65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 7,80 Nr. 15 Oktober 1980

Nun auch 6809

Dem aktuellen Datenverbund zwischen Prozessorsystemen sind in diesem Heft zwei Artikel gewidmet. Weitere werden folgen.

Die Mehrzahl der Beiträge enthält wiederum wertvolle Systementwicklungsprogramme für AIM und CBM. - Im ersten Artikel wird der MC 6809 (Motorola) als der derzeit wohl leistungsfähigste 8-/16-Bit Prozessor vorgestellt. Er hat viele verwandschaftliche Merkmale mit dem 6502. Wir werden seine Programmierung künftig berücksichtigen, denn viele Leser betreiben mehrere Systeme, auch schon mit 6809, so daß auch hier ein Informationsbedürfnis besteht.

Inhaltsverzeichnis

Neu im Programm:

Mini - Digital - Kassettenrekorder

Schneller serieller Speicher, direkter Zugriff
auf maximal 60 kByte in maximal 90 sekunden.
Kassettenlaufwerk komplett vom Rechner aus steuerbar.

Preis pro Laufwerk	350,-- DM
Interface-Elektronik zwischen VIA und Kassettenrekorder	210,-- DM
Betriebsprogramm für diese Kassettenrekorder am 6502, Grundroutinen	90,-- DM
Kassettenoperating-System zum Betrieb dieser Laufwerke am 6809 mit Dateiverwaltung (beliebige Namenslänge etc.)	290,-- DM

Die Preise verstehen sich zzgl. 13 % MWSt.

Weiterhin lieferbar:

CPU-Karte Nico 65 mit 6502 CPU
CPU-Karte Nico 69 mit 6809 CPU
Combo-Karte mit 8 k RAM und 2 VIAs
Video-Karte
Buskarte
EPROM-Programmiergerät

Im Wiehagen 15 - Tel. (05241) 67480
4830 Gütersloh 1

Ein fortschrittlicher Verwandter - MC6809

Man zählt die CPU 6502 zur dritten Generation der Mikroprozessoren, nach Intel 4040/8080 und MC6800. Sie wurde bekanntlich von Designern geschaffen, die an der 6800 mitgewirkt hatten. Mit der 6809 hat Motorola einen Mikroprozessor geschaffen, der mit anderen der vierten Generation zuzurechnen ist. Das Leistungsspektrum ist deutlich erweitert und dürfte derzeit die Spitze unter den Prozessoren mit einem 8-Bit-Datenbus halten. Die CPU ist zwar seit etwa einem Jahr am Markt und nimmt ihren Weg. Gleichwohl ist sie so wenig bekannt, daß wir sie hier einmal aus der Sicht eines 6502-Programmierers ausführlich darstellen wollen. In dieser Sicht wird man viel Familiäres erkennen können, aber ganz deutlich auch den Fortschritt.

Ein Überblick

Der 6909 hat ein Gehäuse mit 40 Pins, darunter die üblichen 16 Adreßbus- und die 8 Datenbuspins. Die CPU wird in Ausführungen mit 1, 1,5 und 2 MHz Maschinentakt geliefert. Der treibende Quarz muß die vierfache Frequenz haben. Die Taktsignale werden on-chip erzeugt und verlassen die CPU an den Pins Q und E, den Ø1 und Ø2 vergleichbar.

Das R/$\overline{W}$-Signal (Lesen/Schreiben) ist gleich, ebenso die Interrupteingänge $\overline{RESET}$, $\overline{NMI}$, $\overline{IRQ}$. Weitere Signalpins werden später abgehandelt.

Wenn man von den zusätzlichen Betriebsweisen mit den weiteren Signalpins einmal absieht, so liegt der deutliche Unterschied zum 6502 darin, daß die CPU 6 für den Benutzer erreichbare 16- Bit breite Register trägt, nämlich Programmzähler, Indexregister X und Y, Hardware Stackpointer, User-Stackpointer und Doppelakkumulator D. Das Statusregister ist in 8 Bit angelegt, ebenso das Direct Page Register DP.

Der Befehlssatz der 6809 umfaßt selbstverständlich ausreichend Instruktionen, die in 16 Bit Breite arbeiten. Die beiden mit fast identischem Instruktionssatz ausgerüsteten Akkumulatoren A und B sind eine Untermenge des Akku D.

Nach Datenbus und Architektur ist der 6809 vor allem ein leistungsfähiger 8-Bit Prozessor (oder man sagt 'Pseudo 16-Bit-Prozessor' - im Gegensatz zum echten 16-Bit-Prozessor, dem 68000). Er ist statt des aus offensichtlich personenbezogenen Gründen von Rockwell, Synertec und Mos-Technology nicht mehr vollendeten 6516 heranzuziehen (das gilt auch bezüglich des vorhandenen 68000 zum zurückgezogenen 65000).

Der Befehlssatz ist dementsprechend vorwiegend auf in 8 Bit Breite wirkende Instruktionen zugeschnitten - aber mit erheblichen Erweiterungenund mit einer Indizierungsmöglichkeit mit 6 direkten und 7 indirekten Adressierungsarten, die für fast alle 8- und 16-Bit-Befehle gelten! Damit ist eine kleine Bombe im Zimmer versteckt, mit der man umzugehen lernen muß.

Dieser Überblick muß wie folgt ergänzt werden: Es gibt 15 bedingte und einen unbedingten Verzweigungsbefehle (gegen 8 bei 6502). Jeder Verzweigungsbefehl ist auch als Long Branch (mit Offset von 2 Bytes) implementiert. Daneben gibt es Branch to Subroutine und Long Branch to Subroutine.

Und: Alle indizierbaren Befehle können mit einem Offset von 1 oder 2 Bytes relativ zum augenblicklichen Stand des Programmzählers adres-

siert werden. Damit und mit den Long Branches erhält man unbeschränkt verschieblichen Maschinencode. Programm-Moduln samt ihren Konstanten- und Arbeitsbereichen sind ohne Umrechnung an jeder Stelle des Speichers ausführbar!

Das Direct Page Register erlaubt es, jede der 256 Speicherseiten als 'Zeropage' mit den kurzen 2-Byte-Befehlen und der schnelleren Ausführungszeit zu benutzen. Das Direct Page Register wird anfangs initialisiert und kann im Programmverlauf je nach Bedarf geändert werden. M.a.W.: Bei Direct-Page-Befehlen wird sein Inhalt automatisch auf dem hohen Adreßbus ausgegeben (A8-A15).

Der Befehlssatz erlaubt binäre Multiplikation zwischen den Operanden in Akku A und Akku B mit Ergebnis in Akku D (Ausführung in 11 Zyklen). Für das Rechnen mit BCD-Zahlen steht der DAA-Befehl zur Verfügung (Decimal Adjust Accu A), 2 Zyklen.

Die Registerinhalte können in jedes andere gleich breite (8 oder 16 Bit) Register übertragen (Transfer) oder gegeneinander ausgetauscht werden (Exchange), einschließlich PC!

Jedes Register (außer DP und Status) kann zur direkten und indirekten Indizierung benutzt werden, z.T mit einem +Offset zum Registerinhalt, und es gibt ein Auto-Inkrement/Auto-Dekrement des Resgisters um 1 oder 2 nach der Befehlsausführung.

Der 6809 ist stack-orientiert und eignet hervorragend zur Übergabe von Parametern. Die Stacks können an jeder Adresse des Speichers implementiert werden, per Ladeanweisung für die Stackpointer oder Registertransfer. (Bei der 6502 ist der Stack immer in page 1) Der Hardwarestack sichert bei Unterprogrammaufrufen und Interrupts die Maschinenregister. Für beide Stacks gibt es die Befehle PUSH und PULL, mit denen jedes beliebige einzelne Register oder Register-Sets gestackt/entstackt werden können. Mit der indizierten Adressierung sind auch auf dem Stack befindliche Registerinhalte noch bequem erreichbar. - Beim 6502 übergibt man Parameter u.a. durch Transporte auf den Stack. Beim 6809 setzt man den User-Stackpointer oder ein Register auf die Parameterliste.

Der Befehlssatz

Um den Leser nicht zu langweilen, werden nicht alle Befehle abgehandelt. Folgende Befehle, die auf eine Speicherzelle (Read Modify Write) oder auf Akku A oder B wirken, sind in ihrer Wirkung bekannt: Die Verschiebebefehle ASL, LSR, ROL und ROR, Decrement und Increment (auch für beide Akkus!). Neu ist ASR mit Erhaltung des Vorzeichen-Bits. Hinzugekommen sind CLR (setze auf 00) und NEG (bilde 1er Komplement), ferner COM (2er-Komplement).

Der TST-Befehl (Test) prüft Speicher oder Akkus auf Vorzeichen und Null. Der vorgenannte Befehlssatz ist komplett für Direct Page, absolut und indiziert vorgehalten.

Hinsichtlich der Akkus A und B sind bekannt das Laden und Abspeichern, der Vergleich, Addieren und Subtrahieren (auch ohne Carry-Einfluß), die logischen Befehle und der fast gleiche Bit-Befehl. Alles in den 4 Adressierungsarten Direktoperand, Direct Page, indiziert und extended (absolut).

Für die 16-Bit breiten Register sind Lade-, Store- und Vergleichsbefehle implementiert, für X und Y auch mit Direktoperand, für die übrigen in den Adressierungsarten Direct Page, indexed und extended. - Die

Stackpointer können auch mit Direktoperanden verglichen werden. - Akku D erhält zusätzliche Rechenhaftigkeit durch Addition, Subtraktion und Vergleich in allen 4 Adressierungsarten.

Weitere Befehle wurden schon in der Übersicht angesprochen. Erwähnenswert sind die gegenüber 6502 zusätzlichen Verzweigungsbefehle für vorzeichenbehaftete und voorzeichenfreie Zahlen, die auf die Kombination von Statusbits reagieren (wenn kleiner, kleiner/gleich, gleich, größer/oder gleich, größer).

Neben den Branches gibt es natürlich auch Sprung- und JSR-Befehle mit Adressierung für Direct Page, absolut (z.B. in Monitor-Routinen) und, man höre, in allen direkten und indirekten Indizierungen. Es ist also ein Leichtes, eine Sprungleiste aufzumachen und in Abhängigkeit von einem mitlaufenden Register indiziert zu springen.

Es gibt keine Befehle zum Setzen oder Löschen von Statusflags. Stattdessen wird das Statusregister mit einem Direktoperanden ge-odert (Hinzuschalten eines odoer mehrerer Flags) oder ge-anded (abmaskieren).

Es gibt drei Befehle für Software-Interrupts, jeder mit eigenem Sprungvektor.

Interrupts und besondere Signale

Die Interruptvektoren sind wie üblich ab $FFF2 aufwärts abgelegt, und zwar für die drei Software-Interrupts, den Fast Interrupt $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$ und $\overline{\text{NMI}}$. Der FIRQ erlaubt eine beschleunigte Interruptbearbeitung, weil er nur PC und Status auf dem Stack ablegt, die übrigen Interrupts legen alle Register ab.

Der NMI ist in bekannter Art flankengetriggert, der IRQ wird durch low level ausgelöst, ebenso der FIRQ. Letzterem ist ein eigenes Flag im Status zugeordnet, das FIRQ zuläßt oder nicht. Er ist auch gegenüber dem IRQ bevorrechtigt. Durch den Befehl CWAI werden die Register schon vor dem Interruptereignis gerettet, der Prozessor wartet auf den Interrupt, die Reaktion wird damit noch schneller.

Unter HALT (Eingangspin) hält der Prozessor ohne Datenverlust an und bringt die Busse in den hochohmigen Zustand (Tri-State). Er kann nur noch auf $\overline{\text{DMA}}$ reagieren. Ein jetzt eintreffender NMI oder RESET werden nicht ausgeführt aber für spätere Abarbeitung gelatcht. - Der Eingang $\overline{\text{DMA/BREQ}}$ hält die CPU ebenfalls an (z.B. auch für Refresh von dynamischen Speichern), sie führt alllerdings alle 16 Takte einen Erholungstakt durch.

Die Ausgangssignale BA und BS (Bus Available, Bus Status) zeigen in ihrer Kombination Betriebsweise und Busverfügbarkeit an. Man kann ihnen ferner ansehen, wann ein Interruptvektor geholt wird. Durch externe Beschaltung kann man mit ihnen also hardware-vektorierte Adressen für das spezielle Interruptereignis auf den Datenbus geben.

Das Signal MRDY (Memory Ready, input) erlaubt die Bedienung von Speichern mit langsamer Datenbereitstellung bis 10 Mikrosekunden.

Zusammenfassung

Der vorstehende Aufsatz ging weniger auf bekanntes Gemeinsames zwischen 6502 und 6809 ein, als vielmehr auf das vom 6809 zusätzlich Gebotene. Und das ist wirklich eine Menge, wenn wir auf die letzten Seiten nooch einmal zurückblicken:

Interne 16-Bit-Struktur in 6 erreichbaren Registern,
Erweiterter Befehlssatz, auch für 2 Bytes breite Operanden,
Durchgehend 6 direkte und 7 indirekte Indizierungen, indizierte Sprünge,
Zugriff auf alle Register und Indizierung mit allen Registern, auch relativ zum Programmzähler,
Adreßunabhängige Programmierung durch Branches, Long Branches und Branches zu Unterprogrammen,
zwei beliebig definierbare Stacks zur bequemen Parameterübergabe oder zur Bereitstellung von Arbeitsbereichen für Module,
Beliebig viele Zeropages,
Erweiterte Test- und Verzweigungsbedingungen,
Erweiterte externe Beeinflussung in Interrupts und in DMA,
Volle Verträglichkeit mit 68er und 65er Interfacebausteinen.

Die Architektur ist darauf abgestellt, daß die am häufigsten benutzten Befehle mit 1-2 Bytes auskommen.

Mit dem 6809 steht eine CPU zur Verfügung, die einen höheren Datendurchsatz mit weniger Befehlsbytes zuläßt. Nach der immer notwendigen Einarbeitung, die auf Bekanntem aufbauen kann, wird man auch effektiver programmieren. Man sollte allerdings nicht ohne einen leistungsfähigen Assembler anfangen, denn einige weniger benutzte Instruktionen können 5 Byte lang sein.

Mit ihrer Ausstattung wird der 6809 höhere Leistungsbereiche abdecken und die Implementierung von Sprachen und den Aufbau von Mehrbenutzersystemen auf sich ziehen.

Seitens des Herausgebers sind ab etwa Januar 1981 6809-Programmierworkshops geplant, in die die Erfahrungen aus zahlreichen 6502-Workshops einfließen: Der programmierende Vortrag wird direkt auf die für die Teilnehmer aufgestellten Monitore übertragen. Thematik: Hardware (kurz), Vorstellung und Erprobung des Befehlssatzes, Assembler- und Interfaceprogrammierung. Das Interesse an einer Teilnahme kann schon jetzt bekundet werden.

Datenblätter zum 6809 bei den Distributoren von Motorola. Entwicklungssystem EXORset 30 mit Tastatur, Bildschirm, 2 Floppies, Betriebssystem, Assembler/Editor und BASIC-M ca. DM 13.000. Kleinere Systeme werden von der Fa. Dohmann in Gütersloh und von Eltec in Mainz geliefert.

R. L. *

****

BRANCHEN-NACHRICHTEN

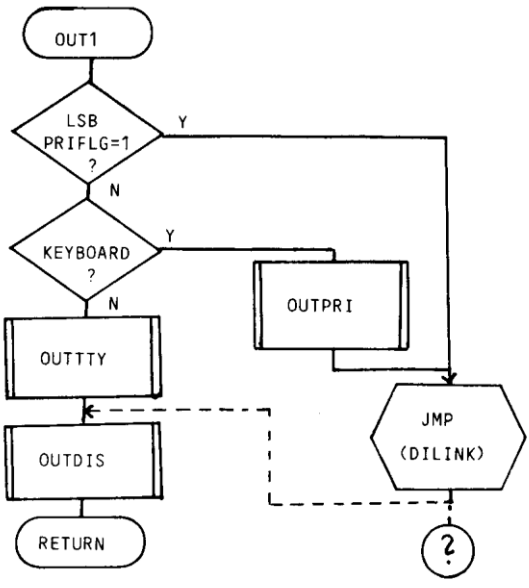
Der Alpha-Monitor für PET 2001 ist ein Zusatzprint mit 2 kROM-Programm, das auf den Expansionsbus des PET 2001 gesteckt wird (Adressenbereich von $B800 bis BFFF). Neben den Fähigkeiten des TIM-Monitors sind enthalten ein Disassembler, TRACE, Druckeraufruf, FIND Bytefolge oder String. Beziehbar bei Fa. PHS, Teichstr. 9, 3000 Hannover 91.

Der Pic-Chip ist ein ROM-Einschubmodul für CBM 3016/3032. Es sind etwa 40 grafische Befehle implementiert, die in das BASIC eingebunden wurden. Zum Lieferumfang gehört eine deutsche Betriebsanweisung von etwa 20 Seiten. Info: Ing.-Büro R. G. Houghton, Arabellastr. 58, 8 München 81, Tel. 089-91 46 28.

Ein- und Ausgabe am AIM 65 (2)

2.2. Die Ausgabe auf das LED-Display

Die Ausgabe auf die Systeminterfaces Thermodrucker, ggfs. TTY-Terminal und LED-Display wurde im Abschnitt 1.5 bei OUTALL bereits angesprochen. Die Programmverzweigung erfolgt in OUT1 ($E97B) dabei in Abhängigkeit vom Printerflag in $A411 und vom Schiebeschalter KB/TTY:



Man muß weiter erklären: Der im DILINK ($A406/07) hinterlegte Sprungvektor zeigt per Initialisierung im cold RESET auf das Unterprogramm OUTDIS.

Wenn im Printerflag das niederwertigste Bit=1 ist, erfolgt normalerweise die Ausgabe nur am LED-Display. OUTDIS wird aber als Schlußroutine auch nach OUTPRI (Thermaldrucker) oder nach OUTTTY erreicht (letzteres jedoch nicht für die Zeichen $0A und $FF, Line-Feed und Nullcharacter). - Auf die Ankoppelungsmöglichkeiten an das DILINK werden wir später eingehen.

OUT1 wird von OUTALL aus angesprungen, wenn keiner der alternativen Ausgabekanäle T, K, P, U oder X vom OUTFLG her angesprochen war. OUTALL hatte das auszugebende Zeichen eingangs auf dem Stack abgelegt. Aus diese Konstruktion folgt: Ein Zeichen kann unabhängig vom Inhalt des OUTFLG $A413 zwingend auf die aktiven Systemeinheiten ausgegeben werden, wenn man die Routine OUTPUT in $E97A aufruft. Hier wird (wie in OUTALL) das auszugebende Zeichen zunächst nur auf den Stack gerettet, der Folgebefehl ist schon die zuvor erwähnte Routine OUT1.

Als **Ausgabetreiber für die Anzeige** auf dem 20-stelligen Display dient die Routine OUTDD1 in $ EF7B. Mit jedem Durchlauf dieses Unterprogrammes wird ein Zeichen geschrieben. Dabei steuert der im Register X übergebene Wert (0 ≤ X ≤ $13) die Schreibstelle an (von links nach rechts. Das im Akku übergebene ASCII-Zeichen gelangt zur Anzeige. Dabei ist zu beachten, daß zu diesen Zeichen $80 logisch addiert werden müssen (ORA #$80), andernfalls gelangt das sog. Cursor-Zeichen (alle 16 Segmente eingeschaltet) zur Anzeige.

Beispiel 2.1.1: Anzeige des Zeichens 'S' in der 4. LED-Stelle.

```
        OUTDD1=$EF7B

        *=$200
        LDX #3          ANSTEUERUNG DER 4. ANZEIGESTELLE
        LDA #'S'        LADE ASCII 'S'
        ORA #$80        LOGISCHE ADDITION IN BIT 7
        JSR OUTDD1      ANZEIGE
WAIT    JMP WAIT        LEERE WARTESCHLEIFE
```

Aufgabe von OUTDD1 ist es mithin, den in X übergebenen Parameter in die Ansteuerung einer Zeichenstelle über den PIA-Port $AC00 umzusetzen. Aus der Beschaltung der Anzeigen ergeben sich folgende Steuerausgaben an Port A:

Stelle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Port $	7B	7A	79	78	77	76	75	74	6F	6E	6D	6C	5F	5E	5D	5C	3F	3E	3D	3C

Beispiel 2.2.2: Da die PIA-Ports beim Einschalten des AIM sowieso initialisiert sind, können wir z.B. den Buchstaben 'S' auch wie folgt in Stelle 0 direkt einschreiben:

```
        LDA #$7B        ANZEIGESTELLE 0
        STA $AC00       NACH PORT A
        LDA #$D3        ASCII 'S' +$80
        STA $AC02       DATENAUSGABE NACH PORT B
WAIT    JMP WAIT        WARTESCHLEIFE
```

Der oben besprochenen physischen Ausgabe ist im Monitorprogramm die Ausgaberoutine OUTDIS in $EF05 übergeordnet. Sie verwaltet einen Datenpuffer DIBUFF=$A438 von 60 Stellen Breite. Ab Adresse $A460 überlappt sich dieser mit dem Ausgabepuffer für den Thermodrucker. Als Läufer zur indizierten Adressierung über Register X ist dem DIBUFF die Zelle CURPO2 in $A415 zugeordnet.

Folgende Programm-Mechanik ist verwirklicht: Übergibt man ein 'CR' im Akku ($0D) und ruft man OUTDIS auf, so wird CURPO2=0 und das Display wird vollständig gelöscht. Ein von 'CR' verschiedenes Zeichen wird an der durch CURPO2 indizierten Stelle in DIBUFF abgelegt und zugleich auch auf dem LED-Display angezeigt.

Beispiel 2.2.3: Löschen des Display und Ausgabe eines Textes lt. Tabelle ab Stelle 15.

```
OUTDIS=$EF05
CURPO2=$A415
CR      =$D     CARRIAGE RETURN
```

```
        *=$400
        LDA #CR
        JSR OUTDIS    LOESCHE ANZEIGE
        LDA #14       ADRESSIERE AB STELLE 15
        STA CURPO2    ZUR ANZEIGEVERWALTUNG
        LDX #0        ZUM ZAEHLEN UND ADRESSIEREN
LOOP    LDA TAB,X     HOLE ANZUZEIGENDES ZEICHEN
        JSR OUTDIS    ANZEIGE IN STELLE
        INX
        CPX #4        SOLLZAHL DER ANZEIGEN ERREICHT?
        BNE LOOP      NEIN
WAIT    JMP WAIT      WARTESCHLEIFE

TAB     .BYT 'TEXTAUSGABE AUF DISPLAY'
```

Im Ergebnis sehen wir, daß das kleine Programm die vier Buchstaben 'TEXT' ab adressierter Stelle rechts in das Display schreibt. Was geschieht nun, wenn wir ab Stelle 15 noch mehr Text ausgeben wollen? Wir ändern an entsprechender Stelle auf CPX $14 (dezimal 20) ab. Nun erscheinen 20 Zeichen ab äußerster linker Stelle auf dem Display. Programmieren wir mit CPX $17, so erscheint nur noch der Schwanz der Textkette in der Anzeige, es wurde gerollt.

Damit ist die Mechanik von OUTDIS aufgezeigt:

a) Zu den ASCII-Zeichen werden automatisch $80 logisch addiert, um ein ausgabefähiges Zeichen zu erzeugen (keinen Cursor).

b) Ein 'CR' löscht das Display insgesamt.

c) Mit jedem ausgegebenen Zeichen wird CURPO2 als Läufer automatisch erhöht.

d) Wenn CURPO2 20 (hex $14), dann wird der Inhalt der Anzeige mit jeder weiteren Einschreibung nach links gerollt. Das besorgt das Unterprogramm OUTD2A.

e) Sobald CURPO2=60 ($3C), sind die weitere Abspeicherung von Zeichen in DIBUFF und das Rollen der Anzeige unterbunden.

f) DIBUFF dient als Textausgabepuffer, über den ein 20 Stellen breites Fenster für die Ausgabe auf das Display gelegt ist.

g) DIBUFF dient zugleich als Tastaturpuffer, wie wir im Zusammenhang mit der Texteingabe im Editor noch sehen werden.

h) OUTDIS gibt die Inhalte von A, X und Y unversehrt zurück.

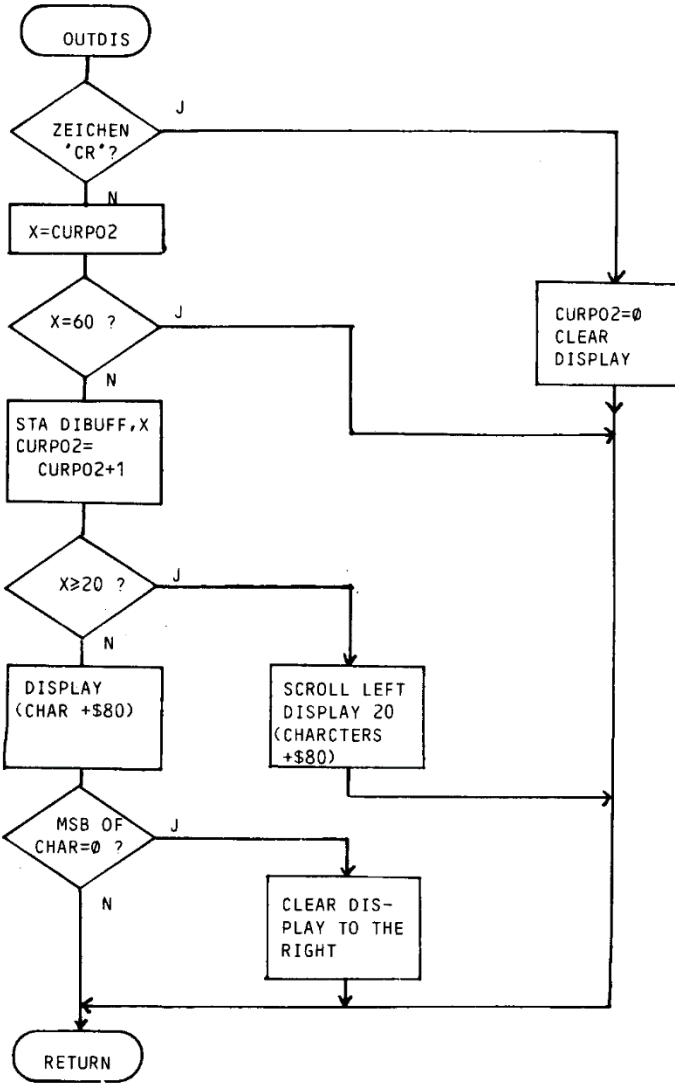
Punkt b) muß wie folgt ergänzt werden: Das Zeichen $8D setzt CURPO2 auf 0 und damit die nachfolgenden Zeichen wieder linksbündig in die Anzeige. Ein Versuch ist leicht ausgeführt.

Für OUTDIS gilt damit der auf der folgenden Seite abgedruckte Programmablaufplan.

OUTDIS enthält eine weitere Mechanik: Ein von 'CR' verschiedenes Zeichen, das kleiner $80 ist, löscht das Display automatisch in allen Stellen rechts von der Schreibstelle:

```
LDA #14       SCHREIBSTELLE 15
STA CURPO2
LDA #'L       SCHREIBE 'L'
JSR OUTDIS    ANZEIGE
LDA #8        NUN SCHREIBSTELLE 9
              STA CURPO2
              LDA #'R         SCHREIBE 'R' LINKS VOM 'L'
              JSR OUTDIS
WAIT          JMP WAIT        WARTESCHLEIFE
```

Programmablaufplan für OUTDIS



Mit dem Programm auf der Vorseite wird das zunächst geschriebene 'L' wieder gelöscht. Programmieren wir aber statt LDA #'R' ein LDA #$D2, so wird das weiter hinten stehende 'L' nicht gelöscht.

Bei der interaktiven Ausgabe von Texten möchte man nicht jedesmal für die unterschiedliche Länge des Textes besonders programmieren, wie noch in Beispiel 2.2.3. Im AIM-Handbuch sind Beispiele enthalten, wie man mit einem Begrenzer-Byte für den Text arbeitet, z.B. mit einem ';'. Das ist nicht die kürzeste und speichersparende Art des Programmierens. Vor der Abrasterung von ASCII-codierten Texten sollte man im letzten Byte vielmehr das Bit 7 als Begrenzer setzen (ORA #$80):

Beispiel 2.2.4: Ausgabe von Text mit Bit 7=1 im letzten Byte.

```
        LDA #CR         CLEAR DISPLAY
        JSR OUTDIS
        LDX #$FF
LOOP    INX             X=Ø ON THE FIRST INSTANCE
        LDA TEXT,X      GET CHAR. TO BE DISPLAYED
        JSR OUTDIS
        BPL LOOP        CHAR. WAS JUST RETURNED WITH PLA
WAIT    JMP WAIT
TEXT    .BYT 'BEISPIE',$CC      CC='L'+$80
```

Der AIM-Monitor enthält mit KEP in $E7AF ein sehr ähnliches Unterprogramm. Es schreibt Meldungen aus einer Tabelle des ROM. Die Auswahl der Meldung erfolgt durch Parameterübergabe in Y. Nach diesem nützlichen Prinzip kann man Meldungen dicht aneinanderreihen und das Unterprogramm allgemein nutzbar machen. Bei sehr großen Texttabellen wird man indirekt Y über eine Pointerzelle adressieren und in der Initialisierung nicht nur das Y-Register, sondern auch die Pointerbasis entsprechend setzen. Dafür folgt weiter unten ein Beispiel.

Zur Löschung des Displays: In vorstehenden Beispielen haben wir das Display mit LDA #'CR', JSR OUTDIS gelöscht, um für Anwenderprogramme die volle Sequenz darzustellen. Wenn wir vom Monitor mit dem G-Kommando (GO) herkommen, ist das nicht nötig, das Gleiche wird auch dort besorgt.

Die Routine OUTDIS enthält für viele Anwendungen (in denen der Text nicht durch DELETE verkürzt wird) eine unnötig zeitraubende Redundanz: Jedes normale ACSII-Zeichen löscht per Durchlauf von OUTD5 alle Anzeigestellen hinter der eigentlichen Schreibstelle. Wir unterbinden das in der folgenden Abänderung gegenüber Beispiel 2.2.4:

Beispiel 2.2.5: Beschleunigte Anzeige (ohne DELETE).

```
        LDX #$FF
LOOP    INX             X=Ø ON THE FIRST INSTANCE
        LDA TEXT,X
        EOR #$80        NO ERASE TO END OF LINE
        JSR OUTDIS
        BMI LOOP
WAIT    JMP WAIT
```

Hier wird aus dem letzten Zeichen $CC durch EOR #$80 ein 'L', das den Rest der Anzeige löscht.

Unter dieser Mechanik brauchte bei der Textausgabe ja eigentlich nur das erste Zeichen 'mit der Löschfunktion für die Folgestellen' gesendet zu werden, und bei Texteingaben der Cursor, sofern zuvor die DELETE-Taste betätigt wurde. Dieser Hinweis mag für zeitkritische Anwendungen nützlich sein, wenn z.B. für schnelle Magnetbandformate Blocknummern etc. ausgeschrieben werden sollen.

Im Beispiel 2.2.3 haben wir gesehen, daß das Ausgabefenster nach 'rechts' verschoben wird (scroll), sobald die Eingabe in die 21. Stelle des DIBUFF gelangt. Aber auch das Zurücksetzen der Schreibstelle bzw. das Zurückrollen einer Textzeile um eine Zeichenbreite nach Betätigen der DELETE-Taste ist implementiert, und zwar in der Routine RDRUB in $E95F. Sie wird im Monitor in allen Fällen benutzt, in denen eine Korrektur der Eingabe möglich sein soll, insbesondere auch vom Editor unter Tastatureingabe. Das eigentlich zurücksetzende Unterprogramm ist PSLS in $E7DC.

Die bisherigen Überlegungen zur Display-Ausgabe sollen in einem Demonstrationsprogramm zur Ausgabe beliebiger Texte aus einer großen Texttabelle zusammengefaßt werden. Der Ansatz der Monitor-Routine KEP reicht dann nicht mehr aus, wenn die Texttabelle größer als eine Speicherseite (page) ist. Bei der nachfolgenden Programmierung fällt etwas schmerzlich auf, daß der AIM-Assembler keine 'conditional assembly' kennt, keine Möglichkeit, die Programmerzeugung von der Bewertung von Assembler-Ausdrücken zur Zeit der Assemblierung abhängig zu machen.

Sobald nämlich im Beispiel 2.2.6 der Abstand zwischen den Anfangsstellen der Texte, z.B. TF-T0, größer als $FF wird, muß im Programm unter MES1 der Befehl INC PNTL+1 gegeben werden (Erhöhung der Page-Adresse im Pointer), andererseits muß der Offset unter TF-T0 zu TF-T0-$100 definiert werden. Der Assembler des AIM macht zum Trost allerdings mit Error 14 (Address not valid) darauf aufmerksam. - Bei Texttabellen von mehr als 2 Speicherseiten muß man mit weiteren Korrekturen arbeiten (z.B. TF-T0-$200 und 2 Abfragen CPX #..., verbunden mit INC PNTL+1).

Der Lösungsansatz in 2.2.6 ist gleichwohl allgemein und kann für jede 6502-Maschine implementiert werden, wenn man statt OUTDIS eine andere Ausgaberoutine verwendet, z.B. OUTPUT oder auch OUTALL beim AIM. Statt der Folge LDY #0 und STY CURPO2 wird man ein CR/LF auf die Ausgabe schreiben (Routine CRCK), das EOR #%10000000 fortlassen und die bei OUT folgenden BMI/BPL miteinander vertauschen. Und bei der empfehlenswerten Anlage als Unterprogramm ab MES1 wird man bei OUT ein RTS schreiben.

Mit der nachfolgenden Routine läßt sich mühelos auch gesperrte Schrift erzielen, wenn man am Label SPERR folgende Befehle einfügt:

```
LDA #$D     lade ein 'CR'
JSR OUTDIS.
```

In der Fortsetzung werden wir Routinen des Thermodruckers behandeln.

Beispiel 2.2.6: Ausgabe von Texten variabler Länge aus großen Tabellen. Im jeweils letzten Textbyte ist Bit 7 als Begrenzer gesetzt. Texte werden von der Tastatur her aufgerufen, dabei sind nur die Tasten 0-9 und A-F zugelassen. Die zugehörigen Texte werden über Pointeradresse und indirekte Indizierung mit Y angesteuert.

```
0000          TEXTAUSGABE AUS TABELLE
0000
0000          ;SYMBOLE
0000 PNTL=$0
0000                                ;BASISPOINTER
0000 RCHEK=$E907
0000                                ;TASTATUR LESEN, OGFS ESCAPE
```

```
0000 OUTDIS=$EF05
0000 CRLF=$E9F0
0000 OUTPUT=$E97A
0000 HEX=$EA7D
0000                                    ;CONVERSION ASCII TO HEX
0000 CURPO2=$A415
0000                                    ;DISPLAY-POINTER
0000                    ---------------------
0000                    MAIN PGM
0000                          *=$900
0900 START
0900                    A94B  LDA #<T0          ;INITIALIZE POINTER
0902                    8500  STA PNTL
0904                    A909  LDA #>T0
0906                    8501  STA PNTL+1
0908 KEY
0908                    2007E9 JSR RCHEK        ;GET CHAR FROM KB
090B                    48    PHA               ;SAVE
090C                    207DEA JSR HEX          ;CONVERT CHAR
090F                    AA    TAX               ;SAVE ADDRESSER
0910                    68    PLA
0911                    B0F5  BCS KEY           ;CHAR WAS NON-HEX
0913                    A000  LDY #0
0915                    8C15A4 STY CURPO2
0918                    2005EF JSR OUTDIS       ;WRITE KEY/CHAR TO DISPLAY
091B                    A9A0  LDA #$A0          ;WRITE A SPACE TO DISPLAY
091D                    2005EF JSR OUTDIS
0920 MES1                                       ;MAIN LOOP TO DISPLAY TEXT
0920                    E00F  CPX #$F           ;SEE IF WE SURPASSED PAGE
0922                    9002  BCC **+4
0924                    E601  INC PNTL+1        ;YES
0926                    BC3B09 LDY OFFSET,X
0929 MES2
0929                    B100  LDA (PNTL),Y
092B                    4980  EOR #%10000000    ;DO NOT ERASE TO END OF DISPLAY
092D                    2005EF JSR OUTDIS
0930                    08    PHP               ;SAVE BIT7 CONDITION
0931                    C8    INY
0932                    D002  BNE **+4
0934                    E600  INC PNTL
0936                    28    PLP
0937                    30F0  BMI MES2          ;CHAR WAS WITHOUT DELIMITER
0939                    10C5  BPL START         ;GET NEXT INPUT & DISPLAY
093B                    ---------------------
093B                    ADDRESSES OF ITEMS RELATIVE TO T0
093B OFFSET
093B                    00      .BYT 0,T1-T0,T2-T0,T3-T0,T4-T0,T5-T0,T6-T0,T7-T0
093C                    10
093D                    20
093E                    30
093F                    40
0940                    4E
0941                    58
0942                    5E
0942                    64      .BYT T8-T0,T9-T0,TA-T0,TB-T0,TC-T0,TD-T0,TE-T0,TF-T0-$100
0944                    68
0945                    6C
```

```
0946            73
0947            78
0948            7F
0949            84
094A            19
094B            ____________________
094B            MESSAGES DELIMITED BY BIT7 =1
094B T0
094B            4D45    .BYT 'MESSAGE ON KEY=',$B0
095A            B0
095B T1
095B            4D45    .BYT 'MESSAGE ON KEY=',$B1
096A            B1
096B T2
096B            4D45    .BYT 'MESSAGE ON KEY=',$B2
097A            B2
097B T3
097B            4D45    .BYT 'MESSAGE ON KEY=',$B3
098A            B3
098B T4         4441    .BYT 'DAS WAR TASTE',$B4
0998            B4
0999 T5
0999            4155    .BYT 'AUF TASTE',$B5
09A2            B5
09A3 T6
09A3            5441    .BYT 'TASTE',$B6
09A8            B6
09A9 T7
09A9            5349    .BYT 'SIEBE',$CE
09AE            CE
09AF T8         414348  .BYT 'ACH',$D4
09B2            D4
09B3 T9         4E4555  .BYT 'NEU',$CE
09B6            CE
09B7 TA         5441    .BYT 'TASTE ',$C1
09BD            C1
09BE TB         554E    .BYT 'UND ',$C2
09C2            C2
09C3 TC         432D    .BYT 'C-TAST',$C5
09C9            C5
09CA TD         5741    .BYT 'WAR ',$C4
09CE            C4
09CF TE
09CF            4549    .BYT 'EIN ',$C5
09D3            C5
09D4                    *=*+$90             TF-T0 NOW >$FF,
0A64 TF                                     OFFSET > 1 PAGE
0A64            4549    .BYT 'EIN ',$C5
0A68            C6
0A69                    .END
```

[WIRD FORTGESETZT]

* R.L.

****

KLEINANZEIGEN DER LESER

EPROM-Schießer für CBM, (s. Artikel P. Trübger, 65xx MICRO MAG 11), komplett mit Steckern für CBM, Software auf Cassette oder Disk, Anleitung (ohne Gehäuse) DM 300,- + MWSt. David Long, Kl. Pfahlstr. 19a, 3000 Hannover. Tel. 0511/31 90 19

Dipl.-Math. A. Quindt, Wiesbaden

Programmveränderung

Das Programm läuft auf dem cbm 3001 und verwendet die Floppy-Disk 3040 (Geräte-Nummer 8).

Beim Programmieren taucht gelegentlich der Wunsch auf, eine Routine aus einem Programm auszulagern und in ein anderes Programm einzubinden. Dies war von Interesse bei Routinen für die Dateiverwaltung, Fehlerkanalabfrage, Verwaltung relativer Dateien, binäres Suchen usw.. Eine solche Möglichkeit ist im cbm 3001 nicht vorgesehen. Es gab Tricks, dies über den Cassettenrekorder zu lösen. Es war aber zu zeitaufwendig und zu umständlich. Ein solcher Weg mußte auch über Floppy-Disk und erheblich schneller möglich sein. Beim Weg über Kassettenrekorder mußten die Programme über CMD und LIST auf Kassette geschrieben werden. Das ist recht umständlich. Diesen Weg kann man natürlichauch über Floppy-Disk gehen, nur gilt hier die gleiche Kritik.

Bei der Floppy-Disk kann man unmittelbar auf die Programme zugreifen wenn man angibt:

```
        OPEN 10,8,10,"0: ....,P,R"        (Lesen)
oder    OPEN 10,8,11,"0: ....,P,W"        (Schreiben).
```

Dabei sind 10 und 11 frei gewählte logische File-Nummern bzw. Sekundäradressen. Das erste und zweite Byte eines solchen Files enthält die Startadresse (lo, hi). - Im Fall der Erzeugung eines BASIC-Programmes auf diesem Weg muß also $01 und $04 übergeben werden. Die Programme werden danach in der gleichen Form abgespeichert, wie sie im Computer gespeichert sind. Also:

> Das 1. und 2. Byte einer Programmzeile enthalten die Chaining-Adresse des Beginns der nächsten Zeile.
>
> Byte 3 und 4: Zeilennummer (lo/hi).
>
> Folgebytes: Befehl verschlüsselt und Text in ASCII.
>
> Letztes Byte einer Zeile: $00.

Danach folgt die nächste Zeile. Als Markierung für das Ende des Programmes wird schließlich $0000 als Chainining vom BASIC gesetzt.

Mit diesem Wissen kann man nun die gewünschten Veränderungen vornehmen. Man kann Programmzeilen lesen und die entsprechenden Zeilen unter neuem Namen auf der Diskette speichern. Man beginnt beim Schreiben mit dem OPEN-Befehl, übergibt danach $0104 (Startadresse), die gewünschten Zeilen, als Endmarkierung $0000 und schließt mit CLOSE das File. Dabei werden allerdings die Bytes 1 und 2 einer Zeile (Chaining) falsch übergeben, weil sie Folgeadressen aus dem Bereich des alten Programms enthalten. Da aber beim Einlesen des Programms mit LOAD diese überprüft und korrigiert werden, gibt es dabei keine Schwierigkeiten. (Ausnahme: Lädt man ein solches bearbeitetes Programm ein und überprüft den Inhalt mit VERIFY, so erhält man natürlich keine Übereinstimmung!) Diese so ausgelagerte Routine kann man nun in ein anderes Programm einbauen. Dies geschieht auf völlig analogem Weg. Da maximal 6 Files gleichzeitig eröffnet werden dürfen (Beschränkung der Floppy), ergibt sich:

1 File für Fehlerkanal,
1 File für entstehendes Programm.

Maximal hat man also 4 Files für zu mischende Programme. - Nach der Größe der Zeilennummern kann man bis zu 4 Programme zu einem neuen Programm zusammenmischen. Analog kann man auch Zeilen in einem Programm zerstören, indem man die Zeilen einliest und die unerwünschten nicht wieder ausschreibt.

Nachteilig an diesem Programm war der Zeitbedarf. Zwar war das Programm schneller und flexibler als die entsprechenden Kassettenrekorder-Lösungen, aber es sollte noch schneller sein. - Die Programmzeilen wurden über GET # eingelesen. Und hier lag das Problem. Der INPUT # -Befehl konnte nicht verwendet werden, da er eine Programmzeile nicht korrekt einliest, sondern z.B. bei einem Komma stoppt.

Ein Ausweg lag in einer entsprechenden Assembler-Routine. Diese mußte:

1. die Floppy über den IEC-Bus ansteuern, die entsprechende Sekundäradresse übergeben,
2. 2 Bytes holen und abspeichern und auf $0000 als Ende des Programmes prüfen (Chaining),
3. 2 weitere Bytes holen und abspeichern (Zeilennummern),
4. weitere Bytes holen, abspeichern, bis $00=Zeilenende,
5. den IEC-Bus wieder freigeben und
6. die gespeicherte Information dem BASIC-Programm zur Verfügung stellen.

Durch die Zeile 22100 wird die gewünschte Sekundäradresse dem Assemblerprogramm übergeben. Die Zeile 22300 speichert den String um (analog einer Anweisung +"") und rettet somit den String vor Überschreiben beim nächsten Aufruf. Die Zeile 22500 bewirkt eine Wiederholung des Lesevorganges, falls der Aufruf der Floppy über den IEC-Bus zu einer Fehlermeldung (Statusvariable ungleich 0) führte. - Das Programm arbeitete jetzt mit zufriedenstellender Geschwindigkeit:

```
100 REM*****************************
120 REM**                           **
140 REM** PROGRAMM VERAENDERUNG     **
160 REM**                           **
180 REM**                           **
200 REM** PROGRAMM VON              **
220 REM** AUGUST QUINT              **
240 REM** HUENEFELDSTR. 19          **
260 REM** 6200 WI-ERBENHEIM         **
280 REM** TEL. 06121/711464         **
300 REM**                           **
320 REM*****************************
400 CLR:AA$=""
450 READI1,I2
470 FORI=I1TOI2:READJ:POKEI,J:NEXT
500 OPEN1,8,15
520 X$=CHR$(0):Y$=X$+X$:Z$=CHR$(1)+CHR$(4):X1$=CHR$(141):X2$=CHR$(137)
(167)
540 X4$=CHR$(58)                                          X3$=CHR$
560 ZZ$=CHR$(A1)+CHR$(A2/2)
600 DIMA$(3),AX(3),D(4)
900 GOTO40000
1000 PRINT"■■■  C O P Y   M E R G E  2 - 4          "
1010 PRINT"■■BITTE LEGEN SIE DIE DISKETTE(N) EIN.
```

```
 1140 PRINT"■■■BITTE GEBEN SIE DIE PROGRAMMNAMEN EIN."
 1150 PRINT"■BEACHTEN SIE :"
 1155 PRINT"BEI GLEICHEN ZEILENNUMMERN WIRD DIE     ZEILE AUS DEM ZUERST"
 1160 PRINT"EINGEGEBENEN PROGRAMM VERWENDET UND DIE ANDEREN ZEILEN WERDEN
GELOESCHT"
 1200 PRINT"■■■BITTE GEBEN SIE DIE ANZAHL DER ZU       KOPIERENDEN PROGRAM
ME AN :"
 1300 INPUTJ%:IFJ%<1ORJ%>4THENPRINT"□";:GOTO1300
 1400 FORI=0TOJ%-1
 1500 PRINT"■";I+1;".PROGRAMM :"
 1600 INPUT"NAME ";A$(I)
 1650 INPUT"LAUFWERK ";D(I):IFD(I)<>0ANDD(I)<>1THENPRINT"□";:GOTO1650
 1700 A%(I)=1:NEXT
 1800 PRINT"■■■NAME DES ZU BILDENDEN PROGRAMMS :"
 1900 INPUTB$
 1950 INPUT"LAUFWERK ";D(4):IFD(4)<>0ANDD(4)<>1THENPRINT"□";:GOTO1950
 1960 PRINT#1,"I"+STR$(D(4))
 1970 FORI=0TO3:IFA%(I)<>0THENIFD(I)<>D(4)THEN1990
 1980 NEXT:GOTO2000
 1990 PRINT#1,"I"+STR$(D(I))
 2000 GOSUB20000
 2050 FORI=0TO3:IFA%(I)=0THEN2300
 2100 GOSUB21000:GOSUB22000
 2200 C$(I)=A1$:NEXT
 2300 OPEN10,8,10,STR$(D(4))+":"+B$+",P,W"
 2400 GOSUB20000
 2500 PRINT#10,Z$;
 2600 FORI=0TO3
 2700 IFA%(I)=0THEN2900
 2800 B(I)=ASC(MID$(C$(I),3,1))+256*ASC(MID$(C$(I),4,1))
 2900 NEXT
 3000 B=67000:J=15
 3100 FORI=0TO3
 3200 IFA%(I)=0THEN3400
 3250 IFB=B(I)THENC$(I)=""
 3300 IFB>B(I)THENB=B(I):J=I
 3400 NEXT
 3500 PRINT#10,C$(J);
 3600 C$(J)=""
 3800 FORI=0TO3
 3900 IFA%(I)=0THEN4300
 4000 IFLEN(C$(I))>0THEN4300
 4100 GOSUB22000
 4150 C$(I)=A1$
 4300 NEXT
 4350 FORI=0TO3
 4400 IFA%(I)<>0THEN2600
 4450 NEXT
 4500 PRINT#10,Y$;
 4600 CLOSE10
 4700 GOSUB20000
 4800 GOTO50000
 5000 PRINT"□■    D E L E T E                    "
 5100 PRINT"■MIT DIESEM PROGRAMM WERDEN IN EINEM"
 5200 PRINT"PROGRAMM ZEILEN ZERSTOERT.■"
 5300 PRINT"BEACHTEN SIE :"
 5400 PRINT"WENN DAS NEUE PROGRAMM DEN GLEICHEN"
 5500 PRINT"NAMEN WIE DAS URSPRUENGLICHE PROGRAMM"
```

```
5600 PRINT"HAT UND AUSSERDEM AUF DER GLEICHEN"
5700 PRINT"DISKETTE SEIN SOLL WIRD DAS"
5800 PRINT"URSPRUENGLICHE PROGRAMM UEBERSCHRIEBEN!■■
6000 INPUT"NAME DES PROGRAMMS ";NA$
6050 INPUT"■LAUFWERK ";D(0):IFD(0)<>0ANDD(0)<>1THENPRINT"■■";:GOTO6050
6100 PRINT"■ZERSTOEREN DER ZEILEN:"
6200 INPUT"VON ";B1
6300 INPUT"BIS ";B2
6400 IFB2<B1THEN5000
6500 INPUT"■NAME DES NEUEN PROGRAMMS ";NB$
6600 INPUT"■LAUFWERK ";D(1):IFD(1)<>0ANDD(1)<>1THENPRINT"■■";:GOTO6600
6650 PRINT#1,"I"+STR$(D(0)):GOSUB20000
6670 IFD(0)<>D(1)THENPRINT#1,"I"+STR$(D(1)):GOSUB20000
6700 I=0:A$(I)=NA$
6800 GOSUB21000
6850 D$=STR$(D(1))
6900 IFNA$=NB$ANDD(0)=D(1)THEND$="@"+D$
7000 OPEN10,8,10,D$+":"+NB$+",P,W":GOSUB20000
7100 PRINT#10,Z$;
7200 GOSUB22000
7250 IFLEN(A1$)<5THEN7700
7300 B=ASC(MID$(A1$,3,1))+256*ASC(MID$(A1$,4,1))
7400 IFB>=B1ANDB<=B2THEN7600
7500 PRINT#10,A1$;
7600 GOTO7200
7700 PRINT#10,Y$;:CLOSE10:GOSUB20000
7800 GOTO50000
8000 PRINT"■■     U N L O A D                              "
8100 PRINT"■■MIT DIESEM PROGRAMM KOENNEN TEILE EINES
8200 PRINT"PROGRAMMS AUSGELAGERT WERDEN, OHNE DAS
8300 PRINT"URSPRUENGLICHE PROGRAMM VERAENDERN ZU   MUESSEN.■"
8400 PRINT"BEACHTEN SIE :"
8500 PRINT"WENN DAS NEUE PROGRAMM DEN GLEICHEN"
8600 PRINT"NAMEN WIE DAS URSPRUENGLICHE PROGRAMM"
8700 PRINT"HAT UND AUSSERDEM AUF DER GLEICHEN"
8800 PRINT"DISKETTE SEIN SOLL WIRD DAS"
8900 PRINT"URSPRUENGLICHE PROGRAMM UEBERSCHRIEBEN!■■
9000 INPUT"NAME DES ALTEN PROGRAMMS ";A$(0)
9050 INPUT"■LAUFWERK ";D(0):IFD(0)<>0ANDD(0)<>1THENPRINT"■■";:GOTO9050
9100 INPUT"■NAME DES NEUEN PROGRAMMS ";A$(1)
9200 INPUT"■LAUFWERK ";D(1):IFD(1)<>0ANDD(1)<>1THENPRINT"■■";:GOTO9200
9300 PRINT"■AUSZULAGERNDE ZEILEN :"
9400 INPUT"VON ";B1
9500 INPUT"BIS ";B2
9600 IFB1>B2THEN8000
9700 PRINT#1,"I"+STR$(D(0)):GOSUB20000
9800 IFD(0)<>D(1)THENPRINT#1,"I"+STR$(D(1)):GOSUB20000
9900 I=0:GOSUB21000
10000 D$=STR$(D(1)):IFD(0)=D(1)ANDA$(0)=A$(1)THEND$="@"+D$
10100 OPEN10,8,10,D$+":"+A$(1)+",P,W":GOSUB20000
10200 PRINT#10,Z$;
10300 IFB=B2THEN10800
10350 GOSUB22000
10400 IFLEN(A1$)<5THEN10800
10500 B=ASC(MID$(A1$,3,1))+256*ASC(MID$(A1$,4,1))
10550 IFB<B1THEN10300
10600 IFB>B2THEN10800
10700 PRINT#10,A1$;:GOTO10300
```

```
10800 PRINT#10,Y$;:CLOSE10:GOSUB20000:CLOSE5:GOSUB20000
10900 GOTO50000
20000 INPUT#1,I1$,I2$,I3$,I4$
20010 IFVAL(I1$)=0THENRETURN
20020 PRINT"■";I1$;",";I2$;",";I3$;",";I4$
20030 STOP:RETURN
21000 D=5+I
21100 OPEND,8,D,STR$(D(I))+":"+A$(I)+",P,R"
21200 GOSUB20000
21300 GET#D,I$:IFI$=""THENI$=X$
21400 GET#D,J$:IFJ$=""THENJ$=X$
21500 IFASC(I$)+256*ASC(J$)=1025THENRETURN
21600 PRINT"■KEIN BASIC PROGRAMM : ";A$(I)
21700 STOP
22000 D=5+I:A1$=""
22100 POKE647,(96+D)
22200 SYS634
22300 A1$=MID$(" "+AA$,2)
22400 IFLEFT$(A1$,2)=Y$THEN22800
22500 IFST=0THEN22700
22600 POKE150,0:GOTO22000
22700 RETURN
22800 CLOSED
22900 GOSUB20000
23000 A%(I)=0:RETURN

40000 PRINT"□■   V  E  R  A  E  N  D  E  R  U  N  G   "
40100 PRINT"■■■■DIESES PROGRAMM SOLL IHNEN HELFEN"
40200 PRINT"BASIC-PROGRAMME, DIE AUF EINER DISKETTE
40300 PRINT"GESPEICHERT SIND ZU VERAENDERN."
40400 PRINT"ES GIBT FOLGENDE MOEGLICHKEITEN:■■"
40500 PRINT"     ■C■OPY MERGE 2-4  ■
40600 PRINT"     ■U■NLOAD    ■
40700 PRINT"     ■D■ELETE    ■
40900 PRINT"■WAS WOLLEN SIE ?"
41000 GETA$
41100 IFA$="C"THEN1000
41200 IFA$="U"THEN3000
41300 IFA$="D"THEN5000
41500 GOTO41000
50000 PRINT"■■■"I1$","I2$","I3$","I4$
50100 CLOSE1:END
60000 DATA 634, 850
60010 DATA 169,1,141,237,2,169,8,133,212,32,182,240,169,106,133,211
60020 DATA 32,40,241,32,140,241,141,240,2,165,150,208,60,32,140,241
60030 DATA 141,241,2,208,5,173,240,2,240,47,24,173,237,2,105,3
60040 DATA 141,237,2,32,140,241,141,242,2,32,140,241,141,243,2,32
60050 DATA 140,241,201,0,240,13,174,237,2,157,240,2,232,142,237,2
60060 DATA 76,185,2,174,237,2,157,240,2,32,127,241,160,2,24,173
60070 DATA 237,2,105,1,145,42,200,173,238,2,145,42,200,173,239,2
60080 DATA 145,42,96,0,240,2,255,0,255,0,255,0,255,0,255,0
60090 DATA 255,0,251,0,255,0,255,0,255,0,255,0,255,0,255,0
60100 DATA 255,0,255,0,255,0,255,0,255,0,255,0,255,0,255,0
60110 DATA 255,0,255,0,255,0,255,0,255,0,255,0,255,0,255,0
60120 DATA 255,0,255,0,255,0,255,0,255,0,255,0,255,0,255,0
60130 DATA 255,0,255,0,255,0,255,0,255,0,155,0,255,0,255,0
60140 DATA 255,0,255,0,255,0,255,0,255,0,155,0,255,0,255,0
READY.
```

LISTUNG DES IN DEN DATA-STATEMENTS ENTHALTENEN MASCHINENPROGRAMMES

```
027A  A9 01     LDA  #01
027C  8D ED 02  STA   02ED
027F  A9 08     LDA  #08
0281  85 D4     STA   D4
0283  20 B6 F0  JSR   F0B6
0286  A9 6A     LDA  #6A
0288  85 D3     STA   D3
028A  20 28 F1  JSR   F128
028D  20 8C F1  JSR   F18C
0290  8D F0 02  STA   02F0
0293  A5 96     LDA   96
0295  D0 3C     BNE   02D3
0297  20 8C F1  JSR   F18C
029A  8D F1 02  STA   02F1
029D  D0 05     BNE   02A4
029F  AD F0 02  LDA   02F0
02A2  F0 2F     BEQ   02D3
02A4  18        CLC
02A5  AD ED 02  LDA   02ED
02A8  69 03     ADC  #03
02AA  8D ED 02  STA   02ED
02AD  20 8C F1  JSR   F18C
02B0  8D F2 02  STA   02F2
02B3  20 8C F1  JSR   F18C
02B6  8D F3 02  STA   02F3
02B9  20 8C F1  JSR   F18C
02BC  C9 00     CMP  #00
02BE  F0 0D     BEQ   02CD
02C0  AE ED 02  LDX   02ED
02C3  9D F0 02  STA   02F0,X
02C6  E8        INX
02C7  8E ED 02  STX   02ED
02CA  4C B9 02  JMP   02B9
02CD  AE ED 02  LDX   02ED
02D0  9D F0 02  STA   02F0,X
02D3  20 7F F1  JSR   F17F
02D6  A0 02     LDY  #02
02D8  18        CLC
02D9  AD ED 02  LDA   02ED
02DC  69 01     ADC  #01
02DE  91 2A     STA  (2A),Y
02E0  C8        INY
02E1  AD EE 02  LDA   02EE
02E4  91 2A     STA  (2A),Y
02E6  C8        INY
02E7  AD EF 02  LDA   02EF
02EA  91 2A     STA  (2A),Y
02EC  60        RTS
02ED  00
02EE  F0 02
```

Pointer auf Anzahl übermittelter Bytes stellen

IEC-Bus Ansteuerung Floppy-Disk Sekundäradr. 10

1 Byte holen und abspeichern

Statusvariable überprüfen

1 Byte holen und abspeichern

Prüfen auf $0000 = Ende des Programms

Zeilennummer holen und abspeichern

1 Byte holen Prüfen auf Ende der Zeile

Byte abspeichern Zähler erhöhen und abspeichern

letztes Byte abspeichern IEC-Bus freigeben Länge des Strings abspeichern in Variable AA$

Pointer auf Beginn des Strings abspeichern in Variable AA$ (definiert als 1. Variable)

Zähler für Anzahl überm. Bytes

Adresse des Beginns der Speicherung der übermittelten Bytes

*

Johann Leitner, Kurfürstenstraße 2, 6000 Frankfurt 90

Zeitanzeige für PET und CBM

Die Lieferung meines cbm 8032 enthielt auch einen RAM-ROM-Test mit Zeitanzeige. Dies gab Anlaß zur Entwicklung des folgenden Programms. Es dient zum Einblenden der laufenden Zeit in den Bildschirm und ist als Teil der Interrupt-Routine konzipiert. Daher wird die "normale" Betriebsart des Automaten nicht beeinträchtigt. BASIC und die direkten Operationen werden lediglich etwas verlangsamt (ca. 0,25%).

Mit POKE 634,0 kann die Anzeige abgeschaltet werden, ohne daß die Zeitinformation verloren geht. Zum Stellen der Uhr ein Programmbeispiel in BASIC:

```
10 PRINTCHR$(19)CHR$(19)CHR$(147)CHR$(17)CHR$(15)
20 AB=634:INPUT"ZEIT (HHMMSS)";Z$:FORI=1TO3
30 POKEAB+I,VAL(MID$(Z$,I*2-1,1))+16*VAL(MID$(Z$,I*2,1)
40 NEXT:POKEAB+4,0
```

(In Zeile 10 wird der Befehl SET TOP - PRINTCHR$(15) - verwendet. Die gegenwärtige Kursorposition wird damit als zukünftige HOME-Position definiert.)

Das Assemblerprogramm wurde in den Pufferbereich des 1. Rekorders gelegt, weil der Puffer für den Rekorder 2 beim cbm 8032 vom Betriebssystem benutzt wird (u.a. für TAB und gewisse Diskoperationen).

Der Befehl CLD in Zeile 28 ist notwendig, weil die Interrupt-Routine des Betriebssystems (unkorrekterweise) die decimal flag als gelöscht voraussetzt.

Das Programm kann relativ einfach zur Alarmuhr ausgebaut werden. Man kann sich dann vom Computer wecken lassen, wenn man beim Programmieren eingeschlafen ist.

```
LINE§ LOC CODE LINE


0001  0000             ;   ZEIT 8032 * V3 * 040980
0002  0000             ;
0003  0000             ;ZEIGT DIE LAUFENDE ZEIT IN DER FORM
0004  0000             ;            HH:MM:SS
0005  0000             ;AUF DEM BILDSCHIRM AN. DER IRQ-VEKTOR
0006  0000             ;($90 UND $91 BEIM CBM, $219 UND $21A BEIM PET)
0007  0000             ;MUSS AUF START EINGERICHTET WERDEN.

0009  0000                     *=634

0011  027A  01         ZEIG    .BYT 1          ;NULL HIER SCHALTET ANZEIGE AB
0012  027B  00         HMSJ    .BYT 0,0,0,0    ;J="JIFFIES"=1/60.SEC
0012  027C  00
0012  027D  00
0012  027E  00
0013  027F  24         VOLL    .BYT $24,$60,$60,$60
0013  0280  60
0013  0281  60
0013  0282  60
```

```
0015  0283                BILD=$8000              ;START ANZEIGE
0016  0283                NORMAL=$E455            ;NORMALE INTERRUPTROUTINE
0017  0283                ;                        (BEIM CBM 8000)
0018  0283                ;NORMAL=$E62E BEIM CBM 3000
0019  0283                ;NORMAL=$E685 BEIM PET

0021  0283  F8            START  SED
0022  0284  A2 03                 LDX §3          ;XR IST ZEIGER IN HMSJ UND VOLL
0023  0286  18            STELL  CLC
0024  0287  BD 7B 02             LDA HMSJ,X
0025  028A  69 01                ADC §1
0026  028C  9D 7B 02             STA HMSJ,X
0027  028F  DD 7F 02             CMP VOLL,X
0028  0292  F0 04                BEQ ZEIGZT
0029  0294  D8                   CLD
0030  0295  4C 55 E4             JMP NORMAL
0031  0298  A9 00      ZEIGZT LDA §0
0032  029A  9D 7B 02             STA HMSJ,X
0033  029D  CA                   DEX
0034  029E  E0 02                CPX §2
0035  02A0  D0 E4                BNE STELL
0036  02A2  A0 07                LDY §7          ;YR IST BILDZEIGER
0037  02A4  AD 7A 02    SCHRB  LDA ZEIG        ;ANZEIGEN?
0038  02A7  F0 24                BEQ STERN       ;-NEIN
0039  02A9  A9 3A                LDA §':         ;-JA
0040  02AB  8D 02 80             STA BILD+2
0041  02AE  8D 05 80             STA BILD+5
0042  02B1  BD 7B 02             LDA HMSJ,X      ;BETRACHTE DIE RECHTE ZIFFER...
0043  02B4  29 0F                AND §$0F
0044  02B6  09 30                ORA §'0
0045  02B8  99 00 80             STA BILD,Y
0046  02BB  88                   DEY
0047  02BC  BD 7B 02             LDA HMSJ,X      ;...UND DIE LINKE
0048  02BF  4A                   LSR A
0049  02C0  4A                   LSR A
0050  02C1  4A                   LSR A
0051  02C2  4A                   LSR A
0052  02C3  09 30                ORA §'0
0053  02C5  99 00 80             STA BILD,Y
0054  02C8  88                   DEY
0055  02C9  88                   DEY
0056  02CA  CA                   DEX
0057  02CB  10 D7                BPL SCHRB
0058  02CD  A2 02      STERN  LDX §2
0059  02CF  D0 B5                BNE STELL       ;(JUMP)
0060  02D1              FINI   .END


ERRORS = 0000


SYMBOL TABLE

SYMBOL VALUE
 BILD     8000    FINI     02D1    HMSJ     027B    NORMAL   E455
 SCHRB    02A4    START    0283    STELL    0286    STERN    02CD
 VOLL     027F    ZEIG     027A    ZEIGZT   0298
END OF ASSEMBLY
```

*

Dipl.-Ing. Klaus-Rüdiger Hase, Recklinghausen

Disassemblierung in den Texteditor

Das nachfolgende Programmpaket mit DISASSEMBLER (into Editor), KILL-LABEL und BYTE-TABELLE gestattet die Reassemblierung bereits im Object-Code vorliegender Programme. Es ist hilfreich, wenn keine entsprechende Dokumentation zur Verfügung steht. Es besteht aus den o.a. genannten drei Teilen, wobei der eigentliche Disassembler mit (A) vom Eingangsverzweiger aufgerufen wird.

Hierbei wird jeder Zeile ein Label vorangestellt, das aus dem führenden Buchstaben "H" und der hexadezimalen Adresse besteht (z.B. H4E32). Absolut- und Relativ-Operanden werden ebenfalls als Labels ausgegeben, wenn der zugehörige Adreßwert im zu disassemblierenden Bereich liegt, andernfalls wird der absolute Adreßwert mit '$' gekennzeichnet.

Bei 0-Page- und Immediateoperanden geschieht dies immer. Wird der Disassembler-File im Editor erstellt, so werden mit dem Programm KILL-EX (A) überflüssige Adreßlabels wieder gelöscht. Dieses Programm kann auch unabhängig aufgerufen werden und dann von TOP oder beliebiger NOWLN-Stellung (KILLAB (N)) arbeiten. - Vom Programm nicht gefundene Labels werden in einer Error-Tabelle zur manuellen Überarbeitung ausgegeben.

Kleinere Datenfelder werden mit ".BYTE $NN" ausgegeben. Da häufig Datenbytes als Branch-Befehle mit nicht existierenden Relativ-Adreß-Labels ausgegeben werden, kann die Ausgabe solcher Relativ-Adreß-Labels mit M='O' (Offset) abgeschaltet werden. Es können größere Datenfelder mit "TABEL" (T) in strukturierter Form erstellt werden. Hierbei sind verschiedene Darstellungen in ASCII, BYTE, WORD, DBYTE mit beliebiger Anzahl zwischen 1 und 29 möglich, sowie auch in Label-Form mit WORD, DBYTE, BYTE/"L".

```
0000                    *=$112
0112            4C0002 JMP ANFANG
0115
0115 ANFANG=$200
0115
0115            ;*** REGISTER ***
0115                    *=$0
0000 REGANF
0000 FIRST              *=*+2           ;START-ADR.
0002 NOWADR             *=*+2           ;LAUFENDE ADR.
0004 LAST               *=*+2           ;END-ADR.
0006 NUMBER             *=*+1           ;ANZAHL D. BYTES/WORDS/...
0007 MODUS              *=*+1           ;BYTE/WORD/ASCII & HEX/LABEL
0008 FORMA              *=*+1           ;F.DISASSM
0009 LMNEM              *=*+1
000A RMNEM              *=*+1
000B BRANCH             *=*+1           ;FLAG F. OFFSET/REL.ADR.
000C XSTO1              *=*+1           ;X-REGISTER
000D STONW1             *=*+2           ;EDIT-ANF.F.DISASSM
000F STONW2             *=*+2           ;NOWLN
0011 ERRZAL             = NUMBER        ;ANZAHL NICHT GEFUNDENER LABELS
0011 ERRFLG             = MODUS
0011 REGEND
```

```
0011
0011          --------------------
0011          ** SYMBOLTABELLE **

0011                *=ANFANG
0200
0200
0200          ;DIESES VERZWEIGER-PROGRAMM GESTATTET DEN EIN-
0200          ;TRITT IN JEDES DER 4 PROGRAMME MITTELS BUCH-
0200          ;STABENKUERZEL UNABHAENGIG VON DER AKTUELLEN
0200          ;VERSION.
0200
0200 JSRIP
0200          2073E9 JSR REDOUT      ;INPUT CODE
0203          2906   AND #6          ;MASKE F. TABELLE
0205          AA     TAX
0206          BD1002 LDA JMPTAB+1,X   ;HI-BYTE ADR.
0209          48     PHA             ;AUF STACK
020A          BD0F02 LDA JMPTAB,X
020D          48     PHA
020E          60     RTS
020F
020F JMPTAB   1602   .WOR ADISAH-1   ;<-- [A] DISASSEMBLER
0211          0F05   .WOR KILLEX-1   ;<-- [K] KILL LABELS (TOP)
0213          7F03   .WOR TABEL-1    ;<-- [T] BYTE,WORD-TABELLE
0215          1205   .WOR KILLAB-1   ;<-- [N] KILL LABELS (NOWLN)
```

Dieses Programm generiert ein Disassembler-Listing, welches für beliebige Ausgabeeinheiten erstellt werden kann und welches vom AIMAssembler verstanden wird. U-OUT schreibt an das Ende eines offenen Editor-Bereiches ein. - Die Branch-Flag ermöglicht Umschalten von Adresse (M="A") auf Offset (M="O"). Absolute Adressen werden innerhalb der Grenzen (FROM/TO) mit Label "HXXXX", außerhalb mit Hex-Adressen "$XXXX" versehen.

```
0217 ADISAH
0217          205F04 JSR INIADR      ;INITIALISIERUNG
021A          2071E8 JSR WHEREO      ;OUTPUT-DEVICE
021D          2013EA JSR CRLOW       ; --> PRINTER
0220          4C2802 JMP ADIS02
0223 ADIS01
0223          20B904 JSR TSTNOW      ;PRUEFE OB NOWADR>ENDADR
0226          901B   BCC ADIS04      ;C=0 --> ENDE
0228 ADIS02   204602 JSR DISASM
022B          AD25A4 LDA SAVPC
022E          38     SEC
022F          65EA   ADC LENGTH
0231          8D25A4 STA SAVPC
0234          8502   STA NOWADR      ;ERHOEHE UM BEFEHLS-LAENGE
0236          9005   BCC ADIS03
0238          EE26A4 INC SAVPC+1
023B          E603   INC NOWADR+1    ;EBENSO
023D ADIS03   20F0E9 JSR CRLF
0240          4C2302 JMP ADIS01
0243 ADIS04
0243 ADIS05   4C2404 JMP TAB110      ;SCHLIESSE FILE
0246
```

```
0246 DISASM
0246            204A03 JSR PCLOUT      ;PROGRAMM-ZAEHLER-LABEL
0249            A000   LDY #$00
024B            2056EB JSR PCLLD       ;BEFEHLS-CODE
024E            A8     TAY
024F            4A     LSR A
0250            900B   BCC DISAS1
0252            4A     LSR A
0253            B017   BCS DISAS3      ;KEIN CODE
0255            C922   CMP #$22
0257            F013   BEQ DISAS3
0259            2907   AND #$07
025B            0980   ORA #$80
025D DISAS1     4A     LSR A
025E            AA     TAX
025F            BD5BF5 LDA MODE,X      ;DECODE N. TABELLE
0262            B004   BCS DISAS2
0264            4A     LSR A
0265            4A     LSR A
0266            4A     LSR A
0267            4A     LSR A
0268 DISAS2     290F   AND #$0F
026A            D004   BNE DISAS4
026C DISAS3     A080   LDY #$80
026E            A900   LDA #$00
0270 DISAS4     AA     TAX
0271            BD9FF5 LDA MODE2,X
0274            8508   STA FORMA       ;HIER FORMAT BESTIMMT,
0276            2903   AND #$03        ;2 LOW BITS = LAENGE-1
0278            85EA   STA LENGTH      ;%00 --> 1BYTE,%01 --> 2BYTE...
027A            98     TYA
027B            298F   AND #$8F
027D            AA     TAX
027E            98     TYA
027F            A003   LDY #$03
0281            E08A   CPX #$8A
0283            F00B   BEQ DISAS7
0285 DISAS5     4A     LSR A
0286            9008   BCC DISAS7
0288            4A     LSR A
0289 DISAS6     4A     LSR A
028A            0920   ORA #$20
028C            88     DEY
028D            D0FA   BNE DISAS6
028F            C8     INY
0290 DISAS7     88     DEY
0291            D0F2   BNE DISAS5
0293            A8     TAY
0294            B9B9F5 LDA MNEML,Y
0297            8509   STA LMNEM
0299            B9F9F5 LDA MNEMR,Y
029C            850A   STA RMNEM
029E            A203   LDX #$03
02A0 DISAS8     A900   LDA #$00
02A2            A005   LDY #$05
02A4 DISAS9     060A   ASL RMNEM
02A6            2609   ROL LMNEM
02A8            2A     ROL A
```

```
02A9            88      DEY
02AA            D0F8    BNE DISAS9
02AC            69BF    ADC #$BF
02AE            C9BF    CMP #$BF
02B0            D00E    BNE DISA10
02B2            A200    LDX #0          ;F. TABELL
02B4            204104  JSR PRITAB      ;PRINT ".BYT $"
02B7            207003  JSR PRIDOL
02BA            2056EB  JSR PCLLD
02BD            4C46EA  JMP NUMA
02C0
02C0 DISA10     20BCE9  JSR OUTALL      ;BEFEHLS-CODE
02C3            CA      DEX
02C4            D0DA    BNE DISAS8
02C6            206703  JSR PRBL2
02C9            A902    LDA #2          ;MASKE F. ABS-ADR.
02CB            2508    AND FORMA       ;SETZE MASKE F.
02CD            8507    STA MODUS       ;LABEL-HEX
02CF            A206    LDX #$06
02D1            A900    LDA #$00
02D3            8D29A4  STA STIV+2      ;FLAG F. ADR.
02D6 DISA11     E003    CPX #$03
02D8            D02A    BNE DISA15
02DA            A4EA    LDY LENGTH
02DC            F026    BEQ DISA15
02DE DISA12     A508    LDA FORMA
02E0            C9E8    CMP #$E8
02E2            2056EB  JSR PCLLD       ;OPERAND --> <A>
02E5            B033    BCS RELADR
02E7 DISA13     48      PHA
02E8            AD29A4  LDA STIV+2
02EB            D010    BNE DISA14
02ED            A507    LDA MODUS       ;F.ABSOLUT-ADR.
02EF            F005    BEQ PRINHX      ;WENN NICHT
02F1            20A404  JSR TSTNOP      ;TESTE OB OPERAND IM BEREICH
02F4            B001    BCS PRINLB
02F6 PRINHX     18      CLC             ;ABSOL. ADR. MIT "$"
02F7 PRINLB     206C03  JSR PRIVOR
02FA            EE29A4  INC STIV+2      ;SET FLAG
02FD DISA14     68      PLA
02FE            2046EA  JSR NUMA        ;OPERAND
0301            88      DEY
0302            D0DA    BNE DISA12
0304 DISA15     0608    ASL FORMA       ;INDEX FUER ...
0306            900E    BCC DISA16
0308            BD7303  LDA CHAR1-1,X
030B            20BCE9  JSR OUTALL
030E            BD7903  LDA CHAR2-1,X
0311            F003    BEQ DISA16
0313            20BCE9  JSR OUTALL      ; SONDERZEICHEN
0316 DISA16     CA      DEX
0317            D0BD    BNE DISA11
0319            60      RTS
031A
031A RELADR     8408    STY FORMA       ;KEINE SONDERZ. B. OPER.
031C            240B    BIT BRANCH      ;PRUEFE OB OFFSET
031E            102C    BPL PCADJH      ;WENN MSB=1
0320
```

```
0320 BRELST                             ;BRANCH MIT RELATIV-OFFSET
0320          A8      TAY               ;TEST +/- , C=1 !!
0321          08      PHP               ;SAVE N-FLAG
0322          100F    BPL BREL01        ;F. POS. VERZWEIGUNG
0324          49FF    EOR #$FF          ;BEI NEG. VERZWEIGUNG
0326          D008    BNE BREL03
0328          68      PLA               ;P-STATUS
0329          297E    AND #%01111110     ;N=0,C=0 --> "+"
032B          48      PHA               ;STATUS-REG.
032C          A901    LDA #1
032E          D005    BNE BREL04        ;IMMER
0330 BREL03   E902    SBC #2            ;C=1
0332          18      CLC
0333 BREL01   6901    ADC #1            ;+2 F. POS./-2 F. NEG.
0335 BREL04   A8      TAY               ;SAVE
0336          A92A    LDA #'*'
0338          20BCE9  JSR OUTALL        ;PROGR.-ZAEHLER
033B          A92B    LDA #'+'          ;F.POSITIV
033D          28      PLP               ;TESTE FLAG F. NEG. ?
033E          1002    BPL BREL02
0340          A92D    LDA #'-'          ;F. NEGATIV
0342 BREL02   20BCE9  JSR OUTALL
0345          98      TYA               ;OFFSET --> <A>
0346          A408    LDY FORMA         ;FORMA = 1 = LENGTH
0348          D09D    BNE DISA13
034A
034A PCLOUT   A9FE    LDA #$FE          ;OFFSET F.ADRESS-LABEL
034C PCAD3H   AC26A4  LDY SAVPC+1       ;HI-BYTE
034F          AA      TAX               ;MSB --> N-FLAG
0350          1001    BPL PCAD4H
0352          88      DEY               ;2-ER KOMPL.
0353 PCAD4H   38      SEC               ;F.FREMDEINSTIEG (ADR.)
0354          6D25A4  ADC SAVPC
0357          9001    BCC PCAD5H
0359          C8      INY
035A PCAD5H   AA      TAX
035B          E8      INX
035C          D001    BNE PRNPCH
035E          C8      INY
035F PRNPCH   38      SEC               ;LABEL "H"
0360          206C03  JSR PRIVOR
0363          98      TYA
0364          2042EA  JSR WRAX
0367 PRBL2    A920    LDA #$20
0369 PRBL0    4CBCE9  JMP OUTALL
036C PRIVOR   A948    LDA #'H'          ;F. LABEL
036E          B0F9    BCS PRBL0
0370 PRIDOL   A924    LDA #'$'
0372          D0F5    BNE PRBL0
0374
0374 CHAR1
0374          2C29    .BYT ',),#(A'
037A CHAR2
037A          59      .BYT 'Y',0,'X',0,0,0
037B          00
037C          58
037D          00
037E          00
037F          00
```

```
0380            ;DIESES PROGRAMM GENERIERT AUS EINER DA-
0380            ;TENTABELLE EINE BYTE,WORD O.DBYTE-TABELLE,
0380            ;DIE FUER DEN ASSEMBER VERSTAENDLICH IST.
0380            ;ES WERDEN ANFANGS- UND END-ADRESSE ABGE-
0380            ;FRAGT, SOWIE MODUS UND BYTE-ZAHL JE ZEILE.
0380            ;MODEN : A -> ASCII, B -> HEX-BYTE,
0380            ;MODEN : D -> DBYTE, W -> WORD.
0380            ;ANZAHL : 01...29, <RTN> = 01,
0380            ;L,<SPACE> = 1* LABEL
0380
0380 TABEL
0380            205F04 JSR INIADR       ;INIT ADRESSEN UND REGS.
0383            2037E8 JSR PSL1         ;"/"
0386            2085E7 JSR GCNT         ;DEZIMALE EINGABE
0389            C92C   CMP #$2C         ;FALLS GROESSER 29
038B            9002   BCC TABE10       ;WENN <= 29
038D            A900   LDA #0
038F TABE10     8506   STA NUMBER
0391            203EE8 JSR BLANK
0394            2071E8 JSR WHEREO       ;OUTPUT WO ?
0397            204A03 JSR PCLOUT       ;AUSGABE ADRESS-LABEL
039A
039A TABE20
039A            ;AB HIER SCHLEIFE
039A            A200   LDX #0
039C            A507   LDA MODUS        ;AUSGABE-ART
039E            C944   CMP #'D'         ;DOPPEL-BYTE
03A0            D002   BNE TABE30
03A2            A206   LDX #TAB2I-TABELL
03A4 TABE30     C957   CMP #'W'         ;WORD ?
03A6            D002   BNE TABE40
03A8            A20C   LDX #TAB3I-TABELL
03AA TABE40     204104 JSR PRITAB       ;ASSEM.-ZUWEISUNG
03AD            A507   LDA MODUS
03AF            C941   CMP #'A'
03B1            F00D   BEQ TABE45
03B3            A506   LDA NUMBER
03B5            8D19A4 STA COUNT
03B8            D016   BNE TABE60
03BA            EE19A4 INC COUNT
03BD            38     SEC
03BE            B011   BCS TABE65
03C0 TABE45
03C0            A927   LDA #$27         ;NUR F. ASCII
03C2            20BCE9 JSR OUTALL
03C5 TABE50     A507   LDA MODUS
03C7            C941   CMP #'A'         ;KEIN "," F. ASCII
03C9            F009   BEQ TABE70
03CB            A92C   LDA #','         ;TRENNUNG F. BYT,DBYT,WOR
03CD            20BCE9 JSR OUTALL
03D0 TABE60     18     CLC
03D1 TABE65     206C03 JSR PRIVOR       ;PRINT '$'
03D4 TABE70     A000   LDY #0
03D6            B102   LDA (NOWADR),Y    ;LADE BYTE
03D8            A607   LDX MODUS
03DA            E057   CPX #'W'          ;BEI WORD TAUSCH
03DC            D00D   BNE TABE75
```

```
03DE           48      PHA               ;F. TAUSCH
03DF           20B304  JSR INCNOW        ;INC NOWADR U. CMP
03E2           B102    LDA (NOWADR),Y     ;NAECHSTES BYTE
03E4           2046EA  JSR NUMA
03E7           68      PLA               ;HIGH VOR LOW
03E8           4C0104  JMP TABE80
03EB TABE75    E041    CPX #'A'
03ED           D006    BNE TABE78        ;AUSSER ASCII
03EF           20BCE9  JSR OUTALL        ;NUR ASCII
03F2           4C0404  JMP TABE90
03F5 TABE78    2046EA  JSR NUMA          ;ALLE HEX O. WORD
03F8           E044    CPX #'D'
03FA           D008    BNE TABE90        ;WENN "BYTE"
03FC           20B304  JSR INCNOW        ;2.BYTE
03FF           B102    LDA (NOWADR),Y
0401 TABE80    2046EA  JSR NUMA
0404 TABE90    20B304  JSR INCNOW
0407           9005    BCC TABE95
0409           2090E7  JSR DONE          ;COUNT-1
040C           D0B7    BNE TABE50
040E TABE95    A507    LDA MODUS
0410           C941    CMP #'A'          ;NUR BEI ASCII...
0412           D005    BNE TAB100
0414           A927    LDA #$27          ;ZUSAETZLICH "'"
0416           20BCE9  JSR OUTALL
0419 TAB100    20F0E9  JSR CRLF
041C           20B904  JSR TSTNOW
041F           9003    BCC TAB110        ;WEITER
0421           4C9A03  JMP TABE20
0424 TAB110    AD13A4  LDA OUTFLG
0427           C955    CMP #'U'          ;BEI EDIT NUR 1* "CR"
0429           F003    BEQ TAB120
042B TAB115    20F0E9  JSR CRLF
042E TAB120    4C0AE5  JMP DU11          ;CLOSE FILE
0431
0431           ;*** UPROS ***
0431
0431 MOVAD2    203404  JSR MOVADR        ;2 MAL MOVE
0434
0434 MOVADR    AD1CA4  LDA ADDR          ;(A,X) POINTER IN <X>
0437           9500    STA FIRST,X
0439           AD1DA4  LDA ADDR+1
043C           9501    STA FIRST+1,X
043E           E8      INX
043F           E8      INX
0440           60      RTS
0441
0441 PRITAB    BD4D04  LDA TABELL,X      ;(A,X) PRINT ANWEISUNG
0444           D001    BNE PRIT10        ;BIS END-ZEICHEN
0446           60      RTS
0447 PRIT10    20BCE9  JSR OUTALL
044A           E8      INX
044B           D0F4    BNE PRITAB
044D
044D TABELL
044D TAB1I     2E42    .BYT '.BYT ',0
0452           00
0453 TAB2I     2E44    .BYT '.DBY ',0
0458           00
```

```
0459 TAB3I    2E57    .BYT '.WOR ',0
045E          00
045F
045F INIADR
045F          ;(A,X,Y) INITIALISIERUNG VON ADR.-REG.
045F
045F          203EE8  JSR BLANK       ;MIT POINTER IN <X>
0462          A900    LDA #0
0464          A211    LDX #REGEND-REGANF    ;UND 0-PAGE-REGS.
0466 INIAD1   9500    STA REGANF,X
0468          CA      DEX
0469          10FB    BPL INIAD1
046B          20C5F8  JSR SETBOT      ;EDITOR --> END
046E          A200    LDX #0
0470          20CB05  JSR SAVNOW      ;NOWLN --> STONW1
0473          A9D9    LDA #<TABOUT    ;INIT U-OUT
0475          8D0A01  STA UOUT
0478          A904    LDA #>TABOUT
047A          8D0B01  STA UOUT+1
047D          20A3E7  JSR FROM        ;ANFANGSADRESSE
0480          A200    LDX #0
0482          203104  JSR MOVAD2      ;IN FIRST UND NOWADR
0485          20D7E5  JSR CGPC0       ;ADDR --> SAVPC
0488          203EE8  JSR BLANK
048B          20A7E7  JSR TO          ;ENDADRESSE
048E          A204    LDX #4          ;ADDR --> LAST
0490          203404  JSR MOVADR
0493          203EE8  JSR BLANK
0496          A003    LDY #3          ;AUSGABE VON "M="
0498          2070E9  JSR KEPR        ;CHR IN <A>
049B          8507    STA MODUS       ;F. BYTAB
049D          C94E    CMP #'N'        ;C=1 ,CHR > M : N --> OFFSET
049F          660B    ROR BRANCH      ;C=0 ,CHR < N : A --> ABSOL.
04A1          4C3EE8  JMP BLANK
04A4
04A4 TSTNOP                           ;(A,Y) TESTE OPRAND
04A4          860C    STX XSTO1       ;SAVE X
04A6          A001    LDY #1          ;POINTER AUF LOW-BYTE
04A8          2056EB  JSR PCLLD       ;LOW-OPERAND
04AB          AA      TAX             ;IN <X>
04AC          C8      INY
04AD          2056EB  JSR PCLLD       ;HI-OPERAND IN <A>
04B0          4CBF04  JMP TSTNW5
04B3 INCNOW                           ;(A) INC NOWADR UND C=1 BEI ENDE
04B3          E602    INC NOWADR      ;LOW-BYTE
04B5          D002    BNE INCN10
04B7          E603    INC NOWADR+1    ;WENN UEBERLAUF
04B9 INCN10
04B9 TSTNOW   860C    STX XSTO1       ; (A) RETTE X
04BB          A602    LDX NOWADR      ;LOW-BYTE
04BD          A503    LDA NOWADR+1    ;VERGLEICH MIT...
04BF TSTNW5   C505    CMP LAST+1      ;ENDE D. LISTE
04C1          9008    BCC TSTNW1      ;WENN KLEINER
04C3          D010    BNE TSTNW2      ;WENN GROESSER
04C5          E404    CPX LAST        ;LOW-BYTE
04C7          F002    BEQ TSTNW1      ;GLEICHHEIT
04C9          B00A    BCS TSTNW2      ;NOWJLAST UNGUELTIG
```

```
04CB TSTNW1  C501    CMP FIRST+1      ;HI-BYTE
04CD         9006    BCC TSTNW2       ;NOW<FIRST UNGUELTIG
04CF         D005    BNE TSTNW3       ;GUELTIG
04D1         E400    CPX FIRST        ;LOW-BYTE
04D3         B001    BCS TSTNW3
04D5 TSTNW2  18      CLC              ;WENN UNGUELTIG --> C=0
04D6 TSTNW3  A60C    LDX XSTO1
04D8         60      RTS              ;FIRST<=NOWADR<=LAST --> C=1
04D9
04D9         ; *** USER-OUT --> EDIT-IN ***
04D9
04D9 TABOUT  9034    BCC TABO30       ;(A) DANN INIT.
04DB         68      PLA
04DC TABOUN  48      PHA
04DD         297F    AND #$7F         ;0. MSB F. EDITOR
04DF         C90A    CMP #LF          ;EINSPRUNG O.UOUT
04E1         F02B    BEQ TABO20
04E3         C97F    CMP #$7F         ;BEI TTY
04E5         F027    BEQ TABO20
04E7         209EEB  JSR PHXY         ;RETTE Y
04EA         A000    LDY #0
04EC         91DF    STA (NOWLN),Y    ;IN EDITOR
04EE         2028F9  JSR AD1          ;NOWLN+1
04F1         98      TYA              ;0 --> ACCU
04F2         91DF    STA (NOWLN),Y    ;END-ZEICHEN
04F4         A5DF    LDA NOWLN
04F6         85E1    STA BOTLN
04F8         A5E0    LDA NOWLN+1
04FA         85E2    STA BOTLN+1
04FC         20F9F8  JSR ATEND        ;VERGL. OB ENDE ?
04FF         900A    BCC TABO10
0501         201DF9  JSR SUB          ;NOWLN-1
0504         A90D    LDA #CR
0506         91DF    STA (NOWLN),Y
0508         4C5CFA  JMP ENDERR       ;WENN EDIT-ENDE
050B TABO10  20ACEB  JSR PLXY         ;RESTORE REGS
050E TABO20  68      PLA
050F TABO30  60      RTS

0510         ;DIESES PROGRAMM LOESCHT UEBERFLUESSIGE
0510         ;ADRESS-LABELS IM EDITOR.
0510
0510 KILLEX  20BCF8  JSR TOPNO        ;(A,X,Y) 1.ZEILE
0513 KILLAB  A200    LDX #0           ;VON NOWLN AB
0515         20CB05  JSR SAVNOW       ;NOWLN --> STONW1
0518 KILL02  20C205  JSR RESNOW       ;STONW1/2 --> NOWLN
051B KILL01  A920    LDA #' '         ;ANFANG F. ALLE OPERANDEN
051D         85EB    STA STRING
051F         A948    LDA #'H'
0521         85EC    STA STRING+1
0523         A002    LDY #2           ;STRING-LAENGE
0525         8C29A4  STY STIV+2
0528         20D705  JSR FC5H         ;SUCHE LABEL ALS OPERAND
052B         9056    BCC DELABL       ;BEI LETZTEM LABEL
052D         A202    LDX #2
052F         20CB05  JSR SAVNOW       ;NOWLN --> STONW2
0532 KILL03  AC15A4  LDY CURPO2       ;CURSOR-POSITION
0535         C8      INY              ;OHNE <SPACE>
0536         A200    LDX #0
```

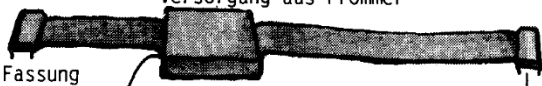
```
0538           8607    STX ERRFLG      ;RESET FLAG
053A KILL04    B1DF    LDA (NOWLN),Y   ;LABEL --> STRING
053C           95EB    STA STRING,X
053E           E8      INX
053F           C8      INY
0540           E005    CPX #5          ;LABEL-ENDE
0542           D0F6    BNE KILL04
0544           8E29A4  STX STIV+2      ;STRING-LAENGE
0547           A200    LDX #0
0549           20C205  JSR RESNOW      ;STONW1 --> NOWLN
054C           B005    BCS KILL08      ;IMMER
054E
054E KILL05    A000    LDY #0
0550           ;SUCHE GUELTIGE ADRESS-LABELS
0550           2013F7  JSR UP1
0553 KILL08    20D705  JSR FC5H        ;SUCHE OBIGEN STRING
0556           B01F    BCS KILL07      ;MARKIERE LABEL
0558           A202    LDX #2          ;RESTORE ALTES NOWLN
055A           20C205  JSR RESNOW
055D           A000    LDY #0
055F           2013F7  JSR UP1         ;NAECHSTE ZEILE
0562           A507    LDA ERRFLG      ; =0, WENN KEIN LABEL GEFUNDEN
0564           D0B5    BNE KILL01      ;NEXT LABEL
0566           A200    LDX #0
0568 KILL06    B5EB    LDA STRING,X    ;AUSGABE DES NICHT
056A           207AE9  JSR OUTPUT      ;GEFUNDENEN LABELS
056D           E8      INX
056E           E005    CPX #5
0570           D0F6    BNE KILL06
0572           2013EA  JSR CRLOW
0575           D0A4    BNE KILL01      ;DA CRLOW MIT 'PLA' <>0
0577 KILL07    C005    CPY #5          ;ADRESS-LABELS
0579           D0D3    BNE KILL05
057B           A940    LDA #'@'        ;MARKIERE ADRESS-LABEL
057D           91DF    STA (NOWLN),Y   ;ALS GUELTIG
057F           8507    STA ERRFLG      ;--> <>0
0581           D0CB    BNE KILL05
0583
0583 DELABL
0583           ;AB HIER WERDEN ALLE LABELS OHNE
0583           ;MARKE '@' GELOESCHT.
0583
0583           A066    LDY #$66
0585           20AFE7  JSR KEP         ;OUTPUT : "LOAD"
0588 DELA01    20B2F8  JSR CFLG        ;SETZE CHANGE-FLG
058B           A200    LDX #0
058D           20C205  JSR RESNOW      ;ANF. D. ASSM-BEREICHES
0590 DELA06    2009F7  JSR UPNO        ;MIT AUSSTIEG --> ENDERR
0593           A000    LDY #0
0595           B1DF    LDA (NOWLN),Y   ;PRUEFE AUF ENDE
0597           D006    BNE DELA03
0599           20F0E9  JSR CRLF
059C           4C5CFA  JMP ENDERR
059F DELA03    C948    CMP #'H'        ;OB LABEL ?
05A1           D0ED    BNE DELA06
05A3           A005    LDY #5          ;STRING-LAENGE
05A5           84E9    STY OLDLEN
05A7           E6E9    INC OLDLEN      ;AUCH <SPACE> LOESCHEN
```

```
05A9            B1DF    LDA (NOWLN),Y
05AB            C940    CMP #'@'        ;MARKE ?
05AD            F00D    BEQ DELA07
05AF            A000    LDY #0
05B1            84EA    STY LENGTH      ;LOESCHE LABEL
05B3            8C15A4  STY CURPO2
05B6            203FF9  JSR REPLAC
05B9            4C9005  JMP DELA06
05BC DELA07     A920    LDA #$20        ;ENTFERNE MARKE WIEDER
05BE            91DF    STA (NOWLN),Y   ;UND ERHALTE LABEL
05C0            D0CE    BNE DELA06
05C2
05C2            ;(A) TRANSPORT-PROGRAMME
05C2
05C2 RESNOW     B50D    LDA STONW1,X    ;STONW1/2 --> NOWLN
05C4            85DF    STA NOWLN
05C6            B50E    LDA STONW1+1,X
05C8            85E0    STA NOWLN+1
05CA            60      RTS
05CB
05CB SAVNOW     A5DF    LDA NOWLN       ;NOWLN --> STONW1/2
05CD            950D    STA STONW1,X
05CF            A5E0    LDA NOWLN+1
05D1            950E    STA STONW1+1,X
05D3            60      RTS

05D4            ;(A,X,Y) SUCHPROGRAMM WIE MONITOR O. ENDERR
05D4
05D4 FC2H       2009F7  JSR UPNO        ;MIT AUSGANG --> ENDERR
05D7 FC5H       A000    LDY #0          ;ZEILENANFANG
05D9            8C15A4  STY CURPO2
05DC FC6H       AC15A4  LDY CURPO2
05DF            A200    LDX #0
05E1 FC7H       B1DF    LDA (NOWLN),Y
05E3            D002    BNE FC8H
05E5            18      CLC
05E6            60      RTS             ;C=0 --> STRING NICHT GEFUNDEN
05E7 FC8H       C90D    CMP #CR
05E9            F0E9    BEQ FC2H
05EB            D5EB    CMP STRING,X
05ED            F006    BEQ FC9H
05EF            EE15A4  INC CURPO2
05F2            4CDC05  JMP FC6H
05F5 FC9H       E8      INX
05F6            C8      INY
05F7            EC29A4  CPX STIV+2
05FA            D0E5    BNE FC7H
05FC            60      RTS             ;C=1 --> STRING GEFUNDEN
```

```
;ASM-LISTE /.12.09.80 DIASM 'DISASSEMBLER MIT BYTE-TAB. V2.6'
.OPT NOL
AD1     =$F928  ADDIN   =$EAAE  ADDR    =$A41C  ADIS01 =$0223
ADIS02 =$0228   ADIS03 =$023D   ADIS04 =$0243   ADIS05 =$0243
ADISAH =$0217   ANFANG =$0200   ATEND   =$F8F9   BLANK  =$E83E
BOTLN  =$00E1   BRANCH =$000B   BREL01 =$0333   BREL02 =$0342
BREL03 =$0330   BREL04 =$0335   BRELST =$0320   CFLG   =$F8B2
CGPC0   =$E5D7  CHAR1   =$0374  CHAR2   =$037A  COUNT  =$A419
CR      =$000D  CRLF    =$E9F0  CRLOW   =$EA13  CURPO2 =$A415
```

Assemble

RAM Versorgung aus Prommer DM 495,--



verbleibt im Prommer

EPROM - Fassung des µP-Systems

automatischer Reset

wird nach dem Laden in EPROM-Fassung des µP-Systems gesteckt

CMOS-RAM mit Pufferbatterie DM 655,--

2732 EPROM

2716 EPROM

System-Übersicht

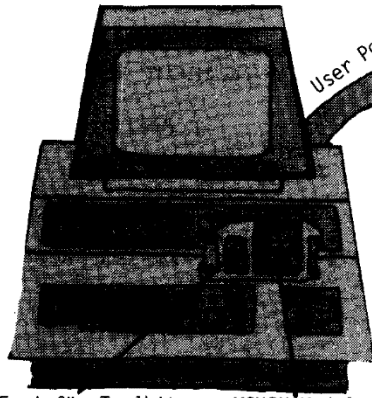
PROMMER DM 799,--

Textool-Fassung

User Port

ROM/RAM 2K/4K-Belegung

Net



Fassung für 2 St. 2K-EPROM DM 36,--

Disketten:

ACY Assembler DM 305,--
TWG Teachware DM 330,--

Manual extra DM 35,--

Bitte rufen Sie uns an, wir geben eine begrenzte Anzahl Manuals kostenlos ab. Anrufbeantworter nachts 08178/3532.

Frei für Toolkit o.a. Bereich $BØØØ-BFFF

MOUSY Modul Bereich $9ØØØ-AFFF DM 799,--

TECHNOFOR LIZENZ- U. PATENTVERWERTUNGSGESELLSCHAFT M.B.H.

ADALBERT-STIFTER-RING 21
D-8026 EBENHAUSEN

TELEFON 08178/3531 und Ø 4337
TELEX 05-27856 tech d

oftware-Entwicklungssystem

für alle 65xx-Projekte

ine Vorteile:
* sehr hoher Komfort
* preiswert
* enorme Reduzierung der Testzeit durch vollständige Software-Testmöglichkeit
* Installation auf COMMODORE CBM-System 3000, diese Anlage ist auch außerhalb der Assembler-Aufgaben vielseitig verwendbar

dware:
* Grundausbaustufe mit CBM 3032 und MOUSY Steckmodul für freie Plätze $9000 und $A000
* Weiterer Ausbau mit COMMODORE Floppy 3040, Drucker (beliebig, jedoch IEC-fähig), EPROM-Programmiergerät (PROMMER) **, EPROM-Emulatoren (MOUSY-RAM) **
** siehe linke Seite

tware:
* 8 K ROM, 2 Seiten RAM mit Steuerlogik in MOUSY Steckmodul
* schneller, problemloser Disketten-Assembler für automatische Assemblierung von Quellendateien
* Disketten Teachware (integriertes Assembler-Lernsystem)

ISY enthält 42 problemlose und z.T. vielfältig anwendbare Grundbefehle:

Assemblieren auf dem Bildschirm oder von Eingabegeräten
Disassemblieren auf dem Bildschirm oder von Eingabegeräten
Testen von Programmen ('execute')
- mit Kontrolle auf Bildschirm
- Anzeige aller Register
- mit Kontrolle auf Drucker
- mit freiem Lauf in bereits getesteten Teilen
- mit Unterdrückung des Ausdrucks in verschiedenen Bereichen
- und Anhalten bei Eintreffen einer gesetzten Bedingung in einem beliebigen Speicherplatz oder Register
- mit voller Interrupt-Simulation
- mit laufender Darstellung einer Anzahl frei wählbarer Speicherplätze
- mit voller Verfügung über die Zeropage
- mit expliziter Darstellung aller Sprünge

automatisches Suchen und Ändern von Sprüngen und Verzweigungen
automatisches Verschieben und Verlegen von Programmen oder Programmteilen
- mit oder ohne Adressumrechnung
- mit Überwachung von Seitengrenzen

automatisches Umadressieren ganzer Bereiche um einen bestimmten Betrag
Ausmessen des echten (!!) zyklusgenauen Zeitbedarfs von Programmteilen (true cycle counter)
Ändern von Registern oder Speichern über Bildschirm
Hex/Dezimal/Hex-Umrechnen
Schneller und rationaller Floppy-Umgang
Lesen und Programmieren von EPROMs

HTIG:

Alles ist auf problemlose Anwendung ausgelegt. Keine Kniffe oder Tricks notwendig.

* CBM RAM immer voll nutzbar
* Seite Ø voll nutzbar
* vollständige Software Simulation
* Dem MOUSY-Konzept liegt eine erprobte Debugging-Strategie zugrunde

Lieferung durch TECHNOFOR oder Ihren COMMODORE-Fachhändler. Ihre Fragen beantworten wir sofort. Bitte schreiben Sie uns oder rufen Sie uns an.

* In Kürze gibt es eine Demo-Cassette für PET/CBM-Besitzer kostenlos. Bestellen Sie gleich heute unter Angabe der Seriennummer Ihres Gerätes sowie Typs. - Die Auflage ist limitiert.

```
DELA01 =$0588  DELA03 =$059F  DELA06 =$0590  DELA07 =$05BC
DELABL =$0583  DISA10 =$02C0  DISA11 =$02D6  DISA12 =$02DE
DISA13 =$02E7  DISA14 =$02FD  DISA15 =$0304  DISA16 =$0316
DISAS1 =$025D  DISAS2 =$0268  DISAS3 =$026C  DISAS4 =$0270
DISAS5 =$0285  DISAS6 =$0289  DISAS7 =$0290  DISAS8 =$02A6
DISAS9 =$02A4  DISASM =$0246  DONE   =$E790  DRB    =$A800
DU11   =$E50A  ENDE   =$05FD  ENDERR =$FA5C  ERRFLG =$0007
ERRZAL =$0006  FC2H   =$05D4  FC5H   =$05D7  FC6H   =$05DC
FC7H   =$05E1  FC8H   =$05E7  FC9H   =$05F5  FIRST  =$0000
FORMA  =$0008  FROM   =$E7A3  GCNT   =$E785  INCN10 =$04B9
INCNOW =$04B3  INIAD1 =$0466  INIADR =$045F  JMPTAB =$020F
JSRIP  =$0200  KEP    =$E7AF  KEPR   =$E970  KILL01 =$051B
KILL02 =$0518  KILL03 =$0532  KILL04 =$053A  KILL05 =$054E
KILL06 =$0568  KILL07 =$0577  KILL08 =$0553  KILLAB =$0513
KILLEX =$0510  LAST   =$0004  LENGTH =$00EA  LF     =$000A
LL     =$E8FE  LMNEM  =$0009  MNEML  =$F5B9  MNEMR  =$F5F9
MODE   =$F55B  MODE2  =$F59F  MODUS  =$0007  MOVAD2 =$0431
MOVADR =$0434  NOWADR =$0002  NOWLN  =$00DF  NUMA   =$EA46
NUMBER =$0006  OLDLEN =$00E9  OUTALL =$E9BC  OUTFLG =$A413
OUTPUT =$E97A  PCAD3H =$034C  PCAD4H =$0353  PCAD5H =$035A
PCLLD  =$EB56  PCLOUT =$034A  PHXY   =$EB9E  PLXY   =$EBAC
PRBL2  =$0367  PRBLO  =$0369  PRIDOL =$0370  PRINHX =$02F6
PRINLB =$02F7  PRIT10 =$0447  PRITAB =$0441  PRIVOR =$036C
PRNPCH =$035F  PSL1   =$E837  REDOUT =$E973  REGANF =$0000
REGEND =$0011  RELADR =$031A  REPLAC =$F93F  RESNOW =$05C2
RMNEM  =$000A  SAVNOW =$05CB  SAVPC  =$A425  SETBOT =$F8C5
STIV   =$A427  STONW1 =$000D  STONW2 =$000F  STRING =$00EB
SUB    =$F91D  TAB100 =$0419  TAB110 =$0424  TAB115 =$042B
TAB120 =$042E  TAB1I  =$044D  TAB2I  =$0453  TAB3I  =$0459
TABE10 =$038F  TABE20 =$039A  TABE30 =$03A4  TABE40 =$03AA
TABE45 =$03C0  TABE50 =$03C5  TABE60 =$03D0  TABE65 =$03D1
TABE70 =$03D4  TABE75 =$03EB  TABE78 =$03F5  TABE80 =$0401
TABE90 =$0404  TABE95 =$040E  TABEL  =$0380  TABELL =$044D
TABO10 =$050B  TABO20 =$050E  TABO30 =$050F  TABOUN =$04DC
TABOUT =$04D9  TAPOUT =$A435  TO     =$E7A7  TOPNO  =$F8BC
TSTNOP =$04A4  TSTNOW =$04B9  TSTNW1 =$04CB  TSTNW2 =$04D5
TSTNW3 =$04D6  TSTNW5 =$04BF  UIN    =$0100  UOUT   =$010A
UP1    =$F713  UPNO   =$F709  WHEREO =$E871  WRAX   =$EA42
XSTO1  =$000C
```

Einige Hinweise: Das Programm KILLEX, das die überflüssigen Labels löscht, kann unter Umständen eine halbe Stunde und länger erfordern, z.B, für 4 K-Byte Objectcode. - Der Speicherbedarf liegt bei 4 K-Byte Objektprogrammen bei ca. 30 KB im Editor. Der erzeugte Text verkürzt sich nach KILLEX erheblich. KILLEX dauert deshalb so lange, weil erst alle Labels abgefragt werden müssen, und zwar so oft, wie Labels vorhanden sind. Die Verarbeitungszeit wächst also quadratisch. Bei der Betriebsart "Offset" wird die geringste Label-Zahl erzeugt, sie ist aber nicht so anschaulich wie 'absolut'.

Zur Inbetriebnahme des Programmes: Start mit der Funktionstaste F=3. Zuvor muß ein Textspeicher in ausreichender Größe per Editor bereitgestellt sein, wenn man auf den späteren Prompt OUT= mit U antworten will (U führt immer in den Editor, RETURN auf die aktivierten Systemeinheiten, wie Display, Thermoprinter, Bildschirm oder auch externer Drucker). - Auf das angezeigte Symbol für die betätigte F3-Taste ist nun einer der folgenden Buchstaben für den Modus einzugeben:

A für Disassembler
K für KILLEX, Beseitigung überflüssiger Labels ab 1. Zeile
N für KILLEX ab zuletzt angesteuerter Zeile (NOWLN)
T für Byte-/Wordtabelle

Die nachfolgenden Prompts FROM und TO geben den Bereich an, in dem das Kommando arbeiten soll. Für den T-Befehl also speziell dort, wo man nach erster Analyse auf eine Tabelle gestoßen ist.

Im Disassembler beeinflußt der nachfolgende Prompt "M=" das Ausgabeformat: M=A schreibt Branches in Label-Form absolut. M=O (=Offset) schreibt die Branches z.B. also relativ als BCC *+$1D.

Unter T-Kommando erzeugt M=W im Arbeitsbereich Adreßworte mit Label und .WOR-Anweisung mit Labels als Operanden. - M=B erzeugt Hex-Bytes mit Label, .BYT-Anweisung und Operanden. - M=A erzeugt entprechend eine .BYT-Anweisung mit ASCII-Stringoperanden. - Auf die Eingabe nach M= folgt als Prompt ein '/', um die Menge der Bytes oder Wörter abzufragen. Höchstzulässing sind 29. Erzeugt wird allerdings nur bis zur durch TO gesetzten Adreßgrenze.

Ein nach dem Disassemblieren (A) mit K beginnender augeschriebener Label weist auf einen nicht gefundenen Label hin, und damit im allgemeinen auf eine Tabelle.

Einige kurze Beispiele:

```
<E>
EDITOR                          INITIALISIERUNG DES EDITORS
FROM=4000     TO=5000
IN=

END
<↑>A FROM=217                   DISASSEMBLER-MODUS AB $217
 TO=245  M=A  OUT=U             BIS 245, BRANCHES MIT ABSOLUTEN LABELS
                                AUSGABE IN DEN EDITOR (U)
H0217 JSR $045F                 DISASSEMBLIERUNGSBEISPIELE AUS DIESEM
H021A JSR $E871                 PROGRAMM, SIEHE ASSEMBLER-LISTE
USW.

<↑>K                            EINSATZ VON KILLEX

JSR $045F                       UNNOETIGE LABELS SIND ENTFERNT
JSR $E871                       ABSOLUTE ADRESSE, WEIL AUSSERHALB FROM/TO
...
JMP H0228                       MIT LABEL, DA INNERHALB
-----

<↑>T FROM=20F                   TABELLEN-MODUS
 TO=216 M=W /L OUT=             TABELLE IM WORD-FORMAT ALS LABEL
                                AUSGABE AUF DRUCKER (WEIL RETURN)

LISTBILD Z.B.
H020F .WOR H0216
.WOR H050F
.WOR H037F
.WOR H0512
-----
<↑>T FROM=374                   FORMATIERUNG VON ASCII-BYTES AUS BEREICH
 TO=379  M=A /  OUT=U           IN DEN EDITOR MIT MAXIMAL 29 BYTES (RETURN)

LISTBILD Z.B.                   MIT M=B WUERDEN HEX-BYTES ERZEUGT.
H0374 .BYT ',),#(A'
```

*

StDir. Peter Rix, 2350 Neumünster

NRINS - AIM Editor mit Zeilennummern

Das nachfolgende Programm gestattet die automatische Vorgabe von 4-stelligen Zeilennummern bei der Texteingabe in den Editor. Die erste zu vergebende Zeilennummer und das Inkrement von Zeile zu Zeile können beim Eintritt in das Programm beim IN=U/(Device), FROM=(erste Zeilennummer) und /= (Zeileninkrement) vorgegeben werden.

Eine besondere Eigenschaft ist das Auto-Insert: Eine Zeile wird automatisch an der Stelle im Textspeicher abgelegt, die ihrer Nummer entspricht. Dieses Einrangieren erfolgt auch dann, wenn die am Zeilenbeginn zunächst angezeigte Nummer durch DELETE und Übertippen abgeändert wurde. Eine solche Zeile landet zeilenrichtig im Textspeicher. Die danach von NRINS für die Folgezeile vergebene Zeilennummer entspricht der überschriebenen, so daß man an der alten Eintragungsstelle fortfahren kann.

Als Eingabedevices sind Tastatur (IN=U/SPACE) und Cassettenrekorder vorgesehen (IN=U/T). Im lezteren Falle werden die vom Tonband kommenden Textzeilen im Zuge des Einlesens numeriert. - Die Bedienung des AIM-Texteditors bleibt ansonsten unverändert.

Von der Monitorebene herkommend sind mit den Funktionstasten F1, F2 und F3 zusätzliche Dienstleistungen implementiert:

F1 Alle Zeilennummern werden aus dem Editbuffer gelöscht. Der Löschvorgang beginnt bei der Kopfzeile des Buffers.

F2 Teillöschung wie vor, bei der zuletzt angezeigten Zeile beginnend.

F3 Renumerieren des Bufferinhaltes ab zuletzt angezeigter Zeile. Die Vorgaben werden über FROM= /= erfragt.

Der Ablauf dieser Funktionen kann in allen Fällen durch SPACE angehalten und durch ESC abgebrochen werden (wichtig bei nur teilweiser Numerierung oder Löschung). Nach einem Programmhalt bei diesen Funktionen kann durch Betätigen einer beliebigen Taste fortgesetzt oder durch Betätigen von SPACE und ESC gemeinsam und folgener Freigabe vor SPACE ab folgender Zeile abgebrochen werden.

Nach Durchlauf der Lösch- oder Renumerierungsroutine wird mit Anzeige der Bufferkopfzeile gestoppt. Alle Editorbefehle stehen dann wieder wie üblich zur Verfügung. - Ein Durchlaufen der Löschroutinen bei unnumerierten Texten ist unkritisch, weil Fehllöschungen durch Syntaxprüfungen weitghend ausgeschlossen werden.

Die Funktionen F1-F3 wurden eingerichtet, um während der Editierung jede Freiheit für die zusammenhängende Neuvergabe von Zeilenummern und für die Wiederherstellung gleicher Nummernikremente zu haben. Die Funktion F1 ist ferner für die Assemblierung nützlich, denn bekanntlich akzeptiert das Assemblerprogramm keine numerierten Zeilen im Quelltext.

Bei der Programmierung wurde besonderer Wert auf die freie Verschieblichkeit des Maschinencodes gelegt. Es ist durchgehend gelungen, ohne absolute Adressierungen auszukommen, die sich auf den Adressenraum des Programmes selbst beziehen würden. Absolute Adressen betreffen also nur ROM-Routinen des AIM sowie die bekannten Vektoren des Moni-

tors. Bei einer Verschiebung brauchen also nur die den Tasten F1-F3 zugeordneten Adressen geändert zu werden. Das Programm ist könnte damit in einen Festwertspeicher übernommen werden, zumal es wiedereintrittsfähig ist, keine Variablen im Adreßraum enthält und nur die Zeropage-Adressen $0D-$13 für seine Zwecke benutzt.

Erwähnenswert ist die Programmierung der F1-Taste mit einem Unterprogrammaufruf. Dadurch wird nach dessen Abarbeitung sofort auch die nachgeschaltete Funktion F2 ausgeführt.

Dem Autor war es aus zeitlichen Gründen bisher nicht möglich, auch ein Block-Move einzurichten, d.h. die Renumerierung eines Textbereiches von ... bis Zeile mit anschließender Umordnung des Textes entsprechend der neuen Nummernfolge. Im Sinne dieser Zeitschrift als Software-Forum wäre es interessant, hierzu Vorschläge aus der Leserschaft zu erhalten.

```
                ---------------------
0000            P. RIX, 06.08.80
0000            ---------------------
0000            SYMBOLE
0000 NRINS=$E50
0000            ;GGFS. START ABAENDERN
0000 UIN=$100
0000                    MONITOR-VEKTOREN
0000 KEYF1=$10C
0000                    F1-TASTE
0000
0000            ;MONITOR-UNTERPROGRAMME UND -ADRESSEN
0000 ADDIN=$EAAE                                    0000 PRPC=$F53C
0000 ADDN6=$EAFD                                    0000 PSL1=$E837
0000 ADDN8=$EB2B                                    0000 RCHEK=$E907
0000 ATBOT=$F8F9                                    0000 RDRUB=$E95F
0000 ATTOP=$F8DB                                    0000 REENTR=$F6CF
0000 BLANK=$E83E                                    0000 REPLAC=$F93F
0000 BLANK2=$E83B                                   0000 SETBOT=$F8C5
0000 CFLG=$F8B2                                     0000 TIBYTE=$ED3B
0000 CLR=$EB44                                      0000 TOPNO=$F8BC
0000 DLNE=$F74C                                     0000 UP=$F6E9
0000 DOW1=$F8E3                                     0000 UPNO=$F709
0000 FROM=$E7A0                                     0000 WHEREI=$E848
0000 KIFLG=$F3B6                                    0000 PRIFLG=$A411
0000 PACK=$E984                                     0000 DIBUFF=$A438
0000 PATC12=$FEF9                                   0000 ADDR=$A41C
0000 PLNE=$F727

0000            ;ZEROPAGE DES EDITORS UND DIESES PROGRAMMES
0000 FLAG1=$D                       ;FLAG STEUERT NR-GENERIERUNG
0000 FLAG2=$E                       ;BETRIEBSART TAPE/TASTAUR/BUFFER
0000 FLAG3=$F                       ;FLAG KENNZEICHNET ZEILENENDE
0000 NUMMR=$10                      ;ZEILEN-#, 2 BYTE
0000 IKREM=$12                      ;SCHRITTWEITE, 2 BYTE
0000 NOWLN=$DF                      ;PARAMETER DES EDITORS
0000 OLDLEN=$E9
0000LENGTH=$EA
0000 SVNEU=$FD
```

```
                ;--------------------
0000            ;BELEGUNG USER-INPUT
0000                    *=UIN
0100            0900    .WOR BFEIN      ;USER-INPUT LINKVEKTOR
010A            ;--------------------
010A            ;BELEGUNG DER TASTEN F1-F3
010A                    *=KEYF1         ;F1: ZUR BUFFER-KOPFZEILE UND F2-FUNKTION
010C            20BCF8  JSR TOPNO
010F            4CC40F  JMP DELETE      ;F2: LOESCHEN AB AKTUELLER ZEILE
0112            4C6D0F  JMP RENUMR      ;F3: NUMERIEREN BUFFER AB AKTUELLER ZEILE
0115            ;--------------------
0115            ;HAUPTPROGRAMM
0115                    *=NRINS
0E50 SORT                               ;EINSORTIEREN ZEILE
0E50            4E11A4  LSR PRIFLG      ;DRUCKER AUS
0E53            20E3F6  JSR DOW1        ;1 ZEILE AUFWAERTS
0E56            A5DF    LDA NOWLN       ;ZEILENADRESSE SPEICHERN
0E58            85FD    STA SVNEU
0E5A            A5E0    LDA NOWLN+1
0E5C            85FE    STA SVNEU+1
0E5E            B01B    BCS EINSZ       ;ZUM EINSETZEN, FALLS KOPFZEILE ERREICHT IST
0E60 AUFW
0E60            20DBF8  JSR ATTOP       ;PRUEFEN AUF KOPFZEILE
0E63            B016    BCS EINSZ       ;FALLS ERREICHT, ZEILE EINSETZEN
0E65            20E3F6  JSR DOW1        ;SONST 1 ZEILE AUFWAERTS
0E68            A000    LDY #$00
0E6A NRVGL                              ;VERGLEICH NEUE NR. UND NR. DER BUFFERZEILE
0E6A            B1FD    LDA (SVNEU),Y   ;(4 ZIFFERN)
0E6C            D1DF    CMP (NOWLN),Y
0E6E            30F0    BMI AUFW        ;1 ZEILE AUFWAERTS, FALLS NEUE # KLEINER
0E70            D038    BNE ABW         ;ABWAERTS, FALLS >
0E72            C8      INY
0E73            C004    CPY #$04
0E75            30F3    BMI NRVGL
0E77            A901    LDA #$01        ;NUMMERN SIND IDENTISCH
0E79            85FE    STA SVNEU+1     ;ZEIGER AUF 1 SETZEN
0E7B EINSZ
0E7B            A5EA    LDA LENGTH
0E7D            C906    CMP #$06

0E7F            301B    BMI ZDEL+7
0E81            D007    BNE *+$09
0E83            AD3DA4  LDA DIBUFF+5    ;WENN ZEILENLAENGE=0 UND 6. ZCH.="/"
0E86            C92F    CMP #"/"        ;ZEILE LOESCHEN
0E88            F00B    BEQ ZDEL
0E8A            20B6F8  JSR KIFLG
0E8D            85E9    STA OLDLEN
0E8F            203FF9  JSR REPLAC      ;ZEILE IN BUFFER EINSCHIEBEN
0E92            2009F7  JSR UPNO        ;1 ZEILE ABWAERTS
0E95 ZDEL
0E95            C6FE    DEC SVNEU+1     ;ZEIGER DEKREMENTIEREN

0E97            D003    BNE *+$05
0E99            204CF7  JSR DLNE        ;ZEILE GLEICHER NR. LOESCHEN
0E9C            20C5F8  JSR SETBOT
0E9F            20E3F6  JSR DOW1        ;ORIGINALZEILE AM BUFFERENDE LOESCHEN
```

```
0EA2          204CF7 JSR DLNE
0EA5          0E11A4 ASL PRIFLG      ;PRINTER-FLAG RESTAURIEREN
0EA8          9077   BCC ZLNR        ;ZUM ZEILEN-NUMMER-GENERATOR
0EAA ABW
0EAA          2009F7 JSR UPNO        ;1 ZEILE ABWAERTS IM BUFFER
0EAD          18     CLC
0EAE          90CB   BCC EINSZ
0EB0          ---------------------
0EB0          SORTIER-PRUEFUNG
0EB0 SRTT
0EB0          A50D   LDA FLAG1       ;NICHT SORTIEREN
0EB2          C9F0   CMP #$F0        ;FALLS NEUINITIALISIERUNG
0EB4          F06B   BEQ ZLNR        ;ODER TAPE-EINGABE
0EB6          A50E   LDA FLAG2
0EB8          C954   CMP #'T
0EBA          F065   BEQ ZLNR        ;SONDERN NUMMER GENERIERREN
0EBC          D092   BNE SORT        ;ZUM EINSORTIEREN
0EBE          ---------------------
0EBE          CARRIAGE RETURN-TEST
0EBE CRTEST
0EBE          C90D   CMP #$0D        ;'CR'?
0EC0          F020   BEQ CREIN
0EC2          A200   LDX #$00        ;KEIN CR
0EC4          860F   STX FLAG3       ;DANN FLAG3=0
0EC6          C003   CPY #$03
0EC8          D017   BNE EDTR+2      ;ZEILENLAENGE <4, DANN ZURUECK Z. EDITOR
0ECA          203EE8 JSR BLANK       ;WENN LAENGE=4, DANN SPACE HINTER NR.
0ECD          A000   LDY #$00
0ECF          20FDEA JSR ADDN6       ;GUELTIGE NR.?
0ED2          B031   BCS NRERR+3     ;ERROR-AUSGABE BEI NEIN
0ED4          E8     INX
0ED5
0ED5 VERGL                           ;VERGLEICH ZEILEN-NR. MIT GENERIERTER NR.
0ED5          B510   LDA NUMMR,X
0ED7          DD1CA4 CMP ADDR,X
0EDA          D020   BNE NRNEU       ;VERZWEIGEN, WENN NR. VERAENDERT WURDE
0EDC          CA     DEX
0EDD          F0F6   BEQ VERGL
0EDF EDTR                            ;ZURUECK ZUM EDITOR
0EDF          A920   LDA #' '
0EE1          60     RTS
0EE2          ---------------------
0EE2          CR-EINGABE
0EE2 CREIN
0EE2          48     PHA
0EE3          20C5F8 JSR SETBOT      ;ZEIGER AUF TEXTENDE SETZEN
0EE6          68     PLA
0EE7          A60F   LDX FLAG3       ;2. 'CR'?
0EE9          F003   BEQ *+$05       ;VERZWEIGEN, WENN 1. CR
0EEB          A000   LDY #$00        ;2. CR. NR. LOESCHEN
0EED          60     RTS             ;ZURUECK ZUM EDITOR
0EEE          C60F   DEC FLAG3       ;1. CR. DANN FLAG3<0
0EF0          C004   CPY #$04        ;ZEILENLAENGE <5?
0EF2          300E   BMI NRERR       ;JA, UNGUELTIGE NR., ERROR-AUSGABE
0EF4          A220   LDX #' '        ;5. ZEICHEN EIN SPACE?
0EF6          EC3CA4 CPX DIBUFF+4
0EF9          D007   BNE NRERR       ;NEIN, UNGUELTIGE NR. ERROR-AUSGABE
0EFB          60     RTS             ;ZURUECK ZUM EDITOR
```

```
0EFC NRNEU
0EFC          A9FE   LDA #$FE       ;FLAG >=0 + ZURUECK Z. EDITOR
0EFE          850D   STA FLAG1      ;NR-GENERATOR WIRD ALTE NR ADC
EN
0F00          D00D   BNE EDTR
0F02          ---------------------
0F02          ERROR-AUSGABE
0F02 NRERR
0F02          202BEB JSR ADDN8      ;AUSGABE
0F05          850D   STA FLAG1      ;FLAG >=0
0F07          D018   BNE ZLNR       ;ZUM ZEILEN-NUMMERN-GENERATOR
0F09          ---------------------
0F09          EINGANG VOM EDITOR
0F09 BFEIN
0F09          9076   BCC NRWAHL     ;VERZWEIGEN B. 1. AUFRUF VON WHEREI
0F0B          A50F   LDA FLAG3      ;ZEILENENDE (CR)?
0F0D          30A1   BMI SRTT       ;FALLS JA: SORTIERPRUEFUNG
0F0F          A50E   LDA FLAG2
0F11          C954   CMP #'T
0F13          F006   BEQ TAPE       ;ZEICHEN AUS BANDPUFFER ODER TASTATUR
0F15          205FE9 JSR RDRUB      ;LESEN + ZUR CR-PRUEFUNG VERZWEIGEN
0F18          18     CLC
0F19          90A3   BCC CRTEST
0F1B TAPE
0F1B          203BED JSR TIBYTE
0F1E          18     CLC
0F1F          909D   BCC CRTEST
0F21          ---------------------
0F21          ZEILEN-NUMMER-GENERATOR
0F21 ZLNR
0F21          2007E9 JSR RCHEK      ;TASTATURABFRAGE AUF 'ESCAPE'
0F24          A60D   LDX FLAG1      ;FALLS FLAG1 >=0
0F26          D00E   BNE NRAUS      ;KEINE NEUE NR GENERIEREN, ALTE NR AUSGEBEN
0F28          F8     SED
0F29          A2FF   LDX #$FF
0F2B          18     CLC
0F2C NEU
0F2C          B513   LDA IKREM+1,X  ;NR UM (DEZIMALE) SCHRITTWEITE ERHOEHEN
0F2E          7511   ADC NUMMR+1,X  ;FALLS FLAG1=0
0F30          9511   STA NUMMR+1,X
0F32          E8     INX
0F33          F0F7   BEQ NEU
0F35          D8     CLD
0F36          ---------------------
0F36          NR-AUSGABE
0F36 NRAUS
0F36          A50E   LDA FLAG2
0F38          C954   CMP #'T    ;PRINTER ABSCHALTEN, FALLS TAPE-EINGABE
0F3A          D003   BNE *+$05
0F3C          EE11A4 INC PRIFLG
0F3F          A610   LDX NUMMR
0F41          A511   LDA NUMMR+1
0F43          2042F5 JSR PRPC+6     ;NR AN DISPLAY/PRINTER AUSGEBEN
0F46          20F8FE JSR PATC12     ;DRUCKER FLAG RESTAURIEREN
0F49          A900   LDA #$00
0F4B          850D   STA FLAG1      ;FLAG1=0: VORBEREITUNG DER FOLGE-NR
0F4D          A005   LDY #$05
0F4F          A50E   LDA FLAG2      ;PRUEFEN BFEHLSQUELLE
0F51          D0BE   BNE BFEIN+8    ;BAND- ODER TASTATUREINGABE, WENN FLAG2 >=0
```

```
                ;--------------------
0F53            ;BUFFER-RENUMERIERUNG, FLAG2=0
0F53 RENR
0F53  20E9F8    JSR ATBOT
0F56  B026      BCS REND        ;ZUM ENDE, FALLS SCHLUSSZEILE ERREICHT IST
0F58  203FF9    JSR REPLAC      ;EINFUEGEN ZEILEN-NR
0F5B  A004      LDY #$04
0F5D  A920      LDA #' '
0F5F  91DF      STA (NOWLN),Y   ;EINFUEGEN 'SPACE'
0F61  2044EB    JSR CLR
0F64  2027F7    JSR PLNE        ;ZEILE AN DISPLAY/PRINTER AUSGEBEN
0F67  2009F7    JSR UPNO        ;1 ZEILE ABWAERTS
0F6A  18        CLC
0F6B  9007      BCC NEXTNR
0F6D            ;--------------------
0F6D            ;EINGANG ZUM MONITOR MIT F3-TASTE
0F6D RENUMR
0F6D  A900      LDA #$00        ;FLAG2=0
0F6F  F016      BEQ NRWAHL+6    ;ZUR WAHL VON START-NR + SCHRITTWEITE
0F71  20B2F8    JSR CFLG
0F74 NEXTNR                     ;VERSCHIEBEN RESTLICHE BUFFERZEILEN
0F74  A000      LDY #$00        ;UM 5 BYTE ZU HOEHEREN ADRESSEN VORBEREITEN
0F76  84E9      STY OLDLEN
0F78  A004      LDY #$04
0F7A  84EA      STY LENGTH
0F7C  10A3      BPL ZLNR        ;ZUM ZEILEN-NR-GENERATOR
0F7E            ;--------------------
0F7E            ;TEXTENDE
0F7E REND
0F7E  4CCFF6    JMP REENTR      ;KOPFZEILE ANZEIGEN
0F81            ;--------------------
0F81            ;INITIALISIERUNG
0F81 NRWAHL
0F81  2037E8    JSR PSL1        ;/SP TASTATUREINGABE
0F84  205FE9    JSR RDRUB       ;/T BANDEINGABE
0F87  850E      STA FLAG2       ;/W WIEDERAUFNAHME
0F89            ;ALTE NR-FOLGE BEI TASTATUREINGABE
0F89  C957      CMP #'W
0F8B  F027      BEQ INIT
0F8D  2044EB    JSR CLR
0F90  20A3E7    JSR FROM        ;EINGABE STARTNUMMER 'FROM=XXXX..'
0F93  B0F8      BCS *-6
0F95  AD1CA4    LDA ADDR
0F98  8510      STA NUMMR       ;SPEICHERN STARTNUMMER
0F9A  AD1DA4    LDA ADDR+1      ;LETZTE 4 ZEICHEN IN 2 BYTE
0F9D  8511      STA NUMMR+1
0F9F INKR
0F9F  20B8E9    JSR BLANK2      ;EINGABE SCHRITTWEITE
0FA2  20E7E9    JSR PSL1        ;'/=XXXX..'
0FA5  20AEE6    JSR ADDIN
0FA8  B0F5      BCS INKR
0FAA  AD1CA4    LDA ADDR
0FAD  8512      STA INKREM
0FAF  AD1DA4    LDA ADDR+1
0FB2  8513      STA INKREM+1
0FB4 INIT
0FB4  A9F0      LDA #$F0        ;FLAG1, FLAG3 INITIALISIEREN
0FB6  8580      STA FLAG1
```

```
0FB8          850F    STA FLAG3
0FBA          2044EB  JSR CLR
0FBD          A50E    LDA FLAG2
0FBF          F0B0    BEQ RENUMR+4    ;ZUR BUFFER-RENUMERIERUNG, FALLS FLAG2=0
0FC1          4C50E8  JMP WHEREI+8    ;SONST ZUR BANDEINGABE-INITIALISIERUNG
0FC4
0FC4          ;ODER TASTATUREINGABE
0FC4          --------------------
0FC4          LOESCHEN BUFFERZEILEN-NUMMERN, F2-TASTE
0FC4 DELETE
0FC4          2044EB  JSR CLR
0FC7          20B2F8  JSR CFLG
0FCA          20FCF6  JSR UP+3
0FCD          B027    BCS DEND        ;ZEILE AM DISPLAY/PRINTER AUSGEBEN
0FCF          ;ZUM ENDE, FALLS SCHLUSSZEILE ERREICHT IST
0FCF          2007E9  JSR RCHEK       ;TASTATURABFRAGE AUF ABBRUCH
0FD2          A000    LDY #0
0FD4          84EA    STY LENGTH
0FD6 ZIFFER
0FD6          B1DF    LDA (NOWLN),Y   ;ZEICHEN AUS BUFFER LESEN
0FD8          C8      INY             ;PRUEFEN AUF GUELTIGE HEXZAHL
0FD9          C90D    CMP #$D         ;DER FORM XXXX -SPACE
0FDB          F014    BEQ NEXTZ       ;FALLS ERFUELLT, ZUM NR-LOESCHEN
0FDD          C920    CMP #' '        ;FALLS NICHT ERFUELLT, NAECHSTE Z

0FDF          F007    BEQ CLEAR
0FE1          2084EA  JSR PACK
0FE4          90F0    BCC ZIFFER
0FE6          B009    BCS NEXTZ
0FE8 CLEAR
0FE8          C005    CPY #5
0FEA          D005    BNE NEXTZ
0FEC          84E9    STY OLDLEN
0FEE          203FF9  JSR REPLAC      ;NR. LOESCHEN (5 ZEICHEN)
0FF1 NEXTZ
0FF1          20F9F6  JSR UP          ;1 ZEILE ABWAERTS UND ZEILE AN DISPLAY/PTR
0FF4          90D9    BCC DELETE+11   ;ZUR TASTATURABFRAGE, WENN NOCH KEIN TEXTEN

0FF6 DEND                             ;TEXTENDE
0FF6 LABEL
0FF6          4CCFF6  JMP REENTR      ;KOPFZEILE ANZEIGEN
0FF9                  .END
```

*

****

KLEINANZEIGEN DER LESER

AIM-Netzteil +5, +24 V DM 150,-. Videokarte anschlußfertig 64x16 Zeichen (s. FUNKSCHAU 6/80) incl. Mod., Zusatzschalt. + Softw. DM 250,-. 32 KByte DM 770,-. M. Roßmöller Tel. 0228-22 48 37.

Z u verkaufen: ISE PC22-Computer. Commodore-BASIC, in schönem Gehäuse mit 24 K Anwenderspeicher, 12 Zoll Schirm, Tastatur und 2x80 Kbyt Floppy-Disk. Neu in originaler Verpackung, alles zus. DM 6750,-. Houghton, Arabellastr. 58, 8000 München 81, Tel. (089) 91 46 28.

Michael Zimmermann, 6102 Pfungstadt

AIM 65 als Simplexfernschreiber am KIM-1

1. Warum den AIM als Fernschreiber benutzen?

Sicher wird es einer Reihe von AIM-Benutzern, die vom KIM her aufgestiegen sind, ähnlich wie dem Verfasser gehen: Mit Benutzung des komfortablen AIM gerät der KIM mehr und mehr in Vergessenheit und verstaubt irgendwo im Regal. Sicher gibt es noch eine Reihe von Möglichkeiten, bei denen auch der KIM seine Fähigkeiten einsetzen kann, ohne daß man als Benutzer auf die vielfältigen Hilfen des AIM zu verzichten brauchte. Hierzu ist in irgendeiner Weise eine Verbindung zwischen AIM und KIM zu realisieren.

Die vom Aufwand her einfachste Verbindung dürfte in einer reinen Programmübertragung vermittels Kassetten bestehen, ein Weg, der hier als zu zeitraubend abgelehnt werden muß. Als bessere Möglichkeit bietet sich die auf beiden Platinen realisierte 20 mA-Schnittstelle an, wobei die Systeme durch einen Optokoppler getrennt werden sollten. Entsprechend seiner Ausstattung ist der AIM als Fernschreiber in dieser Zusammenschaltung prädestiniert. Über die Funktionen eines einfachen Fernschreibers hinaus ergeben sich noch weitere Einsatzmöglichkeiten, besprochen werden sollen.

2. Möglichkeiten des AIM als Fernschreiber zum KIM

Bei einer Verbindung über die 20 mA-Schnittstelle ist der AIM als Fernschreiber zunächst einmal der passive Teil. Dies bedeutet, daß dem Benutzer die KIM-Monitor-Befehle, soweit sie bei der Fernschreibereingabe vorgesehen sind, zur Verfügung stehen. Der Benutzer ist hierbei auf den jeweiligen Ausbau des KIM beschränkt, Erweiterungen für den AIM sind für den KIM hier nicht nutzbar.

Ein Nachteil des AIM als Fernschreiber ist das Fehlen der Lochstreifenperipherie. Dieser Nachteil schlägt schnell in einen Vorteil um, wenn man bedenkt, daß das Ladeformat für beide Maschinen identisch ist. Hierdurch wird es möglich, Speicherinhalte zwischen beiden Maschinen auszutauschen, wobei der KIM sich so verhält, als erhielte er seine Eingaben vom Lochstreifen, bzw. als würde er den Inhalt seines Speichers auf einen Lochstreifen stanzen. Diese Verbindungen lassen sich, wie wir bei der Diskussion des entsprechenden Programmes sehen werden unschwer über User-Input bzw. Output realisieren.

Hier bieten sich vielfältige Möglichkeiten, die über einen reinen Austausch von Speicherinhalten hinausgehen. So ist es z.B. möglich, vom AIM her direkt in den Speicher des KIM zu assemblieren.

3. Beschreibung des Fernschreiber-Programmes

Im Folgenden sollen die Schritte zur Realisierung des AIM-Fernschreibers dargelegt und die erforderlichen Programmschritte dem Leser nahegebracht werden. - Das Hauptprogramm beginnt damit, daß eine Reihe von Vektoren gefüllt werden. Zuerst F3 (Funktionstaste), eben mit der STartadresse das Hauptprogrammes, damit jederzeit ein Restart über die F3-Taste möglich wird.

Sodann wird der User-Input-Vektor mit der Adresse von TRKA gefüllt (Transfer from KIM to AIM). Als letzter Vektor wird OUT bedient, der

auf die Adresse von TRAK (Transfer from AIM to KIM) gesetzt wird. An die Vektoren schließt sich das Setzen der Baudrate an und darauf das Übertragen eines DELETE-Zeichens zum KIM. Dieses ist erforderlich, damit der KIM sich auf die Geschwindigkeit des TTY einstellen kann.

Auf diese Initialisierung folgt die Prüfung, ob und welche Eingabe vorliegt. Zuerst wird geprüft, ob im Empfangsteil der 20 mA-Schleife ein Low-Pegel anliegt. Ist dies der Fall, so wurde ein Start-Bit erkannt. Es wird dann zur Verarbeitung einer TTY-Eingabe verzweigt. - Darauf folgt die Prüfung, ob eine der Tasten betätigt wurde. Ist dies nicht der Fall, so wird zur TTY-Prüfung zurückverzweigt.

Die Tastenabfrage soll hier näher erläutert werden, da diese Form der Prüfung etwas ungewöhnlich erscheint. Leider ist am AIM-Monitor keine Unterroutine realisiert, die eine Prüfung auf eine gedrückte Taste ermöglicht. Auch die Routine GETKEY wartet auf einen Tastendruck und kommt mit dem Wert der betätigten Taste zurück. Da andererseits der TTY-Eingang nicht interruptfähig ist, kann die GETKEY-Routine hier nicht benutzt werden. Bei der Abfrage auf eine gedrückte Taste wird auf die Hardware-Ebene des Tastaturanschlusses heruntergegangen. Die Tastatur liegt zwischen den Ports A und B des RIOT (6532), der Port B ist hierbei durch Pull-up-Wiederstände auf 1 gezogen. Schreibt man nun in Port A $00 hinein, so geschieht solange nichts (d.h. es liegt an Port B weiter $FF an), wie keine Taste betätigt ist. - wird aber eine Taste gedrückt, so wird ihre Verbindung zu Port B durch den Kontakt auf Null gezogen, an Port B liegt also nicht mehr $FF an.

Wurde ein Tastendruck in dieser Form erkannt, so wird in KEYHAN diese Taste eingelesen, hierbei mit GETKY, da GETKEY wartet, bis keine Taste mehr betätigt ist. Dieser Wert wird gesichert und dann abgewartet, bis die Taste wieder freigegeben wurde. - Ist dies der Fall, so wird der Wert wieder hergestellt, und die eigentliche Verarbeitung kann beginnen.

Zuerst wird geprüft, ob es sich um den Wert der F1-Taste handelt. Ist dies der Fall, so wird die Eingabe als Line-Feed interpretiert, weil die eigentliche LF-Taste vorab dekodiert wird und von GETKEY nicht mit einem Wert zurückkommt. Andererseits soll dem Benutzer auch nicht zugemutet werden, daß er die Kombination CTRL-J für LF zu drücken hat. - Diese F1-Taste führt zu einem Löschen der Anzeige und zu einer Umwertung in $0A.

Im Anschluß an die F1-Behandlung wird auf die F2-Taste geprüft, diese bewirkt einen Rücksprung in den Monitor. Alle anderen Zeichen sind gültig und werden über die 20 mA-Schnittstelle übertragen, anschließend wird zu weiterer Prüfung zurückverzweigt.

In der folgenden TTY-Behandlung wird zuerst das Zeichen über die Stromschleife eingelesen. Bei $00 und $0A wird sofort zur Prüfschleife zurückverzweigt. Andere Zeichen werden an die Displayausgabe übergeben. Darauf wird gewartet, bis an der Schnittstelle wieder ein High-Pegel, das Stop-Bit anliegt. Und darauf wird auch in diesem Falle zur Prüfschleife zurückverzweigt.

Es folgen die Transfer-Routinen, zuerst die Übertragung von KIM zu AIM. Beim OPEN dieser Routine (User-Input mit gelöschtem Carry) wird über den Q-Command vom KIM ein Speicherauszug angefordert, die zugehörigen Bereiche sind im KIM vorab entsprechend niederzulegen. Beim eigentlichen User-Input werden die vom KIM ankommenden Zeichen zuerst auf der Anzeige ausgegeben und dann zur Verarbeitung dem User-Input übergeben. - Beim im Programm folgenden Transfer zum KIM wird

beim OPEN dem KIM der L-Command übergeben und damit ein Speicherladen veranlaßt. Beim User-Output selbst werden die Zeichen lediglich über die Stromschleife übermittelt.

4. Betrieb des Fernschreiberprogrammes

Bereits in der Programmbeschreibung wurde auf eine Reihe von Besonderheiten beim Betrieb hingewiesen. Dies betrifft einmal die Funktionstasten:

F1 Substitution für Line-Feed
F2 Rücksprung zum AIM-Monitor
F3 Einsprung in das Fernschreibprogramm.

Weitere Besonderheiten sind bei UIN und UOUT zu beachten: Nahezu problemlos ist der Einsatz von UOUT, der Benutzer wird beim DUMP über FROM-TO auf den entsprechenden Speicherbereich gepromptet. Dieser Bereich muß natürlich im KIM, in den ja geladen werden soll, auch vorhanden sein. Auch die Verwendung im Zusammenhang mit dem Assembler macht keine Probleme. Weitere Anwendungen von UOUT scheinen nicht sinnvoll.

Beim Einsatz von UIN ergibt sich eine Schwierigkeit darin, daß der AIM weitere Eingaben erwartet, die vom KIM nicht mehr gegeben werden. Der AIM hängt sich also auf und muß mit der RESET-Taste zurückgeholt werden. Eine genaue Analyse beider Betriebssysteme zeigt hier eine geringe Inkompatibilität, die aber nur den Schlußsatz betrifft und somit keinen Datenbverlust nach sich zieht. - Weiterhin ist bei UIN zu beachten, daß die entsprechenden Adressen des KIM mit dem zu übertragenden Speicherbereich zu füllen sind und daß dieser Bereich eine Ensprechung im Speicher des AIM hat.

Weitere Probleme beim Austausch zwischen AIM und KIM wurden vom Verfasser nicht festgestellt. Alle Übertragungen wurden mit 300 Baud gefahren, bei höheren Raten ergaben sich Fehler. - Leider stehen dem Verfasser keine weiteren Systeme zur Verfügung um Zusammenschaltungen zu prüfen.

```
0000          ---------------------
0000          MONTIOR-ENTRY-POINTS
0000
0000 RSET           =$E0BF          ;MONITOR-RETURN
0000 GETKY          =$EC43          ;GET KEY
0000 OUTTTY         =$EEA8          ;OUTPUT ACC TO TTY
0000 OUTDP          =$EEFC          ;OUTPUT ACC TO DISPLAY
0000 GETTTY         =$EBDB          ;GET A CHAR FROM TTY
0000          ---------------------
0000          MONITOR-RAM-ADRESSES
0000
0000 UIN            =$108           ;USER INPUT VECTOR
0000 UOUT           =$10A           ;USER OUTPUT VECTOR
0000 KEYF3          =$112           ;F3-KEY
0000 CNTH30         =$A417          ;BAUD RATE AND
0000 CNTL30         =$A418          ;DELAY FOR TTY
0000          ---------------------
0000          MONITOR-I/O-ADRESSES
0000
0000 KEYB00         =$A480          ;RIOT DATA-REGISTER A
0000 KEYB02         =$A482          ;DATA REGISTER B
0000 TTY            =$A800          ;VIA ADRESS
0000
```

```
0000            ;--------------------
                ;SPEED-CONSTANTS HERE 300 BAUD
0000            ;
0000 SPEEDL     =$C2
0000 SPEEDH     =$0C
0000            ;--------------------
0000            ;
0000            ;
0000            ;--------------------
0000            ;MAIN PROGRAMM
0000            ;
0000                    *=$F00
0F00 START      *=*
0F00            ;--------------------
0F00            ;SET VECTORS FOR LATER USAGE
0F00            ;
0F00            A900    LDA #<START     ;SET F3 FOR REENTRY
0F02            8D1301  STA KEYF3+1
0F05            A90F    LDA #>START
0F07            8D1401  STA KEYF3+2
0F0A            A94C    LDA #$4C
0F0C            8D1201  STA KEYF3
0F0F            A982    LDA #<TRKA      ;SET USER INPUT
0F11            8D0801  STA UIN
0F14            A90F    LDA #>TRKA
0F16            8D0901  STA UIN+1
0F19            A990    LDA #<TRAK      ;SET USER OUTPUT
0F1B            8D0A01  STA UOUT
0F1E            A90F    LDA #>TRAK
0F20            8D0B01  STA UOUT+1
0F23            A90C    LDA #SPEEDH     ;SET SPEED
0F25            8D17A4  STA CNTH30
0F28            A9C2    LDA #SPEEDL
0F2A            8D18A4  STA CNTL30
0F2D            A9FF    LDA #$FF        ;TRANSMIT DEL
0F2F            20A8EE  JSR OUTTTY
0F32            ;
0F32            ;GENERAL TEST AND PROCESS LOOP
0F32            ;
0F32 LOOP       *=*
0F32            2C00A8  BIT TTY         ;TEST TTY-INPUT FOR START-BIT
0F35            5035    BVC TTYHAN      ;WE FETCHED A START-BIT AND PROCESS IT
0F37            A900    LDA #$00        ;TEST KEYBOARD, WRITE ZERO IN PORT A
0F39            8D80A4  STA KEYB00
0F3C            AD82A4  LDA KEYB02      ;NOW GET PORT B, PULLED UP, AND SEE
0F3F            C9FF    CMP #$FF        ;IF ALL INPUTS ARE STILL UP
0F41            F0EF    BEQ LOOP        ;THEY ARE ALL UP, NO KEY WAS CLOSED
0F43            ;--------------------
0F43            ;HANDLE KEYBOARD-ENTRY IF KEY CLOSED
0F43 KEYHAN     *=*
0F43            2043EC  JSR GETKY       ;GET KEYBOARD-ENTRY
0F46            48      PHA             ;SAVE CHARACTER
0F47 KEY10      *=*
0F47            A900    LDA #$00        ;WAIT TILL KEY IS RELEASED
0F49            8D80A4  STA KEYB00
0F4C            AD82A4  LDA KEYB02
0F4F            C9FF    CMP #$FF
0F51            D0F4    BNE KEY10
0F53            68      PLA             ;RESTORE KEY
```

```
0F54          C95B   CMP #'[          ;CHECK FOR F1=LF-SUBSTITUTE
0F56          D007   BNE KEY20        ;NO F1, SKIP
0F58          A90D   LDA #$0D         ;CLEAR DISPLAY BY CR
0F5A          20FCEE JSR OUTDP
0F5D          A90A   LDA #$0A         ;NOW LOAD LF FOR PROCESSING
0F5F KEY20            *=*
0F5F          C95D   CMP #']          ;END TTY
0F61          D003   BNE KEY30        ;GO TO FURTHER PROCESSING
0F63          4CBFE0 JMP RSET         ;RETURN TO MONITOR
0F66 KEY30            *=*
0F66          20A8EE JSR OUTTTY       ;NOW SEND CHARACTER TO KIM
0F69          4C320F JMP LOOP         ;AND BRANCH BACK TO LOOP
0F6C          ------------------
0F6C          HANDLE TTY-INPUT
0F6C TTYHAN           *=*
0F6C          20DBEB JSR GETTTY       ;GET CHARACTER
0F6F          C900   CMP #$00         ;NUL GO BACK TO LOOP
0F71          F0BF   BEQ LOOP
0F73          C90A   CMP #$0A         ;LF  GO BACK TO  LOOP
0F75          F0B8   BEQ LOOP
0F77          20FCEE JSR OUTDP        ;OUTPUT CHARACTER ON DISPLAY
0F7A TTY10            *=*
0F7A          2C00A8 BIT TTY          ;TEST TTY-INPUT UNTIL
0F7D          50FB   BVC TTY10        ;WE GOT THE STOP-BIT
0F7F          4C320F JMP LOOP         ;GO BACK TO LOOP
0F82          ------------------
0F82          TRANSFER MEMORY FROM KIM TO AIM
0F82
0F82 TRKA             *=*
0F82          B005   BCS TR10         ;SKIP IF NOT OPEN
0F84          A951   LDA #'Q'         ;START Q-COMMAND ON KIM
0F86          4CA8EE JMP OUTTTY       ;AND SEND IT TO KIM
0F89 TR10             *=*
0F89          20DBEB JSR GETTTY       ;GET CHARACTER
0F8C          20FCEE JSR OUTDP        ;REPEAT IT ON DISPLAY
0F8F          60     RTS              ;AND RETURN TO UIN
0F90          ------------------
0F90          TRANSFER FROM AIM TO KIM
0F90
0F90 TRAK             *=*
0F90          B005   BCS TR50         ;CHARACTER IS NOT OPEN
0F92          A94C   LDA #'L'         ;IF OPEN START L-COMMAND ON KIM
0F94          4CA8EE JMP OUTTTY       ;AND SEND IT TO KIM
0F97 TR50             *=*
0F97          68     PLA
0F98          4CA8EE JMP OUTTTY       ;SEND CHARACTER TO KIM
0F9B                  END
```

*

****

Dem Erstversand dieses Heftes Nr. 15 liegt ein Prospekt der Firma Software-Verbund Mikrocomputer GmbH in München bei.

Dr. Fritz Mayer-Lindenberg, Bielefeld

Musikerzeugung mit dem AIM 65

Unzweifelhaft ist ein Heimcomputer um so interessanter, je vielfältiger seine Ein- und Ausgabemöglichkeiten sind. Dieser Aufsatz beschreibt, wie sich an den AIM 65 ein programmierbarer Musiksynthesizerbaustein anschließen läßt, auf dem der Rechner dreistimmig "spielen" kann. Neben musikalischen Anwendungen erlaubt eine solche Ausgabemöglichkeit z.B. auch, akustische Signale zu erzeugen, die in einem längeren Programmlauf über den jeweiligen Zustand des Rechners informieren.

Bei dem Baustein handelt es sich um den seit einiger Zeit angebotenen AY-3-8910 von General Instruments (Preis ca. DM 33). Er ist für die Zusammenarbeit mit einem Mikroprozessor, den CP 1610 gedacht und erlaubt die Programmierung von "Hüllkurven", Amplituden und Frequenzen der drei Stimmen. Wahlweise kann auch "Rauschen" erzeugt werden. Zudem enthält der Baustein zwei beliebig verwendbare 8-Bit I/O-Tore. Er wird vom Mikroprozessor als eine Gruppe von 16 Speicherstellen angesprochen. Tabelle 1 zeigt die Bedeutung der Adressen.

Leider verwendet der CP 1610 einen gemultiplexten Daten- und Adreßbus und - verglichen mit dem 6502 - relativ komplizierte Bus-Steuersignale, so daß der AY-3-8910 nicht ohne weiteres durch den 6502 angesteuert werden kann. Tabelle 2 zeigt die erforderlichen Bussignale, Bild 1 die Anschlußbelegung der IS.

Tabelle 1: Die Register des AY-3-8910

00 8 Bit Periode A fein
01 4 Bit Periode A grob
02 dito Kanal B
03 "
04 dito Kanal C
05 "
06 5 Bit Periode des Rauschgenerators
07 Enablebits: $\overline{IN}$/OUT Tor 1, $\overline{Tor\ 22}$
$\overline{Rauschen}$ Kanal C, B, A, $\overline{Ton}$ Kanal C, B, A
08 5 Bit Amplitude Kanal A
09 dito Kanal B
0A dito Kanal C
0B 8 Bit Hüllkurvenperiode fein
0C dito grob
0D 4 Bit Auswahl und Triggerung von 8 Hüllkurven, 08-0F
0E Daten Tor A
0F Daten Tor B

Tabelle 2: Bussignale (Auswahl)

Operation	BDIR	BC2	BC1
inaktiv	0	0	0
inaktiv	0	1	0
Adresse schreiben	1	0	0
Daten schreiben	1	1	0
Daten lesen	0	1	1

Eine Möglichkeit, den AY-3-8910 dennoch am AIM zu betreiben, ergibt sich, indem man die Ansteuerung programmgesteuert durch die VIA vornehmen läßt. Dies demonstriert zugleich die Vielseitigkeit der VIA.

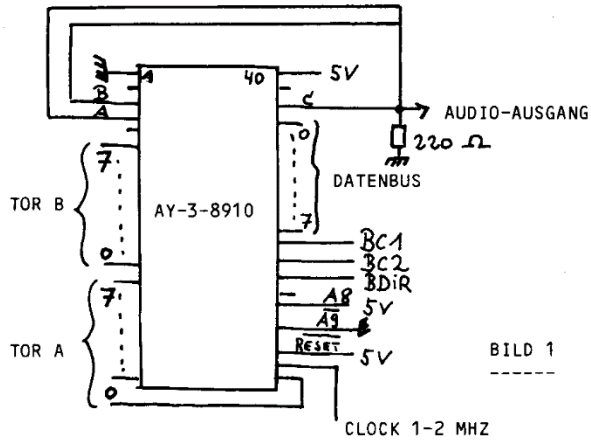


BILD 1

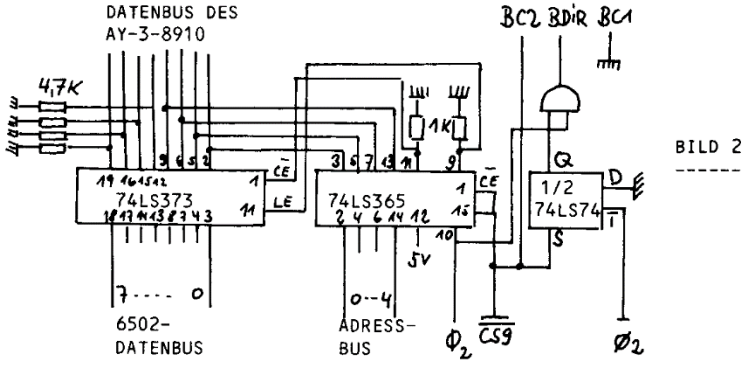


BILD 2

Diese Methode ist natürlich langsam, genügt aber für musikalische Anwendungen. Unten sind entsprechende Unterprogramme zum Setzen der Adresse, Schreiben und Lesen aufgelistet. Der Datenbus des AY-3-8910 ist mit Port A der VIA zu verbinden, die Signale BC1, BC2 und BDIR werden an PB0, PB1 und PB2 erzeugt.

Eine zweite Möglichkeit der Ansteuerung des AY-3-8910 besteht in der Verwendung zusätzlicher Hardware. Bild 2 zeigt eine Lösung, die dadurch mit sehr wenig Aufwand auskommt, daß auf die Möglichkeit, die Register des AY-3-8910 zu lesen, verzichtet wurde. Dies dürfte in vielen Fällen akzeptabel sein. Erweitert man die Schaltung sinngemäß um 1/2 74LS74 und 2 UND-Gatter, so lassen sich damit zwei Synthesizerbausteine ansteuern.

Tabelle 3 soll als Programmierhilfe dienen. Sie enthält 12 2-Byte-Zahlen, die die Periodenlängen der 12 Töne einer Oktave in der temperierten Stimmung sind. Periodenlängen für die oberen Oktavlagen lassen sich daraus durch Division der Werte durch 2^n (Rechtsverschiebung und Rundung) erhalten. Mit Hilfe der Tabelle können die Töne in vier Ok-

taven durch 6 Bits codiert werden. Will man Melodien erzeugen, so sind die musikalischen Daten im Speicher niederzulegen, die dann im musikalischen Takt gelesen und in Daten für den AY-3-8910 umgesetzt werden müssen.

Tabelle 3: Periodenlängen innerhalb einer Oktave

0F D2	0B 2F
0E EE	0A 8F
0E 18	09 F7
0D 4D	09 68
0C 8E	08 E1
0B DA	08 61

ANSTEUERROUTINEN

ADRESSE SCHREIBEN

```
STA A001
LDA A000
ORA #04
STA A000
AND #F8
STA A000
RTS
```

DATEN SCHREIBEN

```
STA A001
LDA A000
ORA #06
STA A000
AND #F8
STA A000
RTS
```

DATEN LESEN

```
LDA #00
STA A003
LDA A000
ORA #03
STA A000
LDA A001
PHA
LDA A000
AND #F8
STA A000
LDA #FF
STA A003
PLA
RTS
```

*

****

INTERACTIVE

INTERACTIVE ist der Titel einer von Rockwell, Kalifornien, speziell für den AIM 65 herausgegebenen etwa 2-monatlichen Zeitschrift. Aus dem Inhalt der ersten Ausgaben. Heft 1: Hinweis auf lieferbare Ersatzteile - Zeitschriften, die den AIM betreuen - Ausgabe einer -Symboltafel für den Assembler - Checksum Program - Datafiles for AIM 65 BASIC - A Couple of 6522 Application Notes - BASIC Real Time Clock. - Heft 2: AIM 65 Graphics (AIMGRAPH, AIMPLOT - Löhr) - Inside BASIC (Butterfield) mit BASIC TOKEN LIST, Zeropage-Belegung, BASIC Entry Points - AIM 65 Sound - Marktübersicht Diskettensysteme für AIM - Disassembler Utility - Offset Loader for AIM - Marktübersicht: Anbieter von AIM-Erweiterungen - Parity Bit Generator Program - BASIC Banner Program.

Bezug: Der Vertrieb in Deutschland wird hier beim Herausgeber vorbereitet. Liefermöglichkeiten ab etwa Mitte November 1980. Abonnements für 6 Ausgaben DM 27,-.

Dipl.-Ing. U. Kornnagel, Ing. grad. G. Krohn und Ing. grad. P. Bach

Datenaustausch

zwischen zwei Mikroprozessorsystemen

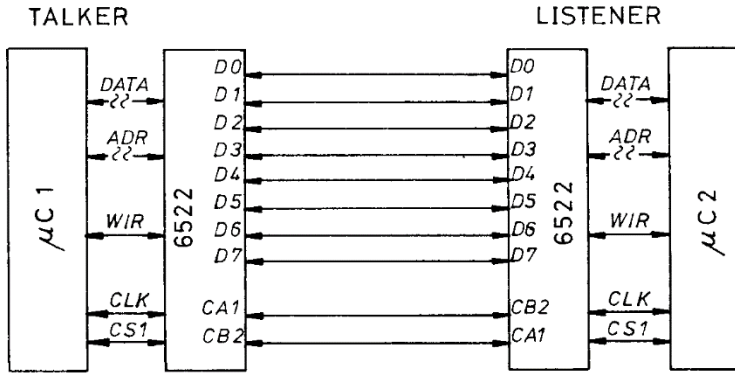
Mit diesem Beitrag soll eine Einführung in Datenaustauschverfahren gegeben werden. Datenaustausch ist für ein Mikroprozessorsystem unbedingt erforderlich, da jedes System E/A-Handling verarbeiten muß. In der heutigen Zeit ist nicht nur die Zentraleinheit, sondern auch die Peripherie mit eigener Intelligenz ausgestattet. Daher wollen wir anhand von Beispielen Datenaustauschverfahren zwischen verschiedenen Systemen auf 6502-Basis besprechen.

Im ersten Teil des Ausfsatzes werden Verfahren mit Hilfe des Versatile Interface Adapters (VIA 6522) dargestellt. Dieser Baustein besitzt u.a. 2x8 bidirektionale E/A-Leitungen und 2x2 Steuerleitungen.

Bei einem Datendialog sind mindestens zwei Geräte beteiligt, ein Sender (TALKER) und ein Empfänger (LISTENER), wobei die Funktion von TALKER auf LISTENER während des Austauschvorganges wechseln kann.

1. 8-Bit Parallelverfahren mit zwei Handshake-Leitungen

Bei diesem Verfahren handelt es sich um einen einfachen Quittungsbetrieb. Der Dialog wird vorbereitet, indem die DDR (Datenrichtungsregister) in der VIA auf Ein- bzw. Ausgabe gesetzt werden.

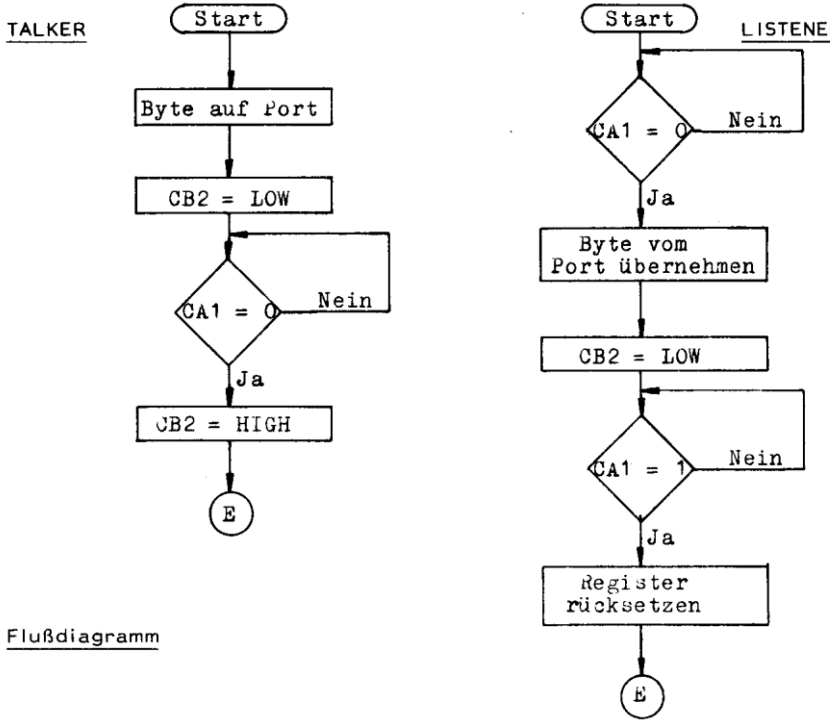


Zu vorstehendem Anschlußschema und zum Flußdiagramm auf der Folgeseite eine Beschreibung:

Bei dem vorliegenden Handshake-Verfahren handelt es sich um einen einfachen Quittungsbetrieb.

1.1. Der TALKER

Der TALKER legt die Daten parallel an den Port an, setzt die Leitung CB2 auf LOW und wartet solange, bis die Daten vom LISTENER übernom-



Flußdiagramm

men worden sind. Die Übernahme wird durch die fallende Flanke am Eingang CA1 angezeigt. Daraufhin zieht der TALKER CB2 auf HIGH, um dem LISTENER mitzuteilen, daß die Datenübertragung des ersten Bytes jetzt beendet ist.

1.2. Der LISTENER

In einer Warteschleife prüft der LISTENER seinen Eingang CA1, ob er auf "0" gesetzt ist. Stellt der fest, daß CA1 auf LOW liegt, werden die Daten vom Port parallel übernommen. Nach erfolgter Datenübernahme wird die Leitung CB2 auf LOW-Potential gezogen, damit der TALKER erkennt, daß alle Daten übernommen wurden. Nun wartet der LISTENER, daß der TALKER seine Leitung CB2 auf hohes Potential bringt. Danach wird die Leitung CB2 des LISTENERS auf HIGH gesetzt. Somit ist der Datenempfang des ersten Bytes vom LISTENER beendet.

1.3. Das Programm

```
DDRA=DATENRICHTUNGSREGISTER A
DRA =DATENREGISTER A
PCR = PERIPHERAL CONTROL REGISTER
IFR = INTERRUPT FLAG REGISTER
BUFF=DATENBUFFER
```

TALKER

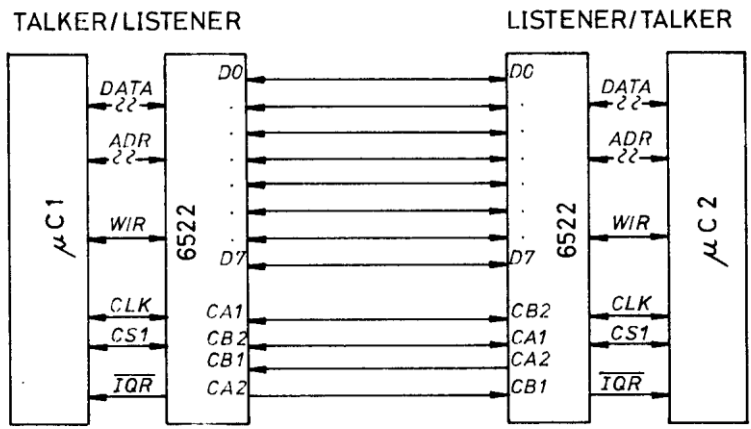
```
        ; PORT A AUF AUSGABE
        LDA #$FF
        STA DDRA

        ; BUFFER AUF PORT A AUSGEBEN
        LDA BUFF
        STA DRA

        ; LEITUNG CB2 AUF LOW
          (110X XXXX)
        LDA PCR
        ORA #%11000000
        AND #%11011111
        STA PCR
        ; WARTE BIS CA1 AUF LOW
        LDA #2
LOOP    BIT IFR
        BEQ LOOP
        ; CB2 AUF HIGH
        LDA PCR
        ORA #%11100000
        STA PCR
        ; ENDE
        RTS
```

LISTENER

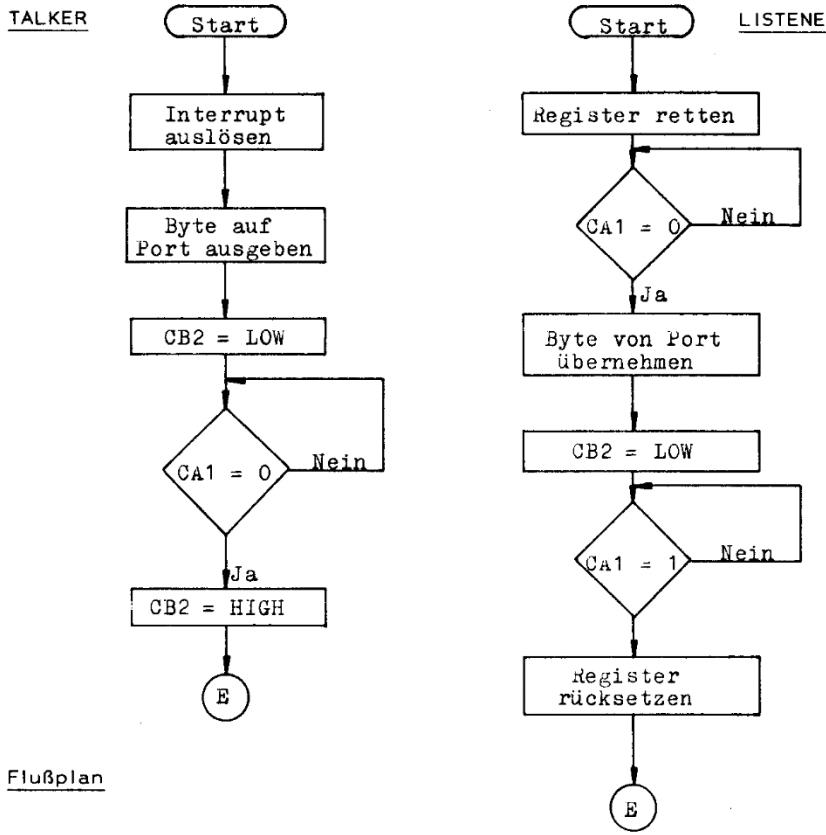
```
        ; PORT A AUF EINGABE
        LDA #0
        STA DDRA

        ; WARTE BIS CA1 AUF LOW
        LDA #2
LOOP1   BIT IFR
        BEQ LOOP1

        ; PORT A LESEN, IN BUFFER LEGEN

        LDA DRA
        STA BUFF
        ;CB2 AUF LOW
        LDA PCR
        ORA #$C0
        AND #$DF
        STA PCR
        ; WARTE BIS CA1 AUF HIGH
        LDA PCR
        ORA #1
        STA PCR
        LDA #2
LOOP2   BIT IFR
        BEQ LOOP2
        ; REGISTER RÜCHSETZEN
        LDA PCR
        AND #$FE
        ORA #$E0
        STA PCR
        ; ENDE
        RTS
```

1.4. Interruptgesteuerte Ein-/Ausgabe

Eine Weiterentwicklung dieser Routinen ist die interruptgesteuerte Ein-



/Ausgabe. Der TALKER verursacht vor der Übertragung beim LISTENER einen Interrupt, damit dieser erkennt, daß Daten anliegen. Diese Methode hat den Vorteil, daß das Gerät selbst erkennt, ob es als TALKER oder als LISTENER arbeiten muß. Hierzu die Beschaltung auf der Vorseite, der nachfolgende Flußplan und schließlich das Programm:



Flußplan

1.5 PROGRAMM

HILFSROUTINEN

```
        ; CA1 = LOW
LOOP1   LDA #2
        BIT IFR
        BEQ LOOP1
        RTS
```

```
        ; CB2 = LOW
CB20    LDA PCR
        ORA #$C0
        AND #$DF
        STA PCR
        RTS
        ; CB2 = HIGH
CB21    LDA PCR
        AND #$FE
        ORA #$E0
        STA PCR
        RTS
```

TALKER

```
        ; PORT AUF AUSGABE
        LDA #$FF
        STA DDRA

        ; INTERRUPT
        LDA PCR
        AND #$FD
        ORA #5
        STA PCR

        ; BYTE AUF PORT
        LDA BUFF
        STA DRA

        ; CB2 = 0
        JSR CB20
        ; WARTE BIS CA1=0
        JSR LOOP1
        ; CB2 = HIGH
        JMP CB21
```

LISTENER

```
        ; REGISTER RETTEN
        PHA
        TXA
        PHA
        TYA
        PHA
        LDA #0
        STA DDRA
        JSR LOOP1
        LDA DRA
        STA BUFF
        JSR CB20
        LDA PCR
        ORA #1
        STA PCR
        JSR LOOP1
        JSR CB21
        PLA
        TAY
        PLA
        TAX
        PLA
        RTI
```

Die Autoren sind
eine Projektgruppe
im Fernmeldeamt 1,
Frankfurt/M.

*

[WIRD FORTGESETZT]

****

Peter W. Arps, Hamburg

INPUT Routine

Um in BASIC über INPUT auch die Zeichen Komma, Anführungsstriche und Doppelpunkt in einen String zu übernehmen, muß man die Tastatur als File eröffnen. Nach INPUT X$ ist mit einer FOR-NEXT-Schleife der Inhalt des Tastaturpuffers bis zur ersten hexadezimalen Null mit der Folge X$=X$+CHR$(PEKK(VAR)) in ASCII umzuwandeln.

Einfacher ist es aber, den Pointer der Stringvariablen 'umzuhängen'. Hierzu wird die Länge der Eingabe im Tastaturpuffer ermittelt und der Variablenpointer auf den Puffer gestellt. Statt der FOR-NEXT-Schleife ist nur noch SYS(xxx)X$ aufzurufen. Um das Programm so kurz wie möglich zu halten, muß die Stringvariable die **erste** Variable des Programms sein.

Nach dem ersten INPUT zeigt die Stringvariable aber immer noch auf den Tastaturpuffer, obwohl sich dessen Inhalt inzwischen verändert hat. Um zu erreichen, daß BASIC die Eingabe in einen echten String umwandelt, ist nach SYS(xxx)X$ der Befehl X$=X$+"" einzufügen.

Das Programm:

```
A2 00       LDX #0      ERMITTLUNG DER EINGABELAENGE
B5 0A       LDA 10,X
F0 03       BEQ +3
E8          INX
D0 F9       BNE -7
A0 02       LDY #2
8A          TXA
```

```
91 7C          STA (124),Y STORE LAENGE IN VAR
C8             INY
A9 0A          LDA #10     POINTER UMHAENGEN
91 7C          STA (124),Y
C8             INY
A9 00          LDA #0
91 7C          STA (124),Y
60             RTS                              *
```

****

BRANCHEN-NACHRICHTEN

Die Leistungsfähigkeit des AIM 65 wird erweitert. Rockwell wird auf der ELECTRONICA in München ein verbessertes System mit folgenden Merkmalen vorstellen: Platine in bisheriger Größe, Steckplätze für 32 kB ROM (bisher 20 kB), bestückbar bis 24 kB dynamisches RAM. Das Display wird 40 Stellen haben (entweder Fluoreszenz- oder LCD-Anzeige). Auch der Thermodrucker arbeitet 40 Stellen breit. Er hat einen eigenen 1-Chip-Controller und entlastet die CPU vom Druckvorgang. - Dem Benutzer steht ein 6551 ACIA (serielle Datenübertragung) und eine Schnittstelle RS232 zur Verfügung. Die Karte enthält Busdriver. - Wie beim AIM, so hat der Benutzer auch hier eine freie Anwender-VIA 6522 und ein Cassetteninterface. Das Betriebsprogramm unterstützt allerdings nicht mehr (das nur noch selten gebrauchte) KIM-Format.

Großes Interesse fand das für die Erweiterung des AIM 65 angekündigte MICROFLEX-System. Es wird ebenfalls in München gezeigt. Lieferbar sind jetzt schon die Karten: 8 kB stat. RAM, PROM-Programmer, Extenderboard, Kartenkäfig mit 4, 8, 16 slots, ACIA-Karte mit 2 St. 6551. Eine CPU-Karte für Anwendersysteme wird hinzukommen. - Bei den Chips ist der Bildschirmcontroller 6545/1 jetzt lieferbar (Version ohne Lichtgriffel).

Die APPLE USER GROUP EUROPE e.V. hat ihre Aktivitäten stark regionalisiert. Lt. Rundschreiben vom Juli 1980 bestehen 14 regionale Gruppen und 9 thematisch ausgerichtete Arbeitsgemeinschaften. Auskunft beim Vorstand Wolfgang Dederichs, Postfach 4069, 4320 Hattingen 13, Tel. 023 24 - 67 412.

SYKO 100 ist die Bezeichnung eines Computers der Fa. System Kontakt GmbH., 7107 Bad Friedrichshall, Siemensstr. 5. Es handelt sich um ein auf dem AIM 65 basierendes Gerät mit 4 k RAM, BASIC-ROMs, Stromversorgung, in einem außerordentlich formschönen Gehäuse. SYKO 100 ist damit ein betriebsfertiger Computer zum Preis von DM 1.750,- + MWSt (unverbindl.).

Ein 12 kBASIC für den AIM 65/Pc 100 ist jetzt bei der GWK-Elektronik fertiggestellt worden (5120 Herzogenrath, Asternstr. 2). Es wird zur ELECTRONICA in München vorgestellt werden, und zwar auf dem Stand von Rockwell/System Kontakt. Im Vorwege war zu erfahren: z.Zt wird noch an der Dokumentation für den Benutzer gearbeitet. Das BASIC kann mit dezimalen, hexadezimalen und binären Argumenten arbeiten, auch in gemischten Ausdrücken, es hat wahlweise automatische Zeilen-Numerierung. Bei Fehlern zeigt ein Cursor die fehlerhafte Stelle an. Speichern und Laden von Programmen wird beschleunigt, weil BASIC-Befehle als Interpreter-Bytes einlaufen.

Eine Treiber-Zwischenplatine für den AIM 65 zur Ausschaltung von Störungen auf den Bussen wurde in Heft 13 auf Seite 53 angekündigt. Die erste Serie wird jetzt aufgelegt und soll zum Selbstkostenpreis von DM 30 (weniger bei größerer Auflage) von Bernhard Kokula, Wredestr. 17, 6700 Ludwigshafen geliefert werden. Tel. 0621-297.261 (tags). Die Platine wird auf den Sockel der CPU gesteckt und nimmt die 6502 nebst Treibern auf.

Die Sprache FORTH für den AIM 65 soll auf der ELECTRONICA von Rockwell vorgestellt werden.

Ein erweitertes Betriebssystem (EPROM 2 kB) für AIM 65/PC 100 wird jetzt von Chr. Streicher, Wittelsbacherstr. 24 in 8034 Germering angeboten Tel. 089-84 12 976 . Aus allen Kommando-Ebenen heraus sind ansprechbar: IEC-Bus für Hörer-Geräte (Plotter, Drucker ..), professionelles Assembler-Listing mit Erstellung einer Symboltafel, dynamische Datenabspeicherung aus dem BASIC, SYS-Befehl, BASIC-Zeilennummern-TRACE, READY-Meldung vom BASIC-Interpreter, Video-Initialisierung für das PC 100-CRT.

****

BUCHBESPRECHUNGEN

Robert Findley: 6502 Software Gourmet Guide & Cookbook, Scelbi Publications 1979. Mit einer Ausgabe für 6502 führt die Fa. Scelbi ihre Serie von Gourmet Guides und Kochbüchern für verschiedene Mikroprozessoren fort. Dem Rezensenten sind weitere Bücher aus dieser Serie leider nicht bekannt, die Urteile über die 6502-Ausgabe sollten deswegen nicht auf die komplette Serie verallgemeinert werden.

Der Titel als solcher läßt einiges an ausgebufften Tricks bei der Programmierung von 6502-Mikroprozessoren erhoffen. Leider wird hier aber nicht aus der Gourmet-Küche geplaudert, sondern höchstens Hausmannskost serviert. Zuerst wird der Befehlsvorrat der 6502 beschrieben. Diese Darstellung erschöpft sich jedoch rein im Verbalen, Diagramme oder Bilder wären zur Verdeutlichung von Adressierungsarten und Befehlen eindrucksvoller gewesen. Die darauf folgenden Beispiele lassen eine Auseinandersetzung mit der Leistungsfähigkeit der 6502 und ihren speziellen Möglichkeiten vermissen. Dies beginnt schon damit, daß sämtliche Datenfelder mit aufsteigenden Schleifen verarbeitet werden und die Möglichkeiten des Y-Registers als Index und Zähler nicht begriffen wurden.

Weiterhin wird grundsätzlich die indirekt-indizierte Adressierung benutzt, sicher die mächtigste Adressierungsform der 6502, aber man vermißt eine Diskussion auch ihrer Grenzen bei der Beschickung mit Parametern. Entsprechend sind auch die als Gourmet-Essen angebotenen Rezepte: Für ein Floating-Point-Package werden mehrere Page Speicherplatz benötigt, Steve Wozniak, sicher ein Meisterkoch auf der 6500, brauchte für ein vergleichbares Produkt, das vor Jahren in Dr. Dobbs Journal veröffentlicht wurde, weniger als eine Page.

Insgesamt macht das Buch den Eindruck, als ob Scelbi, aus welchem Grunde auch immer, seine Serie um eine Ausgabe über die 6502 ergänzen wollte und mit dieser Aufgabe jemanden betraute, der Lösungen, die für die 8080 oder 6800 gedacht waren, 1:1 umgeschrieben hat. Sicher sind diese Programme ausführbar, wie Hausmannskost durchaus sättigend ist, höheren Ansprüchen können sie wegen vollständiger Mißachtung der besonderen Fähigkeiten der 6500-Mikroprozessoren nicht genügen.

Michael Zimmermann

Leo J. Scanlon: 6502 Software Design, Blacksburg Series by Howard W. Sams & Co, Indianapolis 1980, 270 Seiten, DM 29,-, ISBN 0-672-21656-6.
Scanlon ist Documentation Manager bei Rockwell. Sein in Englisch geschriebenes Buch gehört zu den wenigen gelungenen Darstellungen der 6502-Programmierung. Entsprechend dem Titel wird die Hardware eingangs nur kurz vorgestellt. Bei der Erklärung des Instruktionssates und der Adressierungsarten ist bereits eine große Zahl von erklärten Programmierbeispielen enthalten.

Ebenso bei der Unterprogrammtechnik (Move und Zeitverzögerungen), bei der Tabellenverarbeitung (einschl. Sortierung), bei den mathematischen Routinen (einschl. Multiplikation und Division) und bei der Zahlenwandlung. Die Abschnitte Interrupt und Interfacebausteine bedingen zwar mehr Erklärung der Hardware, aber auch sie sind mit Programmierbeispielen versehen, ebenso wie das Schlußkapitel zur Datenein- und Ausgabe am Mikrocomputer. Im letzten Teil werden auch einige Routinen des AIM 65 herangezogen, der Inhalt bleibt wegen der durchlaufenden Erklärungen aber für alle Systembetreiber verständlich. - Das Buch hat mit seinen Beispielen einen hohen Eigenwert, auch gegenüber dem in Heft 14 besprochenen Leventhal (6502 Assembly Language Programming). R.L.

Marvin L. De Jong: Programming & Interfacing the 6502, With Experiments, Blacksburg Series by Howard W. Sams, Indianapolis 1980, 414 Seiten, ISBN 0-672-21651-5.
Als erfahrener Autor hat de Jong ein Buch für Anfänger geschrieben (englisch). Jedem Kapitel stellt er eine Zielsetzung voran und läßt am Schluß Experimente und Fragen folgen, so daß der Leser zu eigener Arbeit angeregt wird. - Wie in keinem anderen der bisher bekannten 6502-Bücher geht der Autor (entsprechend dem Untertitel) auf Außenbeschaltungen des Prozessors und ihre Programmierung ein. Damit - und mit seinen Schritt für Schritt aufbauenden Erklärungen, Schaltbildern und Flußdiagrammen - darf das Buch vor allem digitalelektronisch interessierten Lesern empfohlen werden. R.L.

****

electronica, München

Der Herausgeber ist am Sonntag, den 9.11.80, nachmittags und am Montag, den 10.11.80, vormittags auf dem Messegelände. Verabredungen könnten im Vorwege oder über den Stand von Rockwell/System Kontakt GmbH getroffen werden.

****

Editorial

Mit seinem vermehrten Umfang zeigt dieses Heft, daß der Stoff nicht weniger sondern eher mehr wird. Viele Projekte, die als nützlich und interessant erkannt wurden, konnten noch nicht angefaßt werden. - Der Herausgeber dankt den Autoren für die Vielseitigkeit ihrer Themen. Wie bisher sind solche Artikel aus der Leserschaft sehr willkommen. Dabei interessieren nicht nur spezielle Lösungen und Dienstleistungen für die Programmentwicklung. Es gibt immer wieder Gelegenheit, eine Thematik in Übersichten abzuhandeln (in diesem Heft z.B. zur Datenein- und Ausgabe, zum Datenaustausch). Der überwiegende Teil der Leserschaft ist zwar irgendwie beruflich mit der Prozessorei verbunden, der Einarbeitungsstand ist aber unterschiedlich. Artikel und Programme sollten dem Leser den gedanklichen Mitvollzug ausreichend ermöglichen, insbesondere durch Kommentare und Bedienerhinweise.

Der Ausdruck von Programmlisten mit Nadeldruckern und speziell Thermodruckern (AIM) kann für die Reproduktion im Offsetverfahren bisher nicht voll befriedigen. Die geschlagene Type ist besser. Wegen möglicher Übertragungsfehler solltegleichwohl vom Computer gelistet werden. In diesem Sinne wird hier ein verbessertes Verfahren vorbereitet. Daher ist es erwünscht, daß Autoren neben dem dort gefertigten Ausdruck auch einen Datenträger hierfür einsenden: Cassette für AIM, KIM oder PET/CBM, Diskette für CBM. Die Reinschrift/der Satz von begleitendem Text erfolgt hier.

Beiträge sollten im Vorwege kurz abgesprochen werden. - Künftig werden auch Artikel für den 6809 berücksichtigt.

Das 65xx MICRO MAG wird nach wie vor als 1-Mann-Betrieb geführt. Zu den Aktivitäten des Herausgebers gehört nicht nur die Arbeit an dieser Zeitschrift, sondern ebenso die freiberufliche Entwicklungsarbeit für Auftraggeber und das Abhalten von Workshops. Beim bestehenden Einspannungsgrad können schriftliche Anfragen nur in begrenztem Umfange beantwortet werden.

****

English Summary

The first article describes 'The Progressive Relative' the MC6809 from the viewpoint of a 6502 Programmer. Especially praised are its new features, the many direct and indirect indexing addressing modes with all registers, its position-independent programming methods with long branches and addressing relative to PC, the user stack for passing parms and the 'direct paging' with all pages, the transfer of all registers to another register.

Ein- und Ausgabe am AIM 65 treats the various means to output characters and strings to the LED-display (the first article in this series covered the overall I/O mechanism and input from the keyboard) and explains the driver programs in the monitor. The final example program outputs any desired text at a keystroke in an interactive environment. Text may reside in more than one page.

Programmveränderung allows to merge BASIC pgms from 4 source files (disk) into a new file. Machine code speeds up this cbm service.

Zeitansage für PET und CBM installs an output window for the time on the screen.

Disassemblierung in den Texteditor is an AIM utility to derive a source text with artificial labels and .BYT and else directives from an object file. Allows re-editing in texteditor!

NRINS is an AIM texteditor with automatic line numbering with increment by choice and with insertion of a line at its proper place - as in BASIC. Line numbers may be stripped off in advance to an assembly or for a renumbering.

In the next article **AIM is the TTY terminal to KIM-1** and may dump a program from its assembler (Out=U) to KIM.

Musikerzeugung mit dem AIM 65 makes music with a GI AY-3-8910 synthesizer.

Datenaustausch (to be continued) covers the exchange of data between processors, first by simple handshake, then interrupt driven. Follow will shared memory and IEEE488 bus.

Input Routine for PET allows to 'get' , " and : by INPUT into a string.

****

Software-Haus

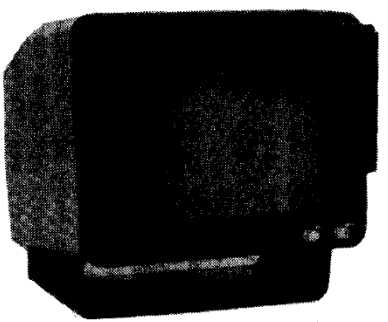
Anwendersysteme benötigen einen eigenen Monitor in Festwertspeichern, mit anwenderspezifischer Initialisierung und Befehlsebene. Für Systeme auf der Basis des AIM 65 stehen in Kürze für beliebige Adreßbereiche assemblierfähige Quelltexte (Cassette) mit Kommentaren zur Verfügung. Leistungsmodule werden unter Berücksichtigung der Interdependenzen nach Kundenspezifikation gruppiert. Anfragen an den Herausgeber.

KEB
COMPUTER

Video-Tastaturen · Monitore · Tischcomputer · Datensichtgeräte

MR 1012

- Als Ausgabegerät für vorhandene Anlagen
- Anschluß für BAS-Signal



- Blendfreier und schwenkbarer Bildschirm, grün
- Beweglicher Blendschutz
- Integriertes Netzteil

Empfohlener Verkaufspreis
DM 1.094,– inkl. MwSt.

KEB
COMPUTER

„Geräte der Datentechnik aus Berlin"

KEB Computer GmbH & Co
Konstruktions-Elemente-Bau KG

Triftstraße 25–35
D-1000 Berlin 27

Telefon 030/432 60 96
Telex 181 489 efe d

KA/UP

PET PicChip

Der PicChip ist ein ROM-Einschubmodul für den cbm3016/32, der dem Befehlsvorrat des PET über 40 höhersprachige parametrisierte BASIC-Kommandos hinzugefügt, um die grafischen Möglichkeiten des PET voll auszuschöpfen.
Komplizierte Formen und Muster - feststehend oder beweglich - können schnell und klar mit einfachen BASIC-Programmen aufgebaut werden. Diese schnellen Kommandos holen ihre Parameter direkt von den BASIC-Variablen X, Y, X0, Y0, X1, Y1, X2, Y2, A1, A2, N und C. Neben nützlichen Möglichkeiten wie ein Dauertaste- ein/aus Kommando, schließen die verfügbaren Funktionen folgendes ein:

Einfache Schreibdichte (40 x 25)

§ Definieren rechteckiges Sichtfenster
§ Fülle Fenster mit Zeichen 'C'
§ Fenster rollen/schieben um N Stellen aufwärts, abwärts, links oder rechts
§ Fenster in upper, lower oder inverted case
§ Fenster auf normal, reverse oder inverted
§ Cursor position lesen/auf X, Y gesetzt
§ Setze Zeichen nach X, Y
§ Kopieren Bildschirm nach/von RAM-Adresse

Doppelte Schreibdichte (80 x 50)

§ Zeichne/lösche Punkt X, Y
§ Zeichne/lösche Linie von X1, Y1 bis X2, Y2
§ Zeichne/lösche senkrecht zur X/Y-Achse
§ Zeichne/lösche fortgesetzt Linie

Feine Schreibdichte (40 x 200 / 25 x 320)

§ Zeichne Wert X mit Auflösung 320
§ Zeichne Wert Y mit Auflösung 200

Hervorragend geeignet für Prozess-Diagramme, mathematische Probleme, Lernprogramme, Bildschirmspiele, bewegliche Schaubilder usw.

Lieferbar für Stecker UD5, UD4 oder UD3.
(Frühere PET-Modelle mit 'New ROMs' benötigen Busadapter.)

PicChip komplett mit Handbuch DM 225,–
nur Handbuch DM 20,–
Preise inkl. MWST

ING.BÜRO
HOUGHTON

Arabellastraße 58
8000 München 81
Tel. (089) 91 46 28

Systemerweiterung für AIM 65/PC 100 auf Europakarten

einheitliches Format 100x160 mm, einheitlich nur +5 Volt, AIM-Expansion-Bus-kompatibel, "S44-Bus", AIM-Software kompatibel, alle Karten auch mit 64-poligem a/c DIN-Stecker lieferbar, Adreßdekodierung auf jeder Karte, anschlußfertig, 1 Jahr Garantie.
Für den PC 100 gibt es z.Z. Sonderausführungen, die mit der im Gerät vorhandenen Stromversorgung auskommen.

z.Zt. lieferbares Programm:

32 kB RAM Modul - die meistgekaufte AIM-Speichererweiterung!
792,- + MWSt = 894,96 DM
teilbestückt mit 16 k: 526,- + MWSt = 594,38 DM

VIDEO INTERFACE 615,- + MWSt = 694,95 DM

ANALOG I/O 8 Bit, 80 kHz max. Abtastrate, mit Aliasingfilter und sample and hold, I/O unabhängig voneinander
370,- + MWSt = 418,10 DM

EPROM-KARTE 24 kB, für 2716 und 2532
ohne EPROMs 180,- + MWSt = 203,40 DM

BUS EXPANSION zusätzlich gepuffert, 6 Steckplätze, erweiterbar
240,- + MWSt = 271,20 DM

ICs (Preise incl. MWSt), jeweils das Spitzenfabrikat (F, NEC, Motorola, TI etc.)

2114L/450 = DM 12,-	2114 CMOS = DM 30,-	4116/200 = DM 15,-
2708L/450 = DM 17,-	2716 = DM 35,-	2532 = DM 89,-
6502 = DM 37,-	6522 = DM 29,-	6532 = DM 37,-
6821 = DM 18,-	6845 = DM 74,-	

EPROM-Löschlampe 6 W, E27 65,- DM

DIPL.-ING. HORST NEUDECKER
INGENIEURBÜRO FÜR MESSTECHNIK

MEHRINGPLATZ 13
1000 BERLIN 61

TEL.: 030- 614 89 00
030- 251 20 00

AIM 65 Video-Interface

Anschlußfertiges 2000-Zeichen Video-Interface einschließlich EPROMresidenter AIM-spezifischer Software.

Ermöglicht volle Zeilenlänge für Textbearbeitung und BASIC-Programme.

Format wählbar: 24 Zeilen zu 80 Zeichen, 27 Zeilen zu 72 Zeichen, andere Formate programmierbar.

Standard-ASCII-Zeichensatz, 96 alphanumerische Zeichen, Groß- und Kleinschreibung mit echten Unterlängen und Grafikzeichensatz in 2k EPROM. Positiv-/Negativ-Darstellung für einzelne Zeichen (Normal/reverse Video) und für das ganze Bild.

Cursor-Bewegungen und Cursor-Funktionen von der Tastatur und per Programm steuerbar. Rollen (scroll) mit variabler Geschwindigkeit.

Vollständig 'transparentes' Video-RAM, zugleich System-RAM. Firmware in 2k EPROM mit Platz für eigene Erweiterungen Zeichengenerator in 2k EPROM und durch Anwender-eigene Software erweiterbares 2 k Video-RAM auf der Interfacekarte. Lichtgriffelanschluß, Ausgang: Video-BAS, 50 Hz, 2 V, 50 Ausgang: Video-BAS, 50 Hz, 2 V, 50 Ohm.

AIM 65 RAM-Modul

Anschlußfertige 32 kByte dynamische Speichererweiterung. Wie die anderen AIM-Systemerweiterungskarten, so ist auch das RAM-Modul ohne zusätzliche Hardware ('motherboard') kompatibel zu Expansion-Connector des AIM 65 oder PC 100. Adressierung in 16 unabhängigen 2k Segmenten. Mehrere Speicherkarten können durch bank select parallel zum Systembus betrieben werden. Eine einzige Stromversorgung mit 5 V und max. 1 A ist für das 32k-Modul ausreichend.

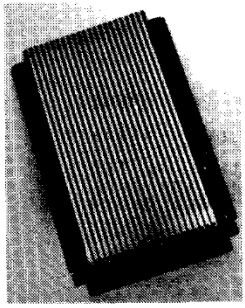
Voll- (32k) und teilbstückt (16k) lieferbar für AIM 65, PC 1oo, SYM, KIM, MCS-alpha. Prospekt anfordern!

DIPL.-ING. HORST NEUDECKER

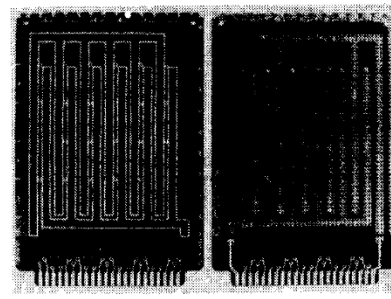
INGENIEURBÜRO FÜR MESSTECHNIK

Verkauf von Leiterplatten, Steckadapter, Relaisplatinen

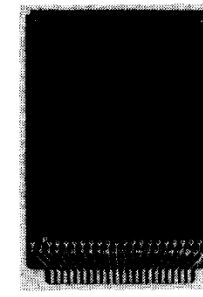
1



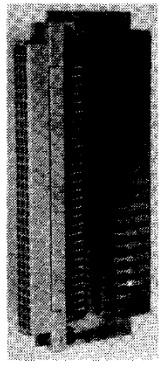
2



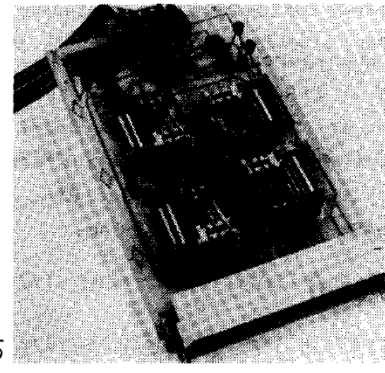
3



4



5



6

	p.St.	bei Abnahme von 3-5	von 6-10
1. 3690 Card Extender	DM 67,95	DM 57,75	DM 50,95
2. 3682-2 DIP Plugboard	DM 34,05	DM 28,95	DM 25,55
3. 3662 DIP Plugboard	DM 31,10	DM 26,45	DM 23,35
4. C 991 Steckadapter	DM 59,50	DM 50,55	DM 44,65
5. E 991 Relaisplatine	DM 210,60	DM 179,05	DM 157,95
6. E 941 dito m. Halter	DM 261,80	DM 221,85	DM 201,35

Alle Preise zuzüglich MWSt und Versandspesen. Lieferung ab Lager Köln, Zwischenverkauf vorbehalten. Weitere techn. Beschreibung auf Anfrage.

STECKER
INGENIEURBURO

5000 KÖLN 60 (Nlehl)
Postfach 60 07 66
Delmenhorster Str. 20
Telefon (02 21) 74 51 12

GWK

FÜR TECHNISC[illegible]LLSCHAFT
[illegible]KTRONIK mbH.

NEU! NEU! NEU! NEU!

D 5120 Herzogenrath
Asternstr. 2
Tel.: 02406/62394

Floppy-Disk für AIM 65/PC 100

Wir haben es !!!

Das Floppy Disk System, das wirklich mit allen Eingabe- und Ausgaberoutinen voll zusammenarbeitet. Dies gilt für

ASSEMBLER - BASIC - EDITOR - MONITOR

Preise:

Doppel - Floppy - Station komplett mit Controller und Software	3.485,-- DM + MWSt
Controller Board mit res. Software	1.248,-- DM + MWSt

Basic Expansion für AIM 65/PC 100

jetzt erweitert auf **4 KByte**

Gegenüber der schon seit einiger Zeit erhältlichen GWK BASIC ERWEITERUNG (2K) sind in der BASIC EXPANSION (4K) die folgenden Funktionen verbessert oder neu aufgenommen worden:

MEMORY SIZE ändert untere und obere Grenze des Speicherbereichs sowie Line Width.

RENUMBER wesentlich verbessert. Von - bis, Schrittweite.

RELOCATE von BASIC-Programmen in anderen Speicherbereich.

START von BASIC-Programmen, die in anderen Speicherbereichen z.B. in einem EPROM abgelegt sind.

CALL von Programmteilen, die von Floppy geladen werden.

GET A$ liest Zeichen von allen Input Devices.

DATA INPUT / OUTPUT jetzt unter Programmsteuerung, Definition von FILENAME und DEVICE durch Programmstatement.

SAVE ist spezifizierbar, z.B. SAVE 50-200.

PRINT USING. Formatierte Zahlenausgabe, rechtsbündig, Festkomma.

USR-Funktionen. Umwandlung von Zahl in String und umgekehrt.

Erhältlich auf: Kassette, Diskette, 2 EPROM's á 2K, EPROM 4K.
Preis: DM 245,-- zuzügl. MWSt und Datenträger.

Besitzer der BASIC-Erweiterung (2K) erhalten die EXPANSION zum Differenzpreis (Software, nicht Datenträger).

65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

HERAUSGEBER:
DIPL.-VOLKSWIRT ROLAND LÖHR
HANSDORFER STRASSE 4
2070 AHRENSBURG
☎ (04102) 55 816

65xx MICRO MAG erscheint zweimonatlich. Beiträge, die nicht besonders gekennzeichnet sind, stammen vom Herausgeber. COPYRIGHT 1980 by Roland Löhr. Alle Rechte vorbehalten, auch die des auszugsweisen Nachdrucks, der Übersetzung, der fotomechanischen Wiedergabe und die der Verbreitung auf magnetischen und sonstigen Trägern. Offsetdruck: Druckartist Gerhard M. Meier, Hamburg 70.

Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 45,- (Inlandsendpreis). Ausland/Foreign via surface mail DM 50,-, USA air $30. Die früher erschienenen Ausgaben dieser Zeitschrift sind nachlieferbar. Einzelpreis für Nos. 1-6 DM 4,-/St., Nos. 7-12 DM 7,80 St. Die Hefte 1-6 sind auch als Sammelband beziehbar (s. Anzeige).

Richten Sie bitte Ihre Überweisungen/Schecks an Roland Löhr, Konto 20/01121 bei der Vereins- und Westbank in Ahrensburg, BLZ 200 300 00. Zuschreibung auch über das Postscheckkonto der Vereinsbank: PSchA Hamburg 2244-207.

Leser-Service des Herausgebers

Thermopapier
für AIM 65/PC 100 in kontrastreicher Spitzenqualität
Packung mit 8 Großrollen à 65 m = 520 m, preiswert DM 50,85

Printerplatte für AIM 65/PC 100 m. Einbauanleitung DM 21,--

Disketten 5,25", softsektoriert, einseitig, single density (z.B. für CBM 3040), vom Hersteller 100%-ig geprüft. 10 St. in Plastik-Archiv-Aufstellbox DM 108,--

Anwenderhandbuch für AIM 65, deutsch
Original Rockwell-Handbuch DM 32,10

Das Buch 1-6 des 65xx MICRO MAG
ca. 230 Seiten, Sammelband der Hefte 1-6 dieser Zeitschr. gebunden, o. Anzeigen. Wertvolles Arbeitsbuch DM 26,--

Microprocessor Systems Engineering
von Camp, Smay, Triska, englisch. Lehrbuch für 65xx/AIM 65 DM 66,--

6502 Software Design v. L.J. Scanlon
emfehlenswert (s.S. 59), engl. 270 S. DM 29,--

Vorstehende Preise sind Endpreise, einschl. Versand. NN + DM 1,50

Programmierworkshops 6502 & 6809
in Ahrensburg b. Hamburg. 16./17.1.81 für 6502, 23./24.1.81 für 6809. Kleine Teilnehmerkreise, intensives Arbeiten. Sonderprospekt beim Herausgeber. - Schulungen in Firmen auf Anfrage.