb:   DFW $b800,$bc00,$9430... ; Daten in Wortbreite (16-Bit)
```

Achtung: Wenn bei der DFB-Anweisung ein 16-Bit-Wert angegeben wird erfolgt keine Fehlermeldung!

- ASC <STZ>Text<STZ>

<STZ> = String-Trennzeichen (" / \ % $)

Achtung: Das STZ kann nicht mehr im Text vorkommen!

" = Text im ASCII-Code einfügen
/ = Inverser ASCII-Code

\ = ASCII-Code, aber (nur) beim letzten Zeichen Bit 7 gesetzt (Ende-kennung)
% = Interner Bildschirmcode
$ = Interner inverser Bildschirmcode

Beispiel:
ASC "ATASCII-Code" oder ASC %Bildschirmcode%

- OPT <expr>

Option beim assemblieren des Quelltextes setzten.

<expr> ist dabei ein Ausdruck, von welchem nur die unteren 8 Bit Verwendung finden. Hier habe ich ein neues Verfahren eingebaut. Bit 7 bestimmt, ob die anderen Bits in dem Assembler internen Option-Flag gesetzt oder gelöscht werden. Am Anfang der Assemblierung ist dieses Flag immer 0.

Beispiel: OPT %x0010011
x = 1 ... Die Bits 0,1 und 4 werden gesetzt!
x = 0 ... Die Bits 0,1 und 4 werden gelöscht!

Die anderen Bits bleiben unbeeinflußt. Dies hat den Zweck, an verschiedenen Stellen während der Assemblierung gezielt bestimmte Bits zu setzen.

Bedeutung der Bits (wenn gesetzt):

6 - Quick mode. Nach der Assemblierung wird sofort gestoppt ohne die Belegung der Symboltabelle anzuzeigen , alle anderen Optionen werden ebenfalls ignoriert (auch die GEN-direktive)!

5 - Non-Query.- Normalerweise wird bei der Ausgabe der Symboltabelle immer nach 16 Zeilen gestoppt und der Anwender muß eine Taste drücken um die nächsten 16 Zeilen angezeigt zu bekommen. Wird dieses Bit gesetzt, scrollt die Ausgabe bis der Anwender eine Taste drückt, nochmaliger Tastendruck läßt das Listing weiterlaufen.

4 - Send Report. Nach dem (fehlerfreien) Assembliervorgang wird der Anwender nach einem Ausgabegerät gefragt. Dies ist normalerweise das unter "Printer-device" voreingestelle Gerät. Auf Diesem wird dann die Symboltabelle ausgegeben, siehe auch nachfolgende Bits.

3 - Head-Line. Wenn Bit 4 (Send Report) gesetzt wurde, wird eine Titelzeile ausgegeben. Der Text muß dazu in einer Kommentarzeile in der 1. Seite stehen. Diese Kommentarzeile muß die erste Zeile im Quelltext sein und als erstes Zeichen ein ">" enthalten! Beispiel:
" ; > Dieses ist die Titelzeile! Hier kann der Programmname stehen."

2 - Liste Makro-Namen. Die definierten Makros werden mit ihrem Status angezeigt.

1 - Liste Symbole. Alle definierten (globalen) Symbole werden mit ihrem Status angezeigt.

0 - Mark labels. Alle Symbole, welche definiert werden solange dieses Bit gesetzt ist, werden intern "markiert", ihr Status in der Symboltabelle ist dann "marked". Alle So markierten (globalen) Symbole können in eine Datei geschrieben werden und in andere Programme übernommen werden! Siehe auch GEN und LIB. Hier macht die Bit-Setz- und Löschmethode eigentlich erst Sinn, so kann gezielt nur dieses Bit beeinflußt werden ohne die anderen, am Anfang gesetzten Optionen zu löschen!

Ach ja: OPT 0 schaltet den Bildschirm ab!

- GEN "<Vollständiger Dateiname>" (z. B. "D8:LIBRARY>SYMBOL.LIB")

GENerate dient zum abspeichern der "markierten" Symbole. Diese können dann in andere Quelltexte mit der LIB-direktive übernommen werden. Achtung, My-DOS unterscheidet zwischen Groß- und Kleinschreibung!

- LIB "<Vollständiger Dateiname>"

Mit LIBrary können Symbole aus anderen Programmen in eigene Programme übernommen werden. Wichtig, wenn man ein Programm in zwei oder mehrere Quelltexte aufteilt. So kann man z.B. eine Unterprogrammbibliothek assemblieren , mit GEN alle Unterprogramm-Label abspeichern und auf diese Routinen dann über ihre Namen von einem anderen Programm aus zugreifen. Zweckmäßigerweise sollten die Libraries in der RAM-Disk abgelegt werden.

- HLT

Hält das Assemblieren in der aktuellen Seite an und macht in der nächsten Seite weiter (wenn möglich).

Den Assemblierungsvorgang kann man mit der ESC-Taste unterbrechen!

Funktionen der Menüleiste des 130XE+ Assemblers:

Die Menüleiste wird vom Editor aus mit der Tastenkombination CONTROL/ ESCape aktiviert. In der obersten Bildschirmzeile erscheint dann anstelle der Titelzeile die Menüleiste. Diese Menüleiste enthält 16 Menüpunkte, aus denen mit den Cursortasten oder den Pfeiltasten "größer als" bzw. "kleiner als" einer ausgewählt werden kann. Der gerade ausgewählte Menüpunkt wird invertiert dargestellt. Um die gewählte Funktion zu aktivieren, ist die <RETURN> - Taste zu drücken. Mit ESCape kommt man wieder in den Editor.

Anmerkungen zur Fileauswahl in der Fileselect-Box:

Sollte irgendeine Diskettenoperation angewählt werden so erscheint eine Datei-Auswahlbox. In dieser Box wird ein Teil des Inhaltsverzeichnisses, die Laufwerksnummer sowie der freie Speicherplatz auf der Diskette angezeigt.

In den ersten beiden Zeilen wird die Verzeichnistiefe bzw. der Direktorypfad angezeigt. Unterverzeichnisse werden mit einem ":SUB" anstelle der Sektorenlänge gekennzeichnet.

Zum Einladen muß ein File mit den Cursortasten ausgewählt werden und <RETURN> gedrückt werden. Der Filename erscheint jetzt in der unteren Zeile der Box. Zum laden drücken Sie nochmals <RETURN> oder wählen Sie ein anderes File. Die Dateiauswahl kann mit der ESCape-Taste abgebrochen werden.

Zum Abspeichern einer Datei wählen Sie den Namen aus dem Inhaltsverzeichnis wie beim laden (zum überschreiben einer bereits existierenden Datei) oder geben Sie einen neuen Namen ein. Dann drücken Sie (nochmals) <RETURN>. Sollte eine abzuspeichernde Datei bereits auf der Diskette vorhanden sein, so werden Sie gefragt, ob Sie die Datei überschreiben oder den Vorgang abbrechen wollen.

Die Laufwerksnummer können Sie mit "SHIFT" & Zahlentaste wählen, also z.B. "SHIFT/8" für die RAM-Disk. Es erscheinen nicht immer alle Filenamen in der Dateiauswahlbox. Quelltexte im internen Format haben den Extender ".ASM", z.B. "D8:FASTLOAD.ASM", Quelltexte im ASCII-Format haben den Extender ".SRC", z.B. "D1:IOLIB.SRC" ! In den jeweiligen Boxen erscheinen daher auch normalerweise nur Dateien mit den jeweils passenden Endungen. Diese Endung (z.B. ".SRC") wird auch in der Dateiauswahlbox angezeigt. Mit der TABulator-Taste kann aber zwischen der normalen Endung und dem kompletten Inhaltsverzeichnis gewechselt werden. Mit SHIFT/TAB kommt man aus einem Unterverzeichnis in das vorherige Verzeichnis (Parent-Funktion).

Nun aber zu den Funktionen in der Menü-Zeile:

1. Inhaltsverzeichnis
2. Quelltext laden
3. Quelltext speichern
4. ASCII-Text laden
5. ASCII-Text speichern
6. Text drucken
7. Suche nach ...
8. Sprung zum Disk Operating System (DOS-Menü)
9. Toolbox -> Assembler-Hilfsprogramme laden und starten.
10. Preferences / Voreinstellungen
11. Block merken ("abfotografieren")
12. Block einfügen ("ankleben")
13. Block ausschneiden (löschen und zwischenspeichern)
14. Block löschen ("wegradieren")
15. Quelltext reparieren / retten (Old-Funktion)
16. Quelltext löschen

1. Inhaltsverzeichnis:

- ohne Worte -

2. Quelltext laden:

Es erscheint die Datei-Auswahlbox. Wählen Sie die gewünschte Datei oder brechen Sie die Funktion mit ESCape ab. Sollte die einzuladende Datei kein Quellcode im internen Format sein, erscheint eine Fehlermeldung und ein evtl. im Speicher stehender Text bleibt erhalten.

Achtung! Es erscheint keine Warnung, falls dieser Menüpunkt versehentlich angewählt wird und ein im Speicher stehender Text noch nicht gesichert wurde!

3. Quelltext speichern:

Es erscheint eine Datei-Auswahlbox. Wählen Sie eine Datei aus oder geben Sie einen Namen über die Tastatur ein. Drücken Sie <RETURN>, der Speichervorgang beginnt.

4. ASCII-Text laden:

Vorgehen wie bei Punkt 2 (Quelltext laden). Der ASCII-Text wird Zeilenweise an der Qursorposition in den evtl. vorhandenen Quelltext eingefügt. Sollte der Speicherplatz in der aktuellen Quelltextebene nicht ausreichen, wird NICHT auf die nächste Seite umgeschaltet.

5. ASCII-Text speichern:

Dateiauswahl wie bei Punkt 3 (Quelltext speichern). Hier gibt es allerdings zwei verschiedene Möglichkeiten: Wenn mit der Blockmarkierungsfunktion ein Textblock markiert wurde (der so markierte Text erscheint invertiert), so wird nur der so markierte Bereich abgespeichert. Sonst wird der Text ab der Cursorposition (Zeile, in der der Cursor steht) bis zum Ende der aktuellen Seite gespeichert.

6. Text drucken:

Textmarkierung siehe Punkt 5 (ASCII-Text speichern). Der Text wird auf dem Drucker oder ein anderes Device geschrieben (wählbar). Der wohl wichtigste Unterschied ist, das bei diesem Menüpunkt alle Leerzeichen mit ausgegeben werden während bei Punkt 5 der Text im "kurzen" ASCII-Format ausgegeben wird. Voreingestellt als Ausgabegerät ist der Drucker bzw. das im Preferences-Menü eingestellte Gerät. Dises kann auch ein beliebiger Schnittstellentreiber sein (z.B. "R:600,8N1" für eine RS-232-Schnittstelle). Die Voreinstellung kann natürlich auch geändert werden.

Beispiel für das Ausgabeformat:

Ausgabe auf Diskette nach Punkt 5 ("_" = Leerzeichen):

```
_Symbol:LDA_(Pointer),Y ; mit Quelle laden
_INY ; nächster Wert
```

Ausgabe nach Punkt 6 auf ein beliebiges Gerät (formatierte Ausgabe):

```
_Symbol:_________LDA (Pointer),Y ; mit Quelle laden
_____________________INY ; nächster Wert
```

7. Suche nach ...

Der Suchbegriff kann eingegeben bzw. geändert werden. Mit drücken von <RETURN> wird ab der Cursorposition gesucht. Bei erfolglosem Suchvorgang steht der Cursor am Textende, ansonsten wird die Zeile, in der eine Übereinstimmung stattfindet, invertiert dargestellt. Soll weiter gesucht werden ist wieder die <RETURN> - Taste zu drücken, ansonsten kann mit ESCape der Suchmodus verlassen werden. Während der Suche kann man mit der <BREAK> - Taste den Suchvorgang abbrechen.

8. Sprung zum Disk-Operating-System:

Das DUP.SYS-Menü wird in den Computer geladen und gestartet.
Achtung, Quelltext sichern nicht vergessen!

9. Toolbox

Hier können (selbstprogrammierte) Hilfsprogramme zum Assembler geladen und gestartet werden. Es können auch "normale" Maschinenprogramme geladen werden. Lesen Sie bitte den gesonderten Abschnitt "Toolbox"!

10. Das Setup-Menü / Voreinstellungen:

Hier können Sie folgende Einstellungen vornehmen:

1 - Printer-Device (Default-Gerät bei Anwählen des Drucker-Symbols).
2 - Farben einstellen (mit "<", ">", "+" und "-").
3 - Den Zeichensatz wählen, der nach dem Booten eingeschaltet sein soll.
4 - Den Modus für die Direktory-Sortierroutine wählen.
5 - Das INVERSE-Menü-Flag setzen (von den Farben abhängig).
6 - Den IO-Sound an- oder abschalten.
7 - Die erweiterte Schreibweise für Binärzahlen einschalten.
8 - Die erweiterte Schreibweise für Hexadezimalzahlen einschalten.
9 - Das Ausgabeformat für die High/Low-Kennzeichnung wählen.
10 - Den Layout-Modus des Editors aktivieren.
11 - Die Warnungen des Assemblers unterdrücken.

Die geänderten Einstellungen können mit "Save Setup" abgespeichert, mit "Exit & use" nur benutzt oder mit "Load Setup" die vorhandene Einstellung geladen werden.

1 - Printer-Device:

Wenn Sie keinen Drucker haben, den Sie über den Gerätetreiber "P:" ansprechen können, aber z.B. ein "RS-232"-Treiber haben, so können Sie diesen vom Monitor aus einladen und initialisieren. Tragen Sie dann hier die Gerätetreiberbezeichnung ein (z.B. "R:300,8N1"). Der hier eingetragene Gerätename wird später auch als Default für die Reportfunktion des Assemblers benutzt.

2 - Farben einstellen:

Stellen Sie hier die Farben ein, welche der Assembler nach dem Booten haben soll. Manche Leute wollen lieber ein popelgrünes Bild, ich bevorzuge schwarz-weiß.

3 - Zeichensatz wählen:

Hier können Sie einschalten, ob sie lieber den normalen Zeichensatz oder den mit den internationalen Sonderzeichen haben wollen. Der Fileselektor aktiviert automatisch den Standartzeichensatz, da er ein paar der Grafikzeichen benötigt. Im Normalfall sollte auch der Standard Zeichensatz eingeschaltet sein. In einer früheren Version des Assemblers wurde der Zeichensatz mit der Symboltabelle überschrieben, was nicht gerade besonders gut aussah. in dieser Version kann der Speicher der Symboltabelle auch in eine RAMDisk-Bank gelegt werden, hier entfällt dann daß flackern. Der Internationale Zeichensatz wird aber nur zum Speichern des Inhaltsverzeichnisses vom Fileselektor benötigt, dann ist aber eh der normale Zeichensatz eingeschaltet. Sollte man den Assembler auf "normal hashing" konfiguriert haben, so sollte der internationale Zeichensatz benutzt werden.

4 - Index Sort:

Hier kann zwischen den drei Modi "Full", "Name" und "None" umgeschaltet werden. Im Modus "Full" werden Unterverzeichnisse vor den Dateien angezeigt während bei "Name" nur nach den Namen sortiert wird. Bei "None" wird natürlich überhaupt nicht sortiert.

5 - Inverse Iconbar?

Abhängig von den gewählten Farben ist es nötig, dieses Flag zu setzen. Dieses hat nur einen optischen Sinn (ausprobieren). Die Auswirkung dieses Flags werden allerdings erst beim nächsten Aufruf der Menüleiste sichtbar.

6 - Serial IO-sound?

Mit diesem Flag kann das Ladegeräusch aus- oder eingeschaltet werden. Einige Speeder berücksichtigen dieses Flag allerdings nicht.

7 - Expand Binary?

Hiermit kann die "erweiterte" Schreibweise von Binärzahlen im Editor eingeschaltet werden. Ist dieses Flag auf "Yes", werden Binärzahlen immer in 8-oder 16-Bit-Gruppen ausgegeben, d.H. ggf. wird mit Nullen aufgefüllt!
Beispiel: %1101 oder %00001101.

8 - Expand Hex?

Ist diese Eintellung aktiviert, werden Hexadezimalzahlen ("Sedezimalzahlen") Zwei- oder Vierstellig angezeigt. Beispiel: $1/$01 oder $123/
$0123.

9 - Expand High/Low?

Erweiterte Schreibweise der High- und Lowbyteselektion einschalten. Die "normale" Schreibweise ist "<" für Lowbyte oder ">" für Highbyte, z.B. "LDA #>Test" läd den Akkumulator mit dem Highbyte des Symboles "Test". Ist diese Option eingeschaltet wird statt "<" ein "L:" ausgegeben und statt ">" ein "H:", also "LDA #H:Test". Der Editor akzeptiert beide Möglichkeiten beim eingeben.

10 - Layout?

Wenn der Layoutmodus eingeschaltet ist, kann man den Cursor im Editor frei und ohne Einschränkungen bewegen. Im Normalmodus wird beim Verlassen einer Zeile der Cursor an das Ende der neuen Zeile gesetzt, wenn die Cursorposition rechts von diesem Ende ist. Einfacher ausgedrückt: Ist die neue Zeile, in die der Cursor bewegt wird, kleiner als die Cursorspalte, so wird der Cursor an das Ende dieser Zeile zurückgesetzt.

11 - Warnings?

Dies ist eine zusätzliche Option für den Assembler. Warnungen sind "Fehler, die man eigentlich auch vernachlässigen kann". Einige Programmierer benutzen z.B. gerne "LDA #-1" anstelle von "LDA #$FF", um den Akkumulator mit $FF zu laden. Dies ist besonders bei Editoren angebracht, die den Quelltext Zeichenweise speichern, um Platz zu sparen. Der Ausdruck "-1" entspricht ja "$FFFF". Dies ergibt aber einen "Operand too big"-Fehler, da mein Assembler halt hier einen Fehler erkennt (ein 8-Bit Register wird mit einem 16-Bit Wert geladen). Andere Assembler mögen sich ja einen Dreck darum kümmern. Ausserdem wird ein "ZP-boundary" Fehler gemeldet, wenn man z.B. bei einer "Indirect-Y" Adressierung einen größeren Wert als $FF angibt. Bei dieser Adressierungsart sollte der Programmierer eigentlich wissen, das es hier keine 16-Bit Adresse gibt! Deswegen wird auch hier kein Fehler ausgegeben, wenn "Warnings?" abgeschaltet wird.

Bei den Punkten 3 - 11 wird die Einstellung mit <SPACE> geändert. In der letzten Zeile wird mit <SPACE> zwischen "Load Prefs", "Save Prefs" und "Quit Prefs" umgeschaltet. Ein Punkt kann mit <RETURN> oder ESCape übersprungen werden (Bei Load, Save oder Quit Prefs führt <RETURN> zur Ausführung der Funktion (z.B. Save Prefs) und dann zu einer Rückkehr in den Editor, während man mit ESCape wieder bei Punkt 1 landet).

Anmerkung zu den folgenden Block-Funktionen:

Da kein Platz mehr im Rechnerspeicher war (ein Block kann bis zu 16 K-Bytes groß sein), wird er als File abgespeichert. Dieses File heißt "CLIP.BLK". Sollte eine RAMDisk vorhanden sein, wird das File in der RAMDisk abgelegt, ansonsten im Hauptverzeichnis auf Laufwerk #1!!!

11. - Block merken

Der im Editor markierte Block wird abgespeichert. Der Text wird nicht geändert.

12. - Block einfügen

Ein vorher abgespeicherter Block wird an der Cursorposition (Zeile) eingefügt. Sollte der Speicher nicht ausreichen, gibt es einen Rüffel!

13. - Block ausschneiden

Der markierte Block wird abgespeichert und aus dem Quelltext entfernt!

14. - Block löschen

Der markierte Block wird (nur) aus dem Quelltext gelöscht.

15. - Quelltext reparieren / retten (Old-Funktion)

Diese Funktion kann nach dem versehentlichem Löschen eines Quelltextes bzw. nach dem Booten aufgerufen werden, um einen noch im Speicher stehenden Quelltext zurückzuholen. Diese Funktion wirkt nur auf die aktuelle (angewählte) Page und muß ggf. auf mehrere Seiten angewendet werden!

16. - Quelltext löschen

Der Quelltext in der aktuellen Seite wird gelöscht! Wird diese Funktion mit gedrückter SHIFT-Taste aufgerufen, so werden ALLE Seiten gelöscht und die Page 1 angewählt!

Allgemeine Anmerkungen:
==========

Bei der PARENT-Funktion des Fileselektors kann es bei einem größeren Pfad passieren, daß in der Menüleiste Grafik-Schrott angezeigt wird. Dieses Verhalten hat aber keinen Einfluss auf die Funktionen; beim erneuten Aufruf der Menüleiste erscheint die Grafik wieder normal. Dieser Fehler läßt sich auch nur beseitigen, wenn für den Grafikpuffer ein eigener Speicherbereich reserviert würde, was aber nicht vorgesehen ist. Zur Zeit teilt sich der Grafikpuffer den Speicher mit anderen Puffern, z.B. Filenamen usw.

- Der Sprung ins DUP-Menü muß mit "Y" bestätigt werden, da eine Rückkehr vom DUP-Menü zum Assembler nicht vorgesehen ist (vom MyDOS). Nun, Backadr. zum Assembler ist $2800 - für alle MEM.SAV-fetischisten. Ein Sprung ins DUP-Menü ist eigentlich auch nicht nötig (ausser zum kopieren, und da wird MEM.SAV so- wieso ungültig), da fast alle dort vorhandenen Funktionen auch mit dem XIO Kommando im Monitor durchgeführt werden können.

Die Monitor Shell V2.1

Bei der Entwicklung des Monitors für diese Version des Assemblers mußte ich einen neuen Weg gehen. Da der zur Verfügung stehende Speicherplatz für den Monitor nur 2 k-Bytes "groß" ist, war es mir nicht möglich, einen komfortablen und speicherresidenten Monitor zu programmieren. Daher besteht der eigentliche Monitor nur aus der "Shell" (Hülle, Umhüllung oder Schale...). Diese Shell hat nur 4 residente Kommandos: DIR, BYE, GO und CONT. Alle anderen Befehle sind als sogenannte "Overlay-Kommandomodule" ausgelegt. Das sind kleine Programme, welche beim Aufruf in den Monitor geladen werden. Wird zum Beispiel das Kommando "LIST" eingegeben versucht die Shell, die Datei "LIST.OVL" im Verzeichnis OVERLAY in der RAMDisk zu öffnen und einzulesen. Sollte es diese Datei nicht geben, gibt die Shell einen "unknown command!"-Fehler aus. Durch diese Technik lassen sich theoretisch bis zu 64 Kommandos erstellen (in ein Verzeichnis passen ja maximal 64 Dateien). Weiterhin werden die benötigten Parameter nicht mehr mit dem Kommando übergeben (z.B. "SAVE $4000 $5FFF D1:TEST.COM"), sondern im Dialogbetrieb abgefragt. Der Benutzer gibt also "SAVE" ein, und der Computer fragt im Dialog nacheinander die Adressbereiche und den Namen ab. Dies ist eine recht komfortable Lösung, wie ich meine.

Anmerkung:

Ein Kommando wird zuerst auf der RAMDisk gesucht (wenn vorhanden). Sollte es nicht in der RAMDisk stehen, wird es "eventuell" auf der EPROMDisk gesucht (wenn konfiguriert), und als letztes von Laufwerk #1! Dabei wird zuerst versucht, das Verzeichnis "OVERLAY" in dem jeweiligen Laufwerk anzusprechen. Sollte es nicht vorhanden sein wird versucht, das Kommando aus dem Hauptverzeichnis des jeweiligen Laufwerks zu laden! Zum "eventuell" bei der EPROMDisk: Sollte der Assembler auf der EPROMDisk installiert sein (wird im Konfigurationsprogramm eingestellt) wäre es natürlich Platzverschwendung, wenn man die Overlay-Kommandomodule zusätzlich in die RAMDisk kopiert. Dann werden also Kommandos zusätzlich auf der EPROMDisk gesucht, sonst (wenn von Disk gebootet wurde) nicht!

Die residenten Kommandos der Monitor-Shell:

- DIR (Directory, Inhaltsverzeichnis)

Nach Eingabe diese Kommandos erscheint die Meldung "Index of". Drückt man nur <RETURN>, so wird das Inhaltsverzeichnis von "D:*.*" ausgegeben. Dieses ist beim MyDOS das "default"-Verzeichnis, d.H. es kann mit einem CIO-funktionscode ein Verzeichnis als aktuelles Verzeichnis gewählt werden, und auf dieses wird dann mit "D:" (ohne Laufwerksnummer!) zugegriffen, siehe auch "PATH". Beispiel: "PATH D3:DEMOS>XE_ONLY" <RETURN> setzt das aktuelle Verzeichnis. Alle Filenameneingaben mit "D:" ohne Nummer beziehen sich auf dieses Verzeichnis. "LOAD D:PAGEFLIP.COM" würde dann das Programm "PAGE- FLIP.COM" von der Diskette in Laufwerk 3 aus dem Verzeichnis "XE_ONLY" laden; dieses liegt im Verzeichnis "DEMOS" und dieses wiederum im Hauptverzeichnis! Nun, man könnte ja auch "LOAD D3:DEMOS>XE_ONLY>PAGEFLIP.COM" <RETURN> eingeben...
Auf die Meldung "Index of" kann ein kompletter Pfadname eingegeben werden. Hierbei ist allerdings die Einschränkung zu beachten, das die Eingaben nur 25 Zeichen lang sein können!

- BYE (Verlassen der Monitor-Shell und Rückkehr in den Editor)

Mit diesem Kommando kommt man wieder in den Editor zurück.

- GO (Starten von Maschinenprogrammen)

Nach diesem Kommando muß auf die Meldung "Address:" die Adresse eingegeben werden, an der ein Maschinenprogramm gestartet werden soll. Die CPU-Register Accu, X und Y werden mit den Systemvariablen AX, XR und YR initialisiert (siehe auch Kommando "REGS"), d. H. es können auch Werte in den Registern an ein Programm übergeben werden.

- CONT (Continue, fortfahren nach Programmunterbrechung)

Mit CONT kann ein Progamm, welches unterbrochen (nicht beendet!) wurde, fort gesetzt werden. Nach der Eingabe fragt der Computer nach der Adresse, an der das Programm fortgesetzt werden soll. Im Allgemeinen ist hier nur <RETURN> zu drücken um an der Abbruchstelle fortzufahren. Die Adressabfrage ist nur für Profis interessant, Assemblerneulinge sollten die Finger von dieser Möglichkeit lassen, da bei unbedachter Anwendung das System abstürzen kann! Beim Unterbrechen wird der Displayliststatus gesaved und die "normale" Dis playlist eingeschaltet. Bei CONT wird die alte Displaylist wieder eingeschaltet,bevor die Programmausführung fortgesetzt

wird. Wurde ein Programm beendet (d. H. mit RTS abgeschlossen), so kann natürlich kein CONT ausgeführt
werden. In diesem Fall erscheint "Can´t continue"!

Ein Programm kann auf zwei arten Unterbrochen werden: Mit dem Maschinen sprachebefehl "BRK" oder, wenn die Interrupts nicht gesperrt wurden, mit der Tastenkombination "SHIFT/BREAK". Es erfolgt die Meldung "Stopped at <hex address>". Wurde das Programm mit einem BRK unterbrochen, so wird mit CONT ohne Adressangabe zu der nächsten Anweisung gesprungen (also hinter den BRK- Befehl!), sonst (bei Angabe einer Adresse, oder wenn das Programm mit SHIFT/ BREAK unterbrochen wurde) wird zur Abbruchadresse gesprungen.

Mitgelieferte Overlay-Kommandodateien:

Hier eine Auflistung der mitgelieferten Overlay-Kommandodateien und eine Kurzbeschreibung:

Kommando: Beschreibung:

PATH Setzten des "Default" Directorys "D:" (für MyDOS). PATH D8: und DIR D:*.* ergibt den Inhalt von D8: bzw. dem evtl. angegebenen Verzeichnis.

XIO Ausführen eines beliebigen CIO-Kommandos. Es wird nach dem Kommando, den beiden Steuerbytes AUX1 und AUX2 sowie nach dem Geräte/Filenamen gefragt. Nach einer Sicherheitsabfrage wird das Kommando ausgeführt.

REGS Anzeige der Register und Statusflags. Alle Register und Flags können geändert werden!

LIST Gibt ein Listing ab der angegebenen Adresse aus (Disassembler). Wird als Startadresse nur <RETURN> gedrückt, nimmt der Computer den aktuellen PC (Program-Counter) als Startadresse. Wird als Endadresse nur <RETURN> ohne weitere Eingabe gedrückt, so gibt der Computer immer 13 Zeilen aus. Auf Tastendruck werden weitere 13 Zeilen ausgegeben, mit ESCape wird abgebrochen. Wenn eine Endadresse eingegeben wurde, so wird ohne Unterbrechung vom Anfang bis zum Ende gelistet, mit einem Tastendruck kann das Listen unterbrochen werden, nochmaliger Tastendruck (außer ESCape!) läßt das Listing weiterlaufen. Der Computer fragt dann noch nach einem Ausgabegerät, wohin das Listing gesendet werden soll. Es kann ein Gerätename oder ein kompletter Dateiname eingegeben werden. Wird nur <RETURN> gedrückt, so erfolgt natürlich keine externe Ausgabe.

SAVE Dieser Befehl speichert einen Speicherbereich als Binärfile ab. Der Computer fragt nach Start- und Endadresse. Eine Datei kann auch "adressversetzt" gespeichert werden. Hierzu ist bei der Frage "Into:" die Zieladresse einzugeben, an die das Binärfile später geladen werden soll. Folgt dem Filenamen ein "/A", so wird an das (existierende) File angehängt!

WRITE Speichert einen Speicherbereich als normale Datei ab, also ohne Binärfileheader! Einzugeben ist die Start- und Endadresse sowie der Filename. Folgt dem Filenamen ein "/A", so wird an die be stehende Datei angehängt (Append-Funktion).

LOAD Mit diesem Kommando kann ein Binärfile geladen werden. Es erfolgt ein "Filetrace", d.H. die Adressbereiche des Files werden angezeigt. Wird nur der Filename eingegeben, so frangt das Programm nach ob die jeweiligen Init- oder Autorunvektorenausgeführt werden sollen. Folgt dem Namen ein "/L", wird nur geladen, d.H. Init/Autorun wird NICHT ausgeführt! Wenn dem Namen ein "/R" folgt, wird nur der Autorunvektor ausgeführt und Init wird ignoriert. Als vorletzte Möglichkeit gibt es "/I", welche alle Initialisierungsaufrufe ausführt, aber den Autorun nicht! Ein

"/A" steht hier für "Automatisch", d. H. Init und Autorun werden automatisch ausgeführt. Wenn dem Namen keine Option folgt, werden Sie jedesmal (wenn ein solcher Init- oder Autorunvektor auftaucht) gefragt, ob er ausgeführt werden soll. Dies muß mit <RETURN> bestätigt werden!

SCAN Durchsucht den angegebenen Speicherbereich nach einer Bytefolge oder einem String. Also Joker kann man beim Suchen nach einer Bytefolge eine $00 eingegeben werden, beim Suchen nach einem String kann der "Klammeraffe" (SHIFT/8) als Joker verwendet werden. Beim Suchen nach einem String wird noch gefragt, ob zwischen Klein- und Großbuchstaben unterschieden werden soll.
Wird diese Funktion nicht gewünscht, muß hier mit "N" geantwortet werden! Ist diese Funktion aktiv, findet sie auch Bildschirmcode. Als Default für die Startadresse wird der aktuelle PC genommen, für die Endadresse ist der Default $BC00. Ist die Suche abgeschlossen, kann in einem anderen Adressbereich weitergesucht werden ohne den Suchstring (bzw. die Bytefolge) nochmal eingeben zu müssen.

EDIT Dies ist ein recht komfortabler Speichereditor. Es kann in Hex editiert werden und in ASCII (oder im internen Bildschirmcode). Der Editor zeigt links die Adressen, in der Mitte jeweils 8 Hexzahlen pro Zeile und rechts die Daten im internen Bildschirmcode oder im ASCII-Code an. Oben wird der aktuelle Modus angezeigt ("Int" für "Interner Bildschirmcode" oder "ASC" für "ASCII-Code". Die Anzeige kann jederzeit mit der Tastenkombination CONTROL/INVERSE zwischen Int und ASC umgeschaltet werden. Mit der TABulatortaste wechselt man zwischen dem "Hexfeld" und dem "Int-" bzw. "ASC-" Feld. Wenn man sich im "Zeicheneditiermodus" befindet, werden keine inversen Zeichen angezeigt (damit man den Cursor besser sieht). Der Cursor wird mit den Cursortasten (mit CONTROL) gesteuert. Mit SHIFT-CONTROL-CursorUp/Down kann um jeweils$80 Bytes gescrollt werden. Wird im Hexfeld <RETURN> gedrückt, wandert der Cursor ein Byte weiter (entspricht "Cursor-Right"), während im ASC/Int Mode der Cursor in die nächste Zeile bewegt wird. Achtung: Eingegebener Text wird gemäß des eingestellten Modes im internen oder im ASCII Code in den Speicher geschrieben! Wird keine Taste gedrückt bzw. keine Eingabe getätigt, so erfolgt automatisch ca. 5 mal in der Sekunde ein Refresh des Bildschirms, d. H. der Bildschirminhalt (Hexwerte und Zeichencodes) werden neu aufgebaut. Somit werden auch interrupt bedingte Änderungen in den Speicherstellen sofort sichtbar. Ansonsten erfolgt ein Neuaufbau des Screens bei jedem Tasten- druck.

READ Dieses Kommando dient zum Einlesen von Daten (nicht-Binärfiles) an eine beliebige Adresse. Es kann die Adresse sowie die maximale Länge angegeben werden.

FILL Füllt einen Speicherbereich mit dem angegebenen Byte. Wird als "Fillbyte" nur <RETURN> gedrückt, so wird mit 0 gefüllt. Wird eine Wert größer als 255 eingegeben (also ein WORT (16 Bit)), so wird in Wortgröße gefüllt!

MOVE Dient zum verschieben (kopieren) eines Speicherbereiches in einen Anderen. Quell- und Zieladresse dürfen sich natürlich dabei überschneiden.

INFO Gibt eine Aufstellung der momentan verfügbaren Kommandos und wartet auf die Eingabe einer Kommandobezeichnung. Wird nur <RETURN> gedrückt so wird eine Kurzerläuterung zu den residenten Kommandos ausgegeben, ansonsten wird ein Hilfstext zum angegebenen Modul angezeigt.
Anmerkung: INFO gibt nur die Kommandos aus, die in dem gleichen
Verzeichnis stehen wie das INFO-Kommando selbst!

CALC Zeigt das Ergebnis des eingegebenen Wertes (bzw. der eingegebenen Formel) hexadezimal, binär und dezimal an.

Batchfile Enhancement V1.7

Geschrieben für MyDOS, getestet mit Version 4.50

Was ist das "Batchfile enhancement"? Nun, ganz einfach: Eine art "Interpreter" für sogenannte Batch-files. PC`s und andere Rechnern können beim Systemstart eine Textdatei "ausführen", in welcher verschiedene Kommandos stehen. So werden z.B. deutsche Tastaturtreiber geladen usw.! Nichts anderes ist das Batchfileenhancement - grob gesagt. Das MyDOS ist ja schön und gut, aber man vermißt doch schon eine Funktion mit der die RAMDisk initialisiert werden kann oder ein Menü dargestellt werden kann.

Das BFE (so nenne ich es mal in Zukunft) wird als AUTORUN.SYS auf eine MyDOS-Diskette kopiert. Es belegt den Speicher von $8000 bis $9FFF. Es sucht als erstes eine Textdatei "D:AUTOEXEC.BAT" und führt die dort enthalteten Kommandos aus. So kann mit COPY eine Datei in die RAMDISK oder mit XCOPY sogar ein ganzes Verzeichnis kopiert werden, ein Menü angezeigt werden, Dateien geschützt werden usw. Befehle können auch bedingt ausgeführt werden. Ein Direktmodus ist ebenfalls vorhanden und alle ROMDisk - Benutzer können auch aufatmen!

Nun zu den einzelnen Befehlen mit den Parametern. Filenamen werden, außer bei RENAME, mit einem Leerzeichen getrennt. Z.B. "COPY D1:README.TXT D8:READ.ME", aber "RENAME D1:alter Name,neuer Name"! Die RAMDisk kann im Batchfile mit RD: angesprochen werden. Ausserdem gibt es eine Variable (%), bei "D%:*.TXT" wird dann die dem %-Zeichen zugewiesene Laufwerksnummer eingesetzt. IF – ENDIF Strukturen sollten der Lesbarkeit halber immer eingerückt werden. Sollte kein AUTOEXEC.BAT auf der Disk sein, oder wurde es ausgeführt, kommt man in den Direktmodus. Beim starten eines Batchfiles befindet sich das System immer im "ECHO ON"-Modus, d.H. die gerade ausgeführten Befehle werden angezeigt. Siehe auch ECHO ... Auch Basicprogramme können gestartet werden.

Alle in Klammern angegebenen Texte usw. sind optional, d.H. nicht zwingend erforderlich. Sie haben nur eine bestimmte Funktion. Es wird nur das erste Zeichen erkannt! "U", "Update" oder "Unparteiisch" sind gleich!

- EXEC <Filename>
Mit EXEC kann eine andere Batch-datei ausgeführt werden. Ein GOSUB ist nicht möglich. Bei EXEC wird automatisch ein ECHO ON ausgeführt.

- ECHO ON
Aktiviert die List-Funktion. Alle Befehle des Batchfiles werden bei ihrer
Ausführung angezeigt.

- ECHO OFF
Schaltet die Anzeige der Befehle ab! (Nur in einem Batchfile sinnvoll).

- ECHO (Text)
Der angegebene Text wird ausgegeben (mit allen Leerzeichen).

- MAKEDIR <DirName>
Erzeugt ein neues Verzeichnis.
Beispiel: MAKEDIR RD:SUBDIR <Return> MAKEDIR RD:SUBDIR>NOCHEINS <Return>

- ASK FOR <KeyList>
Wartet, bis eine der angegebenen Tasten gedrückt wird (oder ESCape).
ECHO Ja oder Nein? <Return> ASK FOR JN <Return>

- IF EXIST <Filename>
Überprüft das Vorhandensein einer Datei und führt die nachfolgenden Befehle bedingt aus (bis zum ENDIF) wenn die Bedingung erfüllt war.

- IF NOT EXIST <Filenname>
Beispiel: IF NOT EXIST D1:DUP.SYS <Return> ECHO Kann nicht ins DOS, keins da!
<Return> STOP <Return> ELSE <Return> EXIT <Return> ENDIF <Return>

- IF ANSWER = <Taste>
Führt die nachfolgenden Befehle bedingt aus, wenn in einer vorangegangenen ASK FOR-Anweisung diese Taste gedückt wurde. Beispiel:
ASK FOR JN <Return> IF ANSWER = J <RETURN> DELETE D1:*.TXT <Return> ELSE <Return> ECHO Nichts wurde gelöscht! <Return> ENDIF <Return>

- IF RAMDISK (-)
IF RAMDISK prüft auf das vorhandensein einer RAMDisk! Ohne die Option "-"
wird auch ein Hardwartest (130XE-kompatibel) durchgeführt, d.H. sollte zwar im DOS eine RAMDisk konfiguriert sein, muß sie hardwaremäßig auch da sein!
IF RAMDISK - prüft nur, ob die RD konfiguriert ist. Dies ist für nicht-130XE
RAMDisks. Ist eine RD da werden die nachfolgenden Befehle bedingt ausgeführt.

- IF NOT RAMDISK (-)
Beispiel: IF NOT RAMDISK <Return> ECHO Keine RAMDisk vorhanden! <Return>
STOP <Return> ENDIF <Return>

- ENDIF
Beendet eine bedingte Ausführung. IF-ENDIF Konstruktionen können übrigens 8-fach verschachtelt werden!

- ELSE
Kehrt die Bedingung auf dem Stapel um (wenn möglich). Zwei mal ELSE ergibt wieder den "Normalzustand". Beispiel:

```
ASK FOR AB
IF ANSWER = A
 ECHO Sie haben die Taste A gedrückt!
ELSE
 ECHO Sie drückten auf die Taste B!
ELSE            ; Nur als Beispiel (eigentlich unsinn)
 ECHO Wieso nicht B?     ; Jetzt wieder Bedingung für "A"!
ENDIF
```

Wenn die Taste A gedrückt wird, erscheint der Text "Sie haben die Taste A gedrückt!" und "Wieso nicht B?"!

- LOAD <Filename> (Chan) (Q/L/I/R)
  Mit Load wird das angegebene Binärfile geladen. Die Optionen sind:
Q(uit) - Wird ein File geladen und mit RTS beendet, wird nicht zum BFE zurückgesprungen sondern in das DUP.SYS (oder ins BASIC). Das File wird geladen und alle Init/Autorun-Vektoren ausgeführt!
L(oad) - Das File wird nur geladen, weder INIT noch RUN werden ausgeführt!
R(un) - Das File wird geladen und der Autorun ausgeführt, Init NICHT!
I(nit) - Alle Initialisierungsaufrufe werden ausgeführt, Autorun NICHT!

Mit (Chan) kann optionell ein Kanal gewählt werden, über den die LOAD- Funktion ausgeführt werden soll. Werte von 0 bis 7 sind möglich, Kanal 0 wird allerdings vom Editor benötigt! (Chan> muß vor der Option stehen (wenn benutzt)! Dies ist z.B. nötig wenn Programme geladen werden sollen, welche das Laden "mittendrin" selbst übernehmen. Dies ist z.B. bei Turbo-BASIC, dem Compiler und dem Runtime-Modul der Fall. Diese setzten vorraus, daß das Programm von Kanal 1 geladen wird. Im Batchfile-Enhancer werden Kanal 5,6 und 7 benutzt, bei XCOPY auch Kanal 4!
Beispiel:

```
LOAD D1:TURBOBAS.COM 1 Quit
```

- COPY <Quelle> (<Ziel>)
Dient zum Kopieren eines einzelnen Files. Dateiangaben im MyDOS-üblichen Format. Siehe auch TYPE. Fehlt <Ziel> wird automatisch der Editor genommen!

- SETDIR <DirName>
Damit kann das "Defaultverzeichnis" von MyDOS angesprochen werden.
Beispiel:
SETDIR RD:TEXTE>PRIVAT <Return> TYPE D:ELKE.TXT P: <Return> entspricht in etwa TYPE RD:TEXTE>PRIVAT>ELKE.TXT P: <Return>! Nach SETDIR kann mit D: auf das Verzeichnis zugegriffen werden.

- FORMAT DISK <Device>
Dient zum formatieren einer Diskette. Achtung: Es findet keine Sicherheitsabfrage statt!
Beispiel: FORMAT DISK D2: <Return>

- TYPE <Quelle> (<Ziel>)
Wurde nur aus optischen Gründen implementiert und sollte zum Anzeigen von Texten usw. benutzt werden. Siehe auch COPY.

- GOTO <Label>
Springt zum angegebenen Label. Siehe auch LABEL! Es können nur Vorwärtssprünge ausgeführt werden!

- LABEL <Label>
Ist das Sprungziel für GOTO. Achtung! <Label> ist Case-Senitive, d.H. Groß- und Kleinschreibung beachten. Vom Label werden nur die ersten acht Zeichen beachtet!

- STOP
Beendet die Ausführung des Batchfiles und geht in den Direktmodus.

- DIR (<FileMask> (<Ziel>))
Zeigt das Inhaltsverzeichnis an. DIR ohne Parameter zeigt das Inhaltsverzeichnis des Default-Directories "D:*.*" an (siehe SETDIR). Soll <Ziel> angegeben werden muß <FileMask> angegeben werden! DIR RD:*.* <Return> zeigt den Inhat der RAMDisk auf dem Bildschirm an, DIR RD:*.TXT P: <Return> druckt es aus!

- BUFFERS = <Size>
Der interne Puffer zum kopieren ist nur ca. 50 Bytes "groß". Damit beim kopieren , beim DIR usw. nicht so häufig zwischen <Quelle> und <Ziel> umgeschaltet werden muß, kann hiermit der Puffer vergrößert werden, und zwar um bis zu 9 K-Bytes. Dieser Buffer geht dann von $9000 bis zum angegebenen Wert. Achtung! Wenn das BASIC eingeschaltet ist oder wird, wird der Buffer automatisch auf 3 KB zurückgesetzt, da ab $9C00 der Bildschirm anfängt! Size ist eine Zahl zwischen 0 und 9! Es kann also z.B ein Programm nach $9800 geladen werden und der Buffer auf 2 kB begrenzt werden (sonst wird evtl. das Programm im Speicher überschrieben).

- INIT RAMDISK (U) (M)
Die RAMDisk wird formatiert und daß DUP.SYS in die RAMDisk kopiert! Wird die Option "U" (für Update) angegeben, so wird die RAMDisk nur formatiert, wenn sich kein DUP.SYS auf ihr befindet! Die Option "M" (für MemSav) erzeugt gleichzeitig die MEM.SAV Datei. Achtung! Bei "U(pdate)" wird nur nach dem File "DUP.SYS" auf der RAMDisk gesucht. Dieses File wird nicht geprüft! Sollte sich also DUP.SYS von einem anderen DOS auf der RAMDisk befinden, wird dies nicht erkannt. Wenn der Computer kurz aus- und wieder eingeschaltet wird kann es passieren, daß das Directory in der RAMDisk erhalten bleibt, aber die Daten natürlich zerstört wurden. In den letzten beiden Fällen MUß der Speicher komplett gelöscht werden. Dazu den XL für ein paar Sekunden ausschalten!

- RAMDRIVE = <DriveNum>
Setzt die Laufwerksnummer für die RAMDisk.

- BASIC OFF
Schaltet das BASIC aus und baut den Editor neu auf.

- BASIC ON
Schaltet das BASIC ein und baut den Editor neu auf.

- LOCK <FileMask>
Schützt die angegebene(n) Datei(en) vor versehentlichem Überschreiben.

- UNLOCK <FileMask>
Hebt den Schreibschutz wieder auf (siehe LOCK).

- DELETE <FileMask>
Löscht die angegebe(n) Datei(en) ohne Sicherheitsabfrage!

- RENAME <alter Name,neuer Name>
Dient zum Umbenennen von einer oder mehreren Dateien!

- EXIT
Verläßt das BFE-Programm und springt ins BASIC oder DUP Menü.

- XCOPY <Maske> <Ziel> (Q) (U)
XCOPY ist wohl einer der mächtigsten Befehle. Er benötigt auch 4 Diskbuffer (d.H. das MyDOS muß auf mindestens 4 gleichzeitig geöffnete Dateien eingestellt sein). XCOPY kopiert nur aus dem "default"-Verzeichnis (D:).
Beispiel: MAKEDIR RD:TEST <Return> SETDIR D1:DEMOS <Return> XCOPY *.COM RD:TEST <Return> kopiert alle D1:DEMOS>*.COM -Dateien in das Verzeichnis TEST auf der RAMDisk! Die Option Q(uiet) bewirkt ein "leises" kopieren (ohne Anzeigen von "Copying RD:TEST>...."), wird die Option U(pdate) angegeben, werden nur die Dateien kopiert, welche noch nicht im Zielverzeichnis vorhanden sind. Bei Maske darf kein Laufwerksbuchstabe angegeben werden.

- APPEND <Quelle> <Ziel>
Die Datei <Quelle> wird an die Datei <Ziel> angehängt bzw. "ankopiert". Die Quelldatei wird nicht verändert.

- BASIC: (<Basic Line>)
Hiermit kann man in den BASIC-Interpreter gehen und eine Zeile übergeben. Das BFE-Programm wird verlassen. Das Basic muß eingeschaltet sein.
Beispiel: BASIC: GRAPHICS 8:COLOR 1:PLOT 0,0:DRAWTO 319,159:PLOT 319,0:DRAWTO 0,159:RUN "D1:MyPRG.BAS" <Return> zeichnet ein "großes X" und startet danach das angegebene Programm.

- ROMDRIVE = <DriveNum>(,EprNum)
Setzt die ROMDisk (David) auf die angegebene Laufwerksnummer. Die Epromnummer läßt sich auch setzen (von 0 bis 7)!
Beispiel:
ROMDRIVE = 3 Setzt die LW-Nummer für die ROMDisk auf 3.
ROMDRIVE = 3 , 0 Wählt zusätzlich noch das 1. Eprom.
ROMDRIVE = , 2 Wählt Eprom 3, die LW-Nummer bleibt unverändert!

Zwischen Parameter und Komma muß mindestens ein Leerzeichen stehen! Wenn von der ROMDisk gebootet wird (bzw. das Batchfile ausgeführt wird), darf im Batchbetrieb weder die LW-Nummer noch das EPROM geändert werden! Wenn dieses gewünscht wird, einfach das Batchfile mit COPY in die RAMDisk kopieren und erst dann mit EXEC (von der RAMDisk) starten!

- SET % = <DriveNum>
Der Pseudo-Variablen "%" wird die angegebene Nummer zugewiesen. Wird im BFE ein Filename mit "D%:..." angegeben, so wird das %-Zeichen durch diese Nummer ersetzt.

- SET % TO ANSWER
Setzt die Pseudovariable "auf die Antwort". Ach, Mist! Beispiel:

```
ECHO Welches Laufwerk soll formatiert werden? A=Abbruch
ASK FOR 012A
IF ANSWER = A
 ECHO Abgebrochen - Gott sei Dank!
  ; Kommentare vergaß ich zu erklären!
ELSE
 SET % TO ANSWER
 FORMAT DISK D%:
```

ENDIF

- CLS
Dieser Befehl löscht nur den Bildschirm!

Ach ja: Mit dem Klammeraffen kann ein Zeilenweises ECHO OFF erzwungen werden!
Beispiel:

```
ECHO ON
@COPY D1:DATA RD:DATA
LOAD D1:MAINPRG.COM Quit
```

Der COPY-Befehl wird nicht angezeigt.

So, das solls erstmal gewesen sein! Gruß an alle Atari 8-Bit Freaks! Auswahlmenüs für BASIC oder Maschinenprogramme sind kein Problem - mit dem BFE!

Atari 8 Bit - Light years ahead!

Code Linker/Packer

Revision 2.3

Mit dem Packer können fast alle Binary-Load-Files in ihrer Länge verkürzt werden. Die zu packenden Programme müssen selbststartend sein. Außerdem können selbstgeschriebene Programme (z.B. Vorspann, Zeichensatz, Hauptprogramm usw.) zu einem einzelnen Programm "zusammengelinkt" werden.

Nach dem Laden meldet sich der Packer mit der Kommandozeile. Es gibt die Kommandos Dir, Load, Info, Pack, Rest. (Restore), Save, Quit, New, Auto-Batch und Toggle auto-compact:on/off. Zum Ausführen der Kommandos braucht nur der invers hervorgehobene Buchstabe gedrückt zu werden.

Zum Verstehen der Anleitung sind Kenntnisse über das Binärfile-Format unbedingt notwendig!

- Dir

Inhaltsverzeichnis ausgeben.

- Load

Binärfile einladen. Das einzuladende Binärfile kann auch mehrteilig sein. Es kann auch eine INIT-Adresse angegeben werden.

Beispiel:
Load - Filename(,Init)

- D:FONT.BIN normales Einladen.

- D:TITEL.COM, Einladen und Init-Vektor an 1. Adresse erzeugen.

- D:TITEL.COM,A800 Einladen und Init-Vektor auf angegebene Adresse erzeugen.

- ,1F00 Init-Vektor erzeugen oder austauschen.

Sollte ein Init-Vektor schon existieren, wird die Adresse ausgetauscht (nur beim letzten Eintrag)!

- Info

Es wird die Speicherbelegung aller geladenen Files angezeigt. Von gepackten Teilen wird die Originallänge und die gepackte Länge angezeigt.

Beispiel:
FROM:8000 TO:82A4 LEN:02A5
Init at:8000
FROM:0400 TO:0477 LEN:0078
FROM:1F00 TO:3EA0 LEN:2FA1 / 22EB Original / gepackte Länge
FROM:00F0 TO:00F6 LEN:0007
Init at:0400
Autorun:1F00

Zum Schluß wird noch die Gesamtlänge angezeigt (normale Filelänge und gepackte Filelänge (mit Entpacker)).

- Pack

Im Speicher stehende ungepackte Programmteile werden gepackt.

- Rest.
- Restore packed Data -
Im Speicher stehende gepackte Programmteile werden entpackt.

- Save
Abspeichern unter dem angegebenen Namen.
Beispiel:
Save - Filename(,Autorun(,Relocator)):

- D:AUTORUN.SYS,A800

Das (gepackte) Programm wird unter dem Namen "AUTORUN.SYS" gepeichert und ein Autostart-Vektor an die angegebene Adresse (Hier A800) erzeugt. Der Entpacker liegt im Kassetten-Buffer (0400-0477)

- D:SPIEL.COM,1F00,8000
Das Programm wird gespeichert, ein Autorun-Vektor nach 1F00 erzeugt sowie der Entpacker nach 8000 reloziert, liegt also beim späteren Einladen des Programmes nicht im Kassettenbuffer (Entpacker enthält absolute Sprünge und Unterprogrammaufrufe)!

- D:SPIEL.COM,,0100
Das Programm wird gespeichert und der Entpacker nach 0100 reloziert (CPU- Stack). Ein Autorun-Vektor wird nicht erzeugt, muß also schon im File vorkommen!!!

Sollte kein Programmteil gepackt sein, so wird auch kein Entpacker mit in das
File geschrieben (normale Link-Funktion).

- Quit
Verlassen des Programmes nach Sicherheitsabfrage (ins DOS).

- New
Löschen des Speichers für einen neuen Packvorgang. Wenn Sie mehrere Programme packen wollen, müssen Sie den Speicher nach dem Speichern wieder löschen, sonst "linken" Sie zwei Programme!

Neue Funktionen in V2.3:

- Auto-Batch
Hier kann das Linken und Packen auch automatisch erfolgen. Dazu muß einmalig eine Textdatei erzeugt werden in der die Befehle stehen, welche normalerweise einzeln eingegeben werden müssen! Siehe auch unteres
Beispiel! Um per Batch zu packen einfach "B" drücken und den Namen des Textfiles eingeben!

- Toggle auto-compact:On/Off
Hiermit kann das automatische zusammenfügen eines mehrteiligen Binärfiles eingeschaltet werden. Besonders bei alten "Kasetten-zu-Disketten"-Kopien kommt es vor, das viele kleine Binärfiles direkt hintereinander im Speicher stehen. Diese Fragmentierung könnte auch von einem alten Assembler oder Compiler stammen. Beispiel:

From:3000 To:30FB Len:00FC
From:30FC TO .... Len:00FC und so weiter.

So gibt es dann viele kleine Binärfiles welche direkt hintereinander im Speicher stehen. Sollte "auto-compact" auf "On" geschaltet sein, wird automatisch aus den einzelnen kleinen Files ein großes gemacht wenn die Adressendirekt hintereinander stehen.

Beispiele:

- Es sollen fertige Programme gepackt werden:

Evtl. Speicher mit "N"ew löschen!

"L" drücken
Link - Filename(,Init)
"D:SPIEL.OBJ"
FROM:8000 TO:BFFF LEN:4000 (Programmdaten)
FROM:02E0 TO:02E1 LEN:0002 (Autorun-Vektor)

"P" drücken
Packing ...

Evtl. mit "I" die Speicherbelegung anzeigen lassen.

"S" drücken
Save - Filename(,Autorun(,Relocator))
D:SPIELP.COM
Saving ...
FROM:0400 TO:0477 LEN:0078 (Entpacker wird geschrieben)
FROM:8000 TO:AC4E LEN:2C4F (Gepackte Daten)
FROM:00F0 TO:00F6 LEN:0007 (Informationen für den Entpacker)
New Vector Jump (Init/Autorun) (Starten des Entpackers)
FROM:02E0 TO:02E1 LEN:0002 (Autostart für Programm)

File saved!

Sollte beim Speichern die Fehlermeldung "Decruncher address in use" erscheinen, belegt das Programm den gleichen Speicher, in dem der Entpacker liegt. In diesem Fall muß eine (unbenutzte) Adresse (Relocator) eingegeben werden, an die der Entpacker geladen werden soll! Beispiel: "D1:SPIEL.COM,,0600" - Der Entpacker liegt in Page 6 ($0600)!

Wenn man ein eigenes Programm geschrieben hat, ergibt sich das Problem, daß es aus mehreren Teilen besteht, z.B. Zeichensatz, Hauptprogramm, Titelvorspann, Spritedaten (`tschuldigung, ich meine Player-Missile-Daten...) usw. und sofort!

Diese Dateien möchte man (sollte man möchten...) gerne in einer einzigen Programmdatei haben, also z.B. nicht extra den Zeichensatz nachladen. Dazu ist es allerdings nötig, das alle Dateien im Binary-Load-Format vorliegen ($FFFF -Startadresse - Endadresse - Daten - ...).

Angenommen, Sie haben folgende Dateien, welche Sie gerne "linken" wollen: Alle Adressangaben sind (mehr oder weniger fiktive) Beispiele:

Name: From: To: Len: Funktion des Programmes:

TITEL.COM 3400 37EA 07EB Titelbild mit DLI-Effekt während des Ladens.
ZEICHENS.BIN 3000 33FF 0400 Zeichensatz (Grafik) für das Spiel.
SPRITES.BIN 3800 3A3F 0240 Daten für Sprite/Player-Missile-Animationen.
DSOUND.BIN 4000 7FFF 4000 Digitalisierter Sound für Effekte im Spiel.
BALLER.OBJ 8000 A498 2499 Das eigentliche Ballerspiel-Programm.

Der Ablauf soll folgendermaßen stattfinden: Zuerst wird das Titelbild eingeschaltet, dann der Zeichensatz, die Sprites und der Sound geladen. Anschließend soll das Spielprogramm geladen werden, und vor dem Starten des Spieles muß der DLI-Effekt abgeschaltet werden. Nun, wie geht

man am besten vor?

1. Schritt:

Zuerst muß natürlich das Titelbild erscheinen. Dieses enthält zweckmäßigerweise am Anfang einen Sprung zur Einschaltroutine sowie einen Sprung zur ausschaltenden Routine ($3400: JMP <an>, $3403: JMP <aus>). Daher wird das Titelprogramm als erstes geladen mit "L" und "D1:TITEL.COM,". Die Adressangabe "3400" kann entfallen, da die Startadresse mit der ersten Ladeadresse übereinstimmt. Nur das Komma ist wichtig, damit ein Initialisierungsaufruf erfolgt!

2. Schritt:

Jetzt laden wir den Zeichensatz mit "L" und "D:ZEICHENS.BIN". Genauso verfahren wir mit den Dateien "SPRITES.BIN" und "DSOUND.BIN"!

3. Schritt:

Nun laden wir das Programm mit "L" und "D:BALLER.COM". Jetzt stellen wir fest, das wir vergessen haben den Init-Vektor nach 3403 anzugeben, damit der DLI-Effekt abgeschaltet wird (im Programm TITEL.COM)! Wir hätten statt "D:BALLER.COM" lieber "D:BALLER.COM,3403" eingeben sollen. Tragen wir den Initaufruf halt mit "L" und ",3403" nach (ohne Filenamen!) ...

4. Schritt:

Nun Packen wir das Programm zweckmäßiger Weise mit "P"...

5. Schritt:

Abspeichern! In unserem Beispiel soll die Startadresse des Programmes einfach mal $8285 sein. Dummerweise kopiert unserer TITEL.COM ein paar Daten in den Kassettenbuffer, so daß der Entpacker dort nicht liegen darf. Legen wir den Entpacker also in den unteren CPU-Stack: "S" und "D1:MEINGAME.COM,8285,0100" eingeben. Das Programm wird abgespeichert. Wenn jetzt als Entpackeradresse statt "0100" z.B. "9000" eingegeben würde, wäre die Fehlermeldung "Decruncher address in use!" die Folge! Der Adressbereich für den Entpacker geht von 0100 bis ca. BB8x. Sollte als Adresse z.B. 0000 oder C000 eingegeben werden, gibt es die Fehlermeldung "Decruncher at illegal address!".

Das File für die "Auto-Batch" Funktion sähe folgendermaßen aus:

```
NY
LD:TITEL.COM,3400
LD:ZEICHENS.BIN
LD:SPRITES.BIN
LD:DSOUND.BIN
LD:BALLER.OBJ,3403
P
SD:MEINGAME.COM,8285,0100
```

So braucht man (wenn z.B, eine Programmänderung vorgenommen wurde usw.) nur noch "B" und den Filenamen des Textfiles einzugeben!