

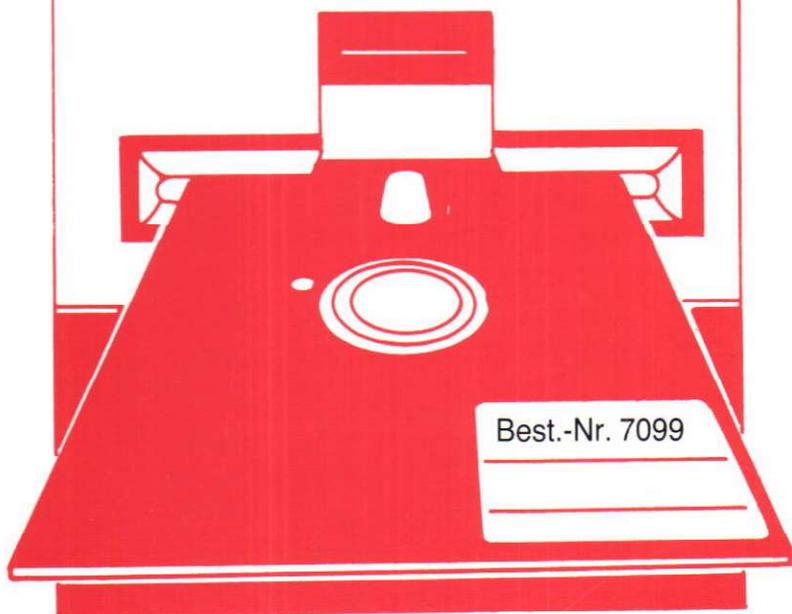
Software

Anwender-Handbuch

ATMAS-II

Makroassembler, Editor und Maschinsprache-Monitor

für ATARI-Computer 400/600XL/800/800XL und 130XE



HOFACKER

Bedienungshandbuch für

A T M A S - I I

**Ein Makroassembler, Editor
und Maschinensprache-Monitor
für ATARI-Computer 400/600XL/800/800XL
und ATARI 130XE
mit Diskettenstation und min. 48KByte RAM**

**Die Programme, Disketten und Bedienungshandbücher
des ATMAS-Systemes
unterliegen dem Copyright der**

**Ing. W. Hofacker GmbH
und
Dipl.-Ing. Peter Finzel**

Jegliche Rechte vorbehalten.

LADEANWEISUNG

=====

- 1.) Entfernen Sie eventuelle Steckmodule aus dem Cartridge-Schacht Ihres ATARI-Computers.
- 2.) Schalten Sie das Diskettenlaufwerk und den Fernseher (bzw. Monitor) ein.
- 3.) Sobald die rote LED des Diskettenlaufwerkes erloschen ist, legen Sie die Programmdiskette mit der Aufschrift oben und dem ovalen Ausschnitt nach hinten in das Laufwerk ein. Schließen Sie die Tür des Diskettenschachtes.
- 4.) Schalten Sie dann Ihren ATARI-Computer ein. Bei XL-Modellen muß während des Einschaltens die OPTION-Taste gedrückt werden.
- 5.) ATMAS-II wird jetzt geladen, nach kurzer Zeit erscheint der Vorspann, dann meldet sich der Editor, der Ladevorgang ist damit beendet.

Falls die Meldung 'ATMAS-II benötigt 48K-Speicher' erscheint, so gibt es mehrere Möglichkeiten:

- a) Es befinden sich noch Steckmodule im Schacht.
- b) OPTION wurde nicht gedrückt (nur bei XL).
- c) Ihr Computer hat zu wenig Speicherplatz. ATMAS-II benötigt mindestens 48KByte.

KURZBEISPIEL FÜR DEMO-PROGRAMM

=====

Wenn Sie sich vor dem Durchlesen des Bedienungshandbuches kurz ein Demoprogramm des ATMAS-II Systemes ansehen möchten, so können Sie das mit den folgenden Befehlen erreichen:

<ESC>RD:DEMO<ESC><ESC>	(Demoprogramm wird geladen)
<CTRL>-Y	(Programm assemblieren)
<ESC>	(zurück in den Editor)
<ESC>U<ESC><ESC>	(Programm starten)
<START>	(Programm beenden)

Anmerkung: <ESC> Escape-Taste drücken.
<CTRL>-Y Control-Taste gedrückt halten und die 'Y'-Taste betätigen.

INHALTSVERZEICHNIS

Einleitung	1
Teil 1 Der ATMAS-Editor	3
1.1 Benutzung des Editors	3
1.2 Texteingabe	5
1.3 Editierbefehle im Textmodus	5
1.4 Befehle der Kommandozeile	10
Teil 2 Der Makro-Assembler	15
2.1 Benutzung des Makro-Assemblers	15
2.2 Eingabeformat	16
2.3 Assemblerdirektiven	20
2.3.1 ORG (Origin)	20
2.3.2 EQU, EPZ (Equates)	21
2.3.3 DFB (Define Byte)	21
2.3.4 DFW (Define Word)	22
2.3.5 ASC (ASCII-String)	22
2.3.6 OUT (Output Listing)	23
2.4 Makrofähigkeit	25
2.4.1 Makrodefinition	25
2.4.2 Makroaufruf, Makroexpansion	26
2.4.3 Lokale Labels	27
2.4.4 Verschachtelte Makros	27
2.4.5 Makros contra Unterprogramme	28

Teil 3	Der Maschinensprache-Monitor	29
3.1	Allgemeine Benutzungshinweise	29
3.2	Befehlsvorrat	30
3.2.1	M - Memory-Dump	30
3.2.2	D - Disassemble	30
3.2.3	C - Change Memory	31
3.2.4	F - Fill Memory	31
3.2.5	B - Blocktransfer	32
3.2.6	G - Goto Address	32
3.2.7	S - Binary Save	33
3.2.8	L - Binary Load	35
3.2.9	E - Editor	35
3.2.10	I - Disketten-Inhaltsverzeichnis ..	35
Teil 4	Beispiele	36
4.1	Demoprogramm Farbscrolling	36
4.2	Makrobibliotheken	38
ANHANG A:	ATMAS-II Memory-Map	
ANHANG B:	Fehlermeldungen	

EINLEITUNG

ATMAS-II ist ein leistungsfähiges Entwicklungssystem für Maschinenprogramme auf Ihrem Atari-Computer. ATMAS-II besteht aus drei integrierten Teilsystemen: Editor, Makro-Assembler und Maschinensprache-Monitor. Diese Komponenten zusammen erlauben eine komfortable und schnelle Programmentwicklung auf Maschinenebene.

Alle drei Teile dieser integrierten Programmierumgebung sind aufeinander abgestimmt und jeweils nur einen Tastendruck voneinander entfernt. Ein komfortabler, bildschirmorientierter Editor, der über Funktionen wie Kopier-Register, wiederholbare Befehlssequenzen und platzsparende echte Tabulatoren verfügt, erleichtert die Eingabe auch umfangreicher Programme. Der Makro-Assembler arbeitet zur Erreichung einer kürzest-möglichen Assemblierzeit mit modernen Hashing-Algorithmen, so daß auch längere Programme in wenigen Sekunden assembliert werden. Schließlich steht Ihnen als Werkzeug zum Testen von Maschinenprogrammen ein Monitor zur Verfügung.

Obgleich sich jemand, der noch nie in Assembler programmiert hat, mit ATMAS-II schnell vertraut fühlen wird, bietet er dem fortgeschrittenen Benutzer viele Möglichkeiten: Beginnend bei den leistungsfähigen Editorfunktionen bis hin zu der Programmierung von Makrobefehlen.

Es sollte noch angemerkt werden, daß dieses Bedienungshandbuch kein Lehrbuch der 6502-Maschinensprache ist und das auch nicht sein kann. Hier wird auf weiterführende Literatur verwiesen, insbesondere auf ein im Herbst '85 erscheinendes Buch vom Autor dieser Bedienungshandbuches, das speziell auf den Atari-Computer zugeschnitten ist. Zusätzlich bieten auch die Beispiele im Teil 4 dieses Handbuches einen kleinen Einblick in die Assemblerprogrammierung des Atari-Computers.



Teil 1: DER EDITOR

Beim ATMAS-II Editor handelt es sich um einen bildschirmorientierten Editor, der ohne jegliche Zeilennummerierung auskommt, sich im wesentlichen ähnlich einem Textverarbeitungssystem verhält. Dadurch entfällt die umständliche Eingabe von LIST-Befehlen wie Sie diese von verschiedenen anderen Editoren kennen (z.B. in BASIC). Sie können sich einfach unter Zuhilfenahme der Cursorstasten durch das Listing bewegen, das dann von oben nach unten bzw. von unten nach oben durch das Textfenster scrollt.

1.1 Benutzung des Editors

Sobald ATMAS-II von der Diskette geladen worden ist, gelangen Sie direkt in den Editor, der gleichsam die Steuerzentrale des ATMAS-II Systemes darstellt. Von hier aus können Sie den Makroassembler sowie den Maschinensprache-Monitor aufrufen, außerdem bringt Sie die <RESET>-Taste stets in den Editor zurück.

Nach dem Booten werden Sie die folgende Bildschirm-Maske des Editors sehen:

```

+-----+
I P:00000 T:17408 C:1024 OK I <---Statuszeile
+-----+
Cursor ->I C I
I I
I Textfenster I
I I
I I
+-----+
I ..... I <---Kommandozeile
+-----+
```

Die Statuszeile

In der Statuszeile werden Ihnen ständig Informationen über Cursorposition, freien Textspeicherplatz, Kopierregister und Editorzustand mitgeteilt.

Aus 'P:nnnnn' können Sie die Entfernung (in Zeichen) des Cursors vom Textanfang entnehmen, 'T:xxxxx' gibt Ihnen Auskunft über den freien Textspeicherplatz, (d.h. Sie können noch maximal 'xxxxx' Anschläge eingeben). 'C:yyyy' informiert Sie über den freien Speicherplatz des Kopier-Registers (des sog. C-Registers) der im noch unbenutzten Zustand 1024 Zeichen umfaßt. Die letzten beiden Zeichen der Statuszeile geben Ihnen eine Zustandsmeldung des Editors wieder, wenn alles in Ordnung ist, werden Sie dort 'OK' finden. Eventuelle Fehler- bzw. Zustandsmeldungen (z.B. wenn das Kopierregister geöffnet ist) werden dort ausgegeben, eine Liste der Meldungen finden Sie im Anhang B.

Textfenster

Die nächsten 21 Zeilen bilden das Textfenster, hier sehen Sie immer den aktuellen Text, den Sie mit den im folgenden besprochenen Editierfunktionen bearbeiten können. In wahrsten Sinne des Wortes ist dieser Bildschirmteil als Fenster in den Text zu verstehen, das u.a. mit den Cursortasten nach oben und unten über den Text verschoben werden kann. Alle Editierfunktionen werden sofort im Textfenster sichtbar.

Kommandozeile

Die letzte Zeile des Bildschirms ist für die Eingabe von Kommandos reserviert. Neben einer Reihe von Direkt-Befehlen, die unmittelbar im Textfenster gegeben werden, gibt es zusätzlich noch einen Kommando-Modus des Editors, der durch die <ESC>-Taste aufgerufen wird. Hier werden in der Regel Befehle eingegeben, die zusätzliche Information benötigen, z.B. ein Wort, nach dem zu suchen ist. Auch Befehle zur Ein-/Ausgabe von Texten werden hier eingegeben.

1.2. Texteingabe

Sie können ohne Vergabe von Zeilennummern gleich mit der Eingabe des Textes beginnen. Jedes eingegebene Zeichen wird an der momentanen Cursorposition abgelegt und der Cursor bewegt sich eine Position nach rechts. Sie können jederzeit eine neue Zeile mit <RETURN> beginnen. Wenn Sie mehr als 38 Zeichen in eine Zeile eingeben, so verschwindet der Cursor von der rechten Bildschirmseite, sobald Sie <RETURN> drücken taucht dieser am Anfang der nächsten Zeile wieder auf. Dieses Verhalten erscheint auf den ersten Blick ungewöhnlich, Sie werden jedoch bald feststellen, daß es bei Assemblerprogrammen durchaus von Vorteil ist. Der wesentliche Teil eines Assemblerprogrammes befindet sich immer in den ersten 30 Zeichen, da dort der Assemblerbefehl zu finden ist. Wenn Sie längere Zeilen editieren wollen, (z.B. das Kommentarfeld), so haben Sie dazu eine Spezialfunktion zur Verfügung (<CTRL>-V, siehe 1.2 / Spezialfunktionen), die maximal 76 Zeichen in zwei Bildschirmzeilen darstellt. Sie werden allerdings feststellen, daß sich der normale 38-Zeichen Bildschirm wesentlich besser zum Editieren von Assemblerprogrammen eignet, da alle Assemblerbefehle im Gegensatz zum Zwei-Zeilenmodus ununterbrochen untereinander stehen.

Wie üblich haben alle Tasten automatische Wiederholungsfunktion, Kleinbuchstaben können mit der <CAPS>-Taste erreicht werden.

1.3 Editierbefehle im Textmodus

Wie bereits angesprochen, bietet Ihnen der ATMAS-II Editor zwei verschiedene Ebenen der Befehlseingabe. Die einfachere der beiden Ebenen bilden die sog. Direkt-Befehle, oder auch Befehle im Textmodus genannt. Diese Kommandos werden entweder durch Spezialtasten, z.B. <BACK SPACE>, oder durch Control-Funktionen ausgeführt. Letzteres bedeutet einfach, daß Sie die <CTRL>-Taste (Bei 600/800XL: <CONTROL>) gedrückt halten, während Sie eine andere Taste betätigen, ganz ähnlich dem, wie Sie in ATARI-BASIC den Cursor führen. Wenn Sie also im folgenden <CTRL>-X lesen, so bedeutet das, daß Sie <CTRL> niedergedrückt halten, und dabei die 'X'-Taste drücken.

Cursor-Steuerung

Im Textmodus haben Sie die Cursor-Funktionen zur Verfügung:

Um den Cursor nach rechts zu verschieben drücken Sie

<CTRL>-<Pfeil rechts>

Eine Cursorbewegung nach links erfolgt mit

<CTRL>-<Pfeil links>

Wenn Sie dabei über das rechte Ende einer Zeile gelangen, so wird der Cursor an den Anfang der nächsten Zeile gesetzt. Fahren Sie über das linke Ende hinaus, so befindet sich der Cursor am Ende der vorherigen Zeile.

Um den Cursor nach oben zu bewegen tippen Sie

<CTRL>-<Pfeil oben>

Der Cursor wird dabei, falls er sich nicht schon am linken Rand befindet, zuerst dorthin zurückgesetzt. In diesem Falle führt erst der zweite Tastendruck nach oben.

Der Cursor kann mit

<CTRL>-<Pfeil unten>

um eine Zeile nach unten an den Anfang der nächsten Zeile gesetzt werden. Bei allen Cursorbewegungen können Sie nicht über Anfang und Ende des Textes hinaus. Im Unterschied zu anderen Editoren werden Sie feststellen, daß der Cursor nicht über den Text läuft, sondern eingefügt wird. Das hat den Vorteil, daß Sie immer genau wissen, wo momentan eingefügt werden kann.

Zeichen löschen

Um ein Zeichen links vom Cursor zu löschen drücken Sie

<BACK-SPACE>

Wenn sich der Cursor dabei am Zeilenanfang befindet, dann wird das Return-Zeichen gelöscht, d.h. die Zeile, in der sich der Cursor befand, wird an die vorherige angehängt. Falls diese eine Leerzeile war und somit nur aus einem Return-Zeichen bestand, wird die Leerzeile gelöscht.

Um ein Zeichen rechts vom Cursor zu löschen tippen Sie

<CTRL>-<BACK-SPACE>

Wie im obigen Fall kann damit auch die nächste Zeile unter dem Cursor an die momentane angehängt werden. Ist diese eine Leerzeile, so wird sie gelöscht.

Zeilen löschen

Wenn Sie eine Zeile löschen möchten, dann drücken Sie

<CTRL>-X

Die Zeile wird von der momentanen Cursorposition bis zum Zeilenanfang gelöscht. Befindet sich der Cursor schon am Anfang einer Zeile, so wird die ganze vorhergehende Zeile gelöscht.

Cursor-Sprünge

<CTRL>-E bringt Sie an den Anfang des Textes

<CTRL>-D bringt Sie ans Ende des Textes

Tabulator

Der ATMAS-Editor verfügt über einen echten Tabulator, der je nach Erfordernis 1 bis 9 Leerzeichen belegt, aber nur als ein Zeichen abgespeichert wird. Um an die nächste vortabulierte Stelle zu gelangen tippen Sie einfach

<TAB>

Wenn Sie später über einen solchen Tabulator mit dem Cursor fahren, werden Sie feststellen, daß der Cursor darüber 'springt', und auch Einfügungen vor einen Tabulator den nachfolgenden Text nicht verschieben. Der Tabulator wird nur als ein Zeichen abgespeichert,

so daß Sie damit gut lesbare Listings mit geringen Speicherplatzbedarf anfertigen können.

Kopier-Register (C-Register)

Das C-Register ist ein zweiter, 1024 Zeichen großer Textbuffer, den Sie zum Transferieren und Kopieren von Textteilen benutzen können. Positionieren Sie zuerst den Cursor auf das untere Ende des zu bewegenden Textteiles, dann öffnen Sie das C-Register mit

<CTRL>-R

Im linken Teil der Statuszeile wird dieser Zustand mit der Meldung 'CR' festgehalten. Wenn Sie jetzt den Cursor rückwärts bewegen, so werden die überlaufenen Textteile in das C-Register geschrieben. Sie können diesen Vorgang in der Statuszeile mit verfolgen, die Zahl der freien Zeichen im Kopier-Register ('C:1024') nimmt mit der Zahl der Cursorbewegungen ab.

Befindet sich der gesamte gewünschte Text im C-Register, so schließen Sie dieses mit dem Befehl

<CTRL>-F

Jetzt können Sie den Inhalt des Kopier-Registers an jeder beliebigen Cursorposition einsetzen, indem Sie

<CTRL>-J

eingeben. Der Inhalt des C-Registers wird dabei nicht zerstört, d.h. es sind auch komfortable Mehrfachkopien möglich. Löschen des Kopier-Registers ist mit dem Befehl

<CTRL>-K

möglich. Das C-Register wird selbsttätig durch Aufruf des Assemblers sowie nach einem Ein-/Ausgabebefehl geschlossen.

Noch ein paar Anmerkungen zum C-Register: Sie können damit Textteile zum Kopieren vorbereiten, wenn Sie diese rückwärts mit <CTRL>-<Pfeil oben> oder <CTRL>-<Pfeil links> durchlaufen. Genausogut können Sie Textblöcke transferieren, indem Sie den Cursor durch die Löschbefehle <BACK-SPACE> und <CTRL>-X rückwärts bewegen. Der gelöschte Text verbleibt im Kopier-Register und kann an beliebigen anderen Stellen wieder eingesetzt werden.

Sie können auch einen Textblock aus verschiedenen Textteilen zusammensetzen, indem das C-Register an verschiedenen Textstellen mehrfach geöffnet und geschlossen wird, ohne dazwischen seinen Inhalt mit <CTRL>-K zu löschen.

Ist das C-Register voll ('C:0000'), erscheint zusätzlich links oben die Fehlermeldung 'C?'. .

Spezialbefehle des Editors

Den Zwei-Zeilenmodus können Sie mit

<CTRL>-V

einschalten. Dadurch werden Zeilen, die länger als 38 Zeichen sind, zweizeilig am Bildschirm dargestellt. Die maximale Zeilenlänge dieses Modus beträgt somit 76 Zeichen, wobei allerdings nur 11 Zeilen dargestellt werden können. Ein weiteres <CTRL>-V bringt Sie in den normalen Ein-Zeilenmodus zurück.

Control-Zeichen, insbesondere Tabulatoren und Return-Zeichen können mit

<CTRL>-T

sichtbar gemacht werden. Alle Control-Zeichen werden 'revers' dargestellt. Ein weiteres <CTRL>-T schaltet in den Normalmodus zurück.

<CTRL>-G

dient zur Wiederholung der in der Kommandozeile stehenden Befehlskette. Lesen Sie dazu bitte Abschnitt 1.4.

Aufruf des Assemblers und des Monitors

Der ATMAS-II Makroassembler wird durch

<CTRL>-Y

aufgerufen. ATMAS-II beginnt nun sofort mit der Assemblierung des im Textbuffer befindlichen Quelltextes (siehe Abschnitt 2), nach Abschluß der Assemblierung bringt Sie ein beliebiger Tastendruck in den Editor zurück.

Der Maschinensprache-Monitor kann durch Eingabe von

<CTRL>-P

aufgerufen werden. Er meldet sich mit 'MONITOR.' und erwartet Ihre Eingabe (s. Abschnitt 3). Zurück in den Editor gelangen Sie durch die Eingabe von 'E'.

1.4 Befehle der Kommandozeile

Außer den Direktbefehlen im Textmodus verfügt der ATMAS-II Editor über einen weiteren leistungsfähigen Editiermodus: Die Kommandozeile. Sie werden hier in erster Linie Befehle finden, die außer dem eigentlichen Aufruf des Befehles noch weitere Angaben benötigen, etwa eine Zeichenkette nach der gesucht werden soll. Weiterhin werden Sie einige Befehle finden, die Sie schon vom Textmodus her kennen, diese können aber im Kommandomodus mit Wiederholungsfaktoren versehen und verkettet werden, eine Art von 'Befehls-Makros'.

Benutzung der Kommandozeile

Sie betreten den Kommandomodus durch die <ESC>-Taste, die als Dollar-Zeichen ('\$') dargestellt wird. Jetzt können Sie einzelne Kommandos (siehe unten) eingeben und dabei die <ESC>-Taste als Trennzeichen benutzen. Die Kommandozeile wird ausgeführt, sobald Sie <ESC> zweimal hintereinander betätigen. Die Ausführung wird mit einem Doppelkreuz hinter der Kommandokette quittiert, Sie befinden sich anschließend wieder im Textmodus. Sollte sich die Kommandozeile als fehlerhaft erwiesen haben, so wird Ihnen das in der linken oberen Ecke durch ein Fehlerkürzel angezeigt.

Alle Befehle (mit Ausnahme der I/O-Kommandos R und W) können mit vorangestellten Wiederholungsfaktoren 2 bis 255 mal ausgeführt werden. Zusätzlich kann eine vollständige Wiederholung einer eingegebenen Kommandozeile vom Textmodus aus mit <CTRL>-G erfolgen.

Eingabefehler in der Kommandozeile selbst können mit <BACK SPACE> korrigiert werden. <CTRL>-X führt zum Löschen der Kommandozeile und zur Rückkehr in den Textmodus.

Editierfunktionen im Kommandomodus

- B : Cursor eine Position zurück
- F : Cursor eine Position vorwärts
- D : Ein Zeichen links vom Cursor löschen
- T : Vom Cursor bis Anfang d. nächsten Zeile löschen

Mit diesen Befehlen ist z.B. eine schnelle Cursor-Bewegung im Text möglich. Öffnen Sie die Kommandozeile mit <ESC> und geben Sie '200F' gefolgt von zweimal <ESC>. Jede Wiederholung der Kommandozeile mit <CTRL>-G bringt Sie nun 200 Zeichen im Textbuffer weiter.

Weitere Editierbefehle:

- H<Hex-Byte>** : Einsetzen eines beliebigen ATASCII-Codes in den Text, z.B. zur Druckersteuerung.
BEISPIEL: \$HOF\$ schaltet EPSON-Drucker auf Engschrift
- S<String>** : Sucht nach der Zeichenkette <String> ab der momentanen Cursorposition bis zum Ende des Textes. Falls nicht vorhanden, erscheint 'S?' in der Statuszeile.
BEISPIEL: \$SLABEL\$ sucht nach Wort LABEL.
- I<String>** : Setzt Zeichenkette <String> an momentane Cursorposition ein. Kann im Zusammenhang mit dem S-Befehl zu einer Such- und Austausch-Funktion ausgebaut werden (s.u.).

Spezialfunktionen:

- J** : Wiederholt die gesamte Kommandozeile. Damit können Sie z.B. eine Such- und Austausch-Funktion bis Textende wiederholen.
- K** : Löscht den gesamten Textbuffer.
VORSICHT: Der eingegebene Text wird gelöscht!
- U** : USER-Befehl. Ruft ein Maschinenprogramm, das ab \$A800 im Speicher stehen muß, per JSR-Befehl auf. VORSICHT! Nur verwenden wenn Sie dort auch ein Programm stehen haben.
- @<n>** : Einstellung der Tabulatorweite auf <n> Zeichen (n von 0 bis 9). Ohne Angabe wird eine Weite von 8 verwendet.
BEISPIEL: @\$6\$:Tabulatorweite 6 Zeichen.
- M** : Rückkehr zum DOS. Mit 'B'- '* ATMAS-II *' kommen Sie wieder in ATMAS-II zurück.

Noch zwei Befehle, die das C-Register betreffen:

- E : Löscht C-Register (wie <CTRL>-K)
- G : C-Register in Text einfügen (<CTRL>-J)

Befehle zur Ein-/Ausgabe von Text

Wenn Sie einen im Textbuffer befindlichen Text auf Diskette abspeichern wollen, so können Sie das mit dem 'W'-Kommando erledigen. Sie brauchen lediglich den gewünschten Namen des Files mit einer Gerätebezeichnung angeben.

BEISPIEL: Sie wollen den momentanen Text unter dem Namen TEST.SRC auf Laufwerk 1 speichern. Dazu geben Sie folgendes ein:

<ESC>WD1:TEST<ESC><ESC>

Die Extension .SRC wird von ATMAS selbst hinzugefügt, Sie können selbstverständlich auch eine andere Extension angeben, etwa \$WD1:TEST.ATM\$ <ESC>.

VORSICHT: Der Text wird immer BEGINNEND bei der
===== momentanen POSITION DES CURSORS aufgezeichnet. Deshalb immer zum Abspeichern des gesamten Textes zuvor <CTRL>-E eingeben.

Auf ähnlich Weise können Sie ein Textfile von der Diskette in den Speicher laden. Dazu wird der Befehl 'R' benutzt, der wie 'W' verwendet wird.

BEISPIEL: Laden des Demo-Files DEMO.SRC von der ATMAS-II Diskette (in Laufwerk 1):

<ESC>RD1:DEMO<ESC><ESC>

Der Extender .SRC wird wiederum automatisch angehängt. Das File wird nun beginnend an der momentanen Cursorposition eingelesen. Daraus ergeben sich zwei Möglichkeiten:

- 1.) Wenn ein neues File geladen werden soll, dann löschen Sie zuvor den Textbuffer vorher mit <ESC> K <ESC> <ESC>.

2.) Sie können aber auch einzelne Files in den Text-buffer einfügen (ähnlich dem Kopier-Register), indem Sie den Cursor an die gewünschte Stelle bringen und dann den Lade-Befehl geben.

Treten während eines Ein-/Ausgabe-Befehles Fehler auf, so wird dies in der Statuszeile durch die Meldung 'RW' angezeigt.

Listings

Ausdrucke des Quelltextes können mit dem 'L' Befehl hergestellt werden. Dazu gibt es mehrere Optionen:

- 'L' : Listing scrollt über den Bildschirm
- 'LO' : Ausgabe auf RS232 Schnittstelle Nr. 1 des 850 Interfacemodules (R1:)
- 'L1' : Ausgabe erfolgt über den normalen Printer-Handler 'P:', also im Regelfall über das Atari 850 Interface-Modul.
- 'L2' : Listing wird über Centronics-Schnittstelle aus den Joystick-Ports 3 und 4 ausgegeben. Diese Möglichkeit besteht nur für 400/800-Computer.

Hinweise zur fortgeschrittenen Editorbenutzung

Die Möglichkeit der Befehlsverkettung in der Kommandozeile ermöglicht sehr komfortable Editierhilfen:

`⌘SLDA⌘3D⌘ILDX⌘J⌘ <ESC>`

Diese Befehlskette würde alle LDA-Befehle ab der Cursorposition in LDX-Befehle umtauschen. Wie funktioniert's? Zuerst wird ein LDA-Befehl gesucht (`⌘SLDA`), der Cursor kann man sich dann hinter dem gesuchten String postiert denken. '`⌘3D`' löscht nun drei Zeichen links vom Cursor (Sie erinnern sich: Der Wiederholungsfaktor!), während '`⌘ILDX`' den String 'LDX' einsetzt. '`⌘J`' wiederum bewirkt, daß die gesamte Kommandozeile solange ausgeführt wird, bis der Text zu Ende ist, also ein 'S?'-Fehler (String nicht gefunden) auftritt.

Teil 2: Der Makro-Assembler

2.1 Benutzung des Makroassemblers

Vom Editor aus wird der ATMAS-II Makroassembler mit

<CTRL>-Y

aufgerufen. Er beginnt dann sofort mit der Assemblierung des Quelltextes, der sich im Textbuffer des Editors befinden muß.

Die Assemblierung wird in drei Durchgänge (sog. passes) unterteilt, wobei der zweite Durchgang seine Aktivität durch ein schnell veränderndes Zeichen in der linken oberen Bildschirmecke anzeigt. Der dritte Pass kann ein Listing des assemblierten Programmes auf dem Drucker oder dem Bildschirm ausgeben (s. OUT-Direktive).

Der Vorgang des Assemblierens wird so lange fortgesetzt, bis entweder der Quelltext zu Ende ist, ein <CTRL>-Z Zeichen entdeckt wird (Assembler-Stop Zeichen) oder ein Fehler erkannt worden ist. Im letzteren Fall wird eine Fehlermeldung (s. Anhang B) am Bildschirm ausgegeben. Ein beliebiger Tastendruck bringt Sie in den Editor zurück, und zwar genau an die Stelle, die den Fehler verursacht hat. Sie können sofort den Fehler verbessern und den Assembler mit <CTRL>-Y neu starten.

Dieses komfortable 'Hand-in-Hand' Arbeiten zwischen Editor und Assembler ist ein wesentlicher Punkt, der ATMAS-II als Entwicklungssystem für Maschinenprogramme so leistungsfähig macht, und doch dabei die Bedienung einfach gestaltet.

2.2 Eingabeformat des Assemblers

Der ATMAS-II Makroassembler kennt alle Opcodes der 6502-CPU, soweit diese im 'MOS-Technology 6502-Programmierhandbuch' beschrieben sind. Ebenso folgt die Schreibweise der Adressierungsarten der in jenem Buch vorgeschlagenen Notation (siehe unten).

Eine Quelltextzeile kann folgende Formate haben:

A) Leerzeile:

Besteht nur aus einem <RETURN>-Zeichen.

B) Kommentarzeile:

Beginnt mit einem '*' in der ersten Spalte.

Beispiel: * Copyright (c) M. Huber

C) Befehlszeile mit Label:

Beginnt mit einem Label, gefolgt von einem Trennzeichen (Leerzeichen oder Tabulator), dann einem 6502-Opcode oder einer Assemblerdirektive, schließlich kann noch nach einem weiteren Trennzeichen ein Kommentar folgen.

Beispiele: LOOP LDA #\$10
 DOSVEC EQU \$000C DOS-EINSPRUNG

D) Befehlszeile:

Muß immer mit einem Trennzeichen beginnen: Um saubere Listings zu bekommen empfiehlt es sich, diese mit einem Tabulator beginnen zu lassen. Danach folgt ein 6502-Opcode oder eine Assemblerdirektive, wie im voranstehenden Fall kann auch hier nach einem weiteren Trennzeichen ein Kommentar stehen.

Beispiele: <TAB>STA COLORO FARBE AENDERN
 <TAB>DFB 100,120,140

Adressierungsarten:

Im folgenden eine kurze Zusammenfassung der Schreibweisen der einzelnen Adressierungsarten, jeweils mit einem Beispiel versehen:

Implied Akku	<OPC>	ASL
Immediate	<OPC> #<AUSDR>	LDA #&FF
Absolut, Zeropage	<OPC> <AUSDR>	STA &600
Relativ	<OPC> <AUSDR>	BNE **4
Absolut X-indiziert	<OPC> <AUSDR>,X	CMP &3000,X
Absolut Y-indiziert	<OPC> <AUSDR>,Y	LDA TABLE,Y
Indirekt-indiziert	<OPC> (<AUSDR>),Y	EDR (&F0),Y
Indiziert-Indirekt	<OPC> (<AUSDR>),X	STA (&F0,X)

<OPC> : gültiger 6502-Assemblerbefehl (Op-Code)

<AUSDR> : arithmetischer Ausdruck

Wie schon erwähnt, ist vor jedem <OPC> ein Label möglich, nach jedem Assemblerbefehl kann ein Kommentar stehen, der mit einem Trennzeichen abgesetzt ist. AUSNAHME: Nach Implied-Akku Befehlen muß das Kommentarfeld durch einen Strichpunkt abgetrennt sein.

LABELS

Labels bestehen aus Buchstaben und Zahlen, wobei das erste Zeichen immer ein Buchstabe sein muß. Die maximale Länge beträgt 8 Zeichen, wobei alle Zeichen signifikant sind.

Gültige Labels:

SCHLEIF
SCHLEIF1
S12345

Ungültige Labels:

1LOOP
SCHLEIFEN (9 Buchstaben!)

Konstante:

ATMAS-II kennt vier Arten von Konstanten: Dezimal-, Hexadezimal-, Binär- und Zeichen-Konstante.

A) Dezimale Konstante

Bestehen einfach aus der Zahl selbst (im Bereich 0 bis 65535).

Beispiele: 10, 100, 3000, 65535

B) Hexadezimale Konstante

Bestehen aus einem Dollar-Zeichen ('\$') und gültigen Hexadezimalziffern (0-9,A-F).

Beispiele: \$10, \$FF, \$A800, \$FFFF

C) Binäre Konstante

Bestehen aus einem Prozent-Symbol mit nachfolgenden gültigen Binärziffern (0,1).

Beispiele: %1, %1001, %11110000

D) Zeichen-Konstante

Werden aus einem Apostroph mit nachfolgenden ASCII (bzw. ATASCII)-Zeichen gebildet. Die Zeichen werden in 7-Bit-Werte umgewandelt (Bit 7 immer Null).

Beispiele: 'X', '#', 'e', ''

Negative Konstante:

Alle Konstanten können durch ein vorangestelltes Minus-Zeichen als negative Zahlen interpretiert werden. Die Darstellung erfolgt im Zweierkomplement (-1 := \$FFFF).

Beispiele: -1, -\$100, -%101101, -'B

Interner Adresszähler:

ATMAS-II führt während der Assemblierung einen internen Adresszähler mit, der jeweils auf den Opcode der gerade übersetzten Anweisung zeigt (bzw. auf das nächste freie Byte nach dem zuletzt berechneten Ausdruck in DFB/DFW Direktiven). Dieser interne Adresszähler kann jederzeit durch das '*'-Symbol abgerufen werden. Dies kann in vielfältigen Weisen benutzt werden:

- A) Speicherplatz reservieren:
 ORG **+\$100 reserviert 256 Bytes
- B) relative Sprünge ohne Label:
 BNE **4 überspringt die nächsten 4 Bytes
- C) Adressversetzte Labels für Software-Stacking
 RETADR EQU *-1 momentane Adresse minus 1

Ausdruck:

Ein Ausdruck kann eine Konstante, ein Label, ein Makroparameter (siehe 2.4), der momentane Adresszähler (*) oder eine arithmetische Verknüpfung derselben sein. Zugelassene Verknüpfungen sind: +, -, *, /, der Rechenbereich beträgt 0-65535. Einige Beispiele dazu:

LDA # \$FF	Konstante
LDA # LABEL1	Label
STA LABEL+\$A0	Verknüpfung
LDA **+\$10	momentaner Adr.-Zähler+Offset

Zusätzlich sind auch Klammern zur Klärung der Priorität möglich:

LDA # (\$A0+5)/10
LDA # LAB1-(Lab1/256)*256

2.3 Assembler-Direktiven

=====

2.3.1 ORG

Zweck : bestimmt den Inhalt des assemblerinternen Adress-Zählers.

Syntax : [(LABEL)] ORG <AL>[,<AP>] [(KOM)]

Beispiel: ORG \$A800

ORG bestimmt die Anfangsadresse des Objektprogrammes, im obigen Beispiel ist der für diese Zwecke reservierte Speicherbereich ab \$A800 gewählt.

In manchen Fällen ist es aber nicht möglich, das Objektprogramm direkt in den gewünschten Speicherbereich zu legen, wenn dieser z.B. von ATMAS-II belegt wird (s. Memory-Map im Anhang!). Hierzu können Sie den zweiten Parameter der ORG-Direktive verwenden, die sogenannte 'physikalische Adresse' <AP>.

ORG \$3000,\$A800

Dieser ORG-Befehl würde ein Programm so assemblieren, daß es an der Adresse \$3000 (logische Adresse, <AL>) lauffähig ist, während der Assemblierung aber im freien Speicherbereich ab der Adresse \$A800 abgelegt wird. Damit das Programm ausgeführt werden kann, muß es an die richtige Stelle verschoben werden, das kann z.B. mit dem SAVE-Befehl des Monitors geschehen, der ein Binärfile so abspeichern kann, daß es beim erneuten Einlesen an den richtigen Speicherplatz kommt. In einfacheren Fällen (z.B. wenn das Programm in dem Adressbereich abgelegt werden soll, in dem ATMAS-II seine Symboltabelle hat) hilft auch der Blocktransfer-Befehl des Monitors.

2.3.2 EQU, EPZ (Equates)

Zweck : Zuordnung eines Wertes zu einem Label.

Syntax : <LABEL> EQU <AUSDR> [<KOM>]
<LABEL> EQU <AUSDR> [<KOM>]

Beispiel:

GRUEN	EQU	\$A0
CIOV	EQU	#\$456
SAVMS	EPZ	#\$58
PLAYER	EQU	PMBASE+1024

Die EQU-Direktive wird benutzt, um einen Label mit einem bestimmten Wert zu definieren, es kann sich dabei sowohl um ein Datum oder eine Adresse handeln. Der erste Label des Beispiels wird z.B. sicher als Datum für einen Immediate-Befehl verwendet werden (LDA #GRUEN), während das zweite Beispiel eine ROM-Einsprungsadresse bezeichnet (JSR CIOV).

EPZ (Equate Page Zero) hat prinzipiell die gleiche Funktion, kann aber verwendet werden, wenn zum Ausdruck gebracht werden soll, daß der Label einen Zeropage-Speicherplatz darstellt. Es besteht aber kein Zwang zur Verwendung von EPZ, da ATMAS-II Zeropage-Adressierungsarten selbsttätig erkennt. EPZ dient hierbei nur der Verdeutlichung, Sie können in allen Fällen auch EQU verwenden.

2.3.3 DFB (Define Byte)

Zweck : definiert den Inhalt einzelner Bytes im Objectcode

Syntax : [<LABEL>] DFB <AUSDR>[,<AUSDR>][<KOM>]

Beispiel:

TABELL	DFB	1,2,4,8,16,32,64,128
	DFB	'A', 'B', 'C
	DFB	'X+128

Die nach DFB folgenden Ausdrücke werden als 8-Bit Werte in den Objektcode abgelegt. Die erste Zeile des Beispielles würde folgende Sequenz erzeugen:

01 02 04 08 10 20 40 80

ATASCII-Werte werden als 7-Bit Werte abgelegt, das höchstwertige Bit ist immer Null.

2.3.4 DFW (Define Word)

Zweck : definiert den Inhalt eines 16-Bit Wortes im Objektcode.

Syntax : [**<LABEL>**] DFW **<AUSDR>**[,**<AUSDR>**...][**<KOM>**]

Beispiel: ADRTAB DFW \$A900,\$AA03,\$AB06
DFW ADRTAB-1,ADRTAB+2

Die nach DFW foldenden Ausdrücke werden als 16-Bit Werte im Objektcode abgelegt. Dabei wird die Reihenfolge der 6502-Adressen benutzt, d.h. zuerst das niederwertige (LSB), dann das höherwertige Byte (MSB). Die erste Zeile des Beispiels würde demnach folgendes erzeugen:

00 A9 03 AA 06 AB

2.3.5 ASC (ASCII-String)

Zweck : fügt einen String in ATASCII- oder Bildschirmcodierung in den Objektcode ein.

Syntax : [**<LABEL>**] ASC **<STZ>****<STRING>****<STZ>**
[,**<STZ>****<STRING>****<STZ>**][**<KOM>**]

<STZ>: String-Trennzeichen, ", /, \, %, \$

Beispiel: TEXT ASC "HALLO"
MELD ASC /DISK-FEHLER/
ASC "MAKRO",\ASSEMBLER\
ASC % Normaler Bildschirmcode %
ASC \$ Inverser Bildschirmcode \$

Strings nach einer ASC-Direktive werden Byte für Byte in den Objektcode übertragen. Abhängig vom Trennzeichen (<STZ>) können verschiedene Funktionen angewählt werden:

- " : Text im ATASCII-Code eintragen
- / : ATASCII-Code, aber Bit 7=1 (Invers)
- \ : ATASCII-Code, beim letzten Zeichen wird Bit 7 gesetzt (Ende-Kennung)
- % : Interner Bildschirmcode wird generiert!
- \$: Bildschirmcode mit Bit 7=1 (Invers)

Jeder String muß von zwei gültigen String-Trennzeichen umschlossen werden. Anfangs- und Endtrennzeichen müssen gleich sein.

Statt dem Anführungszeichen " (<SHIFT>-2) können Sie auch ein beliebiges anderes nicht alphanumerisches Zeichen benutzen, die oben genannten natürlich ausgeschlossen. Denkbar wären hier z. B. !, #, oder &, Bedingung ist lediglich, daß der String mit dem gleichen Trennzeichen abgeschlossen wird. Mit diesem Trick ist es möglich, daß Sie ein Anführungszeichen in den Text bekommen.

In Verbindung mit dem Displaylistkonzept der Atari-Computer gestattet Ihnen der ASC-Befehl die komfortable Programmierung von Titeln und Überschriften, da direkter Bildschirmcode erzeugt werden kann. Das Beispielprogramm ASCDEMO.SRC auf der ATMAS-II Diskette zeigt Ihnen, wie's gemacht wird.

2.3.6 OUT (Output Listing)

Zweck : Ausgabe eines Assembler-Listings

Syntax : [**<LABEL>**] OUT [**L**][**N**][**M**][**P**][**O**][**1**][**2**][**<KOM>**]

Beispiel: OUT L Listing auf Bildschirm
 OUT LP1 Listing auf Drucker
 OUT LNMP1 Listing auf Drucker, Symbole
 keine Makroexpansion

Mit OUT können Sie bestimmen, ob Sie ein Protokoll des Assembliervorganges haben möchten und das zugehörige Ausgabegerät festlegen. Folgende Parameter werden erkannt:

L : Listing wird erzeugt
N : Symboltabelle wird ausgegeben
M : Makros werden NICHT expandiert (ausgedruckt!)
P : Druckeroptionen:
 P0: RS232 Schnittstelle
 P1: Atari-Drucker, Centronics-Schnittstelle (850)
 P2: Joystick-Interface Port 3&4 (nur 400/800)

Nach jeweils 66 Zeilen wird ein Seitenvorschub generiert.

2.4 Makrofähigkeit

ATMAS-II gestattet Ihnen die Verwendung von Makro-befehlen. Darunter versteht man eine Folge von Assemblerbefehlen, denen im Rahmen einer Makrodefinition ein Name zugeordnet wurde. Bei der Verwendung dieses Makro-Namens im Quelltext werden die gesamten, ihm zugeordneten Assemblerbefehle erzeugt, man sagt, der Makro-Befehl wird expandiert. Mit Hilfe von Makros können Sie sich eine Art von zusätzlichen Assemblerbefehlen schaffen, die in Wirklichkeit aus einer Sequenz von einzelnen Maschinenbefehlen bestehen.

2.4.1 Makro-Definition

Bevor Sie ein Makro verwenden können, müssen Sie es zuerst definieren. Zu diesem Zweck dienen die zwei Assembler-Direktiven `MACRO` und `MEND`.

Zum besseren Verständnis zuerst ein Beispiel, das den Basic-Befehl `POKE` imitiert:

```
POKE      MACRO      ADRESS,DATA
           LDA #DATA
           STA ADRESS
           MEND
```

Die Definition wird von der `MACRO`-Direktive mit voranstehendem Namen des Makros (hier: `POKE`) eingeleitet. Dieser Name wird später zum Aufruf verwendet. Nach der `MACRO`-Direktive folgen die sogenannten formalen Parameter, das sind keine Labels im eigentlichen Sinn, sondern nur Platzhalter für die beim Aufruf angegebenen tatsächlich einzusetzenden Parameter. Dieser Vorgang der Parametersubstitution wird bei der Besprechung des Makro-Aufrufes noch genauer erläutert.

Jetzt folgen die Assemblerbefehle, die als Operanten sowohl gewöhnliche Labels als auch die in der `MACRO`-Direktive angegebenen formalen Parameter benutzen können. Abgeschlossen wird die Makro-Definition durch die `MEND` (Makro-Ende)-Direktive

Syntax : <MNAME> MACRO [<FPAR>][,<FPAR>...]

MEND

<MNAME>: Name des Makros, Syntax wie Label
<FPAR> : formale Parameter

2.4.2 Makroaufruf, Makroexpansion

Das in 2.4.1 definierte Makro kann im Quelltext mit

```
POKE 752,1
```

aufgerufen werden. Bei der Assemblierung findet dann folgendes statt: ATMAS-II erkennt, daß es sich bei POKE um ein bereits definiertes Makro handelt und fügt die Maschinenbefehle der Definition in den Objektcode ein. Dabei werden die formalen Parameter der Definition mit den tatsächlichen Werten des Aufrufes ersetzt. Im Beispiel wird folglich ADRESS durch 752, und DATA durch 1 ersetzt. Nach der Makroexpansion ergibt sich folgendes Maschinenprogramm:

```
LDA #1  
STA 752
```

Die allgemeine Syntax für den Makroaufruf lautet:

```
[<LABEL>] <MNAME> [<PAR>][,<PAR>...]
```

Es müssen ebensoviele Parameter (<PAR>) übergeben werden, wie formale Parameter in der Definition angegeben wurden. Als Parameter können Konstante, Ausdrücke und auch Strings verwendet werden. Ein Beispiel zur Verwendung von Strings als Parameter finden Sie im Teil 4.2 (Makro-Bibliotheken).

2.4.3 Lokale Labels

Als nächstes Beispiel betrachten Sie bitte das Listing des unten angegebenen Makros. Es handelt sich dabei um ein Programm zum Löschen eines Speicherbereiches von max. 256 Bytes (einer Page). Da das Programm mit einer Schleife arbeitet, enthält es folgerichtig auch ein Label. Würden Sie dieses Makro zweimal innerhalb eines Programmes aufrufen, so wäre das Label LOOP doppelt verwendet, wodurch der Assembler den Fehler 'SAME LABEL TWICE' meldet.

```
LOESCH    MACRO ADDRESS,LAENGE
          LDY #LANGE
          LDX #0
          LDA #0
LOOP      STA ADDRESS,X
          INX
          DEY
          BNE LOOP
          MEND
```

Auch hier bietet Ihnen ATMAS-II Unterstützung an: Ein Label, das mit dem Happy-'a' Symbol (<SHIFT>-8) endet, wird als lokales Label des Makros angesehen. Im Beispiel: statt LOOP muß der Label LOOP@ verwendet werden (auch im BNE-Befehl!). Intern ersetzt ATMAS-II das Happy-'a' Symbol durch eine vierstellige Zahl, die bei jedem Makroaufruf um eins erhöht wird. Dadurch erzeugt auch zweimalige Aufruf von LOESCH verschiedene Labels (nämlich LOOP0001 und LOOP0002).

2.4.4 verschachtelte Makros

Eine Makrodefinition selbst kann wiederum einen Makroaufruf einschließlich der Übergabe von Parametern enthalten. Als Beispiel kann ein Double-Poke Befehl dienen, der das bereits besprochene POKE-Makro benutzt:

```
DPOKE      MACRO  ADDRESS,WORT
           POKE  ADDRESS,WORT
           POKE  ADDRESS+1,WORT/256
           MEND
```

Diese Verschachtelungstechnik ist nur durch den Hardwarestack begrenzt.

2.4.5 Makros contra Unterprogramme

Sicherlich ist Ihnen bei der Beschreibung der Makros die nahe Verwandtschaft zu Unterprogrammen aufgefallen. Es gibt jedoch eine Reihe von wichtigen Unterschieden, die Sie sich verdeutlichen sollten:

Makros sind universeller in der Benutzung, da Sie über den Mechanismus der Parameterübergabe verfügen. Sie eignen sich daher gut zum Aufbau von Makro- (Programm) Bibliotheken. Solche Makrosammlungen lassen sich leicht mit dem flexiblen ATMAS-II Editor in den Quelltext einbinden. Makros gestatten wesentlich übersichtlichere Assemblerprogramme, man darf allerdings nicht übersehen, daß alle in der Definition des Makros angegebenen Maschinenbefehle bei jedem Aufruf des Makros in das Programm eingesetzt werden. Das bedeutet, wenn Sie das LOESCH-Makro fünfmal im Programm verwenden, daß es ebenso oft in Ihr Programm eingesetzt wird, und natürlich auch dementsprechend Speicherplatz verbraucht.

Die Verwendung von Unterprogrammen ist wesentlich optimaler in Bezug auf Speicherplatz, wobei es allerdings schwieriger ist, die Übergabe der Parameter so elegant wie im Makro zu gestalten. Ein weiterer, in manchen Fällen entscheidender Gesichtspunkt ist die Geschwindigkeit: Während Makros hier die bessere Lösung darstellen, dauert es bei Unterprogrammen etwas länger, da JSR und RTS-Befehle auch Zeit benötigen.

3. Der Maschinensprache-Monitor

Vom Editor gelangen Sie durch die Eingabe von

<CTRL>-P

in den Maschinensprache-Monitor. Der Bildschirm wird gelöscht und das "MONITOR."-Prompt erscheint in der linken oberen Bildschirmcke. Mit dem ATMAS-II Monitor können Sie Maschinenprogramme starten, auf Diskette ablegen, den Speicherinhalt prüfen, verändern und disassemblieren und dabei ein Protokoll am Drucker mitführen. Da der Monitor über eine dialogorientierte Eingabe verfügt, brauchen Sie sich keinerlei komplizierte Befehlssyntax merken.

3.1 Allgemeine Benutzungshinweise

Alle Eingaben innerhalb des Monitors erfolgen in hexadezimaler Schreibweise (ohne vorangestelltes Dollarzeichen!). Sollten Sie sich bei der Eingabe vertippen, so kann diese jederzeit mit 'X' abgebrochen werden.

Wenn Fragen im Dialog auftreten, werden diese mit 'Y' (Yes) positiv beantwortet, jeder andere Tastendruck (auch <RETURN>) beantwortet die Frage verneinend. Auch hier ist ein Abbruch mit 'X' möglich.

Filennamen müssen immer im Standard-Atari Format eingegeben werden, d.h. zuerst einen Gerätenamen (D:, D1:, D2:...) dann der Filename mit max. 8 Zeichen, der nach einem Dezimalpunkt noch eine max. 3 Zeichen lange Erweiterung haben darf.

Beispiele: D:TEST.OBJ, D2:CODE.COM, D1:FILE

3.2 Befehlsvorat des Monitors

Die Befehle des Maschinensprache-Monitors bestehen aus einfachen Buchstaben, die Parameter, soweit nötig, werden im Dialog abgefragt.

3.2.1 M - Memory-Dump

Mit dem M-Befehl können Sie einen Speicherbereich in hexadezimaler Schreibweise auf Bildschirm oder Drucker ausgeben. Weiterhin können Sie eine zusätzliche Darstellung des Speicherinhaltes in ASCII-Zeichen erhalten.

Beispiel: Ihre Eingabe:	Bildschirm:
M	DUMP
D000	FROM: D000
D080	TO: D080
Y	ASCII?Y
<RETURN>	PRINT?

Nun wird der Speicherbereich von \$D000-D080 sowohl in hexadezimaler Schreibweise als auch in ASCII ausgegeben. Hätten Sie bei der Frage ASCII? einfach <RETURN> gedrückt, so würde nur die hexadezimale Schreibweise ausgegeben. Antworten Sie auf die Frage PRINT? mit 'Y', dann müssen Sie sich für einen Ausgabekanal entscheiden:

- (1): wählt die serielle Schnittstelle R1: des ATARI-Interfaces als Ausgabekanal
- (2): Ausgabekanal ist die Centronics-Schnittstelle des ATARI-Interfaces (normaler 'P:'-Printer-Handler).
- (3): Ausgabe über Joystickinterface Port 3&4 (400/800)

3.2.2 D - Disassemble

Mit dem D-Befehl können Sie den Inhalt eines Speicherbereiches als 6502-Befehle rückübersetzen (disassemblieren) lassen. Wie bei M können Sie die Ausgabe auf den Drucker umlenken.

Beispiel:	Ihre Eingabe	Bildschirm
	D	DISASSEMBLER
	D000	START?D000
	<RETURN>	PRINT?

Jetzt wird der Speicherinhalt ab der Adresse \$D000 in disassemblierter Form ausgegeben, wobei immer nach einer Füllung des Bildschirms unterbrochen wird. Durch Drücken einer beliebigen Taste (außer 'X') wird der nächste Bildschirm ausgegeben. 'X' beendet die Disassemblierung. Bei Ausgabe auf Drucker findet keine Unterbrechung des Listings statt, der Ausdruck kann mit <RESET> abgebrochen werden.

3.2.3 C - Change Memory

Das C-Kommando erlaubt Ihnen die Veränderung von Speicherinhalten. Die Eingabe erfolgt in hexadezimaler Schreibweise.

Beispiel:	Ihre Eingabe	Bildschirm
	C	CHANGE
	A900	ADDRESS?A900
	FF	A900 => FF
	00	A901 => 00
	X	A902 =>
		MONITOR.

Sie können gezielt einzelne Bytes oder auch zusammenhängende Speicherblöcke in hexadezimaler Form eingeben. Durch die Eingabe von 'X' kommen Sie wieder in den Befehls-Eingabemodus des Monitors zurück.

3.2.4 F - Fill Memory

Mit dem F-Befehl können Sie einen Speicherbereich mit einem gewünschten Wert vorbesetzen.

Beispiel:	Ihre Eingabe:	Bildschirm:
	F	FILL
	A900	FROM:A900
	AFFF	TO:AFFF
	FF	WITH:FF

Wirkung: Der Speicherbereich \$A900 bis \$AFFF wird mit dem Wert \$FF gefüllt.

3.2.5 B - Blocktransfer

Das B-Kommando erlaubt die Verschiebung ganzer Speicherblöcke. Dies kann nützlich sein, wenn Sie bei der Assemblierung mit ATMAS-II eine unterschiedliche logische und physikalische Adresse gewählt haben. Der B-Befehl benötigt Anfangs- und Endadresse des zu verschiebenden Bereiches, sowie die Anfangsadresse des Zielbereiches.

Beispiel:	Ihre Eingabe	Bildschirm
	B	BLOCKTRANSFER
	AB00	FROM:AB00
	A900	TO:A900
	AC00	INTO:AC00

Wirkung: Der Speicherblock von \$AB00 bis \$A900 wird in den Speicherbereich von \$AC00 bis \$AD00 kopiert.

3.2.6 G - Goto Address

Mit dem G-Befehl können Sie ein Maschinenprogramm starten, dessen Einsprungadresse Sie eingeben müssen.

Beispiel:	Ihre Eingabe	Bildschirm
	G	GOTO
	AB00	GOTO AB00

ACHTUNG: Sie müssen selbst dafür Sorge tragen, daß ein ausführbares Maschinenprogramm an der angegebenen Stelle steht!

Die Kontrolle wird an den Monitor zurückgegeben, wenn das Maschinenprogramm entweder mit einem RTS (Return from Subroutine) oder BRK (Break-Befehl, Hex-Byte \$00) endet. Im letzteren Fall bekommen Sie die Registerinhalte und die Prozessor-Flags angezeigt, eine hervorragende Möglichkeit, um ein Programm nach Fehlern zu durchsuchen.

3.2.7 S - Binary Save

Um ein vom Assembler erzeugtes Maschinenprogramm auf Diskette abzuspeichern, können Sie entweder ins DOS gehen (über Editor, <ESC> M <ESC> <ESC>) oder den Monitor-Befehl S benutzen. Der Monitor speichert Maschinenprogramme so ab, daß Sie vom DOS-II (oder dazu kompatiblen DOS-Versionen) wieder geladen werden können.

Beispiel:

Ihre Eingabe	Bildschirm
S	SAVE
AB00	FROM:AB00
AF00	TO:AF00
<RETURN>	INTO:
D:TEST.OBJ	FILENAME(D:FN.EXT)? D:TEST.OBJ

Dieses Beispiel bewirkt, daß der Speicherbereich von \$AB00 bis \$AF00 als File TEST.OBJ im Laufwerk 1 abgespeichert wird. Der Filename muß immer im Standard-Atari Format eingegeben werden.

Der SAVE-Befehl des ATMAS-II Monitors hat noch einige Zusätze aufzuweisen, die über den Standard hinausgehen:

A) Adressversetztes Abspeichern: Sie können ein Programm so abspeichern, daß es beim erneuten Einlesen in einen anderen Speicherbereich geladen wird. Das ist sehr nützlich, wenn Sie Programme mit getrennter logischer und physikalischer Adresse (s. ORG-Direktive) assembliert haben.

Sie müssen dazu bei den Fragen FROM bzw. TO den physikalischen Adressbereich angeben (wo das Programm momentan abgelegt wurde) und bei der Frage INTO die logische Adresse (an die das Programm geladen werden soll) angeben.

Beispiel:

Ihre Eingabe	Bildschirm
S	SAVE
A800	FROM:A800
A780	TO:A780
4000	INTO:4000
D:TEST.OBJ	Filename(D:FN.EXT)?D:TEST.OBJ

Dieses Beispiel würde ein Programm, das von \$A800 bis \$A780 im Speicher steht, so auf die Diskette schreiben, daß es beim erneuten Einlesen im Bereich von \$4000 bis \$4180 liegt. Hatten Sie das Programm mit ORG \$4000,\$A800 assembliert, so ist es jetzt ein lauffähiges Maschinenprogramm. ACHTUNG: Sie sollten dieses File nicht mehr im Monitor einlesen, da sonst ATMAS-II überschrieben würde!

B) Append-Option: Wenn Sie als letztes Zeichen des Filenames ein Größer-Zeichen ('>') eingeben, so wird das File an ein eventuell bereits bestehendes mit gleichem Filenamen angehängt. Sie können damit Compound-Files (aus mehreren Blöcken zusammengesetzte Files) oder Files mit INIT und RUN-Adresse erzeugen.

Beispiel:

Ihre Eingabe	Bidschirm
S	SAVE
AA00	FROM:AA00
AB00	TO:AB00
<RETURN>	INTO:
D:TEST.OBJ>	FILENAME (D:FN.EXT)?D:TEST.OBJ>

Das File TEST.OBJ (von vorhin) wird in diesem Beispiel um den Speicherblock von \$AA00 bis \$AB00 verlängert. Mit derselben Methode können Sie auch RUN und INIT-Adressen an ein File anfügen: Sie tragen die RUN-Adresse (LSB, MSB) in die Adressen \$02E0, \$02E1 (INIT: \$02E2, \$02E3) ein und hängen den jeweiligen 'Speicherblock' (der nur aus zwei Bytes besteht FROM: 02E0 TO: 02E1) an das File an.

3.2.8 L - Binary Load

Analog zum Save-Befehl kann hier ein Binär-File von der Diskette geladen werden. Es kann sich dabei um ein vom Save-Befehl erzeugtes oder um ein DOS erzeugtes Binär-File handeln, auch zusammengesetzte Compound-Files werden geladen. RUN und INIT Sprünge werden nicht ausgeführt.

Beispiel:

Ihre Eingabe	Bildschirm
L	LOAD
D:TEST.OBJ	FILENAME (D:FN.EXT)? D:TEST.OBJ
	FROM: 5000 TO: 5180
	FROM: AAOO TO: ABOO

Der Load-Befehl gibt Ihnen gleich an, in welchen Speicherbereich das Binär-File geladen wurde. Im Beispiel wurde ein File geladen, welches ähnlich zu dem im Save-Beispiel erzeugten ist.

3.2.9 E - Editor

Mit dem E-Kommando gelangen Sie zurück in den Editor, der Cursor befindet sich noch an der Stelle, wo Sie den Editor verlassen haben.

3.2.10 I - Disketteninhaltsverzeichnis

Der I-Befehl zeigt Ihnen alle Filenamen des Laufwerks 1 auf dem Bildschirm an.

4. Beispiele

4.1 Demoprogramm Farb-Scrolling

Das nachfolgende Beispiel soll Ihnen das Zusammenspiel der einzelnen Komponenten von ATMAS-II verdeutlichen. Von Vorteil wäre es, wenn Sie den Teil 1 (Der Editor) dieses Handbuches schon durchgelesen hätten und mit dem Editor schon etwas vertraut sind. Teil 2 und 3 wären nicht notwendigerweise Voraussetzung, tragen aber sicherlich zum besseren Verständnis bei.

Obwohl das nachfolgende Demo-Programm auf der ATMAS-II Diskette enthalten ist (DEMO.SRC) sollten Sie es dennoch von Hand eintippen, um sich besser mit dem Editor vertraut zu machen. Falls Schwierigkeiten auftreten sollten, können Sie dann immer noch auf das fertige File zurückgreifen.

```
*****
* ATMAS-II Demo: Farb-Scrolling PF85
*
* Abbruch durch START-Taste
*****
```

```
COLPF2    EQU $D018           Farbregister
WSYNC     EQU $D40A           Synchron.
VCOUNT    EQU $D40B           Rasterzeile
RTCLK     EQU $14             VBI-Uhr
CONSOL    EQU $D01F           Fkt.-Tasten
```

```
*
* Programm liegt im USER-Bereich
* (ab $A800)
*
```

```
                ORG $A800           res. Platz

                LDA #8             Abfrage der
                STA CONSOL         START-Taste vorbereiten
SCRCOL          CLC
                LDA VCOUNT        Bild-Zaehler
                ADC RTCLK          plus Raster-Zeile
                STA WSYNC          synchronisieren
                STA COLPF2        in Farbregister
                LDA CONSOL        Funktionstasten
                AND #1            START-Taste?
                BNE SCRCOL        nein, weiter-->
                RTS
```

Zur Erinnerung einige Hinweise zum Eintippen des Programmes: Kommentarzeilen beginnen immer mit einem Stern in der ersten Spalte. Ebenfalls müssen Zeilen, die mit einem Label versehen sind in der ersten Spalte beginnen. Bei allen anderen Zeilen ist es von Vorteil am Zeilenanfang einen Tabulator zu verwenden.

Beispiel:

Die erste 'EQU' Zeile sollten Sie so eintippen:

```
COLPF2<TAB>EQU<LEERZ.>#$D01B<RETURN>
```

Nachdem Sie das ganze Programm eingetippt haben, können Sie die erste Assemblierung durch die Eingabe von <CTRL>-Y beginnen. Sie sehen den Copyrightvermerk des Assemblers und, wenn der Assembler keinen Fehler erkannt hat, die Endadresse des erzeugten Objektprogrammes.

Sollten Fehler aufgetreten sein, so werden Ihnen diese im Klartext am Bildschirm präsentiert. Sie können diesen mit Hilfe der Fehlertabelle im Anhang B auf die Spur kommen. Als Ursachen kommen hier nur Tippfehler in Frage.

Zuerst drücken Sie nun eine beliebige Taste um wieder in den Editor zurück zu kommen. Der Cursor befindet sich sofort in der fehlerhaften Zeile. Ein kleiner Tip: Manchmal hilft die Eingabe von <CTRL>-T um zu sehen, ob vielleicht ein Tabulator an der falschen Stelle sitzt (Eine Zeile, die nur aus einem Tabulator oder einem Leerzeichen besteht wird als SYNTAX-ERROR gemeldet!).

Meldet der Assembler keinen Fehler, so können Sie daran gehen, das gerade assemblierte Programm zu starten. Dazu betätigen Sie zuerst eine beliebige Taste um in den Editor zurück zu gelangen, und geben anschließend <CTRL>-P zum Aufruf des Monitors ein.

Im Monitor können Sie sich das vom Assembler erzeugte Programm ansehen. Zu diesem Zweck geben Sie 'D' (Disassemble) gefolgt von 'A00' und <RETURN> ein. Sie müßten jetzt ein disassembliertes Listing des Programmes am Bildschirm sehen. Drücken Sie jetzt 'X' um das Disassembler-Listing abubrechen.

Zum Starten des Programmes geben Sie 'G' (GOTO) und die Startadresse des Programmes, in unserem Falle 'A800' ein. Wenn Sie alles richtig gemacht haben, dann müßten Sie jetzt 128 Farben über den Bildschirm laufen sehen. Das Farbspektakel kann mit <START> oder <RESET> abgebrochen werden. Wenn Sie <RESET> drücken, kommen Sie automatisch in den Editor zurück, mit <START> wird das Programm ordnungsgemäß beendet, und die Kontrolle geht an den Monitor zurück. 'E' reaktiviert schließlich den Editor.

4.2 Makro-Bibliotheken

Auf der ATMAS-II Diskette finden Sie zwei weitere Files IOLIB.SRC sowie GRAFLIB.SRC, deren Listings auf den nächsten Seiten wiedergegeben sind. Es handelt sich dabei um sogenannte Makro-Bibliotheken, d.h. sie enthalten (neben einem kleinem Demo) nur Makro-Definitionen. Die Files enthalten reichlich Kommentar, so daß eine genaue Beschreibung nicht nötig ist.

GRAFLIB.SRC enthält Makros die ähnlich den BASIC-Befehlen GRAPHICS, COLOR, PLOT und DRAWTO arbeiten. Ein Demo können Sie sich ansehen, wenn Sie GRAFLIB.SRC laden, assemblieren und mit dem Monitor an der Adresse \$A800 starten.

IOLIB.SRC enthält Makros, die ähnlich den BASIC-Befehlen OPEN, CLOSE, PRINT und INPUT arbeiten, sowie zwei weitere, BGET und BPUT, die Laden und Abspeichern von binären Dateien erlauben. Auch hier ist ein interessantes Demo enthalten, das Sie analog zur Vorgehensweise bei GRAFLIB bekommen.

In den Makro-Bibliotheken finden Sie Beispiele zur Verschachtelung von Makros und zur Übergabe von String-Parametern. Verstehen Sie die beiden Files als Anregung, was mit ATMAS-II machbar ist.

Selbstverständlich sind Erweiterungen und Verbesserungen Ihrer persönlichen Intuition überlassen.

```
*****
*          GRAFLIB.SRC
*
*          Makro-Bibliothek
*
*          GRAPHIK
*
*          Fuer ATMAS-II
*          PETER FINZEL
*****
*
* IOCB-Struktur:
*
ICCOM      EQU $342
ICSTA      EQU $343
ICBAL      EQU $344
ICBAH      EQU $345
ICBLL      EQU $348
ICBLH      EQU $349
ICAX1      EQU $34A
ICAX2      EQU $34B

CIOV       EQU $E456

* CIO-Befehle

COPEN      EQU 3
CCLSE      EQU 12
CGTXT      EQU 5
CPTXT      EQU 9
CGBIN      EQU 7
CPBIN      EQU 11
CDRAW      EQU $11

* ATARI Graphik-Variable

ATACHR     EQU $2FB
ROWCRS     EQU $54 CURSOR-
COLCRS     EQU $55 POSITION

*
* GRAPHICS-Befehl
*
* Aufruf: GRAPHICS <stufe>
*
* <stufe> 0 bis 15 (XLs)
*          0 bis 11 (400/800)
*
GRAPHICS   MACRO STUFE
           JMP GRI@

DEV@      ASC "G:"
```

```
GR10      LDX ##60
          LDA #CCLSE           ZUERST KANAL 6
          STA ICCDM,X         SCHLIESSEN
          JSR CIOV
          LDA #STUFE          JETZT NEUE GRAPHIK
          STA ICAX2,X         STUFE ANWAEHLEN
          AND ##F0
          EOR ##10
          ORA ##0C
          STA ICAX1,X
          LDA #COPEN
          STA ICCDM,X
          LDA #DEVE
          STA ICBAL,X
          LDA #DEVE/256
          STA ICBAL,X
          JSR CIOV
          MEND
```

```
*
* Auswahl der Zeichenfarbe
*
* Aufruf: COLOR <farbe>
*
* <farbe> von 0 bis 255, je nach
*   Graphikmodus, muss eine
*   Konstante sein.
*
```

```
COLOR     MACRO COL
          LDA #COL
          STA ATACHR
          MEND
```

```
*
* Positionierung des Cursors
*
* Aufruf: POSITION <x>,<y>
*
* <x>,<y> je nach Graphikmodus, beide
*   muessen Konstante sein
*
```

```
POSITION  MACRO X,Y
          LDA #X
          STA COLCRS
          LDA #X/256
          STA COLCRS+1
          LDA #Y
          STA ROWCRS
          MEND
```

```
* Graphik-Punkte setzen
*
* Aufruf: PLOT <x>,<y>
*
* <x>,<y> je nach Graphikmodus,
*       muss sich um Konstante
*       handeln
*
PLOT      MACRO X,Y
          POSITION X,Y
          LDX #$60          KANAL 6
          LDA #CPBIN
          STA ICCOM,X
          LDA #0
          STA ICBLL,X
          STA ICBLH,X
          LDA ATACHR
          JSR CIOV
          MEND
```

```
*
* Graphik-Linien ziehen
*
* Aufruf: DRAWTO <x>,<y>
*
* <x>,<y> je nach Graphikmodus
*       Konstante
*
DRAWTO    MACRO X,Y
          POSITION X,Y
          LDX #$60          KANAL 6
          LDA #CDRAW
          STA ICCOM,X
          LDA #CCLSE
          STA ICAX1,X
          LDA #0
          STA ICAX2,X
          JSR CIOV
          MEND
```

```
*****  
* Demo-Programme f. Graphik-Bibliothek  
*  
* zeichnet Raute in GRAPHICS 7  
*****
```

```
*  
* befindet sich im reservierten  
* Speicherplatz fuer Objektcode  
*
```

```
ORG $A800
```

```
GRAPHICS 7+16  
COLDR 1  
PLOT 79,0  
DRAWTO 159,47  
DRAWTO 79,95  
DRAWTO 0,47  
DRAWTO 79,0
```

```
ENDLOS JMP ENDLOS  
*  
* Abruch mit <RESET>  
*
```


*
* Name : OPEN
* Zweck : oeffnen eines Files
* Aufruf:
* OPEN <Num>,<Aux1>,<Aux2>,<Filename>
* Beispiel: OPEN 1,4,0,"D:TEST.OBJ"
*

```
OPEN      MACRO KANAL,AUX1,AUX2,FILENAME
          JMP OP1@
FNAME@   ASC FILENAME
          DFB EOL
OP1@     KANNUM KANAL
          LDA #AUX1
          STA ICAX1,X
          LDA #AUX2
          STA ICAX2,X
          LDA #COPEN
          STA ICCOM,X
          LDA #FNAME@
          STA ICBAL,X
          LDA #FNAME@/256
          STA ICBAL,X
          JSR CIOV
          MEND
```

*
* Name : CLOSE
* Zweck : File schliessen
* Aufruf : CLOSE <Num>
* Beispiel: CLOSE 1
*

```
CLOSE    MACRO KANAL
          KANNUM KANAL
          LDA #CCLSE
          STA ICCOM,X
          JSR CIOV
          MEND
```

*
* Name : PRINT
* Zweck : Ausgabe eines mit 'ASC'
* definierten Textes, muss
* mit EOL beendet werden
* Aufruf : PRINT <Kanal>,<Label>
* Beispiel: PRINT 0,TEXT1
*

```
PRINT    MACRO KANAL,LABEL
          KANNUM KANAL
          LDA #CPTXT
          STA ICCOM,X
          LDA #LABEL
          STA ICBAL,X
          LDA #LABEL/256
          STA ICBAL,X
          LDA #127
          STA ICBLL,X
          LDA #0
          STA ICBLH,X
          JSR CIOV
          MEND
```

max. Laenge

```
*
* Name      : PRINTS
* Zweck     : direkte Ausgabe eines
*            Strings auf den Bildschirm
* Aufruf    : PRINT <String>
* Beispiel: PRINTS "HALLO"
*
PRINTS    MACRO STRING
          JMP PR2@
PR1@     ASC STRING
          DFB EOL
PR2@     PRINT 0,PR1@      obiges Makro!
          MEND
*
* Name      : INPUT
* Zweck     : String einlesen
* Aufruf    : INPUT <Kanal>,<Label>
* Beispiel: INPUT 0,TEXT1
*
INPUT     MACRO KANAL,LABEL
          KANNUM KANAL
          LDA #CSTXT
          STA ICCOM,X
          LDA #LABEL
          STA ICBAL,X
          LDA #LABEL/256
          STA ICBAH,X
          LDA #127          max. Laenge
          STA ICBLL,X
          LDA #0
          STA ICBLH,X
          JSR CIOV
          MEND
*
* Name      : BGET
* Zweck     : Einlesen eines Datenblockes
*            der Laenge L ab Adresse A
* Aufruf    : BGET <Num>,<L>,<A>
* Beispiel: BGET 1,$B000,$100
*
BGET     MACRO KANAL,LAENGE,BUFFER
          KANNUM KANAL
          LDA #CGBIN
          STA ICCOM,X
          LDA #LAENGE
          STA ICBLL,X
          LDA #LAENGE/256
          STA ICBLH,X
          LDA #BUFFER
          STA ICBAL,X
          LDA #BUFFER/256
          STA ICBAH,X
          JSR CIOV
          MEND
```

* Name :BPUT
* Zweck :Speichern eines Datenblockes
* der Laenge L ab Adresse A
* Aufruf :BPUT(Num),<L>,<A>
* Beispiel:BPUT 1,\$B000,\$100
*

```
BPUT    MACRO KANAL,LAENGE,BUFFER
        KANUM KANAL
        LDA #CPBIN
        STA ICCOM,X
        LDA #LAENGE
        STA ICBLI,X
        LDA #LAENGE/256
        STA ICBLH,X
        LDA #BUFFER
        STA ICBAL,X
        LDA #BUFFER/256
        STA ICBAH,X
        JSR CIOV
        MEND
```

* Demo-Programm I/O-Bibliothek
* zeigt Inhaltsverzeichnis des
* Laufwerks 1 an.

```
        ORG $A800
        PRINTS "Inhaltsverzeichnis Laufwerk 1;"
        OPEN 1,6,0,"Di:*.*)"

NEXT    INPUT 1,BUFFER    Dir-Zeile einlesen
        BMI ENDE        End of File?
        PRINT 0,BUFFER    und ausdrucken
        JMP NEXT        naechste Zeile

ENDE    CLOSE 1        Fertig!
        RTS

BUFFER  ORG **20        Platz freihalten
```

* START DES DEMOS:
* Mit <CTRL>-Y assemblieren,
* Monitor mit <CTRL>-P aktivieren
* und mit 'G'oto A800 starten.

Anhang A ATMAS-II Memory-Map
=====

0000 - 007F: *Betriebssystem-Zeropage*
0080 - 0085: ***** frei für Benutzer *****
0086 - 00DF: *Editor/Monitor Zeropage*
00E0 - 00FD: *Assembler-Zeropage, aber benutzbar*
(wird von ATMAS-II gelöscht)
00FE - 047F: *Stack, Vektoren, IOCBs ...*
0480 - 05FF: ***** frei für Benutzer *****
0600 - 06FF: ***** frei (PAGE 6) *****
0700 - <LOMEM>: DOS, <LOMEM> DOSII:=\$1F00
DOSXL:=\$2200
<LD> - 27FF: ***** frei für Benutzer *****
2800 - 4AFF: *ATMAS-II*
4B00 - 4BFF: *Zeilenbuffer*
4C00 - 5FFF: *Symboltabelle*
6000 - 63FF: *Kopierregister*
6400 - A7FF: *Textbuffer*
A800 - <MEMTOP>: ***** frei für Benutzer *****
<Me> - BFFF: *Display-List, Screen-RAM*

Für Objektcode stehen Ihnen mehrere Bereiche zur Verfügung:

A) \$600 bis \$6FF: die berühmte Page 6, besonders geeignet, wenn Sie *USR-Programme* für BASIC schreiben wollen.

B) \$1F00 bis \$27FF: dieser Bereich geht von der *DOS-Obergrenze* (die in <LOMEM>, \$2E7-\$2E8 zu finden ist) bis zum Beginn des *ATMAS-II* Programmes. Das die *DOS-Obergrenze* von DOS zu DOS verschieden ist, müssen Sie hier etwas vorsichtig sein, wenn Sie das Standard *DOS-II* verwenden, dann beginnt dieser Bereich bei \$1F00

C) \$A800 bis \$BC3F: Standard-Bereich für Objekt-Code, reicht bis zum Anfang des *Bildschirmspeichers* (bzw. der *Display-List*).

Zusätzlich besteht die Möglichkeit, den Speicherplatz der *Symboltabelle* (\$4C00-\$5FFF) nach dem Assemblieren als Platz für *Felder*, *Zeichensätze* oder als *Player-Missile-Bereich* zu nützen. Die *Symboltabelle* wird schließlich nach der *Assemblierung* nicht mehr benötigt.

Anhang B: Fehlermeldungen

1.) Assembler-Fehlermeldungen:

SYNTAX ERROR: Formatfehler, z.B. wenn Zeilennummer verwendet, oder Label beginnt mit Ziffer, Zeile besteht nur aus Tabuالتor oder Space

NAME UNKNOWN: Fehler im Befehlsfeld; z.B. SRA statt STA

UNDEFINED EXPRESSION: Fehler im Operantenfeld. Tritt bei fehlerhaften arith. Ausdrücken und nicht definierten Labels auf.

ADDRESSING ERROR: Adressierungsart paßt nicht zum Befehl, z.B. STA # $\$6FF$

IMPOSSIBLE BRANCH: Verzweigungsbefehle (BNE, BCS...) reichen nur +127 bzw. -128 Bytes weit
JMP verwenden!

DIVISION BY ZERO: Division durch Null, z.B. LDA #100/0

NUMBER-ERROR: Fehler in der Zahlendarstellung, z.B. LDA # $\%30$

WRONG DELIMITER: Trennzeichen bei ASC-Befehl falsch oder unterschiedlich. <SHIFT>-2 verwenden!

NO ASCII: ASCII-Zeichen nach Zeichen-Konstante fehlt, z.B. LDA #'<RETURN>

LINE TO LONG: eine Zeile darf nicht länger als 127 Zeichen sein.

MACRO ERROR: Fehler in der Makro-Definition bzw. im Makro-Aufruf (Parameter)

ORG ERROR: Fehler in der ORG-Direktive (z.B. ORG fehlt)

TOO MANY LABELS: Symboltabelle ist voll

OPCODE DIFFERENT: Pass 3 erkennt anderen Opcode als Pass 2, z.B. wenn sich das Programm durch zweiten, fehlerhaften ORG-Befehl selbst überschreibt.

2. Editor-Fehlermeldungen:

RW : Fehler bei Disk-Ein/Ausgabe
CO : Kommandozeile zu lang
E? : Fehler in Kommandozeile
H? : fehlerhaftes Hex-Byte
I? : Textbuffer ist voll (T:00000)
L? : Gerätefehler bei Listingausgabe
S? : String nicht gefunden (Suchfunktion)
T? : falscher Tabulator-Wert (nur 1-9)
C? : Kopier-Register ist voll (C:0000)
#? : Wiederholungsfaktor ist falsch (nur 2-255)

OK : Es liegt kein Fehler vor, C-Reg. geschlossen
CR : Kopier-Register ist offen

3. Monitor-Fehlermeldungen:

ADR ERROR: fehlerhafte Adresse beim Laden oder Speichern eines Programmes. Tritt der Fehler bei LOAD-Befehl auf, so ist das gewünschte File nicht im Binärformat (z.B. ein Text-File).

FEHLERCODE: <80-FF>: Betriebssystemfehler, laut DOS-Handbuch. Fehlernummern sind hexadezimal.

ATMAS-II REFERENZKARTE

EDITOR-BEFEHLE im Textmodus

<CTRL>-E : Cursor an den Textanfang
<CTRL>-D : Cursor an das Textende
<CTRL>-R : Kopier-Register öffnen
<CTRL>-F : Kopier-Register schließen
<CTRL>-J : Kopierregister in Text einsetzen
<CTRL>-K : Kopierregister löschen
<CTRL>-V : Umschalten 1-Zeilen und 2-Zeilenmodus
<CTRL>-T : Umschaltung Control-Zeichendarstellung
<CTRL>-G : Wiederholung der Kommandozeile

<CTRL>-Y : Aufruf des ATMAS-II Makroassemblers
<CTRL>-P : Aufruf des Maschinensprache-Monitors

EDITOR-BEFEHLE der Kommandozeile

B : Cursor eine Position zurück
F : Cursor eine Position vorwärts
D : Zeichen links vom Cursor löschen
T : Von Cursorposition bis Zeilenende löschen
H<Hexbyt>: beliebigen ASCII-Code in den TEXT einfügen
S<String>: Sucht nach Zeichenkette <String>
I<String>: Zeichenkette <String> in Text einsetzen
J : Wiederholt Kommandozeile
K : Löscht Textbuffer
U : User-Befehl, startet Maschinenpgm ab \$AB00
&<n> : Setzt Tabulatorweite auf <n> Zeichen
M : Rückkehr zum DOS
E : Löscht Kopier-Register

G : Kopier-Register in Text einfügen
W<D:FN> : Text ab Cursorposition unter <FN> speichern
R<D:FN> : Text ab Cursorposition von File <FN> lesen
L<0,1,2> : Listing des Quelltextes ausgeben (1:=Drucker)

ASSEMBLER-DIREKTIVEN:

ORG : Anfangsadresse Objektcode festlegen
EQU : Konstanten definieren
EPZ : Zeropagekonstante definieren, nicht obligatorisch!
DFB : Byte-Werte in Objektcode einfügen
DFW : Wort-Werte in Objektcode einfügen
ASC : Texte in ASCII und Bildschirmcode ablegen
OUT : Kontrolle des Ausgabelistings
MACRO : Einleitung der Makrodefinition
MEND : Abschluß der Makrodefinition

MONITOR-BEFEHLE:

M : Memory-Dump-Befehl
D : Disassemblieren
C : Change-Memory, Speicher editieren
F : Fill Memory, Speicherblock mit Wert füllen
B : Blocktransfer
G : Goto, Maschinenprogramm starten
S : Binary-Save, Maschinenprogramm speichern
L : Binary-Load, Maschinenprogramm laden
E : Rückkehr zum Editor
I : Inhaltsverzeichnis Laufwerk 1 anzeigen

HOFACKER

Ing. W. HOFACKER GmbH

Tegernseer Straße 18

D-8150 Holzkirchen

Telefon: (0 80 24) 73 31 ★ Telex: 52 69 73

Lesen Sie diesen Text, bevor Sie beginnen !

Das Ihnen hier vorliegende Programm und das dazugehörige Handbuch wurden vom Autor und vom Verlag mit großer Sorgfalt erstellt und geprüft. Trotzdem lassen sich Fehler nie ganz vermeiden. Der HOFACKER VERLAG sieht sich deshalb gezwungen, weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für eventuelle Folgeschäden, die auf Programmfehler oder fehlerhafte Angaben in der Anleitung zurückzuführen sind, zu übernehmen. Bei eventuellen Fehlern sind wir für jeden schriftlichen Hinweis dankbar.

COPYRIGHT BY ING. W. HOFACKER © 1985

Der Käufer von Software erwirbt mit seinem Kaufpreis ausschließlich eine nicht exklusive, an seine Person gebundene Lizenz. Kopien dürfen nur für die Archivierung angefertigt werden.

Eine Weitergabe an dritte Personen ist grundsätzlich verboten. Auch die unentgeltliche Weitergabe führt zu Schadensersatzansprüchen des Verlages.

Gedruckt in der Bundesrepublik Deutschland.
Irrtum sowie alle Rechte vorbehalten.

HOFACKER