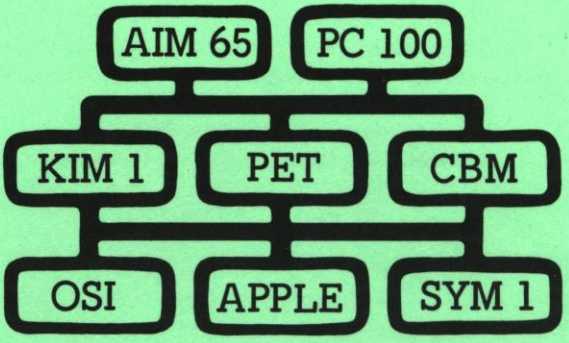
65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 9,50 Nr. 28 Dezember 1982



Inhaltsverzeichnis

DUO Plott Interface

für MX80 F/T und
ITOH 8510
und COMMODORE-Rechner 30/40/80

Paralleles IEEE 488 - Interface, genormter IEEE-Stecker und Kabel
kompletter Zeichensatz des CBM-Computers
zwei Geräteadressen für Groß-/Grafikmodus und Textmodus
alle Funktionen der Drucker bleiben erhalten, Floppy-Kompatibilität
Deutsche Umlaute, ß und Paragraph
Problemloser Einbau, inklusive deutsches EPSON-Handbuch
Komplettpreis einschl. Kabel, CBM-Grafiksatz und Handbuch incl. MWSt
für EPSON DM 398,--
für ITOH DM 398,--

Umrüstsatz MX80 F/T auf **MX80, Typ 3**

Aus Ihrem EPSON MX-80 F/T wird der MX80 Typ 3
Einzelnadelansteuerung, erweiterter Befehlsatz, Elite-Schrift
Preis inkl. MWSt DM 98,--
Komplettpreis Interface, Kabel, Handbuch und Umrüstsatz incl. MWSt DM 450,--
Komplettpreis wie vor, einschl. MX80, Typ 3, inkl. MWSt DM 1.798,--

Für COMMODORE

Deutsche Tastatur mit Original-Tastensätzen (keine Aufkleber)
inklusive deutschem Zeichengenerator
für 8032 DM 98,--
für 8032 und 8096 DM 98,--
nur 8096. ohne Tasten DM 98,--

Speichererweiterung auf **798**

LOS-Kompatibel, inkl. MWSt DM 898,--

ISAM-Routinen
Datenhaltungsprogramm, Indexed Sequential Accesss Method
siehe Besprechung in Heft 23 des 65xx MICRO MAG, DM 298,--

Stellberg Computer-Systeme

COMMODORE EPSON C–ITOH SOFTWARE INTERFACE

Blindenaaf 36 5063 Overath Tel.: 022 06 - 66 44

Dipl.-Ing. Rüdiger Wollenberg und Dipl.-Phys. A. Redder

REMON - Redigierter Monitor (AIM)

1.0 Einleitung

1.1 Ausgangsüberlegungen

Das Monitor-Betriebssystem des Rockwell AIM 65 bzw. des Siemens PC 100 hat von Anfang an immer wieder Wünsche nach Verbesserung und individueller Anpassung geweckt. Dies betrifft in erster Linie die Erweiterung des Ein-Ausgabebereiches, Implementierung zusätzlicher Monitorbefehle und Optimierung des Editors. Es sind dazu seit Bestehen dieser Zeitschrift eine ganze Reihe von Programmen veröffentlicht worden, die häufig auf geradezu kunstvolle Weise über die frei belegbaren Funktionstasten, die User-I/O-Vektoren oder den DILINK-Vektor an das Monitorprogramm adaptiert wurden.

Weil der Preis für 2532-EPROMs inzwischen unter DM 20,- gesunken ist, liegt der Gedanke nahe, eine Umprogrammierung des Monitor-Betriebssystems vorzunehmen, um eigene spezielle Programme einzubinden. Dies geschieht für manche Routinen, ohne auf irgend eine Betriebssystem-Funktion verzichten zu müssen (Repeat-Funktion, Unterdrückung der Editor-Prompts), weil durch eine geschicktere Programmierung eine effektivere Speichernutzung möglich ist. Außerdem gibt es zwei größere ungenutzte Speicherbereiche /2/ mit einer Gesamtlänge von 131 Bytes, die für kleinere Änderungen ausreichen.

Bei größeren Eingriffen (Kommando-Erweiterungen, Video-Initialisierung und -Betriebsprogramm, Kleinschreibung) ist ein Verzicht auf Routinen des früheren Betriebssystems unabwendbar. Die Autoren sahen als entbehrliche Systemroutinen das Laden und Abspeichern mit dem KIM Bandformat an. Wenn diese Routinen für den Betreiber unverzichtbar sind, müssen andere Kompromisse geschlossen werden. Wer z.B. kein Terminal an seinem AIM 65 betreiben will, dem stehen durch Freiwerden einiger TTY-Routinen nochmals ca. 150 Bytes zur Verfügung. Die dann nutzbaren Speicherbereiche sind in der Tabelle des Kapitels 2 aufgeführt, aber von den Autoren bisher nicht

überschrieben worden. Reichen die freigesetzten Speicherräume nicht aus, muß ggfs. eine Auslagerung in ein zusätzliches EPROM vorgenommen werden.

Der aufgezeigte Weg ist bereits von einigen Herstellern beschritten worden, die selbst eigene Monitorprogramme anbieten. Mit diesem Artikel sollen jedoch einem breiten Heer von Anwendern die Möglichkeiten und Grenzen der Veränderung des Betriebssystems dargestellt werden.

1.2 Randbedingungen

Was die Hardwareänderungen am AIM 65 angeht, sind sehr viele Möglichkeiten denkbar. Um möglichst viele Betreiber anzusprechen, soll als Basis von einem Minimalsystem ausgegangen werden. Es ist eine Betriebsbereitschaft unter Verwendung des REMON zu fordern, die keine externe Peripherie fordert. Deswegen wird auch das 16-Segment-Display weiter aktiv gehalten. Allerdings wird bei einem implementierten Video-Betriebssystem die Option eines ausschaltbaren Displays gezeigt.

Um auch weiterhin mit den vorhandenen Dienstprogrammen kompatibel zu sein, wie Assembler, BASIC, FORTH usw., die z.T. sehr stark auf Betriebssystem-Programme zurückgreifen, dürfen die Einsprungspunkte der meisten Unterprogramme nicht verändert werden.

Um für Erweiterungen Platz zu haben, wird auf das Laden und Speichern im KIM Bandformat verzichtet, das wohl für die meisten Benutzer entbehrlich ist. Damit werden im Bereich hex A4xx fünf Speicherzellen frei, die als Flags und Link-Vektoren weitere Verwendung finden.

1.3 Vor- und Nachteile

Die Vorteile eines selbst geänderten Monitorprogramms sind:

- ein auf die Bedürfnisse zugeschnittenes Betriebssystem mit implementiertem Video-Betrieb
- Einsparung von zusätzlichem EPROM-Speicherplatz
- Beseitigung von offensichtlichen Fehlern des Betriebssystems
- verbesserter Rechnereinsatz als Editiersystem
- zusätzliche Monitor- und Editor-Befehle sind leicht anhängbar
- Implementierung von Hilfsroutinen

Zu den Nachteilen gehören:

- gute Kenntnisse des Monitorprogramms sind erforderlich
- gute Programmierkenntnisse sind nötig, um die meist kleinen Speicherlücken effektiv zu nutzen
- schlechte Testmöglichkeiten bei fehlerhaften Änderungen
- unter Umständen Kompatiblitätsschwierigkeiten mit selbstgeschriebenen Programmen oder zukünftiger Systemsoftware
- eine äußerst sorgfältige Dokumentation der vorgenommenen Änderungen ist erforderlich

1.4.0 Vorgenommene Änderungen

1.4.1 Printerflag OFF

Das automatische Anlaufen des Druckers nach einem Power On wurde unterdrückt, denn das Ausschalten des Druckers gehört bei vielen Betreibern zu den ersten Handgriffen nach dem Einschalten des AIM. Erreicht wird dies durch eine Änderung des Wertes für PRIFLG in der Initialisierungstabelle INTAB3.

1.4.2 Fehlerbeseitigung

Folgende Fehler des Betriebssystems wurden beseitigt:

In /1/ wurde auf eine fehlerhafte Interruptbehandlung hingewiesen. Danach wird bei einem Tape-Dump der Interrupt nicht wieder aktiviert.

Nach Herrn Rix /3/ gibt es bei der Tape-Laderoutine Schwierigkeiten mit der Rückkehr ins Monitorprogramm, wenn der letzte Block mehr als 76 Bytes umfaßt. Die Ursache liegt in einer fehlerhaften Vorbesetzung einer Zählschleife.

Herr Beland /4/ wies darauf hin, daß der AIM in seiner Routine HEX das Sonderzeichen ' @ ' als

Ziffer akzeptiert. Die an gleicher Stelle vorgeschlagene Änderung des Herausgebers wurde übernommen. Dieser wies weiter darauf hin, daß die Kassettenlaufwerke nach Einlesen eines Files wieder gestartet werden, was oft unerwünscht ist.

Viele Benutzer, die ein Terminal am AIM betreiben, haben Schwierigkeiten mit der automatischen Baudratenermittlung. Dies ist auf eine falsche Subtraktion zurückzuführen.

1.4.3 Repeat-Funktion

Bei längerem Tastendruck erscheint das Zeichen wiederholt auf dem Display. Dies ist besonders beim M-Befehl, bei Editieren, beim Printervorschub und beim Löschen interessant. An den Umstand, daß die Tasten nur kurz betätigt werden dürfen, gewöhnt man sich schnell.

1.4.4 Groß- und Kleinschreibung

Will man mit dem System Textverarbeitung betreiben, ist eine Kleinschreibung unabdingbar. Es wurde ein Vorschlag von Herrn Schröder /6/ aufgenommen und mit einigen Befehlen derart erweitert, daß eine Caps-Lock-Einrichtung entsteht. Dies geschieht durch Setzen und Löschen des Bits 6 in Speicherstelle A408.

Im Gegensatz zur Shift-Lock-Funktion, die sich auch auf die Ziffern und Sonderzeichen auswirkt, werden bei Caps-Lock nur die Alpha-Zeichen großgeschrieben.

1.4.5 Kleinschreibung und Befehlsunterdrückung im Editor

Gerade im Editor wird der Kleinschreibmodus häufig eingeschaltet sein. Um nicht unnötige Fehlermeldungen zu bekommen, wenn ein Befehl versehentlich in Kleinbuchstaben eingetippt wird, wurde der Bedienungskomfort an dieser Stelle erweitert, indem auch kleingeschriebene Befehle als gültig akzeptiert werden. Ehe der Editor die Befehlsdecodierung vornimmt filtert eine Routine das im Akkumulator übergebene Zeichen und stellt es auf Großbuchstaben um.

Die Repeat-Funktion erlaubt beispielsweise ein zeilenweises Durchfahren des Textes mittels der Befehle 'U' und 'D'. Dabei stört im bisherigen Editor aber, daß die Dastellung in der Ausgabe von den Kommandozeilen (Editor-Prompts) unterbrochen wird. Diese wurden hier unterdrückt.

1.4.6 Kleingeschriebene Monitorbefehle

Weil das Umschalten der Groß-/Kleinschreibung häufig aus der Systemebene heraus vorgenommen wird, war es wünschenswert, Monitorkommandos auch kleingeschrieben eingeben zu können, also ohne gedrückte Shift Taste. Hierzu wird ebenfalls die erwähnte Filterroutine vor der Befehlsdecodierung des Monitors aufgerufen.

1.4.7 Erweiterung der Monitorbefehle

Der Herausgeber hatte uns vorgeschlagen, zusätzliche Monitor- und Editorkommandos im Anschluß an die bestehende Befehlskette zuzulassen. Es ist dann möglich, durch Verifizieren einer Speicherstelle weiteren RAM- oder EPROM-Speicher anzuspringen. Dies geschieht jetzt über einen MOLINK-Vektor, ähnlich dem DILINK-Vektor bei der Zeichenausgabe. - Als Beispiel einer Monitorerweiterung wurde der Befehl '—' eingeführt. Er erlaubt ein rückwärtiges Absuchen von Speicherstellen, antivalent dem Space-Kommando.

1.4.8 Erweiterung der Editorbefehle

Die Möglichkeit der Befehlserweiterung wurde mit Hilfe eines weiteren Vektors im RAM vollzogen, unserem EDLINK-Vektor.

1.4.9 KIM-Magnetbandformat

Damit sich das System bei der versehentlichen Anwahl des Ein- und Ausgabekanals 'K' (KIM-Format) nicht aufhängt, wurde dieses Bandformat vollständig eliminiert. Es wäre aber auch möglich, diese Speicherstellen so zu programmieren, daß eine benutzer-eigene Dump- und Laderoutine angesprungen wird. Beispielsweise könnte beim Load-Befehl ein verschiebliches Laden und beim Dump eine Druckerausgabe implementiert sein. (Anm. d. Hrsg.: für eine solche könnte man auch einen anderen mnemonischen Namen als 'K' implementieren.)

1.4.10 VIDEO

In Anlehnung an eine Veröffentlichung in /5/ hat der Verfasser eine Video-Platine mit dem Controller 6545 aufgebaut. Das für sie geschriebene Treiberprogramm hat folgende Teile:

- Initialisierungsprogramm
- Initialisierungstabelle
- Anzeigeprogramm ähnlich einer Schreibmaschine
- Delete-Routine
- Scroll-Routine

Weil kein zusammenhängender Speicherplatz im Monitorbereich frei ist, wurden die Bestandteile einzeln für sich in den Speicher eingefügt. Dabei wird nach dem Einschalten die Initialisierung des Video-Controllers automatisch durchlaufen und er DILINK-Vektor auf das Anezigeprogramm umgesetzt.

1.4.11 Delete-Funktion

Das Monitorprogramm ist in einigen Teilen unstrukturiert und kompliziert aufgebaut. So wird z.B. das Display-Treiberprogramm in eine Anzeige- und in eine Delete-Routine gesplittet. Die Delete-Taste wird daher nicht als hex 7F über das DILINK ausgegeben, sondern wirkt direkt auf die Delete-Routine des Displays.

Nach dem o.g. Konzept wurde auch das Video-Treiberprogramm aufgebaut. Um eine synchrone Arbeitsweise zu erreichen, wurde das Video-Delete an die PSLS-Routine angebunden. Die darzustellenden ASCII-Zeichen werden selbstverständlich über den DILINK-Vektor zum Videoprogramm geleitet. Weil zusätzlich die Cursorposition des Video-Interfaces aus dem Cursorpointer in A415 des Displays abgeleitet wird, ergibt sich ein gleiches Erscheinungsbild für beide Ausgangsmedien. Dies ist angenehmer bei den Meldungen der Magnetband-Lade- und Speicherkommandos sowie bei den ON/OFF-Nachrichten des Druckers.

1.4.12 Display ON/OFF

Im Zusammenhang mit einem Video-Interface ist das Ein- und Ausschalten des LED-Displays auf dem AIM nützlich. Das geschieht durch Setzen oder Löschen eines Bit im Axxx-Bereich.

Die Einbindung darf dabei nicht bei der Treiberroutine OUTDIS einsetzen, weil diese ja auch die Verwaltung des Display-Buffers ab hex A438 übernimmt, der von einigen Monitorkommandos als Zwischenspeicher für Adressen genutzt wird. Es wurde daher die OUTDD1-Routine umgangen, die die Anzeigeeinheit über den Peripheriebaustein 6520 treibt.

2.0 Übersicht über geeigneten Speicherplatz

Unter den o.g. Randbedingungen ergeben sich für Manipulationen am Betriebssystem die unten aufgeführten Adreßbereiche (alle Angaben in hex). Es sind auch TTY-Routinen angegeben, die für weitergehende Änderungen benutzt werden können.

Speicher von	bis	Byte	AIM 65-Routinen	REMON–Routinen	Einschränkungen
E11B	E13D	35	KB/TTY, Bit Rate Measurement	Automatische Initialisierung	Keine TTY-Eingabe
E3A4	E3FC	89	LOAD, KIM-Format	Eigene Loadroutine	Kein KIM1-Lesen
E425	E43A	22	GETID, KIM-Format	---	Kein KIM-Format
E587	E5D3	77	DUMP, KIM	Eigene Dumproutine	Kein KIM-Dump
E926	E93A	21	Scans TTY	---	Keine TTY-Eingabe
E986	E98E	9	TTY-Test	---	dito
EBDB	EC0E	52	Char. from TTY	---	dito
EC23	EC37	21	DEHALF-Subroutine	---	dito
EE40	EE66	39	KIM-Format	Delete, Videotab.	Kein KIM-Format

EF6D	EF75	9	NOPs	Display ON/OFF	--
EFB2	EFFF	78	--	Help-Routinen	--
F255	F28A	54	KIM-Format	Video-Display	Kein KIM-Format F252: JMP OUTTA1
F2BE	F2D6	25	KIM-Format	Video-Initialisierung	Kein KIM-Format F21D: 3 mal NOP
FD4E	FD57	10	TTY-Test	--	Keine TTY Ausgabe, FD4B: JMP FD58
FE7B	FE82	8	PATCH1, Adjust Baudrate	--	Korr. Adjust-Routine
FEBC	FEE7	44	READ EDIT. Commands	--	Unterdrückte Prompts
FEEF	FEF7	9	PROMPT-Routine	--	Keine TTY Ausgabe, Bei E7C2 jetzt: BEQ F8 CMP #$4C BEQ F4
FFC5	FFF9	53	--	Video-Scroll	--

3.0 Utilities

3.1 Printerflag OFF

Die Speicherzelle E765 in der Initialisierungstabelle für das Monitor-RAM muß zu 00 programmiert werden.

3.2 Fehlerbeseitigung

Der Interrupt nach einem Tape-Dump wird wieder aktiviert, indem ein CLI (hex 58) in die Speicherzelle E523 gebracht wird. - Die Rückkehr in das Monitorprogramm nach einem LOAD wird gewährleistet, wenn der Zählindex in LOAD4 bei E322 auf 02 reduziert wird.

Der Klammeraffe (@) wird als Sonderzeichen ausgeblendet, wenn in EA91 ein 41 steht.

Das Anlaufen der Kassettenlaufwerke am Ende eines Ladens wird verhindert, indem man in die Zellen ab E529 5mal NOPs schreibt (hex EA).

Die automatische Baudratenermittlung ist sichergestellt, wenn PATCH1 durch folgende Routine ersetzt wird:

```
$EFB2 38          HELP1 SEC
$EFB3 E9 2C             SBC #44
$EFB5 8D 18 A4          STA CNTL30
$EFB8 B0 03             BCS *+5        ;Kein Übertrag
$EFBA CE 17 A4          DEC CNTH30
$EFBD 60                RTS
```

Weil dieses Maschinenprogramm größer als PATCH1 ist, wird es ab Stelle EFB2 programmiert. Dann muß die Einsprungsadresse in E13C auf diesen Wert geändert werden. Ggfs. kann man den Platz von PATCH1 jetzt anders verwerten.

3.3 Repeat-Funktion

Um diese sinnvolle Funktion einzuführen, wird aus der Routine ROONEK die Utility HELP7 angesprungen. Die Speicherstellen von ECEF bis ECFF werden überschrieben mit.

```
$ECEF 20 9E EB    ROONEK JSR PHXY
$ECF2 A2 20              LDX #$20       ;Warte ca. 32 * 5 ms
$ECF4 20 EA EF           JSR HELP7
$ECF7 EA                 NOP
```

```
$ECF8 EA                NOP
$ECF9 EA                NOP
$ECFA EA                NOP
$ECFB EA                NOP
$ECFC EA                NOP
$ECFD 20 AC EB          JSR PLXY
                ROO1    ...
```

Bei gedrückt gehaltener Taste erfolgte bisher keine Rückkehr aus ROONEK. Das Rückkehren aus den Repeatprogramm wird jetzt jedoch zusätzlich ausgelöst, wenn die Warteschleife HELP7 abgelaufen ist.

```
$EFEA AD 82 A4      HELP7   LDA DRB2
$EFED 0D 7F A4              ORA ROLLFL
$EFF0 49 FF                 EOR #$FF
$EFF2 F0 06                 BEQ RETURN      ;Andere Taste
$EFF4 20 36 ED              JSR $ED36       ;Warte ca 5 ms
$EFF7 CA                    DEX
$EFF8 D0 F0                 BNE HELP7
$EFFA 60            RETURN  RTS
```

3.4 Groß- und Kleinschreibung

Die Routine GETKEY liefert das Zeichen der gedrückten Taste im Akkumulator ab und zeigt im X-Register, ob zugleich Control oder Shift betätigt wurde. Mit dem Unterprogramm KLEIN /6/ auf Adresse EFBE wird eine Auswertung durchgeführt:

```
                    HELP2 = KLEIN
$EFBE 48            KLEIN   PHA
$EFBF 8A                    TXA
$EFC0 D0 0A                 BNE GROS        ;Shift oder Control
$EFC2 68                    PLA
$EFC3 48                    PHA
$EFC4 29 40                 AND #$40
$EFC6 F0 04                 BEQ GROS        ;Keine alphan. Zeichen
$EFC8 68                    PLA
$EFC9 09 20                 ORA #$20
$EFCB 48                    PHA
$EFCC 68             GROS   PLA
$EFCD 60                    RTS
```

READ ist eine der Basisroutinen, um ein Zeichen von der Tastatur zu lesen. Sie wird nicht nur vom Monitor, sondern häufig auch von den Interpretern und Compilern aufgerufen. Daher ist die Kleinschreibroutine hier angebunden durch Ersetzen des Befehls

```
$E94A 20 40 EC      READ1   JSR GETKEY  in
$E94A 20 CE EF      READ1   JSR HELP3
```

Die Kleinschreibung erfolgt nur, wenn Bit 6 in der Speicherstelle A408 im Monram gesetzt ist.

```
$EFCE 20 40 EC      HELP3   JSR GETKEY
$EFD1 2C 08 A4              BIT $A408
$EFD4 50 03                 BVC *+5
$EFD6 20 BE EF              JSR KLEIN
$EFD9 60                    RTS
```

3.5 Kleinschreibung und Befehlsunterdrückung im Editor

Für eine komfortable Befehlsinterpretation müssen Kleinbuchstaben zu Befehlen in Großbuchstaben umgewandelt werden. Das geschieht mit FILT1. Bei Aufruf von FILTER wird zusätzlich ein Zeichen mit Echo auf dem Display gelesen.

```
                        HELP4 = FILTER
$EFDA 20 96 FE          FILTER JSR RED1
$EFDD C9 40             FILT1  CMP #$40
$EFDF 30 02                    BMI *+4
$EFE1 29 DF                    AND #$DF
$EFE3 60                       RTS
```

Um die Anbindung an den Editor zu erreichen, wurde die Dekodierroutine CD02 ab Speicherstelle FA8F umgeschrieben:

```
$FA8F 20 DD EF          CD02   JSR FILT1
$FA92 DD AC FA          LABEL1 CMP COMTBL,X
$FA95 F0 09                    BEQ CFND1
$FA97 CA                       DEX
$FA98 10 F8                    BPL LABEL1
$FA9A 20 E7 EF                 JSR HELP6    ;Aussprung Befehlserweit.
$FA9D 4C 24 EA                 JMP CRCK
$FAA0 20 1A FF          CFND1  JSR PATC15+3
```

Zur Erklärung des Befehls JSR HELP6 siehe Abschnitt 3.8.

Die Routine PATCH8 liest ein Zeichen von der Tastatur und echot es auf der Anzeige mit dem Kleiner- und Größerzeichen, den sog. Karotten. Ersetzt man in FA8A den Befehl JSR PATCH8 in

```
FA8A    20 3C E9                JSR READ
```

dann wird die Ausgabe unterdrückt und damit auch der Platz vom PATCH8 freigesetzt.

3.6 Kleingeschriebene Monitorbefehle

Damit die Command-Decodierroutine im Monitor auch kleingeschriebene Befehle erkennt, muß lediglich in E18A der Unterprogrammaufruf von Filter erfolgen:

```
E18A    20 DA EF                JSR FILTER
```

3.7 Erweiterung der Monitor-Befehle

Weil das KIM-Format entfällt, stehen fünf Bytes im Bereich A4xx des MONRAM zur Verfügung, wobei

A408 Bit 6 zur Umschaltung Groß/Klein und
Bit 7 als Schalter für Display ON/OFF dient.
Bit 0 bis 5 stehen dem Anwender zur Verfügung.
A40A, A40B werden als Aussprungvektor für Monitorbefehle benutzt (MOLINK)
A40C, A40D dito für Editorbefehle (EDLINK).

Die Initialisierung der fünf Bytes nach einem Power On geschieht über eine Änderung der entsprechenden Bytes in der Initialisierungstabelle des Monitor-ROMs:

```
E75C 00   Großbuchstaben, Display ON
E75E D4   (MOLINK) = QM
E75F E7   höherwertiges Byte von QM
E760 D4   (EDLINK) = QM
E761 E7
```

Der Aussprung aus der Befehlsdecodierung des Monitors erfolgt bei E19E mit einem Unterprogrammaufruf. Das aufgerufene Unterprogramm besteht nur aus dem Befehl JMP (VEKTOR). Dadurch wird ein indirekter Unterprogrammaufruf simuliert wie JSR (VEKTOR). Die Link-Vektoren bei VEKTOR sind durch die Initialisierung auf die Fragezeichen-Ausgaberoutine gerichtet. Weil

der Aussprung bei dem ehemaligen Aufruf von QM erfolgt, sieht der Benutzer keinen augenfälligen Unterschied bei der Befehlsdecodierung:

```
E19E    20 E4 EF          JSR HELP5
EFE4    6C 0A A4   HELP5   JMP (MOLINK)
```

Diese Maßnahme gestattet, daß jede Erweiterung auch im Fehlerfall mit einem RTS abgeschlossen werden kann.

Als Beispiel einer externen Monitorroutine, einer Befehlserweiterung, wird hier ein "–"-Kommando vorgestellt. Dieser Befehl erlaubt ein rückwärtiges Absuchen des Speicherraumes.

```
INIT    LDA#<EXMON      ;Umsetzen des MOLINK
        STA MOLINK
        LDA #>EXMON
        STA MOLINK+1
        BRK

EXMON   CMP #'-'        ;Gültiger Befehl?
        BNE FEHLER

        JSR BLANK       ;"-" Command
        LDY #4
        JSR BAKADR      ;ADDR = ADDR - 4
        JSR $E615       ;Zeige die nächsten 4 Speicherstellen
        RTS             ;Zurück zum Monitor bei gültigem Befehl

BAKADR  LDA ADDR        ;Hilfsroutine
        STY ADDR
        SEC
        SBC ADDR
        STA ADDR
        BCS *+5
        DEC ADDR+1
        RTS

FEHLER  JSR QM          ;Ausgabe des Fragezeichens
        RTS             ;Zurück zum Monitor im Fehlerfall
```

3.8 Erweiterung der Editorbefehle

Der EDLINK-Vektor wurde bereits in 3.7 dargestellt. Abschnitt 3.5 enthielt bereits die Verwertung kleingeschriebener Befehle, auch Zusatzbefehle, in Form des JSR HELP6. Das Unterprogramm HELP6 besteht dabei, äquivalent dem HELP5, nur aus dem Befehl

```
EFE7    6C 0C A4          HELP6 JMP (EDLINK)
```

Die Initialisierungsroutine setzt den Vektor in EDLINK ebenfalls auf die Routine QM. Eine Be fehlserweiterung könnte danach folgendermaßen aufgebaut sein:

```
INIT    LDA#<EXEDIT     ;Umsetzen des EDLINK
        STA EDLINK
        LDA #>EXEDIT
        STA EDLINK+1
        BRK

EXEDIT  CMP #'-'         ;Gültiger Befehl?
        BNE FEHLER

        ...              ;Anwenderroutine, z.B. "Change to End" aus
```

```
                    ;MICRO MAG 18, S. 38, die eine Zeichenkette
                    ;in sämtlichen Zeilen gegen eine andere
                    ;austauscht.

        RTS         ;Rücksprung zum Editor bei gültigem Befehl

FEHLER  JSR QM      ;Ausgabe des Fragezeichens
        RTS         ;Rückkehr zum Editor im Fehlerfall
```

Sowohl den Vektor MOLINK, als auch EDLINK, kann man auch direkt über die Tastatur ändern, weil beide Vektoren ja nur bei Befehlen angesprungen werden, die nicht zum Grundvorrat gehören. Dies unterscheidet sie erheblich vom Vektor in DILINK, der bekanntlich mit Vorsicht zu genießen ist und den man nur über ein kleines Programm ändern kann.

3.9 KIM-Magnetbandformat

Die bisher dargestellten Utilities konnten alle im früher ungenutzten Speicherbereich von EFB2 bis EFFF untergebracht werden. Dabei wurden nur 73 von 76 zur Verfügung stehenden Bytes belegt. Platz von mehr als 300 Byte gewinnt man aber erst, wenn man das KIM-Format eliminiert. Damit das System auch dann noch funktioniert, und damit die fünf Zellen im MONRAM frei werden, sind dabei folgende Änderungen im Speicher durchzuführen:

E474	4x NOP (hex EA)	
E860	8x NOP	
E889	5x NOP	
E9A1	3x NOP	
E9CC	4x NOP	
EE3B	A9 C7	LDA #$C7
EE3D	EA	NOP
EE8E	A9 C7	LDA #$C7
EE90	EA	NOP
EE94	C9 C7	CMP #$C7
EE96	EA	NOP
EE9A	C9 C7	CMP #$C7
EE9C	EA	NOP
F21D	EA EA EA	3x NOP
F252	4C 90 F2	JMP F290

3.10 Video-Betriebsprogramm

Für einen Video-Betrieb müssen Initialisierungs-, Anzeige-, Delete- und Scrollroutinen in den Monitor eingebunden werden. Wegen ihrer Komplexität wurde der Delete-Routine ein eigener Abschnitt eingeräumt. - Treiberprogramme für Video werden bei den einzelnen Anwendern stark voneinander abweichen. Gleichwohl soll exemplarisch eine mögliche Anbindung an das Monitorprogramm vorgestellt werden. Dabei war es das Bemühen, die Lücken möglichst vollständig auszufüllen.

Als Bildformat werden 16 Zeilen je 64 Zeichen verwendet = 1024 Bytes. Es wird immer in die 16. Zeile geschrieben. Die Cursorposition innerhalb dieser Zeile wird vom Display-Cursor CURPO2 in A415 abgeleitet. Die Anzeigeroutine übernimmt ein Zeichen aus dem Akkumulator und speichert es in das Video-RAM ab 7BC0 + Inhalt von CURPO2.

```
$F255 20 9E EB      VIDEO JSR PHXY     ;Save X, Y
$F258 48                  PHA          ;Save Accu
$F259 48                  PHA
$F25A A0 0F               LDY #15      ;Select Videocontroller
$F25C 8C 00 A3            STY VIDADR   ;Cursorregister low
$F25F AD 15 A4            LDA CURPO2   ;Display - Cursor
```

```
$F262 09 C0             ORA #$C0
$F264 A8                TAY
$F265 68                PLA
$F266 29 7F             AND #$7F
$F268 C9 0D             CMP #$0D
$F26A F0 0A             BEQ CR
$F26C C0 FC             CPY #$FC    ;Keine Ausgabe, wenn CURPO2 = 60
$F26E F0 0E             BEQ EXIT
$F270 99 00 7B          STA VIDRAM,Y;Display Charactor
$F273 C8                INY
$F274 D0 05             BNE *+7
$F276 20 C5 FF      CR  JSR SCROLL
$F279 A0 C0             LDY #$C0
$F27B 8C 01 A3          STY VIDREG  ;Andere Cursorposition
$F27E 68            EXIT PLA
$F27F 20 AC EB          JSR PLXY
$F282 4C 05 EF          JMP OUTDIS  ;Zur Display - Routine
```

Es muß jetzt nur noch der DILINK-Vektor auf den Anfang dieser Routine gerichtet werden. Dazu wird der Wert in der Initialisierungstabelle des AIM 65 geändert:

E75A 55 F2

Die Scroll-Routine füllt die 17 (nicht sichtbare) Zeile mit Blanks und schiebt den Bildschirminhalt um eine Zeile höher, d.h. den Bildspeicherinhalt um 64 Adressen tiefer. Eine entsprechend große Speicherlücke liegt bei FFC5:

```
$FFC5 A0 3F         SCROLL LDY #63      ;64 Blanks
$FFC7 A9 20                LDA #' '
$FFC9 99 00 7C             STA $7C00,Y
$FFCC 88                   DEY
$FFCD 10 FA                BPL *-4
$FFCF A0 00                LDY #00
$FFD1 B9 40 78             LDA $7840,Y
$FFD4 99 00 78             STA $7800,Y
$FFD7 C8                   INY
$FFD8 D0 F7                BNE *-7
$FFDA B9 40 79             LDA $7940,Y
$FFDD 99 00 79             STA $7900,Y
$FFE0 C8                   INY
$FFE1 D0 F7                BNE *-7
$FFE3 B9 40 7A             LDA $7A40,Y
$FFE6 99 00 7A             STA $7A00,Y
$FFE9 C8                   INY
$FFEA D0 7F                BNE *-7
$FFEC B9 40 7B             LDA $7D40,Y
$FFEF 99 00 7B             STA $7B00,Y
$FFF2 C8                   INY
$FFF3 D0 F7                BNE *-7
$FFF5 60                   RTS
```

Der Videocontroller 6545 verlangt nach einem Reset oder Power On eine Initialisierung von 16 Registern. Die Routine dazu liegt bei den Autoren ab F2BE. Die Initialisierungstabelle mit den Parametern liegt hinter der Delete-Routine ab EE55.

```
                    VIDTAB = $EE55
$F2BE A2 00         INIT   LDX #0       ;Init. Controller
$F2C0 8E 00 A3             STX VIDADR
$F2C3 BD 55 EE             LDA VIDTAB,X
```

```
$F2C6 8D 01 A3          STA VIDREG
$F2C9 E8                INX
$F2CA E0 10             CPX #16
$F2CC D0 F4             BNE *-10
$F2CE 20 C5 FF          JSR SCROLL    ;Löschen des Schirms
$F2D1 CA                DEX
$F2D2 D0 FA             BNE *-4
$F2D4 EA                NOP
$F2D5 EA                NOP
$F2D6 60                RTS
```

Diese Routine füllt die Speicherlücke komplett. Die letzten drei Byte ermöglichen allerdings einen absoluten Sprung zu weiteren Initialisierungsroutinen, die dann natürlich mit einem RTS abzuschließen wären. Als geeignete Stelle für die Anbindung der Video-Initialisierung erwies sich das Unterprogramm PAT21 ab FF72. Der zweite Aufruf von CRLF wurde herausgenommen und das Unterprogramm um die drei Speicherstellen verschoben, damit eine geeignete Implementierung vorgenommen werden konnte:

```
$FF72 20 BE F2      PAT21   JSR INIT     ;Aussprung zur User -
$FF75 A0 00                 LDY #0       ;Initalisierung
$FF77 B9 88 FF      PAT21A  LDA POMSG,Y
$FF7A F0 06                 BEQ PAT21B
$FF7C 20 7A E9              JSR OUTPUT
$FF7F C8                    INY
$FF80 D0 F5                 BNE PAT21A
$FF82 20 F0 E9      PAT21B  JSR CRLF
$FF85 4C 82 E1              JMP START    ;Starte Monitor
```

Zur Vervollständigung wird hier noch die Initialisierungstabelle für Video aufgeführt:

```
EE55    50 40 40 07 1E 00 10 18
EE5D    00 09 69 69 00 00 03 C0
```

3.11 Delete-Funktion

Für diese gibt es im Monitor die Routine PSLS ab E7DC. Wenn die Cursorposition kleiner als 20 ist, wird ein Blank auf das Display geschrieben, ansonsten wird das Display nur gescrollt. Es ist sinnvoll, auch für das Videoprogramm eine eigene Delete-Routine zu schaffen:

```
$EE40 AD 15 A4      VIDRUB  LDA CURPO2
$EE43 F0 0F                 BEQ RETURN
$EE45 CE 15 A4              DEC CURPO2
$EE48 09 C0                 ORA #$C0
$EE4A AA                    TAX
$EE4B CA                    DEX
$EE4C 8E 01 A3              STX VIDREG
$EE4F A9 20                 LDA #' '
$EE51 9D 00 7B              STA VIDRAM,X
$EE54 60            RETURN  RTS
```

Die Einbindung in das Monitorprogramm geschieht über eine Änderung des Befehls

```
E7E4    20 02 EF        DEC CUROP2 in:
E7E4    20 40 EE        JSR VIDRUB.
```

Gleichzeitig kann man noch einen weiteren Mangel der Display-Delete-Routine beheben, indem man verhindert, daß das für das Display bestimmte Blank über den DILINK-Vektor gegeben wird. Dazu ändert man:

```
E7F0    20 02 EF        JSR OUTDP1 in:
E7F0    20 05 EF        JSR OUTDIS.
```

3.12 Display ON/OFF

Besonders beim Video-Betrieb taucht der Wunsch auf, daß Display über einen Softwareschalter an- und abschalten zu können. Dazu wird eine Routine angegeben, die nur dann OUTDD1 in EF7B anspringt, wenn Bit 7 in Zelle A408 gelöscht ist. OUTDD1 treibt die 16-Segment-LEDs. Ist das Bit gesetzt, erfolgt eine Verzweigung auf ein RTS und damit eine sofortige Rückkehr zum Aufrufer, ohne daß die LEDs bedient werden. Um ein zusätzliches Byte zu gewinnen, wird aus einem JMP-Befehl ein Branch-Befehl gemacht. Das Programm wurde in eine Lücke der vorhandenen Display-Routine gebracht, in der 9x NOP steht.

```
$EF6A F0 0A          BEQ OUTD7     ;Ende OUTDIS - Routine
$EF6C 2C 08 A4       BIT $A408     ;Flag
$EF6F 30 09          BMI *+11      ;Display OFF
$EF71 48             PHA           ;Normaler Aufruf von OUTDD1
$EF72 8A             TXA
$EF73 48             PHA
$EF74 10 08          BPL *+10      ;Verzweige immer
```

Um zu erreichen, daß die OUTDD1-Routine immer über die Display ON/OFF-Verzweigung läuft, werden drei Bytes ersetzt durch:

```
EF7B    4C 6C EF          JMP EF6C
```

4.0 Schlußbemerkungen

Bei Abfassen des Artikels ist den Autoren erst richtig bewußt geworden, wieviel Änderungen und Verbesserungen sich im Laufe der Entwicklungszeit angesammelt haben. Gute Dokumentation ist daher ein unbedingtes Muß, um ein späteres Chaos zu vermeiden. Wir haben uns bemüht, die Änderungen so klar darzustellen, damit auch weniger geübte Anwender keine Schwierigkeiten haben, die Eingriffe nachzuvollziehen. Abgesehen von der Einbeziehung des Video-Interfaces sind die anderen Utilities relativ unabhängig voneinander, so daß der Benutzer eine Wahl treffen kann, was er übernehmen möchte.

Für Systembetreiber, die keinen Eprommer haben oder die sich die Mühe der vielen Eingriffe nicht machen wollen,bieten die Autoren einen EPROM-Satz mit folgenden Spezifikationen an:

Printerflag OFF bei Power On – Fehlerbeseitigung – Groß- und Kleinschreibung über Bit 6 in A408 – Kleinschreibung und Promptunterdrückung im Editor – Befehlserweiterung über MOLINK und EDLINK – KIM-Format herausgenommen – Display ON/OFF – Möglichkeit der User-Initialisierung in den Zellen FF72-FF74, die zu FF gestzt sind und mit einem Subroutine-Aufruf überprogrammiert werden können –Alle in der Tabelle bei 2.0 genannten Speicherbereiche sind zu FF gesetzt, um eigene Implementierungen zu ermöglichen, außer den Bereichen für TTY-Routinen und denen der hier dokumentierten HELP-Utilities in EFB2 bis EFFA. (Bestellungen der zwei REMON-EPROMs Typ 2532 zu zus. DM 88,-- an Dipl.-Ing. R. Wollenberg, Stockumer Str. 234, 4600 Dortmund 50. Der Versand erfolgt per Nachnahme).

Mit diesem Beitrag werden viele der Beschränkungen, die durch das Betriebssystem des AIM 65 auferlegt waren, aufgehoben. Wir hoffen, daß viele Benutzer die geschaffenen Möglichkeiten nutzen, um leistungsfähige Software für eine Monitor- und Editorerweiterung zu entwickeln.

Wir danken dem Herausgeber für seine nützlichen und hilfreichen Hinweise.

5. Literatur

/1/ AIM Spezial (1), Das Buch 1-6 des 65xx MICRO MAG, S. 193
/2/ AIM Spezial (2), 65xx MICRO MAG 8, S. 42
/3/ AIM Spezial (8), 65xx MICRO MAG 13, S. 52
/4/ AIM Spezial (9), 65xx MICRO MAG 20, S. 61
/5/ Zimmermann, M.: Gedanken zum Video-AIM, 65xx MICRO MAG 8, S. 16
/6/ AIM Spezial (7), 65xx MICRO MAG 12, S. 43

#

AIM Spezial (13)

Die in diesem Heft im Artikel REMON vorgeschlagenen Änderungen am AIM-Monitor wurden beim Herausgeber praktisch simultan vollzogen (ohne den Video-Teil). Nach diesen Bearbeitungen hier einige Bemerkungen und weitere Anregungen:

Die Änderungen wurden von den Autoren Wollenberg und Redder sehr sorgfältig vorbereitet. Sie sind lauffähig und dort mit BASIC, hier mit FORTH erprobt worden, jedoch noch nicht mit PL/65 oder mit 'Instant PASCAL'. Es ist aber kein Grund zu sehen, daß für letztere Beschränkungen einträten. Monitor und Editor waren bisher schon ein Fleckerlteppich und bleiben es natürlich auch weiter, weil die Einsprungspunkte für die Sprachen bewahrt werden mußten. Und wahrscheinlich wird man auch fast alle Anwenderprogramme, soweit sie Systemroutinen benutzten, ohne Änderungen einsetzen können.

Mit REMON gelangt der inzwischen vier Jahre alte AIM 65 auf eine neue Leistungstufe. Zunächst einmal ist das Arbeiten in Monitor und Editor durch den Tasten-Repeat jetzt komfortabler und schneller (keine unnötigen Prompts). Der Repeat ist nicht nur für formatierte Quelltexte (Einrükkungen) in den höheren Sprachen nützlich, sondern auch in der allgemeinen Textbearbeitung, zu der auch der AIM zunehmend herangezogen wird, natürlich auch mit der jetzt fest implementierten Kleinschreibung.

Von mindestens gleicher Bedeutung ist die jetzt bequem eröffnete Möglichkeit, erweiterte Befehlsebenen des Anwenders sowohl für den Monitor wie auch für den Editor aufzumachen: Nach der vergeblichen Entschlüsselung eines dem AIM ursprünglich nicht bekannten Befehls erfolgt jetzt immer ein indirekter Sprung über einen vom Anwender veränderbaren RAM-Vektor in den Zellen A40A/0B bzw. A40C/0D. Viele der in dieser Zeitschrift schon veröffentlichten Dienstprogramme können jetzt in kürzerer und eleganterer Form in Anwender-Betriebssysteme oder Programme eingebunden werden. Neue werden hinzukömmen und sollten veröffentlicht werden. Für solche Zwecke haben die Autoren des REMON bereits viel nicht mehr benötigten Speicherplatz freigeräumt.

Der Herausgeber möchte für den Bedarfsfall weitere Veränderungen anregen: Viele Anwender verzichten bereits auf den Gebrauch des Thermodruckers. Wenn man ihn stilllegt, kommt man erstens ohne die 24 V Versorgungsspannung aus, zweitens gewinnt man weitere mehr als 730 Byte Platz in den Festwertspeichern, und zwar durch Entfall der Druckerroutinen von F000-F18A und der Zeichenmuster für den Drucker von F2D7-F420. Dazu sind Änderungen an folgenden Stellen zu empfehlen:

```
$F000 60          OUTPRI  RTS ;(VORSICHTSHALBER)
```

Später kann hier bei F000 eine Ausgaberoutine des Anwenders beginnen, die auf einen 'richtigen' Drucker z.B mit IEEE-488- oder Centronix-Schnittstelle arbeitet. Für solche Anwendungen könnte man die System-VIA, soweit sie den Thermodrucker bedient hat, für das Interfacing heranziehen. In Hinsicht auf eigene Druckroutinen hat der Herausgeber auf seinem System alle Routinen belassen, die den Drucker ein- oder ausschalten oder die den Pfad der Ausgabe betreffen (WHEREO, OUTALL).

Die Reduzierung des Printerprogrammes auf ein vorläufiges RTS macht Änderungen an einigen anderen Stellen notwendig oder empfehlenswert. Die nicht mehr in dieser Form benötigte Initialisierung der System-VIA ab Zeile 368 des Listings (bei E0C7) stört zunächst nicht, wird aber bei Hardware-Änderungen zu berücksichtigen sein. In PATC18 ist zu ändern;

```
$FF4B EA EA EA          NOP, DREIMAL
```

In der Routine PSLS ändert man zweckmäßig:

```
$E802 20 AC EB          JSR PLXY ;ROUTINE IST GEKÜRZT
$E805 60                RTS
```

Die Zellen E806 bis E836 werden damit für andere Verwendung frei. In der GETKEY-Routine kann man den Papiervorschub (LF-Taste) z.B. mit dem genormten Zeichen für das 'ß' belegen oder die Sonderbehandlung einfach ausblenden mit:

```
$ECEC 4C 40 EC          JMP GETKEY
```

Damit werden zugleich die Stellen EC38-EC3F frei. Das Abkoppeln des Thermodruckers befreit zugleich 9 Zellen im Monitor-RAM, nämlich A474-A47C. Diese kann man jetzt im 'geschützten Bereich' für Flags benutzen oder z.B. als Pointer auf weitere Textspeicher, die man zirkular anwählt.

Es wurde auch schon erprobt, diese Zellen, die ja direkt an den Display-Buffer anschließen, zur Vergrößerung der zulässigen Zeilenlänge zu benutzen, indem in der Routine OUTDIS und dreimal hinter dem Label IN02A nicht auf dezimal 60, sondern auf eine größere Zahl, z.B. 64 prüft. Es läuft zwar, die Änderung ist jedoch nicht unbedingt zu empfehlen, weil z.B. der Assembler nur Zeilenlängen bis 60 akzeptiert, so daß hier bei den eigentlich wünschenswerten längeren Zeilen Konflikte entstünden. Wer größere Zeilenlängen haben möchte, sollte lieber den Tastatur-Eingabepuffer in einen anderen Bereich legen, z.B. ab hex 200.oder 300. Sowohl für BASIC, wie auch für FORTH, bereitet das 'Hochlegen' der Sprache keine besonderen Umstände.

Die Dienstleistung VIDRUB, das Zurücksetzen des Cursors über ein über DILINK betriebenes Terminal, ist nach einem Vorschlag von Herrn Steder einfach wie folgt zu erzielen, wenn das Terminal hex 7F als DELETE haben will:

```
$E7E7 A9 7F          LDA #$7F     DELETE-ZEICHEN
$E7E9 20 EF 02       JSR OUTDP1   ÜBER DILINK AUSGEBEN
$E7EC CE 15 A4       DEC CURPO2
$E7EF AE 15 A4       LDX CURPO2
$E7F2 E0 14          CPX #$14     20 ZEICHEN?
$E7F4 B0 05          BCS $E7FB
$E7F6 4C 02 E8       JMP PSLØØ
$E7F9 EA             NOP
$E7FA EA             NOP
$E7FB 20 F8 FE       JSR PATC12   ;ALTES PROGRAMM AB HIER
```

Benutzern eines Video-Interface ist zu empfehlen, im Zuge einer Änderung zugleich für die Befehle 'M' und '/" des Monitors (aufschlagen und verändern von Speicherstellen) folgende neuen Parameter für eine 16 Stellen breite Ausgabe/Veränderung zu programmieren:

```
$E24D A2 10          LDX #16
$E2C1 C0 10          CPX #16
$E610 A0 10          LDY #16
```

R.L.

Erweiterter Befehlssatz: CMOS-CPU R65C02

Zur Lieferung ab Januar 1982 hat Rockwell International CMOS-Versionen der bewährten 6502-CPU angekündigt. Zusammen mit dem ebenfalls von da ab lieferbaren Peripheral Interface Adapter PIA R65C21 und dem ab etwa März lieferbaren Peripheral Interface Adapter Timer PIAT R65C24 wird nun der Aufbau stromsparender CMOS-Microcomputer mit 2, 3 und 4 MHz Takt möglich.

Von diesem Aspekt abgesehen: Die neue CPU hat einen erweiterten Befehlssatz, der für viele Programmirungen effektivere Lösungen erlaubt. Er wird weiter unten besprochen. Zunächst noch zur Hardware: Die CPU R65C02 ist pinkompatibel mit der 6502. Am Adreß- und Datenbus können Standard TTL-Lasten betrieben werden, so daß eigentlich für die meisten Computer ein Umstecken der CPU möglich sein sollte. Es wird zwei weitere Versionen der CPU geben, die R65C102, die eine externe Clock erfordert und die R65C112 als Slave Processor, ebenfalls mit externer Clock.

OP CODE MATRIX

0
BRK
0 Implied
1 7

—OP Code
—Addressing Mode
—Instruction Bytes; Machine Cycles

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	MSD
0	BRK Implied 1 7	ORA (IND, X) 2 6			TSB ZP 2 5	ORA ZP 2 3	ASL ZP 2 5	RMB0 ZP 2 5	PHP Implied 1 3	ORA IMM 2 2	ASL Accum 1 2		TSB ABS 3 6	ORA ABS 3 4	ASL ABS 3 6	BBR0 ZP 3 5**	0
1	BPL Relative 2 2**	ORA (IND), Y 2 5*	ORA (IND) 2 5		TRB ZP 2 5	ORA ZP, X 2 4	ASL ZP, X 2 6	RMB1 ZP 2 5	CLC Implied 1 2	ORA ABS, Y 3 4*	INC Accum 1 2		TRB ABS 3 6	ORA ABS, X 3 4*	ASL ABS, X 3 7	BBR1 ZP 3 5**	1
2	JSR Absolute 3 6	AND (IND, X) 2 6			BIT ZP 2 3	AND ZP 2 3	ROL ZP 2 5	RMB2 ZP 2 5	PLP Implied 1 4	AND IMM 2 2	ROL Accum 1 2		BIT ABS 3 4	AND ABS 3 4	ROL ABS 3 6	BBR2 ZP 3 5**	2
3	BMI Relative 2 2**	AND (IND, Y) 2 5*	AND (IND) 2 5		BIT ZP, X 2 4	AND ZP, X 2 4	ROL ZP, X 2 6	RMB3 ZP 2 5	SEC Implied 1 2	AND ABS, Y 3 4*	DEC Accum 1 2		BIT ABS, X 3 4*	AND ABS, X 3 4*	ROL ABS, X 3 7	BBR3 ZP 3 5**	3
4	RTI Implied 1 6	EOR (IND, X) 2 6				EOR ZP 2 3	LSR ZP 2 5	RMB4 ZP 2 5	PHA Implied 1 3	EOR IMM 2 2	LSR Accum 1 2		JMP ABS 3 3	EOR ABS 3 4	LSR ABS 3 6	BBR4 ZP 3 5**	4
5	BVC Relative 2 2**	EOR (IND), Y 2 5*	EOR (IND) 2 5			EOR ZP, X 2 4	LSR ZP, X 2 6	RMB5 ZP 2 5	CLI Implied 1 2	EOR ABS, Y 3 4*	PHY Implied 1 2			EOR ABS, X 3 4*	LSR ABS, X 3 7	BBR5 ZP 3 5**	5
6	RTS Implied 1 6	ADC (IND, X) 2 6†			STZ ZP 2 3	ADC ZP 2 3†	ROR ZP 2 5	RMB6 ZP 2 5	PLA Implied 1 4	ADC IMM 2 2†	ROR Accum 1 2		JMP Indirect 3 5	ADC ABS 3 4†	ROR ABS 3 6	BBR6 ZP 3 5**	6
7	BVS Relative 2 2**	ADC (IND, Y) 2 5*†	ADC (IND) 2 5†		STZ 2 4	ADC ZP, X 2 4†	ROR ZP, X 2 6	RMB7 ZP 2 5	SEI Implied 1 2	ADC ABS, Y 3 4*†	PLY Implied 1 2		JMP (IND), X 3 6	ADC ABS, X 3 4*†	ROR ABS, X 3 7	BBR7 ZP 3 5**	7
8	BRA Relative 2 3	STA (IND, X) 2 6			STY ZP 2 3	STA ZP 2 3	STX ZP 2 3	SMB0 ZP 2 5	DEY Implied 1 2	BIT IMM 2 2	TXA Implied 1 2		STY ABS 3 4	STA ABS 3 4	STX ABS 3 4	BBS0 ZP 3 5**	8
9	BCC Relative 2 2**	STA (IND, Y) 2 6	STA (IND) 2 6		STY ZP, X 2 4	STA ZP, X 2 4	STX ZP, Y 2 4	SMB1 ZP 2 5	TYA Implied 1 2	STA ABS, Y 3 5	TXS Implied 1 2		STZ ABS 3 4	STA ABS, X 3 5	STZ ABS, X 3 5	BBS1 ZP 3 5**	9
A	LDY IMM 2 2	LDA (IND, X) 2 6	LDX IMM 2 2		LDY ZP 2 3	LDA ZP 2 3	LDX ZP 2 3	SMB2 ZP 2 5	TAY Implied 1 2	LDA IMM 2 2	TAX Implied 1 2		LDY ABS 3 4	LDA ABS 3 4	LDX ABS 3 4	BBS2 ZP 3 5**	A
B	BCS Relative 2 2**	LDA (IND), Y 2 5*	LDA (IND) 2 5		LDY ZP, X 2 4	LDA ZP, X 2 4	LDX ZP, Y 2 4	SMB3 ZP 2 5	CLV Implied 1 2	LDA ABS, Y 3 4*	TSX Implied 1 2		LDY ABS, X 3 4*	LDA ABS, X 3 4*	LDX ABS, Y 3 4*	BBS3 ZP 3 5**	B
C	CPY IMM 2 2	CMP (IND, X) 2 6			CPY ZP 2 3	CMP ZP 2 3	DEC ZP 2 5	SMB4 ZP 2 5	INY Implied 1 2	CMP IMM 2 2	DEX Implied 1 2		CPY ABS 3 4	CMP ABS 3 4	DEC ABS 3 6	BBS4 ZP 3 5**	C
D	BNE Relative 2 2**	CMP (IND), Y 2 5*	CMP (IND) 2 5			CMP ZP, X 2 4	DEC ZP, X 2 6	SMB5 ZP 2 5	CLD Implied 1 2	CMP ABS, Y 3 4*	PHX Implied 1 2			CMP ABS, X 3 4*	DEC ABS, X 3 7	BBS5 ZP 3 5**	D
E	CPX IMM 2 2	SBC (IND, X) 2 6†			CPX ZP 2 3	SBC ZP 2 3†	INC ZP 2 5	SMB6 ZP 2 5	INX Implied 1 2	SBC IMM 2 2†	NOP Implied 1 2		CPX ABS 3 4	SBC ABS 3 4†	INC ABS 3 6	BBS6 ZP 3 5**	E
F	BEQ Relative 2 2**	SBC (IND), Y 2 5*†	SBC (IND) 2 5†			SBC ZP, X 2 4†	INC ZP, X 2 6	SMB7 ZP 2 5	SED Implied 1 2	SBC ABS, Y 3 4*†	PLX Implied 1 2			SBC ABS, X 3 4*†	INC ABS, X 3 7	BBS7 ZP 3 5**	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

— New Opcode

†Add 1 to N if in decimal mode.
*Add 1 to N if page boundary is crossed.
**Add 1 to N if branch occurs to same page;
Add 2 to N if branch occurs to different page.

Der erweiterte Befehlssatz ist auf der folgenden Ganzseite in hexadezimaler Matrix dargestellt, wobei die neuen Befehle stärker umrandet sind. Wenn wir die Spalten dieser Tabelle vorwiegend von links nach rechts durchgehen, so finden wir zunächst den BRA (BRanch Always, relativ). In Spalte zwei ist eine neue indirekte Adressierungsart für arithmetische und logische Befehle eingeführt: Ein Pointer in der Zero Page enthält ohne irgendwelchen Offset die effektive Adresse.

In den Spalten 4 und C ist nun endlich auch der BIT-Befehl mit Register X indizierbar und in Spalte 9 sogar mit Direktoperand ausführbar: Teste in einer Zelle ein bestimmtes Bit, hervorragend für Interfaceregister und Flags, die in den Akku geladen wurden. Neu sind ferner TSB und STZ, Test and Reset Bit bzw. Store Zero to memory location (Spalten 4 und C), ebenfalls interessant für Flags und Interfacebausteine.

Spalte 7 enthält die Befehle RMBx und SMBx, nämlich Reset Memory Bit und Set Memory Bit in der Zeropage. Diese Befehlsgruppe ist vor allem für Anwendersysteme interessant, die Interfacebausteine in der Zero Page haben. Zur Philosophie siehe z.B. auch Heft 27 zu MC6805/68705.

Spalte F enthält die Befehlsgruppe BBRx und BBSx, Branch on Bit Set or Reset, verzweige, wenn ein in der Zeropage adressierte Bit gesetzt oder nicht gesetzt ist. Es sind 3-Byte-Befehle mit Opcode, danach offensichtlich folgender Adresse in der Zeropage (Interface-Register oder Flag-Byte) und der Verzweigungsweite im 3. Byte.

In Spalte A kann der Akku jetzt um 1 inkrementiert oder dekrementiert werden, die Register X und Y können ohne Umwege auf den Stack gebracht und zurückgeholt werden.

Von besonderem Interesse wird der indirekte mit Register X indizierte Sprung in eine Sprungtabelle sein (ON X*2 GOTO – JMP (ind),X), der den R65C02 in die Nähe des 6809 führt (Spalte C): Der Programmierer stellt mit Hilfe einer Tabelle, die er mit X überstreicht, fest welches Merkmal zutrifft, multipliziert dann X mit 2 (leider hierfür immer noch kein Rollierbefehl in X, wie es der 6805 schon hat!) und springt dann indirekt indiziert mit X in die Verzweigungstabelle.

Nachdem sich Rockwell vor mehr als vier Jahren entschloß, die 6502 als Second Source herzustellen, liegt jetzt die erste bemerkenswerte Erweiterung des Befehlssatzes vor. Über entsprechende Daten der neuen CPUs 6509 und 6510 von Commodore ist nichts bekannt.

Ein neuer Befehlssatz erfordert einen angepaßten Assembler, schließlich kann man neue Opcodes nicht per .BYT-Anweisung einprogrammieren. Es ist noch nicht bekannt, in welcher Form ein erweiterter Assembler durch den Lieferer zur Verfügung stehen wird. Bei ausreichendem Interesse ist es aber möglich, eine entsprechende Erweiterung unter dem Assembler des FORTH hier kurzfristig fertigzustellen, natürlich voll symbolisch und mit den üblichen Assembler-Directives sowie mit .OPTLIS und .OPTNOL.

Nach dem vorläufigen Datenblatt hat der neue Interfacebaustein R65C24 wie eine VIA zwei Ports mit je 8 Pin, 4 Control- Handshakeleitungen, eine CNTR-Leitung, offensichtlich zum Zählen, sowie einen 16-Bit-Zähler mit Vorspeicher (latch).

R.L.#

Dr. Werner Sänger, 8510 Fürth-Poppenreuth

Speichererweiterung für AIM 65

Einfach und preiswert

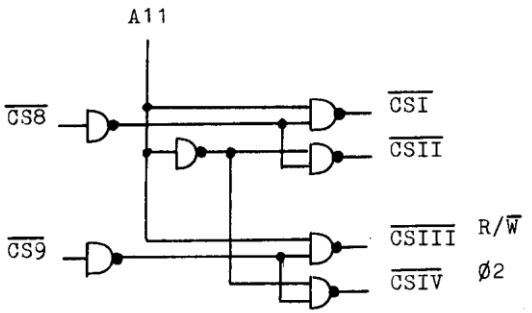
Ohne viel Aufwand läßt sich für den Bereich 8000-9FFF (8k) eine Speichererweiterung für RAM- (6116), EPROM- (2716) oder vier zusätzliche I/O-Bausteine aufbauen.

Schaltung für 6116 RAM-Bausteine

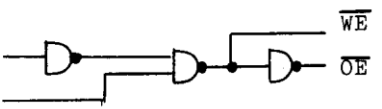
Die Decodierung für vier 2k-Bereiche wird durch $\overline{CS8}$, $\overline{CS9}$ und A11 erreicht. Dafür sind lediglich zwei 74LS00 ICs nötig. $\overline{OE}$ und $\overline{WE}$ werden von einem weiteren 74LS00 erzeugt. Eine Datenbuspufferung war beim Autor nicht nötig, wurde aber wegen späterer Erweiterungen nachträglich eingebaut (1mal 74LS245).

Aufbauvorschlag

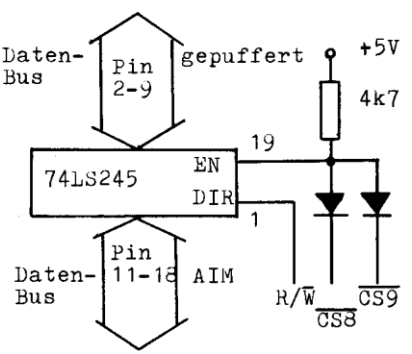
Der Elektor-Verlag hat im Septemberheft 1982 eine 8k EPROM-Platine für vier EPROMs 2716 veröffentlicht. Diese Platine (Preis ca. DM 10,-) läßt sich mit geringfügigen Änderungen (Drahtbrücken) auch für 6116 RAMs verwenden. Da diese Platine für die Aufnahme einer 31-poligen Messerleiste vorbereitet ist, verbindet man sie am einfachsten über eine entsprechende Federleiste mit der Decodierplatine. Somit kann man verschiedene Platinen wahlweise am AIM 65 betreiben. Für 6116 RAMs ist leicht eine Batteriepufferung möglich (4,5 V-Flachbatterie). Es werden lediglich zwei Dioden benötigt. Die Daten bleiben nach Ausschalten der Stromversorgung erhalten, da nun die Batterie die RAMs mit Strom versorgt.



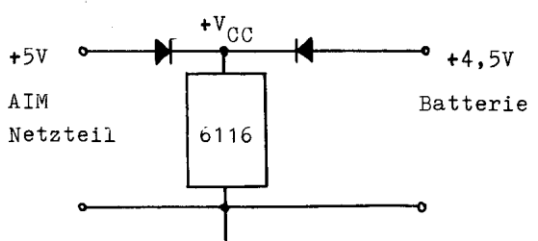
Decodierung mit zwei 74LS00

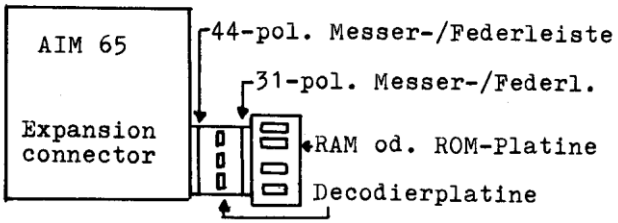


Erzeugen von OE und WE für 6116 RAM



Datenbus-Pufferung und Freigabe bei CS8 oder CS9





#

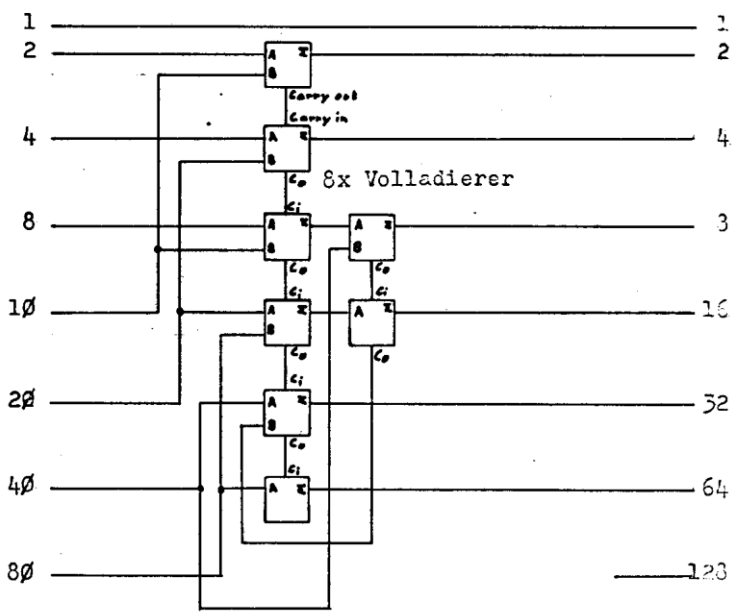
Johannes Baumann, CH-1700 Fribourg

Codewandler

Mit einem Mikroprozessor, hier dem 6502, rechnen wir normalerweise im natürlichen Binärcode oder in BCD (8–4–2–1). Nun finden wir aber Produkte mit sehr verschiedenen Codes. Z.B. ist es sinnvoll, Meßvorrichtungen (Drehgeber etc.) im Gray-Code zu verwenden. Für die oft nötige Codewandlung gibt es je nach Code verschiedene Möglichkeiten:

1. Arithmetische Lösungen

Beispiel 1.1 BCD (8-4-2-1) in natürlichen Binärcode. Hardware-Lösung:



Diese Lösung kann mit zwei IC's realisiert werden. Nach dem gleichen Prinzip lassen sich alle BCD-Codes, die jedem Bit einen Wert zuordnen, umwandeln. Dies sind z.B. Aiken-Code, Jump-at2-Code, Jump at-8-Code, 2 aus 5 (7-4-2-1-0 oder 8-4-2-1-0, Achtung: spezielle Verarbeitung der Null). Wo große Geschwindigkeiten nötig sind, ist eine hardwaremäßige Lösung vorteilhaft.

Genau das Gleiche kann man aber auch softwaremäßig realisieren:

```
ØØØØ  Eingang in BCD (8-4-2-1). Der Inhalt dieser Adresse wird
      durch das Programm verändert.
ØØØ1  Resultat im natürlichen Binärcode.
A     Der Inhalt des Akkumulators wird durch das Programm
                                                    verändert.
Ø2ØØ  18      PØ CLC
Ø2Ø1  D8         CLD          Berechnung in binär
Ø2Ø2  A5 ØØ      LDA ØØ       lade Eingang in BCD
Ø2Ø4  29 ØF      AND #ØF      Uebernahme der vier niedrigen Bit
Ø2Ø6  85 Ø1      STA Ø1
Ø2Ø8  46 ØØ      LSR ØØ       schiebe nach rechts/ teile durch zwei
Ø2ØA  18         CLC
Ø2ØB  A5 ØØ      LDA ØØ
Ø2ØD  29 78      AND #78      isoliere die vier höherwertigen Bit
Ø2ØF  65 Ø1      ADC Ø1       und addiere die vier niederwertigen Bit
Ø211  85 Ø1      STA Ø1       Zwischenresultat in Adresse Ø1
Ø213  46 ØØ      LSR ØØ       schiebe zweimal nach rechts/ teile
Ø215  46 ØØ      LSR ØØ       durch vier
Ø217  18         CLC
Ø218  A5 ØØ      LDA ØØ
Ø21A  29 1E      AND #1E      isoliere die vier höherwertigen Bit
Ø21C  65 Ø1      ADC Ø1       addiere das Zwischenresultat
Ø21E  85 Ø1      STA Ø1       speichere das Schlussresultat
Ø22Ø  6Ø         RTS          Ende
```

Beispiel 1.2 Exzess 3 in BCD und umgekehrt

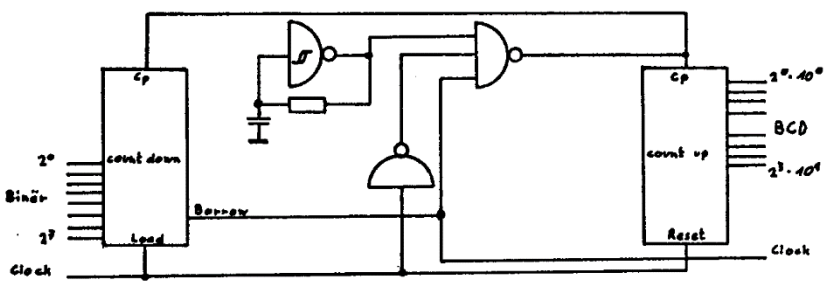
Einen Exzess-Code in seinen ursprünglichen Code zu bringen, ist natürlich sehr einfach, denn Exzess heißt Überschuß, und dieser Überschuß wird einfach abgezählt.

```
ØØØØ  Exzess 3
ØØØ1  Das Resultat in BCD (8-4-2-1)
A     Der Inhalt des Akkumulators wird durch das Programm zerstört
Ø2ØØ  18         CLC
Ø2Ø1  D8         ClD          Berechnung in binär
O2O2  A5 OO      LDA OO        lade den Exzesscode
Ø2Ø4  E9  32     SBC #32      subtrahiere den Ueberschuss
Ø2Ø6  85 Ø1      STA Ø1       speichere das Resultat
Ø2Ø8  6Ø         RTS          Ende
```

```
BCD in Exzess 3
ØØØØ  BCD (8-4-2-1)
ØØØ1  Exzess 3
A     Der Inhalt des Akkumulators wird durch das Programm zerstört
Ø2ØØ  D8       CLD          Rechnung in binär
Ø2Ø1  18       CLC          Carry auf Null setzen
Ø2Ø2  A5 ØØ    LDA ØØ       lade BCD
Ø2Ø4  69 33    ADC #33      addiere den Ueberschuss
Ø2Ø6  85 Ø1    STA Ø1       speichere das Resultat
Ø2Ø7  6Ø       RTS          Ende
```

2. Codewandlung mit Hilfe von Zählern

Bei dieser Methode wird ein Zähler mit dem bekannten Code gesetzt und zählt von da auf Null, während ein zweiter Zähler im anderen Code hochzählt. Wenn der erste Zähler auf Null ist, steht im zweiten das Resultat. Dafür das Prinzip:



Für die praktische Ausführung eines Binär-BCD-Wandlers werden ungefähr 8 IC's benötigt.

2.1 Demonstrationsprogramm

32 Bit Binärcode in BCD umwandeln. Das folgende Programm ist nicht für praktische Anwendungen gedacht, es soll vielmehr die Grenzen dieser Methode zeigen.

```
ØØØØ - ØØØ3   Eingang im natürlichen Binärcode. Niedrigstes Bit,
              Bit, Ø in ØØØØ. Diese Adressen sind nach Ausführung
              mit Null gefüllt.
ØØØ4 - ØØØ8   Ergebnis in BCD 8-4-2-1. Niedrigstes Bit, Bit Ø
              in ØØØ4.
A             Der Inhalt des Akkumulators wird verändert.
Ø2ØØ  F8        INIT  SED      rechne in BCD
Ø2Ø1  A9 ØØ           LDA  #ØØ setze den Zähler
Ø2Ø3  85 Ø4           STA  Ø4  des Ergebnisses auf Null
Ø2Ø5  85 Ø5           STA  Ø5
Ø2Ø7  85 Ø6           STA  Ø6
Ø2Ø9  85 Ø7           STA  Ø7
```

```
Ø2ØB  85 Ø8        STA  Ø8     Zähler, der binär tiefzählt
Ø2ØD  A9 ØØ  PØ    LDA  #ØØ    lade Null
Ø2ØF  C5 ØØ        CMP  ØØ     vergleiche das niederwertigste Byte
Ø211  DØ 13        BNE  P4     wenn es nicht gleich Null, springe P4

Ø213  C5 Ø1        CMP  Ø1     vergleiche das zweite Byte
Ø215  DØ ØD        BNE  P3     wenn ungleich Null, springe auf P3
Ø217  C5 Ø2        CMP  Ø2     vergleiche das dritte Byte
Ø219  DØ Ø7        BNE  P2     wenn ungleich Null,springe auf P2

Ø21B  C5 Ø3        CMP Ø3      vergleiche das höchstwertige Byte
Ø21D  DØ Ø1        BNE P1      wenn ungleich Null, springe auf P1
Ø21F  6Ø           RTS         Ende
Ø221  C6 Ø3  P1    DEC Ø3      dekrementiere das höchstwertige Byte
Ø223  C6 Ø2  P2    DEC Ø2      dekrementiere Byte 3
Ø225  C6 Ø1  P3    DEC Ø1      dekrementiere Byte 2
Ø227  C6 ØØ  P4    DEC ØØ      dekrementiere das niederwertigste Byt
Ø229  18           CLC         Zähler, der BCD hochzählt
Ø22A  A5 Ø4        LDA Ø4      lade das niederwertigste Byte (BCD)
Ø22C  69 Ø1        ADC #Ø1     addiere 1
Ø22E  85 Ø4        STA Ø4      speichere das Resultat
Ø23O  9Ø DC        BCC PØ      wenn kein Ueberlauf, springe auf PØ
Ø232  A5 Ø5        LDA Ø5      lade nächst höheres Byte
Ø234  69 ØØ        ADC #ØØ     addiere den Carry
Ø236  85 Ø5        STA Ø5      speichere das Resultat
Ø238  9Ø D4        BCC PØ      wenn kein Ueberlauf, springe auf PØ
Ø23A  A5 Ø6        LDA Ø6      lade nächst höheres Byte
Ø23C  69 ØØ        ADC #ØØ     addiere den Carry
Ø23E  85 Ø6        STA Ø6      speichere das Resultat
Ø24Ø  9Ø CC        BCC PØ      wenn kein Ueberlauf, springe auf PØ
Ø242  A4 Ø7        LDA Ø7      lade nächst höheres Byte
Ø244  69 ØØ        ADC #ØØ     addiere den Carry
Ø246  85 Ø7        STA Ø7      speichere das Resultat
Ø248  9Ø C4        BCC PØ      wenn kein Ueberlauf, springe auf PØ
Ø24A  A5 Ø8        LDA Ø8      lade höchstwertiges Byte
Ø24C  69 ØØ        ADC #ØØ     addiere den Carry
Ø24E  85 Ø8        STA Ø8      speichere das Resultat
Ø25O  9Ø BC        BCC PØ      springe auf PØ
```

Mit vorstehendem Programm kann man ganz schön die Grenzen der Methode zeigen. Wenn wir annehmen, daß wir 10 Zyklen brauchen, um einen Schritt weiterzuzählen, so kommen wir auf eine Zählfrequenz von 100 kHz (bei einem Quarz mit 1 MHz). Daraus ergeben sich folgende maximale Rechenzeiten:

Für 8 Bit 0,0026 Sek
Für 16 Bit 0,66 Sek
Für 32 Bit 11 Std 55 Min.

2.2 Wandlung von 32 Bit Binärcode in BCD

Bei vorherigen Beispiel haben wir 1 2 3 ... gezählt. Man kann aber auch zählen, wie man es bei Banknoten macht. Man beginnt bei den 1000-ern, 500-ern usw. bis hin zu den kleinen. Das folgende Programm macht im Prinzip das Gleiche. Es werden doch keine Scheine gezählt, sondern jedes Byte wird mit seiner Wertigkeit auf Null gezählt. Der Wert, der sich beim Retourzählen ergibt, wird im anderen Code in die entsprechenden Adressen addiert. Die Adressen sind dabei identisch mit denen im Beispiel 2.1.

```
Ø2ØØ  F8        INIT  SED       Initialisierung
Ø2Ø1  A9 ØØ            LDA #ØØ   setze den BCD-Zähler auf
Ø2Ø3  85 Ø4            STA Ø4    Null
Ø2Ø5  85 Ø5            STA Ø5
Ø2Ø7  85 Ø6            STA Ø6
Ø2Ø9  85 Ø7            STA Ø7
Ø2ØB  85 Ø8            STA Ø8
Ø2ØD  C5 Ø3      PØ    CMP Ø3    wenn höchstes Byte des Binär-
Ø2ØF  FØ 24            BEQ P1    zählers gleich Null, Sprung nach P1
Ø211  C6 Ø3            DEC Ø3    zählt höchstwertiges Byte runter
Ø213  A9 16            LDA #16
Ø215  65 Ø4            ADC Ø4    addiert 16777216 in den BCD- Zähler
Ø217  85 Ø4            STA Ø4
Ø219  A9 72            LDA #72
Ø21B  65 Ø5            ADC Ø5
Ø21D  85 Ø5            STA Ø5
Ø21F  A9 77            LDA #77
Ø221  65 Ø6            ADC Ø6
Ø223  85 Ø6            STA Ø6
Ø225  A9 16            LDA #16
Ø227  65 Ø7            ADC Ø7
Ø229  85 Ø7            STA Ø7
Ø22B  A9 ØØ            LDA #ØØ
Ø22D  65 Ø8            ADC Ø8
Ø22F  85 Ø8            STA Ø8
Ø231  A9 ØØ            LDA #ØØ
Ø233  FØ D8            BEQ PØ    Sprung nach PØ
```

```
Ø235  A9 ØØ     P1   LDA #ØØ
Ø237  C5 Ø2          CMP Ø2     wenn zweithöchstes Byte gleich
Ø239  FØ 22          BEQ P2     Null, Sprung nach P2
Ø23B  C6 Ø2          DEC Ø2     zählt zweithöchste Byte runter
Ø23D  A9 36          LDA #36
Ø23F  65 Ø4          ADC Ø4     addiert 65536 in den BCD-Zähler
Ø241  85 Ø4          STA Ø4
Ø243  A9 55          LDA #55
Ø245  65 Ø5          ADC Ø5
Ø247  85 Ø5          STA Ø5
Ø249  A9 Ø6          LDA #Ø6
Ø24B  65 Ø6          ADC Ø6
Ø24D  85 Ø6          STA Ø6
Ø24F  9Ø E4          BCC P1     wenn kein Ueberlauf, Sprung nach P1
Ø251  A9 ØØ          LDA #ØØ
Ø253  65 Ø7          ADC Ø7     addiere den Carry
Ø255  85 Ø7          STA Ø7
Ø257  9Ø DC          BCC P1     wenn kein Ueberlauf, Sprung nach P1
Ø259  E6 Ø8          INC Ø8     Inkrementiere höchstes Byte
Ø25B  DØ D8          BNE P1     Sprung nach P1
Ø25D  A9 ØØ     P2   LDA #ØØ
Ø25F  65 Ø1          CMP Ø1     wenn mittleres Byte gleich
Ø261  FØ 24          BEQ P3     Null, Sprung nach P3
Ø263  C6 Ø1          DEC Ø1     zählt mittleres Byte runter
Ø265  A9 56          LDA #56
Ø267  65 Ø4          ADC Ø4     addiere 256 in den BCD-Zähler
Ø269  85 Ø4          STA Ø4
Ø26B  A9 Ø2          LDA #Ø2
Ø26D  65 Ø5          ADC Ø5
Ø26F  85 Ø5          STA Ø5
Ø271  9Ø EA          BCC P2     nach vollendeter Addition, retour
Ø273  A9 ØØ          LDA ØØ     nach P2
Ø275  65 Ø6          ADC Ø6
Ø277  85 Ø6          STA Ø6
Ø279  9Ø E2          BCC P2
Ø27B  A9 ØØ          LDA ØØ
Ø27D  65 Ø7          ADC Ø7
Ø27F  85 Ø7          STA Ø7
```

```
Ø281  9Ø DA        BCC P2
Ø283  E6 Ø8        INC Ø8
Ø285  DØ D6        BNE P2
Ø287  A9 ØØ   P3   LDA #ØØ
Ø289  C5 ØØ        CMP ØØ     wenn niederwertigstes Byte, des
Ø28B  FO 2Ø        BEQ P4     Binärzählers gleich Null, Sprung nach
Ø28D  C6 ØØ        DEC ØØ     zählt niederwertigstes Byte runter
Ø28F  A9 Ø1        LDA #Ø1
Ø291  65 Ø4        ADC Ø4     und addiert eins in den BCD-Zähler
Ø293  85 Ø4        STA Ø4
Ø295  9Ø FØ        BCC P3     nach vollendeter Addition, retour
Ø297  65 Ø5        ADC Ø5     nach P3
Ø299  85 Ø5        STA Ø5
Ø29B  9Ø EA        BCC P3
Ø29D  65 Ø6        ADC Ø6
Ø29F  85 Ø6        STA Ø6
Ø2A1  9Ø EØ        BCC P3
Ø2A3  65 Ø7        ADC Ø7
Ø2A5  85 Ø7        STA Ø7
Ø2A9  E6 Ø8        INC Ø8
Ø2AB  DØ DA        BNE P3
Ø2AD  6Ø      P4   RTS        Ende
```

3. Codewandlung mit den logischen Grundfunktionen

Jeden Code kann man in jeden anderen umwandeln, wenn man jedes Bit mit jedem beliebigen Bit mit NAND und NOR verknüpfen kann. Mit einem Mikroprozessor wird man das nicht häufig machen. Diese Methode ist leichter hardwaremäßig zu realisieren.. Das Problem läßt sich am leichtesten mit einem Karnaugh-Diagramm (Veitch-Diagramm) analysieren. Das Vorgehen wird hier nicht erklärt. Die beiden nachfolgenden Beispiele sind sehr einfach und schön, da diese Transformationen anders schwer zu machen sind.

3.1 Natürlicher Binärcode in Graycode 16 Bit

```
ØØØØ  Binärcode low
ØØØ1  Binärcode high
ØØØ2  Graycode low
ØØØ3  Graycode high
A     Im Akkumulator befindet sich nach Ausführung der Inhalt
      von ØØØ2.
Ø2ØØ  18     CLC
Ø2Ø1  A5 Ø1  LDA Ø1          lade höherwertiges Byte
Ø2Ø3  4A     LSR A           schiebe es nach rechts
```

```
Ø2Ø4  45 Ø1  EOR Ø1       EXOR Bit Ø mit Bit 1, Bit 1 mit Bit 2 ..
Ø2Ø6  85 Ø3  STA Ø3       speichere das Resultat
Ø2Ø8  A5 ØØ  LDA ØØ       lade niederwertiges Byte
Ø2ØA  6A     ROR A        rotiere nach rechts
Ø2ØB  45 ØØ  EOR ØØ       EXOR
Ø2ØD  85 Ø2  STA Ø2       speichere das Resultat
Ø2ØF  6Ø     RTS          Ende
```

3.2 Graycode in Binärcode 16 Bit

```
ØØØØ  Graycode low. Nach Ausführung mit Ø gefüllt.
ØØØ1  Graycode high. Nach Ausführung mit Ø gefüllt.
ØØØ2  Natürlicher Binärcode low
ØØØ3  Natürlicher Binärcode high
ØØØ4  Zwischenresultate. Nach Ausführung  höchstens Bit 1 oder
      Ø, alle Anderen Ø.
ØØØ5  Identisch ØØØ4.
A     Der Inhalt des Akkumulators ist nachher gleich ØØØ4
```

```
Ø2ØØ  A2 1Ø     INIT  LDX #1Ø  Initialisierung  für 16 Bit
Ø2Ø2  A5 Ø1           LDA Ø1   lade das höherwertige Byte
Ø2Ø4  29 8Ø           AND #8Ø  und isoliere das höchstwertigste Bit
Ø2Ø6  85 Ø4           STA Ø4   speichere es als Zwischenresultat
Ø2Ø8  85 Ø5           STA Ø5
Ø2ØA  18              CLC
Ø2ØB  Ø6 ØØ     PØ    ASL ØØ   schiebe alle 16 Bit des Gray-
Ø2ØD  26 Ø1           RØL Ø1   codes eine Stelle nach links
Ø2ØF  Ø6 Ø5           ASL Ø5   schiebe das Zwischenresultat
Ø211  26 Ø2           ROL Ø2   in den Binärcode und ihn um
Ø213  26 Ø3           ROL Ø3   eine Stelle nach links
Ø215  A5 Ø1           LDA Ø1   lade das höherwertige Byte
Ø217  29 8Ø           AND #8Ø  und isoliere das höchstwertige Bit
Ø219  45 Ø4           EOR Ø4   Exor mit dem vorhergehenden Bit
Ø21B  85 Ø4           STA Ø4   speichere das Ergebniss als
Ø21D  85 Ø5           STA Ø5   Zwischenresultat
Ø21F  CA              DEX
Ø22Ø  DØ E9           BNE PØ   wiederhole bis alle 16 Bit bearbeite
Ø222  6Ø              RTS      Ende
```

4. Codewandlung mit Hilfe einer Tabelle

Dieses Prinzip sorgt für eine sehr schnelle Transformation irgendeines Codes in irgend einen anderen. Dabei macht der Prozessor das Gleiche, wie man es bei manueller Bearbeitung anfaßt: Er braucht eine Tabelle, nimmt den bekannten Code als Adresse und holt sich von dort den neuen Code.

```
ØØØØ   original Code
ØØØ1   gesuchter Code
Ø3ØØ   bis O3FF Tabelle
A      Im Akkumulator befindet sich nachher der gesuchte Code
X      Im  X-Register befindet sich nachher der original Code

Ø2ØØ   A6 ØØ       LDX  ØØ       original Code in X-Register
Ø2Ø2   BD ØØ Ø3     LDA  Ø3ØØ,X   suche neuen Code aus der Tabelle
Ø2Ø5   85 Ø1        STA  Ø1       und schreibe ihn in Adresse Ø1
Ø2Ø7   6Ø           RTS           Ende
```

Dieser Artikel will keine abgeschlossene Arbeit sein, sondern vor allem zu eigenen Ideen anregen. Für die Bewertung eines Codewandlers noch einige Kriterien:

Übersichtlichkeit. Ist das verwendete Prinzip leicht verständlich?
Geschwindigkeit. Ist das Programm im ungünstigsten Fall schnell genug?
Speicherplatz. Hat es genügend Platz im Programm- und Arbeitsspeicher? #

Bernhard Kokula, 6700 Ludwigshafen

Prozeßtechnik mit Mikroprozessoren (6)

Puls-Proportionalregler (PPR)

Die Puls-Ausgabe erfolgt an den Stellmotor eines integrierenden Stellgliedes, zB. ein Mischventil. Es wird propotional zur Regelanweichung gesteuert. Es ist ein Regler, weil er Istwert und Sollwert vergleicht und entsprechend der Regelabweichung auf das Stellglied einwirkt, mit dem Ziel, die Regelabweichung zu minimisieren.

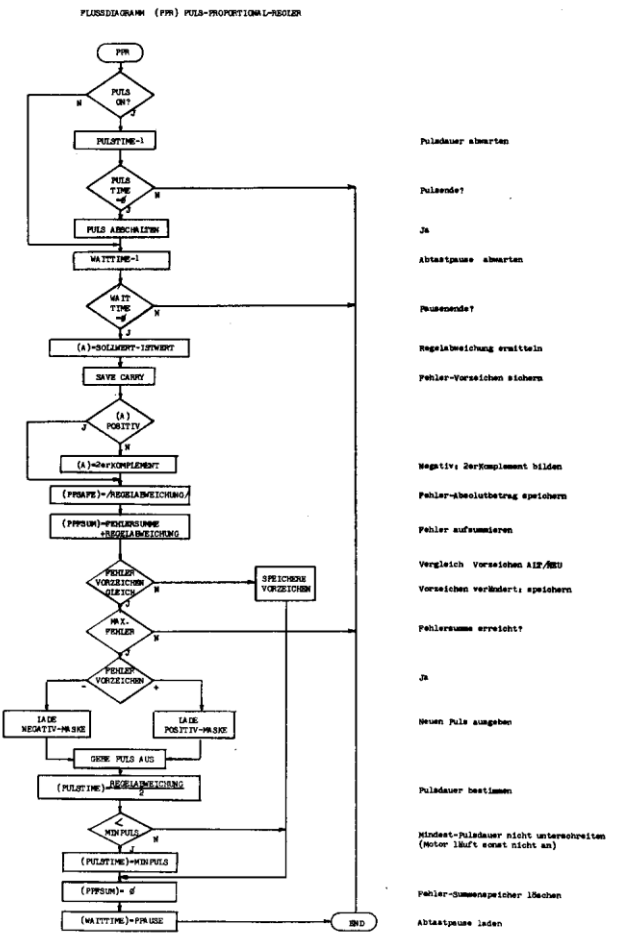
Beispiel: Eine Heizungsreglung soll mit einem Mischventil durch Ändern des Verhältnisses von Warm- und Kaltwasser die Vorlauftemperatur (Istwert) auf den gewünschten Sollwert regeln. Das Modul Dreipunktregler (DPR) ist dazu nicht in der Lage. Wohl kann es den Stellmotor links und rechts laufen lassen, er würde es aber immer gleich bis zu seiner Endstellung tun, weil der Istwert die erreichte Änderung so schnell gar nicht mitbekommt. Hier muß der Regler etwas mehr 'Intelligenz' und 'Zeitgefühl' mitbringen, um diese Regelaufgabe mit großer Totzeit in den Griff zu bekommen. Der (PPR) kann das.

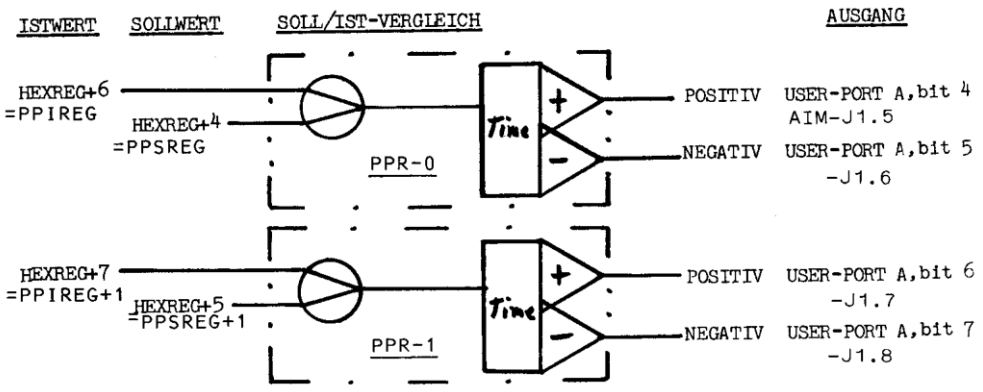
Jeder Meßzyklus des (PPR) beginnt mit der individuellen Abtastpause, die der Totzeit der Regelstrecke anzupassen ist und dem Istwert Gelegenheit gibt, vom Ergebnis des letzten Reglereingriffes etwas mitzubekommen. Das Zeitmaß ist dabei ein leicht verbesserter A/D-Wandler (ADW), der bei jeder Messung ganz abläuft und konstant 65 ms dazu braucht. Nach Ablauf der Abtastpause erfolgt ein Vergleich des neuen Istwertes mit dem Sollwert, gegebenenfalls nach Glättung mit dem Modul FILTER. Das Ergebnis ist die Regelabweichung nach Betrag und Vorzeichen. Beide werden gespeichert.

Solange keine Vorzeichenänderung auftritt, wird die Regelabweichung summiert in den Fehler-Summen-Speicher PPFSUM. Wenn dieser einen wählbaren Grenzwert erreicht, ist der nächste Puls fällig: Ein kleiner Fehler ergibt eine große Pause, ein großer Fehler eine kurze Pause (=schnelle Pulsfolge). Die Richtung des Pulses wird vom Fehlervorzeichen in PPSIGN bestimmt, die Pulsdauer wird vom Betrag der Regelabweichung bestimmt: Großer Fehler = langer Puls und umgekehrt. Damit der Stellmotor sicher anläuft, wird ein individueller Minimalpuls PPPULS nicht unterschritten. Nach Ablauf des Pulses beginnt ein neuer Meßzyklus mit der Abtastpause.

Das Modul (PPR) kann 1 bis 4 verschiedene Regelkreise bedienen und hat für jeden eigene Konstanten und Variablenregister. Es wird wie andere Module mit (X) angewählt. Das kleine Programm PPRTST zeigt seine Verwendung.

Hier das Block-Schaltbild des (PPR), Puls-Proportional-Regler und auch das Flußdiagramm für den logischen Aufbau. Der Übersichtlichkeit zuliebe sind dort alle Kontrollen der Variablen für Über- oder Unterlauf weggelassen, es ist auch so nach lang genug.





Puls-Proportional-Regler

Wer den (PPR) für nur eine Regelstrecke benutzen will, kann die indizierte Adressierung (,X) einfach weglassen. Er braucht dann auch nur für einen Regler die Konstanten und Masken bereitzustellen. Die Variablen-Bereitstellung erfolgt ja automatisch durch den Assembler mit dem Ausdruck PPZAHL=1.

Wenn man nun ein Programm aus Modulen in ein EPROM schießen will, so wird es so nicht laufen. Grund: Alle Zuweisungen, die der Assembler in der Zeropage vornimmt, fehlen nach dem Einschalten. Daher ist im EPROM ein kleines Programm nötig, das bei der Initialiserung die Zeropage beschreibt. -

```
0253        0094 ;-----------------------------------------
0253        0095 ;PAG 'MODUL (PPR) PULS-PROPORTIONAL-REGLER'
0253        0096 ;-----------------------------------------
0253        0097
0253        0098 ;ZUM REGELN VON 1 BIS 4 PROZESSGROESSEN
0253        0099 ;MIT PULS-AUSGABE 'NEGATIV-PAUSE-POSITIV'
0253        0100 ;JEDER REGLER WIRD MIT (X) ANGEWAEHLT,
0253        0101 ;ER HAT EIGENE IST- UND SOLLWERTE, SOWIE EIGENE
0253        0102 ;PULS- UND ABTASTPAUSEN-DAUER.
0253        0103 ;DEN ZEITTAKT BESTIMMT DER (ADW) MIT 65MS
0253        0104 ;DIE AUSGAENGE FUER 'OEFFNEN BEI NEGATIV' UND
0253        0105 ;'SCHLIESSEN BEI POSITIV' KOENNEN AUF BELIEBIGEN
0253        0106 ;BIT LIEGEN, ABER IM SELBEN PORT. DIE ZUORDNUNG
0253        0107 ;GESCHIEHT MIT MASKEN. HIER SIND 2 (PPR) VORGESEHEN.
0253        0108
0253        0109 ;##### (PPR)-VARIABLEN & KONSTANTEN'
0253        0110
0253        0111 PPZAHL =2              ;ANZAHL DER PP-REGLER
0253        0112
0253        0113 ;----- ZUORDNUNG DER EINGABEREGISTER ZU DEN REGLERN
0253        0114 PPSREG =HEXREG+4       ;SOLLWERTE =ADW.MESSTELLEN 4 U.5
0253        0115 PPIREG =PPSREG+PPZAHL  ;ISTWERTE  =ADW.MESSTELLEN 6 U.7
0253        0116
0253        0117        * =VARADW       ;VARIABLEN-LINK VOM ADW.4
00B9        0118 PPFSUM *=*+PPZAHL      ;FEHLERSUMMEN STORE
00BB        0119 PPSIGN *=*+PPZAHL      ;FEHLERVORZEICHEN STORE
00BD        0120 PPWAIT *=*+PPZAHL      ;ABTASTPAUSE STORE
00BF        0121 PPWORK *=*+PPZAHL      ;PULSDAUER STORE
```

```
00C1          0122 PPSAFE *=*+1            ;TEMP.STORE
00C2          0123 VARPPR =*               ;LINK ZUM NAECHSTEN MODUL
00C2          0124
00C2          0125 ;----- KONSTANTEN KOENNEN IN EPROM ODER RAM LIEGEN
00C2          0126 ;      HIER SIND SIE IM RAM UM SIE AENDERN ZU KOENNEN
00C2          0127        * =PROADW        ;PROGRAMM-LINK VON ADW.4
0253 FF       0128 PPFULL .BYT $FF         ;PPR0 FEHLERSUMME MAX
0254 FF       0129        .BYT $FF         ;PPR1
0255 05       0130 PPAUSE .BYT 05          ;PPR0 ABTASTPAUSE
0256 0A       0131        .BYT 10          ;PPR1
0257 01       0132 PPPULS .BYT 01          ;PPR0 PULSDAUER X 65 MS
0258 0A       0133        .BYT 10          ;PPR2
0259          0134
0259          0135 ;----- MASKEN ZUM EIN/AUS-SCHALTEN DER 'WORK'-BIT
0259 10       0136 PPPOS  .BYT %00010000   ;PPR0 POSITIV EIN-MASKE
025A 40       0137        .BYT %01000000   ;PPR1
025B 20       0138 PPNEG  .BYT %00100000   ;PPR0 NEGATIV EIN-MASKE
025C 80       0139        .BYT %10000000   ;PPR1
025D CF       0140 PPOFF  .BYT %11001111   ;PPR0 PULS END-MASKE
025E 3F       0141        .BYT %00111111   ;PPR1
025F          0142
025F          0143 ;----- INITIALISIERUNG <PPR>
025F          0144 UDRA   =$A001           ;DATA REG A
025F          0145 PPPORT =UDRA            ;USER-VIA PORT A
025F          0146
025F          0147 ;----- AUSGABE PORT VORBEREITEN
025F AD03A0   0148 INIPPR LDA PPPORT+2     ;DATA-DIRECTION-REG.
0262 09F0     0149        ORA #%11110000   ;BIT 4-7 'OUT'
0264 8D03A0   0150        STA PPPORT+2
0267          0151
0267          0152 ;----- ARBEITS REGISTER VORBEREITEN
0267 A900     0153        LDA #0
0269 A207     0154        LDX #PPWORK+PPZAHL-PPFSUM-1 ;ALLE AUF '00' SETZEN
026B 95B9     0155 IN1    STA PPFSUM,X
026D CA       0156        DEX
026E 10FB     0157        BPL IN1
0270 60       0158        RTS
0271          0159
0271          0160
0271          0161 ;++++++++++ PROGRAMM <PPR>
0271          0162
0271 B5BF     0163 PPR    LDA PPWORK,X     ;PULSDAUER STORE
0273 F00D     0164        BEQ PP1          ;ZEIT ABGELAUFEN
0275 D6BF     0165        DEC PPWORK,X
0277 D05E     0166        BNE PP9          ;ZEIT LAEUFT NOCH
0279          0167
0279          0168 ;----- PULSDAUER DIESES <PPR>,X IST ABGELAUFEN
0279 AD01A0   0169        LDA PPPORT
027C 3D5D02   0170        AND PPOFF,X      ;MIT ABSCHALT MASKE
027F 8D01A0   0171        STA PPPORT       ;ANSTEHENDEN PULS ABSCHALTEN
0282          0172
0282          0173 ;----- INDIVIDUELLE ABTASTPAUSE JEDES REGLERS
0282 D6BD     0174 PP1    DEC PPWAIT,X     ;<PPR>,X WARTEZEIT UM?
0284 1051     0175        BPL PP9          ;NEIN
0286          0176
0286          0177 ;----- ZEIT DES <PPR>,X IST UM, NAECHSTE MESSUNG FAELLIG
0286 38       0178        SEC
0287 B584     0179        LDA PPSREG,X     ;SOLLWERT
```

```
0289 F5B6    0180        SBC PPIREG,X    ;MINUS ISTWERT
028B 08      0181        PHP             ;RETTE CARRY, '1' BEI IST < SOLL
028C B004    0182        BCS PP2         ;ISTWERT < SOLWERT?
028E 49FF    0183        EOR #$FF        ;BILDE EINERKOMPLEMENT
0290 6901    0184        ADC #1          ;KORREKTUR ZUM ZWEIERKOMPLEMENT
0292         0185
0292         0186 ;----- SOLL/IST-ABWEICHUNG BEWAHREN & SUMMIEREN
0292 85C1    0187 PP2    STA PPSAFE      ;WIRD PULS LAENGE
0294 18      0188        CLC
0295 75B9    0189        ADC PPFSUM,X    ;ADDIERE FEHLER
0297 9003    0190        BCC PP3         ;KEIN UEBERLAUF
0299 BD5302  0191        LDA PPFULL,X    ;MAX WERT
029C         0192 PP3
029C 95B9    0193        STA PPFSUM,X    ;ZU FEHLERSUMMEN STORE
029E         0194
029E         0195 ;----- BESTIMME FEHLER-RICHTUNG
029E 68      0196 PP4    PLA             ;HOLE CARRY IN BIT 0
029F 2901    0197        AND #%00000001  ;ISOLIERE CARRY
02A1 A8      0198        TAY             ;BEWAHRE ERGEBNIS
02A2 55BB    0199        EOR PPSIGN,X    ;VERGLEICHE ALT/NEU
02A4 F004    0200        BEQ PP5         ;SIND GLEICH
02A6         0201
02A6         0202 ;----- FEHLERRICHTUNG HAT SICH GEAENDERT
02A6 94BB    0203        STY PPSIGN,X    ;SETZE NEUES VORZEICHEN
02A8 D024    0204        BNE PP8         ;LOESCHE FEHLERSUMMEN STORE
02AA         0205
02AA         0206 ;----- IST MAX.FEHLERSUMME ERREICHT?
02AA B5B9    0207 PP5    LDA PPFSUM,X    ;VERGLEICHE FEHLERSUMME
02AC DD5302  0208        CMP PPFULL,X    ;MIT HOECHSTWERT
02AF 9026    0209        BCC PP9         ;NOCH NICHT ERREICHT
02B1         0210
02B1         0211 ;----- HOESTWERT ERREICHT, PULS STARTEN
02B1 BD5B02  0212        LDA PPNEG,X     ;NEGATIV-PULS MASKE
02B4 B4BB    0213        LDY PPSIGN,X    ;FEHLER-VORZEICHEN?
02B6 D003    0214        BNE PP6         ;NEGATIV
02B8 BD5902  0215        LDA PPPOS,X     ;POSITIV-PULS MASKE
02BB 0D01A0  0216 PP6    ORA PPPORT      ;BIT EINSCHALTEN
02BE 8D01A0  0217        STA PPPORT
02C1         0218
02C1         0219 ;----- PULSDAUER LADEN
02C1 A5C1    0220        LDA PPSAFE      ;ENTHAELT REGELABWEICHNUNG
02C3 4A      0221        LSR A           ;HALBIEREN
02C4 DD5702  0222        CMP PPPULS,X    ;KLEINSTE PULSLAENGE
02C7 B003    0223        BCS PP7         ;IST NICHT KLEINER
02C9 BD5702  0224        LDA PPPULS,X    ;NIMM KLEINSTWERT
02CC 95BF    0225 PP7    STA PPWORK,X
02CE         0226
02CE         0227 ;----- FEHLERSUMMEN LOESCHEN
02CE A900    0228 PP8    LDA #0
02D0 95B9    0229        STA PPFSUM,X
02D2         0230
02D2         0231 ;----- ABTASTPAUSE STORE NEU LADEN
02D2 BD5502  0232        LDA PPAUSE,X
02D5 95BD    0233        STA PPWAIT,X
02D7 60      0234 PP9    RTS             ;FERTIG
02D8         0235
02D8         0236
02D8         0237 PR03   =*
```

```
02D8          0240 ;------------------------
02D8          0241 ;++++++++++ HAUPTPROGRAMM
02D8          0242 ;------------------------
02D8          0243
02D8          0244 PPRTST
02D8          0245 ;-----(ADW) & (PPR) INITIALISIEREN
02D8 200002   0246        JSR INIADW
02DB 205F02   0247        JSR INIPPR
02DE          0248
02DE          0249 ;-----ARBEITSSCHLEIFE
02DE A200     0250 LOOP1   LDX #0
02E0 86B8     0251        STX XPOINT
02E2 201A02   0252 LOOP2   JSR ADW          ;A/D-WANDLER AUFRUFEN
02E5          0253
02E5          0254 ;----- DANACH IMMER BEIDE (PPR) BEARBEITEN
02E5          0255 ;      (ADW) IST ZEITTAKT MIT 65MS
02E5 A200     0256        LDX #0
02E7 207102   0257 LOOP3   JSR PPR          ;REGLER AUFRUFEN
02EA E8       0258        INX             ;NAECHSTEN REGLER
02EB E002     0259        CPX #PPZAHL     ;ALLE BEARBEITET?
02ED 90F8     0260        BCC LOOP3       ;NEIN
02EF          0261
02EF          0262 ;----- NAECHSTEN A/D-WANDLER ANWAEHLEN
02EF E6B8     0263        INC XPOINT
02F1 A908     0264        LDA #ZYKLUS     ;ALLE (ADW) BEARBEITET?
02F3 C5B8     0265        CMP XPOINT
02F5 90EB     0266        BCC LOOP2       ;NEIN
02F7 B0E5     0267        BCS LOOP1       ;JA, DANN VON VORNE
02F9          0268
02F9          0269
02F9          0270 ;----- START TASTE (F1) AKTIVIEREN
02F9          0271        *=$010C
010C 4CD802   0272        JMP PPRTST
010F          0273
010F          0274        .END
  ERRORS= 000
```

#

Dipl.-Ing. (FH) Rudolf Kirchgäßner, 6831 Brühl

Vom Code zum Text: UNASS

Ein komfortabler Unassembler für den MAE

Ein Disassembler für den 6502 ist nichts neues: Der Hex-Code eines Maschinenprogramms (MPG) wird wieder lesbar, aber Zeile für Zeile muß einem Assembler mittels Editor neu eingegeben werden. Ein Unassembler dagegen bereitet den Code so auf, daß wieder ein assemblierfähiges File entsteht. Dabei werden alle Absolut-Adressen in symbolische Adressen gewandelt. Komfortabel wird dieses Verfahren dann, wenn immer wiederkehrenden Adressen für ROM Routinen oder Zero-Page-Adressen Namen gegeben werden. Dies erhöht die Lesbarkeit und Änderungsfreudigkeit beträchtlich.

Generierungsprogramm UNASS/LABGEN

Dieses Programm enthält DATA-Zeilen mit Label-Namen und den Hex-Adressen für verschiedene BASIC-Versionen sowie eine Bereichszahl. Diese gibt an, wieviele Speicherstellen sich auf dieses Label beziehen sollen.

TOPRAM,86,34,34,34,2 bedeutet also:

TOPRAM	symbolischer Name für RAM-Obergrenze
86	Adresse hex 86 für Serie 2001
34	Adresse hex 34 für Serien 3001, 4001 und 8001
2	die Speicherstelle 87 bzw. 35 soll als TOPRAM+1 decodiert werden.

LABGEN wandelt die Hex-Adressen in Dezimalzahlen, sortiert nach aufsteigenden Adressen und schreibt für jedes Betriebssystem ein File "UNASS.LABx001" auf Diskette. Da diese Generierung nur selten läuft, kann die langsame Geschwindigkeit toleriert werden. - Das hier abgedruckte Programm UNASS/LABGEN enthält nur einige Beispiele für die Zuordnungen zu Labels. Dabei spielen urheberrechtliche Dinge mit. Im praktischen Fall wird man mit eigenen Unterlagen oder mit ROM-Listings arbeiten (s.u.).

Unassembler UNASS

UNASS liest zunächst die ROM-Labels aus "UNASS.LABx001". Danach erfolgt eine zweimalige Disassemblierung. In Pass 1 werden alle Operanden-Adressen und Sprungziele daraufhin untersucht, ob sie mit ROM-Adressen übereinstimmen. Wenn nein, werden sie in einer Programm-Labeltabelle abgelegt. Der zweite Pass schreibt das Textfile. Dabei wird jede Zeile, die Sprungziel ist, mit einem Label versehen. Alle Absolut-Adressen werden mit ihren sprechenden Namen ausgegeben, soweit sie in der ROM-Tabelle stehen, ansonsten werden symbolische Namen gebildet aus dem Buchstaben "Z" und der jeweiligen Hex-Adresse. Besonders behandelt werden Programmabschnitte, die mit ".W" gekennzeichnet wurden. UNASS versteht dies als Kennzeichen für Wordfiles und legt den Hex-Code als .BY-Statement ab. Auf Sprungmarken wurde verzichtet, da diese Bereiche ohnehin nachbearbeitet werden müssen.

Bedienung

Das unbekannte Maschinenprogramm wird zunächst mit einem schnellen Disassembler inspiziert. Dabei bildet man Blöcke von ungefähr 1K Größe und sucht nach Bereichen, die keinen vernünftigen Code ergeben. Hier handelt es sich meistens um Texte (ASII-Zeichen), Konstanten und Spruntabellen. Nach dem Verschieben des MPGs in einen geschützten Bereich (notfalls mit POKE 52,x und 53,y) kann die Unassemblierung beginnen. UNASS werden nach Eingabe der Modulanzahl die Adressen der MPG-Bereiche mitgeteilt, und zwar zuerst die gegenwärtige Startadresse im geschützten Bereich, dann die logische Start- und Endadresse des MPG.

Anpassung an den MAE-Assembler

Der vorliegende Unassembler ist auf den MAE/MACROTEA zugeschnitten. Diese Assembler bieten die Möglichkeit, verschiedene Quellcode-Files zu einem Programm zu übersetzen. Dazu muß zunächst ein Control-File definiert werden, das die Namen der einzelen Teile enthält. UNASS schreibt also ein Control-File, anschließend die verschiedenen Textfiles und schließlich ein File mit den Programm-Labels. - Die File-Syntax des MAE ist:

AA 00 30 87 4D AA	: 20 43	xx xx xx xx xx xx :	xx xx 00 00 00
Start Ende	dez. Line-No.	Text, wobei im letzten	File-Ende
Textfile	hier: 4320	Byte Bit 7 gesetzt ist	

Die Differenz zwischen Ende- und Startadresse gibt dem MAE die Länge des Textfiles an. Hier wurden diese Adressen so gewählt, daß Control- und Textfile in den Textspeicher passen (die genauen Werte setzt MAE ein nach ED / ' / ' /). Eine sofortige Assemblierung der erzeugten Files ist also möglich und ergibt wieder das Original. Voraussetzung ist ein File LABEL.x001, das alle im "UNASS.LABx001" definierten Labels in der bekannten MAE-Form enthält (also TOPRAM .DE $ 34).

Die Vielseitigkeit dieses Verfahrens zeigt sich, wenn man vor dem Assemblieren das ROM Labelfile einer anderen Basic-Version vorschreibt: Das Ergebnis ist dann ein umgestelltes MPG. Zur Korrektur von Sprungtabellen oder weniger gebräuchlichen ROM-Routinen sind natürlich Feinabstimmungen notwendig.

UNASS kann leicht mit einem BASIC-Compiler übersetzt werden, wenn die Routine "SYS 826: PRINT 4, X ;" in den Zeilen 36, 135 und 148 ersetzt wird durch:

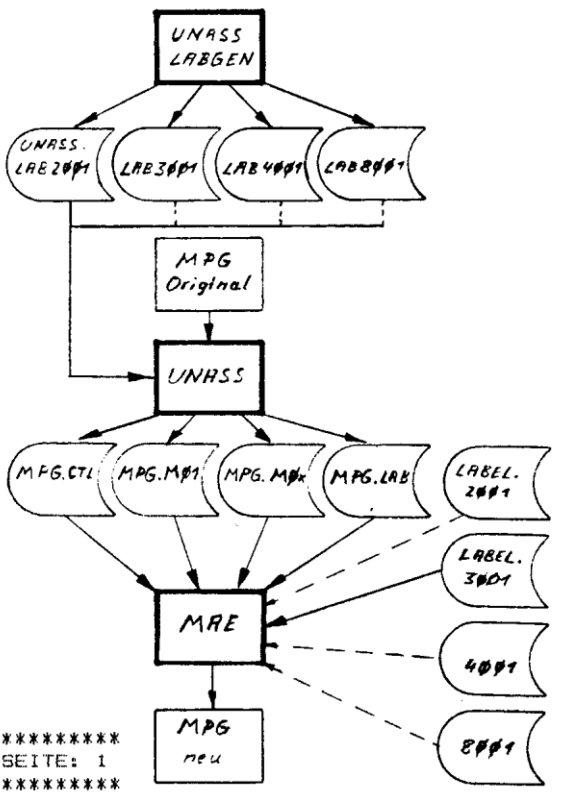
```
PRINT#4,LEFT$(X$,LEN(X$)-1)CHR$(128+ASC(RIGHT$(X$,1)));
```

Alle numerischen Variablen, außer B, C, D, E, K, S, U und L und außer dem Array C ( ), können als Intergervariable deklariert werden.

Eine Diskette mit allen benötigten Files (ca. 200 Labels) ist bei I.-M. Koch, jetzt in 6367 Karben 1, Tel. 06306–42404 erhältlich.

Literatur:

Koch, H.-J.: Speicherbelegung und ROM-Routinen, Garbsen, 1981
Moser, C.: MAE, Eastern Hourse Software, Winston Salem, N.C.
Reateo West: Programming then PET/CBM, London 1982




```
**************************
UNASS/LABGEN  SEITE: 1
**************************
```

```
0 PRINT"CLR.RVS.UNASS/LABGEN FOR 650XDOWN.
1 INPUT"OUTPUT DISK IN DRIVE   OLEFT.LEFT.LEFT.";F$
2 IF F$<>"0" AND F$<>"1"GOTO 1
3 D$=CHR$(13):DIM A$(200),C(200),L%(200),X$(3)
4 FOR I=0 TO 3:N=-1:RESTORE
5 READ A$:IF A$="*"GOTO 13
6 IF N>200 THEN PRINT"RVS.NO ROOM FOR SPECIAL LABEL":END
7 FOR X=0 TO 3:READ X$(X):NEXT:L$=X$(I):READ B%:IF L$="" THEN 5
8 N=N+1:L=0:FOR J=1 TO LEN(L$):L%=ASC(L$):L%=L%-48+(L%>64)*7:L$=MID$(L$
  ,2):L=16*L+L%:NEXT
9 IF N<1 THEN J=N:GOTO 12
```

```
10 FOR J=0 TO N-1:IF C(J)<L OR J=N-1 THEN NEXT:GOTO 12
11 FOR K=N-1 TO J STEP -1:A$(K+1)=A$(K):C(K+1)=C(K):L%(K+1)=L%(K):NEXT:
   K=J:J=N:NEXT:J=K
12 C(J)=L:A$(J)=A$:L%(J)=B%:GOTO 5

13 X=I+2:IF X=5 THEN X=8
14 OPEN 8,8,8,F$+":UNASS.LAB"+STR$(X)+"001,S,W"
15 FOR J=0 TO N:PRINT#8,A$(J)D$MID$(STR$(C(J)),2)D$MID$(STR$(L%(J)),2)
   ;:NEXT
16 PRINT#8,"*":CLOSE 8:NEXT:PRINT"OK":END

17 :
18 '   *** ROM LABELS ***
19 '   ** NAME,2001,3001,4001,8001,LENGTH **
20 :
21 DATA   ABS,DB2A,DB64,CD8E,CD8E,1
22 DATA   ACPTR,F187,F18C,F1C0,F1C0,1
23 DATA   ADD,D73F,D776,C9A0,C9A0,1
24 DATA   ADRFP,DB1B,DB55,CD7F,CD7F,1
25 DATA   AND,CED9,CECB,C089,C089,1
26 DATA   ARGBYT,D676,D678,C8D4,C8D4,1
27 DATA   ARGFLG,5E,07,07,07,1
28 DATA   ARGINT,D0A0,D090,C2E0,C2E0,1
29 DATA   ARGKLA,CE05,CDEC,BEE9,BEE9,1
30 DATA   ARGREA,CCA4,CC8B,BD84,BD84,1

**********************
UNASS  SEITE: 1
**********************

0 GOTO 38

1 '   *** BINARY INSERT ***
2 L=D-X:FOR J=0 TO V:IF D>=C(J) THEN IF D<=C(J)+L%(J) THEN J=V:NEXT:
                                                              RETURN
3 NEXT:J=W:A=1
4 J=INT(J/2):IF J=0 THEN 8
5 I=B%(A+J):IF L<I THEN 4
6 IF L>I THEN A=A+J:GOTO 4
7 RETURN

8 IF B%(A)<L THEN A=A+1:GOTO 8
9 IF B%(A)=L THEN RETURN
10 FOR J=W TO A STEP -1:B%(J+1)=B%(J):NEXT
11 B%(A)=L:W=W+1:RETURN

12 '   *** TEST FOR ROM LABEL ***
13 IF C(J)<=L THEN IF L<=C(J)+L%(J) THEN 16
14 IF C(J)>L THEN 18
15 IF J<V THEN J=J+1:GOTO 13
16 X$=A$(J):IF C(J)<L THEN PRINT#4,X$"+";:X$=MID$(STR$(L-C(J)),2)
17 GOTO 19

18 PRINT#4,"Z";:GOSUB 24
19 X$=X$+T$(I):RETURN

20 '   *** WRITE LINE # ***
21 T=T+16:IF T>144 THEN T=0:N=N+1:IF N>9 THEN N=0:Z=Z+1
22 PRINT#4,CHR$(T)CHR$(Z*16+N);:RETURN
```

```
23 '   *** DECIMAL TO HEX ***
24 IF L<Y THEN H=2:L=L/16:GOTO 26
25 H=4:L=L/4096
26 FOR J=1 TO H:L%=L:X$=X$+CHR$(48+L%-(L%>9)*7):L=16*(L-L%):NEXT:
   RETURN
27 '   *** HEX TO DECIMAL ***
28 L=0:FOR J=1 TO 4:L%=ASC(L$):L%=L%-48+(L%>64)*7:L$=MID$(L$,2):
   NEXT:RETURN                                          L=16*L+L%:

29 '   *** MODULE VALUES ***
30 L$=U$(F):GOSUB 28:K=L:T=-16:N=0:Z=0
31 L$=S$(F):GOSUB 28:S=L
32 L$=E$(F):GOSUB 28:E=L:RETURN

33 '   *** BUILD .BY IN WORD FILE ***                    O=D-U
34 D=K-S+E:FOR U=K TO D:GOSUB 21:PRINT#4," .BY";:O=8:IF U+O>D THEN
35 X$="":FOR I=U TO U+O:PRINT#4,X$;:X$=" $":L=PEEK(I):GOSUB 24:NEXT
36 SYS 826:PRINT#4,X$;:U=U+O:NEXT:RETURN

37 '   *** INITIALIZATION ***
38 PRINT"CLR.RVS.UNASSEMBLER FOR 650XDOWN.
39 PRINT"BASED ON A DISASSEMBLER DEVELOPED BY
40 PRINT"THE SILICON VALLEY PET USERS' GROUP.
41 PRINT"ORIGINAL CBM VERSION BY BILL SEILER.
42 PRINT"ALTERED FOR MAE BY JAMES STRASMA
43 PRINT"DOWN.AS OF OCTOBER 6, 1980
44 PRINT"DOWN.UPDATE DEC 15, 1981 BY HANS-JOACHIM KOCH
45 PRINT"DOWN.UPDATE OCT 10, 1982 BY R. KIRCHGAESSNERDOWN.DOWN.
46 :
47 '   PLEASE SHARE UPDATES WITH:
48 '     ^^^^TED USERS' GROUP

49 '   C/O JIM STRASMA
50 '   3838 BENTON DRIVE
51 '   DECATUR, IL. 62526
52 :
53 INPUT"DOWN.OUTPUT DISK IN DRIVE   0LEFT.LEFT.LEFT.";X$
54 IF X$<>"0" AND X$<>"1"GOTO 53
55 INPUT"DOWN.COMPUTER: RVS.20FF.001, RVS.30FF.001, RVS.40FF.001, RVS.80FF.0
   01   3LEFT.LEFT.LEFT.";L:STOP
56 IF L<1 OR L>4 AND L<>8 THEN 55
57 INPUT"CLR.PROGRAM-NAME";L$:K$="$"+X$+":"+L$:IF LEN(K$)>13 THEN 57
58 INPUT"DOWN.# OF MODULES";G:G=G-1:IF G>20 THEN 58
59 PRINT"DOWN.REAL    LOGIC     PROGRAM=P":PRINT"START START  END  DATAFILE=
   .WDOWN."
60 D=0:B=0:W=2:Q=10000:R=1000:HH=100:ZZ=10:X=32767:Y=256:N$=CHR$(0):P$=CHR$
   (174)
61 DIM N$(60),M%(255),B%(1000),U$(20),S$(20),E$(20),F$(20),A$(200),C(200),L%
   (200)
62 DIM L$(3),T$(5):L$(0)="":L$(1)="#":L$(2)="(":L$(3)="*"
63 T$(0)="":T$(1)=",X)":T$(2)=")",Y":T$(3)=",X":T$(4)=",Y":T$(5)=")"
64 B%(1)=-X:B%(2)=X:FOR F=0 TO G
65 INPUT U$(F),S$(F),E$(F),X$
66 F$(F)=K$+".M"+RIGHT$("0"+MID$(STR$(F+1),2),2):IF X$=".W" THEN F$(F)=F$
   (F)+".W":GOTO 68
67 IF X$<>"P" THEN PRINT"UP.";:GOTO 65
68 IF E$(F)<S$(F) THEN PRINT"UP.";:GOTO 65
69 IF LEN(U$(F))<>4 OR LEN(S$(F))<>4 OR LEN(E$(F))<>4 THEN PRINT"UP.";:
   GOTO 65
70 NEXT
71 PRINT"CLR.READ ROM LABELS
72 OPEN 4,8,8,"UNASS.LAB"+STR$(L)+"001,S,R"
73 N=200:FOR I=0 TO N:INPUT#4,X$:IF X$="*" THEN CLOSE 4:I=N:GOTO 75
74 V=V+1:A$(I)=X$:INPUT#4,X$:C(I)=VAL(X$):INPUT#4,X$:L%(I)=VAL(X$)-1
75 NEXT
```

```
76 '  *** READ MNEMONICS AND CODES ***
77 FOR I=0 TO 60:READ N$(I):NEXT
78 FOR I=0 TO 255:READ M%(I):NEXT
79 FOR I=826 TO 850:READ N:POKE I,N:NEXT
80 PRINT"WRITE CONTROL FILE
81 OPEN 4,8,8,K$+".CTL,P,W":T=-16:N=0:E=0
82 PRINT#4,CHR$(170)N$CHR$(48)CHR$(115)CHR$(50)CHR$(170);
83 GOSUB 21:PRINT#4,";  '"MID$(K$,4)".CTL"CHR$(167);:GOSUB 21:PRINT#4,P$
   ;
84 GOSUB 21:PRINT#4," .C"CHR$(212);:' CONTROL TABLE
85 GOSUB 21:PRINT#4," .O"CHR$(211);:' OBJECT MODULE STORE
86 GOSUB 21:PRINT#4," .C"CHR$(197);:GOSUB 21:PRINT#4,P$;:' CONTINUE IF
   ERRORS
87 GOSUB 21:PRINT#4," .FI "CHR$(34)MID$(K$,4)".LAB"CHR$(162);:' FILE "NAME"
88 FOR F=0 TO G:GOSUB 21:PRINT#4," .FI ";
89 PRINT#4,CHR$(34)MID$(F$(F),4)CHR$(162);:NEXT
90 GOSUB 21:PRINT#4," .FI "CHR$(34);
91 PRINT#4,"LABEL."CHR$(L+48)"001"CHR$(162);
92 GOSUB 21:PRINT#4,P$;
93 GOSUB 21:PRINT#4," .E"CHR$(206);:GOSUB 21:PRINT#4,N$N$N$;:CLOSE 4:'
   END
94 PRINT"RVS.PASS 10FF.: BUILD LABEL TABLE
95 FOR F=0 TO G:IF RIGHT$(F$(F),2)=".W" THEN 107
96 GOSUB 30:FOR O=0 TO E-S:U=K+O
97 L=M%(PEEK(U)):B=INT(L/Q):IF B=0 THEN 106:' ONE BYTE
98 P=INT(L/R)-B*ZZ:IF P=1 THEN 105:' IMMEDIATE
99 D=PEEK(U+1):IF B=2 THEN D=PEEK(U+2)*Y+D:GOTO 104
100 IF INT(L/ZZ)-B*R-P*HH>8 THEN 104:' NO BRANCH
101 IF D<127 THEN 103:' BRANCH ON
102 D=D-Y:' BRANCH BACK
103 D=S+O+2+D
104 GOSUB 2
105 O=O+B
106 NEXT
107 NEXT
108 PRINT"RVS.PASS 20FF.: WRITE MODULE FILE
109 FOR F=0 TO G:PRINT F$(F):GOSUB 30:OPEN 4,8,8,F$(F)+",P,W"
110 PRINT#4,CHR$(170)N$CHR$(48)CHR$(135)CHR$(77)CHR$(170);
111 GOSUB 21:PRINT#4,";  '"MID$(F$(F),4)CHR$(167);:GOSUB 21:PRINT#4,P$;
112 GOSUB 21:PRINT#4,";  "S$(F)" TO "LEFT$(E$(F),3)CHR$(128+ASC(RIGHT$(E$
    (F),1)));
113 GOSUB 21:PRINT#4,P$;
114 '  .BA = BEGIN OF ASSEMBLY
115 GOSUB 21:PRINT#4," .BA $"LEFT$(S$(F),3)CHR$(128+ASC(RIGHT$(S$(F),1)));
116 '  .MC = MOVE CODE
117 GOSUB 21:PRINT#4," .MC $"LEFT$(U$(F),3)CHR$(128+ASC(RIGHT$(U$(F),1)));
118 GOSUB 21:PRINT#4,P$;:IF RIGHT$(F$(F),2)=".W" THEN GOSUB 34:GOTO 137
119 A=1:C=B%(A)+X:FOR O=0 TO E-S:GOSUB 21:L=S+O
120 IF C>L THEN 123
121 IF C<L THEN A=A+1:C=B%(A)+X:GOTO 120
122 X$="":GOSUB 24:PRINT#4,"Z"X$;:B%(A)=-X
123 U=K+O:L=M%(PEEK(U)):B=INT(L/Q):P=INT(L/R)-B*ZZ
124 M=INT(L/ZZ)-B*R-P*HH:IF M=0 THEN X$="":L=PEEK(U):PRINT#4," .BY ";:GOTO
    134
125 IF B=0 THEN X$=" "+N$(M):GOTO 135
126 I=L-B*Q-P*R-M*ZZ:D=PEEK(U+1):IF B=2 THEN D=PEEK(U+2)*Y+D:GOTO 131
127 IF M>8 THEN 131
128 IF D<127 THEN 130
129 D=D-Y
130 D=S+O+2+D
131 PRINT#4," "N$(M)" "L$(P);:L=D:X$="":IF P<>1 THEN J=0:GOSUB 13:GOTO 135
132 IF L<ZZ THEN X$=CHR$(L+48):GOTO 135
133 IF L>31 THEN IF L<92 THEN PRINT#4,"'";:X$=CHR$(L):GOTO 135
134 PRINT#4,"$";:GOSUB 24:X$=RIGHT$(X$,2)
135 SYS 826:PRINT#4,X$;:IF M>56 THEN GOSUB 21:PRINT#4,P$;:' SPACE LINE
136 O=O+B:NEXT
137 GOSUB 21:PRINT#4,N$N$N$;:CLOSE 4:NEXT
138 PRINT"WRITE PROGRAM LABEL FILE
139 OPEN 4,8,8,K$+".LAB,P,W":T=-16:N=0:Z=0
140 PRINT#4,CHR$(170)N$CHR$(48)CHR$(135)CHR$(77)CHR$(170);
141 GOSUB 21:PRINT#4,";  '"MID$(K$,4)".LAB"CHR$(167);
142 GOSUB 21:PRINT#4,P$;:GOSUB 21:PRINT#4,P$;
143 GOSUB 21:PRINT#4,";  *** LABELS **"CHR$(170);
```

```
144 GOSUB 21:PRINT#4,P$;:GOSUB 21:PRINT#4,P$;
145 FOR I=2 TO W-1:IF B%(I)=-X THEN 149
146 '  .DE = DEFINE EXTERNAL LABELS
147 B=B%(I)+X:GOSUB 21:PRINT#4,"Z";:X$="":L=B:GOSUB 24:PRINT#4,X$" .DE $";:
    X$="":L=B:GOSUB 24
148 SYS 826:PRINT#4,X$;
149 NEXT:GOSUB 21:PRINT#4,N$N$N$;:CLOSE 4:PRINT"OK":END

150 '  *** MNEMONICS ***
151 DATA ,BEQ,BNE,BCC,BCS,BMI,BPL,BVC
152 DATA BVS,LDA,STA,LDX,STX,LDY,STY,CMP
153 DATA CPX,CPY,INC,DEC,JSR,AND,ADC,SBC
154 DATA ORA,EOR,BIT,ASL,LSR,ROL,ROR
155 DATA INX,INY,DEX,DEY,CLC,CLD,SEC,SED
156 DATA PLA,PHA,TAX,TXA,TAY,TYA,TSX,TXS,CLI
157 DATA SEI,PHP,PLP,CLV,NOP,"ASL A","LSR A","ROL A","ROR A"
158 DATA JMP,RTS,RTI,BRK
159 '  ***   MNEMONIC CODES:
160 '  *** 1.     LENGTH - 1 OF COMMAND
161 '  *** 2.     POINTER TO PREFIX ARRAY L$()
162 '  *** 3.+4.  POINTER TO MNEMO ARRAY MX()
163 '  *** 5.     POINTER TO SUFFIX ARRAY T$()
164 DATA 600,12241,,,,13240,13270,
165 DATA 490,11240,530,,,20240,20270,
166 DATA 10060,12242,,,,13243,13273,
167 DATA 350,20244,,,,20243,20273,
168 DATA 20200,12211,,,13260,13210,13290,
169 DATA 500,11210,550,,20260,20210,20290,
170 DATA 10050,12212,,,,13213,13293,
171 DATA 370,20214,,,,20213,20293,
172 DATA 590,12251,,,,13250,13280,
173 DATA 400,11250,540,,20570,20250,20280,
174 DATA 10070,12252,,,,13253,13283,
175 DATA 470,20254,,,,20253,20283,
176 DATA 580,12221,,,,13220,13300,
177 DATA 390,11220,560,,22575,20220,20300,
178 DATA 10080,12222,,,,13223,13303,
179 DATA 480,20224,,,,20223,20303,
180 DATA ,12101,,,13140,13100,13120,
181 DATA 340,,420,,20140,20100,20120,
182 DATA 10030,12102,,,13143,13103,13124,
183 DATA 440,20104,460,,,20103,,
184 DATA 11130,12091,11110,,13130,13090,13110,
185 DATA 430,11090,410,,20130,20090,20110,
186 DATA 10040,12092,,,13133,13093,13114,
187 DATA 510,20094,450,,20133,20093,20114,
188 DATA 11170,12151,,,13170,13150,13190,
189 DATA 320,11150,330,,20170,20150,20190,
190 DATA 10020,12152,,,,13153,13193,
191 DATA 360,20154,,,,20153,20193,
192 DATA 11160,12231,,,13160,13230,13180,
193 DATA 310,11230,520,,20160,20230,20180,
194 DATA 10010,12232,,,,13233,13183,
195 DATA 380,20234,,,,20233,20183,
196 '  *** BIT 7 OF THE LAST BYTE BECOMES 1 ***
197 DATA 160,2,177,42,72,200,177,42,133,1,200
198 DATA 177,42,133,2,104,168,136,177,1,9,128,145,1,96

6667 BYTES SPEICHERBEDARF (STAT.)
```

Roland Frank, 8000 München 19

Ausdruck von Kalendern

Das nachstehende FORTH-Programm gliedert sich in ein Treiberprogramm für den AIM 65 zum Betrieb einer parallelen Druckerschnittstelle/Centronix (hier für einen Epson TX-80B) und ein Kalenderprogramm, dessen Algorithmus nicht nur, wie hier geschehen, zum Ausdrucken von Kalenderblättern auf einem parallel angeschlossenen Drucker oder dem Thermodrucker des AIM 65 benutzt werden kann, sondern auch in anderen Umgebungen.

Das Treiberprogramm hat folgende Worte:

OUTIT gibt ein Zeichen im Akku über Port A der User-VIA an den Drucker aus.

INIT initialisiert diesen Port für die Druckerausgabe. PA0–PA7 werden als Ausgänge geschaltet und CA1 und CA2 für den automatischen Handshake (Pulse-Mode). Der Anschluß zwischen Computer und Drucker sieht dabei wie folgt aus:

PA0.....D0	PA5.....D5
PA1.....D1	PA6.....D6
PA2.....D2	PA7.....D7
PA3.....D3	CA1.....ACKNLG
PA4.....D4	CA2.....STROBE
Computer Drucker	Computer Drucker

Für den Anschluß anderer Drucker/Schnittstellen brauchen nur diese beiden Worte geändert zu werden.

DRUCKER–AN. Dieses Wort setzt den DILINK-Vektor auf OUTIT und ruft INIT auf.

DRUCKER–AUS setzt den DLINK-Vektor wieder auf das Anzeigeprogramm des Monitors zurück.

Nun zum eigentlichen Kalenderprogramm:

Wie errechnet man, welcher Wochentag zu einem bestimmten Datum gehört? Nach der letzten Kalenderreform durch Papst Gregor XIII gelten folgende Regeln, um zu bestimmen, ob ein Jahr ein Schaltjahr ist: Alle Jahre, deren Jahreszahl durch vier teilbar ist, sind Schaltjahre, mit Ausnahme der nicht durch 400 teilbaren Säkularjahre. Mit diesen Regeln kann man die Anzahl der Tage ab einem imaginären Nullpunkt mit folgender Formel berechnen:

$$T = (J - 1) \times 365 + \mathrm{INT}\left(\frac{J-1}{4}\right) - \mathrm{INT}\left(\frac{J-1}{100}\right) + \mathrm{INT}\left(\frac{J-1}{400}\right)$$

Dabei ist J das gewünschte Jahr und T ist die Anzahl der Tage bis zum 1. Januar dieses Jahres. Teilt man T nun modulo 7, so gibt der Rest folgendermaßen den Wochentag an: 0 entspricht Montag, 1 Dienstag usw. bis 6 = Sonntag. Damit hat man schon alle Information, um einen Kalender erstellen zu können. Es bleibt nur noch die Frage der Gültigkeit dieser Formel.

Da die Kalenderreform 1582 im Oktober stattgefunden hat, ist also 1583 das erste Jahr. Nach oben gilt diese Formel unbegrenzt, wenigstens solange der Gregorianische Kalender Gültigkeit hat. Hier zeigt sich aber nun, daß eine Reform dieses Kalenders in 'absehbarer' Zeit notwendig werden wird, und zwar aus folgendem Grund: Beim Gregorianischen Kalender ergibt sich eine Länge des Jahres von 365,2425 Tagen. Das Jahr hat aber in Wirklichkeit 365,2422 Tage. Dadurch ergibt sich alle 3333 Jahre ein Fehler von einem Tag, der irgendwann ausgeglichen werden muß. Man könnte nun z.B in allen Jahren, deren Jahreszahl durch 3200 teilbar ist, den Schalttag wiederum ausfallen lassen, wodurch sich der Fehler auf einen Tag pro 80 000 Jahre verringern würde.

Um keine fünfstelligen Jahreszahlen als Kopf drucken zu müssen, habe ich im Programm den Bereich auf 1583–1999 eingeschränkt. Bei Kalendern ab dem Jahre 3200 sollte man beachten, daß evtl. ein Fehlertag dabei ist (später mehrere bei 6400 bzw. 9600). Sollte die Reform noch zu unseren Lebzeiten durchgeführt werden, was unwahrscheinlich ist, kann man die Worte RECHNUNG und ?SCHALT wie folgt ändern:

```
: ?SCHALT
>R R 3200 MOD 0= 0= R 400 MOD 0= AND R 100 MOD 0= 0= R>
4 MOD 0= AND OR ;

: RECHNUNG
DUP ?SCHALT 2 TAGE +! DUP 1- >R R 4 / R 100 / - R 400 /
+ R 3200 / - 0 R> 365 U* D+ ;
```

Zur Bedienung des Programms

Gestartet wird mit dem Wort Kalender. Das Programm fragt daraufhin mit 'OUT ?' ab, wohin der Ausdruck gehen soll. Mit 'D' geht er zum parallel angeschlossenen Drucker, bei allen anderen Zeichen zum Thermodrucker. Danach fragt es, für welches Jahr der Kalender gedruckt werden soll. Bei Ausgabe an den Thermodrucker wird zusäztlich noch nach dem Monat gefragt. Beantwortet man das mit '0', so wird ein Kalender für das ganze Jahr gedruckt, sonst nur für den geforderten Monat. Auf dem parallelen Drucker immer immer für ein ganzes Jahr.

*Zum Schluß noch zwei Hinweise: Am Ende des Befehlswortes Jahr steht die Sequenz 4 LEERZEILEN ** ** CR CR DRUCKER-AUS. Statt der vier Leerzeilen kann man hier individuelle Texte einfügen, wenn man beachtet, daß die Zeile 80 Zeichen breit ist. - Man könnte den Kalender auch im Hexadezimalsystem ausgeben mit HEX KALENDER. Alle Angabe sind dann, wie sonst auch, weiter dezimal einzugeben.*

```
ASSEMBLER HEX

CODE OUTIT
( AUSGABE EINES ZEICHENS IN AKKU UEBER PORT A AN DRUCKER )
PHA,
BEGIN,
  A00D LDA, 2 # CMP,
0= UNTIL,
PLA, 7F # AND, D # CMP,
0= IF,
  A # LDA,
THEN,
A001 STA, RTS,
END-CODE

CODE INIT
( INITIALISIERUNG DER USER-VIA PORT A FUER DRUCKERAUSGABE )
FF # LDA, A003 STA, A # LDA, A00C STA, 7F # LDA, A00E STA,
1B # LDA, A001 STA, NEXT JMP,
END-CODE

: DRUCKER-AN ( --- )
( DRUCKERAUSGABE EINSCHALTEN )
CR INIT ' OUTIT A406 ! CR ;

: DRUCKER-AUS ( --- )
( DRUCKERAUSGABE AUSSCHALTEN )
EF05 A406 ! ;

CODE PRINTER-AN
( THERMODRUCKER EINSCHALTEN )
SEC, A411 ROR, NEXT JMP,
END-CODE

CODE PRINTER-AUS
( THERMODRUCKER AUSSCHALTEN )
```

```
A411 ROL, NEXT JMP,
END-CODE


DECIMAL FORTH


: " ( --- ADDR COUNT )
( COMPILIERT EINEN STRING INS DICTIONARY UND HINTERLAESST )
( ANFANGSADDRESSE UND LAENGE AUF DEM STACK )
34 WORD HERE DUP C@ DUP 1+ ALLOT SWAP 1+ SWAP ; IMMEDIATE

: +++ ( X ADDR[0] --- ADDR[X] COUNT[X] )
SWAP 4 * + DUP @ SWAP 2+ @ SWAP ;

: $ARRAY
( ADDR[X],COUNT[X] .. ADDR[0],COUNT[0],X+1 --- ; COMP. )
( N --- ADDR[N],COUNT[N] ; RUN-TIME )
<BUILDS 2 * 0 DO , LOOP DOES> +++ ;

: ARRAY1
( N[X] .. N[0],X+1 --- ; COMP.-TIME )
( X --- ADDR[N[X]] ; RUN-TIME )
<BUILDS 0 DO , LOOP DOES> SWAP 2 * + ;

: ARRAY2
( N[X] .. N[0] X+1 --- ; COMP-TIME )
( --- ADDR[N[0]] ; RUN-TIME )
<BUILDS 0 DO , LOOP DOES> ;

31 30 31 30 31 31 30 31 30 31 28 31  0 13 ARRAY1 TAGE
 0  0  0  0  0  0  0  0  0  0  0  0  0 13 ARRAY1 TAG-1


 2  3 16 15 14  3  2  7 ARRAY2 NULL
 2  0  0  0  0  1  0  7 ARRAY2 EINS
 7  6  0  5  4  3  2  7 ARRAY2 ZWEI
 2  3  4  5  0  5  7  7 ARRAY2 DREI
 5  5  7  9  8 18  5  7 ARRAY2 VIER
 2  3  4  4 11 10  7  7 ARRAY2 FUENF
 2  3  3 12  6  0  5  7 ARRAY2 SECHS
 6  6  6  0  5  4  7  7 ARRAY2 SIEBEN
 2  3  3  2  3  3  2  7 ARRAY2 ACHT
 6  0  5 13  3  3  2  7 ARRAY2 NEUN
 3  3  7  3  3  8  0  7 ARRAY2 AAA
11  3  3 11  3  3 11  7 ARRAY2 BBB
 2  3 10 10 10  3  2  7 ARRAY2 CCC
17  9  3  3  3  9 17  7 ARRAY2 DDD
 7 10 10 11 10 10  7  7 ARRAY2 EEE
10 10 10 11 10 10  7  7 ARRAY2 FFF

FFF EEE DDD CCC BBB AAA
NEUN ACHT SIEBEN SECHS FUENF VIER DREI ZWEI EINS NULL
16 ARRAY1 ZA

: ZAHL ( N --- ADDR )
ZA @ ;
```

```
"      ****     " " ******      " " ****     **   "
" **   **  **   " " **     ****  " " ****   **   "
" **   ****     " " ********    " " **           "
" **     **     " "   **  **    " " **********   "
"   **         " " "       **   " " "        **  "
" **       **  " " "    ******  " " "    ****     "
"      **       " 19 $ARRAY STERNCHEN

" DEZEMBER " " NOVEMBER " " OKTOBER  " " SEPTEMBER"
" AUGUST   " " JULI     " " JUNI     " " MAI      "
" APRIL    " " MAERZ    " " FEBRUAR  " " JANUAR   " "   "
13 $ARRAY MONATE

" SO" " SA" " FR" " DO" " MI" " DI" " MO" "   "
8 $ARRAY WOCHENTAG

: GET-NUMB ( N1 --- N2 )
( HOLT DIE ZAHL N2 MIT MAXIMAL N1 ZIFFERN VON DER TASTATUR )
PAD 1+ SWAP EXPECT 0 0 PAD (NUMBER) 2DROP ;

: GET-YEAR ( --- N )
0 BEGIN
  DROP CR ." JAHR? " 4 GET-NUMB DUP 1582 > OVER 10000 < AND
UNTIL ;

: GET-MONTH ( --- N )
0 BEGIN
  DROP CR ." MONAT? " 2 GET-NUMB DUP -1 > OVER 13 < AND
UNTIL ;

: ?SCHALT ( N --- F )
>R R 400 MOD 0= R 100 MOD 0= 0= R> 4 MOD 0= AND OR ;

: RECHNUNG ( N1 --- N1 D )
DUP ?SCHALT 2 TAGE +! DUP 1- >R R 4 / R 100 / - R 400 / + 0
R> 365 U* D+ ;

: FILLARRAY ( D --- )
13 0 DO
  I TAGE @ 0 D+ 2DUP 7 M/MOD 2DROP I 1+ TAG-1 !
LOOP 2DROP ;

: ** ( --- )
80 0 DO
  42 EMIT
LOOP CR ;

: *** ( --- )
." **     " ;

: **** ( --- )
."      **" ;

: DRUCK ( N1 N2 --- N1 )
ZAHL OVER 2 * + @ STERNCHEN TYPE ;

: DRUCKEZAHL ( N --- )
BASE @ DUP DUP * * /MOD SWAP BASE @ DUP * /MOD
SWAP BASE @ /MOD
7 0 DO
```

```
  I *** 11 SPACES 5 PICK DRUCK 4 QKCK DRUCK OVER DRUCK
  3 PICK DRUCK 9 SPACES **** CR DROP
LOOP 2DROP 2DROP ;

: LEERZEILEN ( N --- )
0 DO
  *** 68 SPACES **** CR
LOOP ;

: DRUCKEKOPF ( N --- )
** ** 2 LEERZEILEN DRUCKEZAHL 2 LEERZEILEN ** ** ;

: ZEILE ( N1 N2 --- N1 )
OVER WOCHENTAG TYPE DUP TAGE @ SWAP TAG-1 @ 3 PICK SWAP -
6 0 DO
  DUP 1 < OVER 4 PICK > OR IF
    3 SPACES
  ELSE
    DUP 3 .R
  THEN 7 +
LOOP 2DROP ;

: JAHR ( N --- )
DRUCKER-AN DRUCKEKOPF 1 LEERZEILEN
12 0 DO
  3 LEERZEILEN I ***
  4 1 DO
    DUP I + MONATE TYPE 15 SPACES
  LOOP ." **" CR 1 LEERZEILEN
  8 1 DO
    I ***
    4 1 DO
      OVER I + ZEILE 4 SPACES
    LOOP ." **" CR DROP
  LOOP DROP
3 +LOOP
4 LEERZEILEN ** ** CR CR DRUCKER-AUS ;

: MONAT ( N1 N2 --- N1 )
CR ."    " DUP MONATE TYPE OVER 6 .R CR CR
8 1 DO
  I OVER ZEILE DROP CR
LOOP DROP CR ;

: KALENDER ( --- )
28 2 TAGE ! BASE @ >R DECIMAL CR ." OUT? " KEY
GET-YEAR RECHNUNG FILLARRAY SWAP 68 = IF
  R> BASE ! JAHR
ELSE
  GET-MONTH R> BASE ! CR PRINTER-AN CR CR -DUP IF
    MONAT
  ELSE
    13 1 DO
      I MONAT
    LOOP
  THEN DROP PRINTER-AUS
THEN ;
FINIS
```

Ausschnitt aus dem hexadezimalen Kalender für 1983

JANUAR							FEBRUAR							MAERZ						
MO		3	A	11	18	1F	MO		7	E	15	1C		MO		7	E	15	1C	
DI		4	B	12	19		DI	1	8	F	16			DI	1	8	F	16	1D	
MI		5	C	13	1A		MI	2	9	10	17			MI	2	9	10	17	1E	
DO		6	D	14	1B		DO	3	A	11	18			DO	3	A	11	18	1F	
FR		7	E	15	1C		FR	4	B	12	19			FR	4	B	12	19		
SA	1	8	F	16	1D		SA	5	C	13	1A			SA	5	C	13	1A		
SO	2	9	10	17	1E		SO	6	D	14	1B			SO	6	D	14	1B		

#

StDir. Peter Rix, OStDir. Heiko Wolgast, 2350 Neumünster

Multiplikation und Division im FIG-FORTH

Die Verfasser erhielten mit dem FIG-FORTH 1.0 für CBM 8032 (Lowinski) unerklärbar falsche Resultate bei Programmen, die auf dem AIM 65 einwandfrei liefen. Eine Analyse ergab, daß die Assembler-Routinen für U* und U/ Fehler enthalten. Beide Routinen 'vergessen' Übertragsbits bei Schiebeoperationen.

Da beide Routinen von arithmetischen Operationen wie *, /, MOD, /MOD, */MOD */, M/, M/MOD benutzt werden, ist eine Fehlerbeseitigung dringend geboten.

Die AIM 65-Routinen für U* und U/ sind fehlerfrei, passen aber wegen ihrer größeren Länge nicht an den Platz der CBM-Routinen. Da beide Routinen für die Arbeitsgeschwindigkeit von FORTH von entscheidender Bedeutung sind, folgen nachstehend für die Reparatur von CBM-FORTH aber auch für die Laufzeitoptimierung des AIM-FORTH fehlerfreie, kürzere und schnellere Routinen:

```
decimal
code u* tya, n sty, 16 # ldy, clc,
        begin, cs
          if, clc, pha, n lda, 0 ,x adc,
              n sta, pla, 1 ,x adc,
          then,
          .a ror, n ror, 3 ,x ror, 2 ,x ror,
          dey, 0<
        until,
        1 ,x sta, n lda, 0 ,x sta, next jmp,
end-code

code u/ 16 # lda, n sta, 4 ,x asl, 5 ,x rol,
        begin, 2 ,x rol, 3 ,x rol, 1 # lda, .a ror,
          n 1+ sta, 2 ,x lda, 0 ,x sbc, tay,
          3 ,x lda, 1 ,x sbc, n 1+ ror, 0= not
          if, 2 ,x sty, 3 ,x sta, sec, then,
          4 ,x rol, 5 ,x rol, n dec, 0=
        until,
        inx, inx, ' swap jmp,
end-code

code u/ xsave stx, 6 # ldy,
        begin, 0 ,x lda, n ,y sta, inx, dey, 0= until,
        16 # ldx, n 2+ asl, n 1+ rol,
        begin, n 4 + rol, n 3 + rol, 1 # lda, .a ror,
          n sta, n 4 + lda, n 6 + sbc, tay, n 3 + lda,
          n 5 + sbc, n ror, 0= not
          if, n 4 + sty, n 3 + sta, sec, then,
```

```
        n 2+ rol, n 1+ rol, dex, 0=
    until,
     xsave ldx, n 2+ lda, 2 ,x sta, n 1+ lda,
     3 ,x sta, n 4 + lda, 4 ,x sta, n 3 + lda,
     5 ,x sta, pop jmp,
end-code
```

Die 1. und die 2. Routine ersetzen die CBM-Routinen platzsparend. Die Routinen 1 und 3 können die AIM-Routinen im ROM ersetzen, beide sind im Mittel 10 bis 30% schneller als die ROM-Routinen. #

Makro-Assembler unter FORTH

Ein 'normaler' Assembler erzeugt aus jeder Zeile des Programm-Quelltextes einen Maschinenbefehl, wenn man einmal von den Assembler-Anweisungen absieht. Ein Makro-Assembler erzeugt mit dem Aufruf eines Makros eine ganze Folge von Maschinenbefehlen. Makros werden vom Anwender für häufig ähnlich wiederkehrende Programmsequenzen als abstrakte Gerüste definiert, denen für den Fall des Aufrufes 'Parameter' mitzuteilen sind, die bei der Generierung in die Programmsequenz eingebunden werden. Parameter können dabei alles mögliche sein, z.B. Direktwerte für LDA, LDX usw., Adressen oder andere Operanden. Makros haben damit eine gewisse Ähnlichkeit zu Unterprogrammen, die man ja auch für wiederkehrende Dienste benutzt. Nur: Unterprogramme haben unveränderlichen Code, auch wenn sie sich ihre Parameter aus Maschinenregistern oder 'Briefkästen' beschaffen. Und die Befehle JSR und RTS kommen nur dann in Anwendung, wenn das Makro ein ganzes Unterprogramm erzeugen soll. Normalerweise generiert man ein Makro in die laufende Programmebene ein.

Die Parameterübergabe an Makros erfolgt bei Stellungsmakros (den zumeist verwendeten) durch eine durch den Aufbau des Makros vorgeschriebene und stets einzuhaltende Menge und Folge von Operanden im Quelltext. Makro-Assemblierung unter Assembler des FORTH bedeutet damit die Erzeugung von Befehlssequenzen (in Maschinensprache), wobei die einzusetzenden Parameter nach Menge und umgekehrt polnischer Operandenabfolge auf dem Datenstack übergeben werden.

Zum Gebrauch von Makros ist zu bemerken, daß sie wegen der Menge, Folge und Art der zu übergebenden Parameter der Dokumentation bedürfen. Man ist daher gezwungen, sein 'Makro-Handbuch' aufgeschlagen neben den Computer zu legen. Bei kurzen und einfachen Befehlsfolgen programmiert man im allgemeinen schneller ohne Makros. Sie haben ihren Wert für immer wieder mit verschiedenen Parametern gebrauchte längere oder komplexe Sequenzen.

Der nachstehende Vorschlag zur Implementierung von Makros soll keinen Katalog aller möglichen nützlichen und überflüssigen Makros bringen, sondern in erster Linie das Prinzip zeigen und besonders herausstellen, wie anpassungsfähig FORTH ist: 'Es lernt nicht nur einzelne Worte zu sprechen, sondern ganze Sätze'.

Assembler stellen in FORTH-Implementierungen wohl meistens ein eigenes VOCABULARY dar, das mit dem Befehlswort ASSEMBLER hereingerufen wird. Wenn man nun programmieren will, benutzt man das Wort CODE und einen Namen für den Code. Das ist der Normalfall. Nun möchte man andererseits gelegentlich die Dienstleistungen, die der Assembler oder ein anderes Vokabular erbringt, erweitern. Für diese Fälle ruft man herein: ASSEMBLER DEFINITIONS. Im nachstehenden Listing geschieht das in den ersten Zeilen.

Der Leser möge vergegenwärtigen, daß ihm alle Dienstleistungen des rudimentären FORTH weiterhin zur Verfügung stehen, auch wenn er sich in einem anderen Vokabular bewegt. Oder anders: Wird ein Befehlswort im eingeschalteten Vokabular nicht gefunden, so geht die Suche in FORTH weiter. In den nachfolgenden Beispielen wird also ein gemischter Gebrauch von FORTH und Assemblerworten gemacht.

Das erste neue Wort ist bereits ein kleines Makro: NJ erzeugt den Befehl NEXT JMP, . Die vier folgenden Worte dienen der Additions- und Subtraktionsvorbereitung.

Es folgen in 4er-Gruppen typische Sequenzen in den Adressierungsarten. So erzeugt z.B. (+S) den ADC- und STA-Befehl in absoluter Adressierung. Im weiteren Verlauf sehen wir, daß Makros wiederum andere Makros benutzen, z.B. bei Wort (L+S) wird (+S) benutzt.

Im weiteren Verlauf des Listings werden auch typische Enden einer Schleife gebildet, z.B erzeugt (DEX0) die Befehlskette DEX und BNE. Voraussetzung für die Benutzung solcher Makros ist zunächst einmal, daß die Branch-Befehle implementiert werden, die ja im strukturierten Assembler des FORTH fehlen. Und dann gehören auch MARK für des Setzen eines Rücksprung-Labels und BACK für die Ablage der Verzweigungsweite in den Branchbefehl dazu. Mit weiteren Befehlsworten kann man natürlich auch Vorwärtssprünge generieren..

Am Ende der Liste finden sich dann einige Beispiele für volle Schleifenstrukturen, die durch Makros erzeugt werden, um zu zeigen, was sich aus den Grundmakros machen läßt.

An die Listung der Implementierung schließ sich das Protokoll einer Terminalsitzung an. Der Leser beachte, daß die Parameter in umgekehrt polnischer Reihenfolge an die Makros übergeben werden.

```
( MACROS AS OF 7.7.82)


FORGET TASK : TASK ;
HEX
ASSEMBLER DEFINITIONS
: NJ NEXT JMP, ;

: ARC ROT CLC, ;
: ARS ROT SEC, ;
: AOC OVER CLC, ;
: AOS OVER SEC, ;

( PRIMITIVE MACROS)

: (+S) ADC, STA, ;
: (+SX) ,X ADC, ,X STA, ;
: (+SY) ,Y ADC, ,Y STA, ;
: (+SIY) )Y ADC, )Y STA, ;

: (L+S) LDA, (+S) ;
: (L+SX) ,X LDA, (+SX) ;
: (L+SY) ,Y LDA, (+SY) ;
: (L+SIY) )Y LDA, (+SIY) ;

: (-S) SBC, STA, ;
: (-SX) ,X SBC, ,X STA, ;
: (-SY) ,X SBC, ,Y STA, ;
: (-SIY) )Y SBC, )Y STA, ;

: (L-S) LDA, (-S) ;
: (L-SX) ,X LDA, (-SX) ;
: (L-SY) ,Y LDA, (-SY) ;
: (L-SIY) )Y LDA, (-SIY) ;


( BRANCHING)

: BNE, D0 C, ; ( STORE OPCODE AT CURRENT LOCATION)
```

```
: BPL, 10 C, ;
: BMI, 30 C, ;
: BVC, 50 C, ;
: BVS, 70 C, ;
: BCC, 90 C, ;
: BCS, B0 C, ;

( CONTROL)

: BACK MARK - 1- C, ; ( BRANCH TO LAST HERE)
( SET A MARK TO BRANCH BACK)
: MARK HERE ROT ROT OVER ;

( TYPICAL LOOP-ENDINGS)

: (DEX0) DEX, BNE, BACK ; ( BRANCH BACK ON >0)
: (DEX-) DEX, BPL, BACK ; ( BRANCH BACK ON POSITIVE)

: (DEY0) DEY, BNE, BACK ;
: (DEY-) DEY, BPL, BACK ;

: (INX) INX, # CPX, BNE, BACK ; ( COUNT UP UNTIL EQUAL)
: (INY) INY, # CPY, BNE, BACK ;

( MACROS WORKING ON WORDS - 2 BYTES)

: INC2 ( ADDR -- ) ( INCREMENT ANY WORD BY 1)
DUP INC, BEQ, 2 C, 1+ INC, ;

: ADA ( ADD ACCU CONTENTS TO ANY WORD)
DUP DUP CLC, (+S)
1+ DUP 0 # (L+S) ;
: SUBA ( SUBTRACT ACCU CONTENTS FROM ANY WORD)
DUP DUP ( DUP ADDRESS)
SEC, FF # EOR, (+S) ( LOW BYTE ADD COMPLEMENT)
1+ DUP FF # (L+S) ;


( MACROS WORKING ON FIELDS OF DATA)
( WHOLE LOOPING CONSTRUCTS)

: ADD-LOOP-Y0 ( Y DRECREMENTED TO 0)
( COUNT ADDR1 ADDR2 -- CONTENTS OF ADDR2 ADDED INTO ADDR1)
ARC
# LDY,    ( FROM COUNT)
MARK
(L+SY)    ( BODY OF LOOP)
(DEY0)
;

: ADD-LOOP-X0 ( X DECREMENTED TO 0)
( AS IN ADD-LOOP-Y-)
ARC
# LDX,
MARK
(L+SX)
(DEX0)
;
FORTH DEFINITIONS
FINIS
```

Protokoll einer Terminalsitzung
Eingabe von 3 Zeilen Assembler-Quelltext

```
ON ASSEMBLER HERE . 901 OK
( INCREMENT A WORD BY 1) OK
33 INC2 OK
( INCREMENT IN ABSOLUTE ADDRESS) OK
2345 INC2 OK
( BUILD A LOOP, COUNTING X DOWN) OK
3 3456 FADE ADD-LOOP-XO OK
```

Kontrolle des erzeugten Codes

```
<K>*=901
/16
 0901 E6 INC 33
 0903 F0 BEQ 0907
 0905 E6 INC 34
 0907 EE INC 2345
 090A F0 BEQ 090E
 090C EE INC 2346
 090F 18 CLC
 0910 A2 LDX #03
 0912 BD LDA 3456,X
 0915 7D ADC FADE,X
 0918 9D STA 3456,X
 091B CA DEX
 091C D0 BNE 0912
```

R.L.#

FORTH (5)

Im 4. Teil dieser Serie wurden der innere Sprachaufbau, das Vokabular, neue Vokabulare und die sog. 'defining words' behandelt, die neue Datentypen nach Wunsch des Benutzers schaffen. In den laufenden Teil gehörten damit weitere Hinweise auf den Compiler, seine Sekurität (Überwachung erlaubter Anweisungsfolgen) sowie der Hinweis auf die hierfür benutzten Variablen. Da es jedoch nicht darum geht, hier einen Compiler aufzubauen, sondern in erster Linie darum, den Leser möglichst bald mit den am meisten benötigten Befehlsworten vertraut zu machen, ist die Abfolge etwas umgestellt: Es werden zunächst die Ein- und Ausgabe zusammenfassend dargestellt (siehe auch Abschnitt 3.5 in Heft 25). Danach Worte zur Ausgabeformatierung. Einen besonderen Abschnitt wird die Massenspeicherung einnehmen und erst danach folgen weitere Hinweise zum FORTH System.

14. Eingabebefehle

HANG	Dieser Befehl bedeutet ein 'WAIT'. Der Programmablauf ist unterbrochen, bis irgendeine Taste gedrückt wird. Kann zum Anhalten bei einer Störung benutzt werden oder um zu definierter Zeit zu starten.
KEY (—ch)	Wartet auf einen Tastendruck und hinterläßt den ASCII-Wert des Zeichens auf dem Datenstack (Einzelzeicheneingabe).
?IN	Bei AIM die Ausführung des Unterprogrammes WHEREI, um vom Benutzer die aktive Eingabeeinheit abzufragen.
GET (—ch)	Holt ein Zeichen von der mit ?IN bestimmten Eingabe und echot es auf die aktive Ausgabe.
READ (adr n —)	Liest n Zeichen von der aktiven Eingabe ab Speicheradresse adr ein, z.B. PAD 4 READ READ akzeptiert auch z.B. ein Carriage Return und legt es ab.
EXPECT (adr n ——)	erwartet n Zeichen von der Tastatur. Ein vorzeitiges CR bricht die Eingabe ab, wird nicht gespeichert, stattdessen eine Null als Stringbegrenzer.
CHAIN (n Name)	Bei AIM die Verbindung (linking) von Eingabefiles auf Tonband: 0 CHAIN Eingabe vom gleichen Recorder

1 CHAIN von Tape Device 1
2 CHAIN von Tape Device 2

?TERMINAL (—flag)	prüft, ob am Terminal/Tastatur eine Taste gedrückt ist, hinterlegt das Flag=1 für TRUE oder 0 für FALSE und geht dann weiter.
?TTY (—flag)	Prüft den KB/TTY-Umschalter des AIM 65 und hinterläßt Flag.
BAUD (n —)	Setzt die Baudrate für serielle Datenübertragung des AIM 65 auf n
SOURCE	Anweisung, daß ein Programm compiliert werden soll, aus dem Memory oder aus externen Medien des AIM
FINIS	Beendet den zu compilierenden Text, schaltet in den normalen Eingabemodus zurück

15. Ausgabebefehle

. (n —)	(dot). Der Punkt druckt eine einfach genaue Zahl (16 Bit) vom Datenstack auf die aktive Ausgabe Wenn das höchstwertige Bit gesetzt ist, wird die Zahl als negative Integerzahl ausgegeben. Wo das stört, z.B. bei hexadezimalen Adressen, schreibt man 0 D. Damit Ausgabe einer positiven doppelt genauen Zahl
D. (d —)	Ausgabe einer doppelt genauen Zahl (32 Bit) als Ziffernfolge. Eine mögliche Ausgabeformatierung wird weiter unten beprochen.
?OUT	Beim AIM der Ansprung der Routine WHEREO, um die aktive Ausgabe abzufragen und zu setzen
BL (—ch)	Bringt eine hex 20 (SPACE) auf den Datenstack, zur Ausgabe z.B. mit:
EMIT (ch —)	Gibt ein ASCII-Zeichen vom Datenstack auf die aktive Ausgabe
CR	Gibt ein CR/LF auf die aktive Ausgabe ab
SPACE	Dito einen Zwischenraum
SPACES (n —)	Gibt n Spaces an die aktive Ausgabe
TYPE (adr n —)	Überträgt ab Adresse adr einen String von n Zeichen auf die aktive Ausgabe.
COUNT (adr —adr+1 n)	Wird als Vorbereitung von TYPE angewandt, wenn die Länge eines Strings zunächst nicht bekannt ist aber als erstes Byte vor dem String steht.
PUT (ch —)	Ausgabe eines Zeichens an die mit ?OUT bestimmte Ausgabe
WRITE (adr n —)	Gibt n Zeichen ab Adresse adr auf die mit ?OUT bestimmte Ausgabe
CLOSE	AIM-Wort, das die Tape-Files schließt und die E/A wirder auf die Standardeinheiten zurücksetzt.
."	(dot-quote), Ausgabe eines String-Literals an die aktive Ausgabe. Hinter dem ersten Anführungszeichen muß mindestens 1 Space folgen. Der String muß auch nach rechts mit Anführungsstrichen abgeschlossen werden, z.B " AIM 65"
–TRAILING (adr n1 — adr n2)	Abschneiden rechtsbündiger Spaces von einem String. Wird als Vorbereitung zu TYPE benutzt und verändert ggfs. den COUNT von n1 zu n2

Wenn wir einmal von der noch darzustellenden Massenspeicherung absehen, bietet FORTH bereits mit diesen Befehlen für die Ein- und Ausgabe sowohl Einzelzeichenverarbeitung, als auch die Behandlung von Textketten, und zwar an Plätzen, die der Anwender bestimmen kann (Datenstack oder Memory allgemein). Das hat den Vorteil, daß man mit definierten Input- und Output-Buffern arbeiten kann.

(WIRD FORTGESETZT) R.L. #

St.Dir. Peter Rix, 2350 Neumünster

FORTH mit Fließkommaarithmetik

und mathematischen Funktionen

Das standardisierte FIG-FORTH verfügt nur über 16 Bit- bzw. 32 Bit-Integerarithmetik. Bei der Flexibilität und Offenheit der Sprache ist der Gedanke einer Erweiterung um Fließkomma-Verarbeitungsmöglichkeiten aber naheliegend. Als geeignetes Fließkommapaket bietet sich der mathematische Teil des Microsoft-Basicinterpreters geradezu an. Die Microsoft-Routinen enthalten bei einem Kernumfang von nur 2,5 KByte ein vollständiges Funktionenpaket, haben mit 40 Bit-Fließkommazahlen und neun signifikanten Dezimalstellen eine brauchbare Rechengenauigkeit, sind in der Ausführung schnell und haben sich vieltausendfach langjährig bewährt.

Problem einer FORTH-Erweiterung ist die Einbettung des Fließkommapaketes in die Sprache mit systemgerechter Datenorganisation und mit geeigneten Befehlsworten. Einige bekanntgewordene Fließkommapakete, so u.a. von Rockwell, haben in dieser Hinsicht Mängel. Sie arbeiten entweder gar nicht stapelorientiert und überlassen alle Operandenbewegungen der Phantasie bzw. der Verzweiflung des Benutzers oder sie legen Integer- und Fließkommazahlen gemischt auf dmselben Stapel ab, was die Übersichtlichkeit der Programmierung und die Arbeitsgeschwindigkeit erheblich beeinträchtigt.

Der Verfasser hat, um in der täglichen Arbeit nicht ständig auf BASIC zurückgreifen zu müssen, und weil die vorliegenden Lösungen ihn nicht befriedigten, auf der Microsoft-Grundlage eine eigene Fließkomma-Erweiterung geschaffen. Für den AIM 65 ist das gesamte Paket in einem 4 KByte-EPROM auf dem Steckplatz Z24 (Adresse hex D000) untergebracht und arbeitet mit dem AIM-FORTH zusammen. Es kann jedoch auch von jedem Assembler-Programm aus benutzt werden, weil die Einsprungpunkte und die zu übergebenden Argumente beschrieben sind.

Die nachfolgende Beschreibung dieses Fließkomma-FORTH soll einerseits zur Diskussion über den angemessenen Weg zu einer wünschenswerten Standardisierung anregen, andererseits sollen die kommerziellen Softwareanbieter darauf aufmerksam gemacht werden, daß Leistung und Anwenderfreundlichkeit ihrer Programme den berechtigten Ansprüchen der Benutzer trotz hoher Preise noch keineswegs immer gerecht werden.

Grundeigenschaften

Fließkomma-FORTH hat einen eigenen 6 Register tiefen Stapel. Oberstes Stapelregister und Fließkomma-Akkumulator sind identisch. Im Stapel werden nicht die Daten bewegt, sondern es wird nur der Zeiger auf sie in einem Byte verstellt. Dieser Stapel ist optimal schnell, außerdem entfallen alle Rangierereien zwischen Daten, Adressen und Flags, weil der Parameterstapel ('Datenstack') unbeeinträchtigt parallel betrieben wird.

Das beim Parameterstapel sinnvolle Prinzip, bei jedem Zugriff einen Wert zu verbrauchen, ist bei einem reinen Arithmetikstapel begrenzter Tiefe nicht zweckmäßig. Gewöhnlich muß mit angesprochenen Werten weitergerechnet werden, oder Müllablagen auf dem Stapel stören nachfolgende Rechnungen nicht. Der Fließkommastapel arbeitet deshalb wie der Automatikstapel der HP-Taschenrechner.

> Stapelwerte werden nur durch arithmetische Operationen - nicht durch Abspeichern oder durch Anzeigen - verbraucht Bei Ablage eines 7. Stapelwertes wird der 1. Stapelwert aus dem Stapel herausgeschoben und geht verloren. Bei Verbrauch von Werten wird der Stapel vom unteren Ende her mit Nullen aufgefüllt.

Terme bis zu 4 schwebenden Operationen (Klammerebenen) können ohne Umstellung oder Mehrfachabrufe auf dem Stapel berechnet werden. Innerhalb der Berechnung sind noch Fallunterscheidungen möglich, weil Vergleiche den Stapelinhalt nicht ändern. Das reicht für alle praktisch vorkommenden Fälle und geht weit über die Komfortebne programmierbarer Taschenrechner hinaus.

Die neuen FORTH-Worte sind bis auf wenige Ausnahmen als schnell ausführende Code-Definitionen implementiert. Fließkommastrings werden während der Compilierung in das binäre Fließkommaformat übersetzt und bei der Programmausführung nur noch abgerufen. Das Fließkomma-FORTH wird dadurch sehr schnell.

Vergleiche mit rechenintensiven BASIC-Programmen haben ergeben, daß die Ausführungszeiten ca. 30% kürzer sind. Oder anders ausgedrückt: Fließkomma-FORTH ist etwa 40% schneller als das mit den gleichen Mathematikroutinen arbeitende BASIC.

Bei der Ausgabeformatierung hat der Anwender jede Freiheit in der Wahl der Feldbreite, Nachkommastellenzahl und Darstellungsform als Fließkommazahl (F-Format) oder Exponentialzahl (E-Format). Es ist selbstverständlich, daß auch eine korrekte Ausgaberundung durchgeführt wird.

Bei den auftretenden Fehlerarten:

DIVIDE BY ZERO!
OVERFLOW!
UNDEFINED ARGUMENT!
NUMBER TOO LARGE!

bleiben bis auf den obersten fehlerauslösenden Stapeleintrag alle Fließkommawerte erhalten. Der Anwender kann zu einer eigenen Fehlerbehandlung verzweigen.

(Wird fortgesetzt)

FORTH-Splitter (1)

Die verbotene Division durch 0 wird vom AIM-FORTH und anderen nicht beanstandet, Man probiere z.B.:

```
20 0 / .        Man erhält -1
```

Bei Arbeiten mit FORTH möchte man häufig in den Text-Editor des AIM zurück, um Berichtigungen oder Ergänzungen in den Quelltext einzugeben. Die übliche Tastenfolge ist ESC (aus FORTH) und 'T' (Top-Zeile des Editors). Folgende kompilierten Wörter kürzen den Weg ab:

```
HEX  F6CF VARIABLE  EDI  ( Wiedereintrittsadresse des Editors)
: T EDI EXECUTE ;
```

Wenn man jetzt in FORTH den Befehl 'T' gibt, ist man sofort im Editor-Modus.

EXECUTE ist definiert als: " Führe das FORTH-Wort aus, dessen Codefeldadresse sich auf dem Datenstack befindet".Die Variable EDI simuliert mit ihrem Inhalt nun, daß F6CF ein FORTH-Wort sei. Der Aufruf von 'T' führt zur Ausführung des Codes ab F6CF. In gleicher Weise kann man auch andere Routinen zur Ausführung bringen.

Maschinensprachliche Routinen des Monitorprogrammes und auch Routinen des Anwenders enden meist mit RTS. Wenn man solche mit EXECUTE aufruft, erhält man keine systemgerechte Rückkehr zu FORTH. Der richtige Abschluß wäre mit JMP NEXT (Code 4C 5A B0), so wie man es bei mit CODE erzeugten FORTH-Wörtern macht. Im Falle des Editors liegen die Dinge anders. Er ist nicht mit einem RTS oder mit einem JMP NEXT abgeschlossen, sondern endet immer mit der Decodierung der Funktionstasten, bis ein ESCAPE oder QUIT erfolgt. Insofern bleibt man mit dem neuen Befehl 'T' solange im Editor-Modus, bis der Editor gezielt verlassen wird. Es wird sicher möglich sein, mit der in diesem Heft beschriebenen Erweiterung der Editor-Befehlsebene auch eine Funktionstaste mit JMP NEXT zu belegen, so daß man direkt zu FORTH zurück könnte.

R.L.#

Jürgen Rüd, 7800 Freiburg

NEC 8023 B-C

Ein universeller Drucker

Die meisten Besitzer eines Heim- oder Tischcomputers werden sich über kurz oder lang mit dem Gedanken beschäftigen, sich einen Drucker anzuschaffen. Auf dem Markt gibt es ein großes Angebot von Typenrad- oder Kugelkopfdruckern mit einem schönen korrespondenzfähigen Schriftbild. Daneben sind die Matrixdrucker im Schriftbild meist nicht so überzeugend, doch gibt es hier Ausnahmen. Thermo- und Metallpapierdrucker, die zur Klasse der Matrixdrucker gehören, sind zwar billig, haben meistens aber nur 40 oder 80 Zeichen pro Zeile, kein überzeugendes Schriftbild und benötigen außerdem noch ein spezielles Papier, das recht teuer ist.

Der hier besprochene NEC PC-8023B-C hebt sich nach Ansicht des Autors sehr positiv von anderen Druckern ab. Mit seinem relativ guten Schriftbild eignet er sich ebenso für die Textverarbeitung wie mit seiner Schnelligkeit auch für den Ausdruck großer Datenmengen.

Der Papiertransport wird entweder von einer Friktionsführung (Gummiwalze für Einzelblatteinzug) oder durch eine Traktorführung bewältigt (für Endlospapier mit Randlochung). Die Friktionsführung positioniert auch nach vielen Blättern noch sauber. Ein großes Plus ist ferner die Abreißkante, die ein stets sauberes Abtrennen gewährleistet.

Über DIL-Schalter können 5 internationale Zeichensätze ausgewählt werden (japanisch, amerikanisch, englisch, deutsch und niederländisch). Der Drucker beherrscht 96 ASCII-Zeichen, 160 JIS-Zeichen, 64 Grafikzeichen und 14 europäische Zeichen. Die alphanumerischen Zeichen werden in einer Matrix von 9x7, die grafischen in einer von 8x8 dargestellt und haben ein sauberes Schriftbild. Damit eignet der Drucker hervorragend zur Textverarbeitung. Er druckt 100 Zeichen pro Sekunde, arbeitet bidirektional, mit abschaltbarer Druckwegoptimierung.

Ein großer Vorteil ist ferner die hochauflösende Grafik des Druckers. Jede der 8 Nadeln kann einzeln angesteuert werden. Damit hat man die Möglichkeit, beliebige graphische Sonderzeichen im Text zu definieren und kann z.B. mathematische Funktionen grafisch darstellen. Dieses Plotten wird dadurch erleichtert, daß das Papier per Programm in Schritten von 1/144 Zoll bis zu 99/144 Zoll vorwärts und rückwärts transportiert werden kann. Damit wird der Drucker zu einem fast vollwertigen Ersatz eines Plotters.

Per Programm lassen sich auch verschiedene Schriftarten innerhalb einer Zeile wählen. Dazu nachstehend einige Schriftproben.

Für die Beschriftung von Formularen ist der Drucker mit einem horizontalen und einem vertikalen Tabulator ausgerüstet. Das ermöglicht ein PRINT USING, das viele Rechner nicht haben. - Übertragene Daten können gelöscht werden, soweit sie noch nicht gedruckt sind. Weitere per Programm steuerbare Funktionen:

Formfeed - Cancel (Löschen bereits übertragener Daten) - Backspace - hor. Tabulator - programmierbarer vert. Tabulator - wählbarer Zeichenabstand bei Proportionalschrift - verstärkter Druck - einstellbarer Zeilenvorschub von 1/144 bis 99/144 Zoll - Unterstreichung - einstellbarer linker Rand.

Der NEC ermöglicht es, bis zu vier Drucker auf einer Device-Adresse zu benutzen. Der gewünschte Drucker wird per Programm angesprochen. Die Geräteadresse des Druckers läßt sich über DIL-Schalter einstellen.

Der 1K große Eingabepuffer des Druckers erlaubt ein überlappendes Arbeiten mit dem Rechner.

Auch die folgenden Funktionen können über DIL-Schalter eingestellt werden: Zeilen pro Seite (66 oder 72) - Gültigkeit der Geräteadressen - Druck wenn Zeile voll mit oder ohne Linefeed - Auswahl des druckauslösenden Codes - Wagenrücklauf mit oder ohne Zeilenvorschub - Darstellung der Null mit oder ohne Querstrich - Pica oder Propotionalschrift - 7- oder 8-Bit Interface -Druckerzustand beim Einschalten (Select oder Deselect). - An der Frontseite finden sich Taster für Select, Linefeed und Formfeed sowie Leuchtdioden für den Betriebszustand. - Beim Erreichen des Papierendes hält der Drucker an, signalisiert es auf eine LED und auf das Interface.

Der mechanische Aufbau macht einen stabilen Eindruck. Der Drucker arbeitet relativ leise, ein Pfeifen, das durch den Transformator verursacht wird, fällt auf die Dauer nicht weiter auf.

Als Schnittstelle hat der NEC ein Centronix-Interface. Man kann ihn auch an einen anderen parallelen Port anschließen oder an einen seriellen (3 Output- und 1 Inputleitung).

Zum Drucker gehört ein englisches und ein deutsches Handbuch. Die Steuerbefehle sind ausführlich und leicht verständlich erklärt. Mit den beschriebenen Anschlüssen und Zeitdiagrammen kann man ihn an jeden Computer anschließen, z.B. an den Userport des CBM oder beim ATARI an den Gameport.

Alles in allem bietet dieser Drucker bei einem Preis von ca. DM 1.900,- ein recht günstiges Preis-Leistungsverhältnis und ist damit einer näheren Betrachtung wert.

Schmalschrift	mit bis zu 136 Zeichen pro Zeile
breite Schmallschrift	mit bis zu 68 Zeichen pro Zeile
Pica-Schrift	mit bis zu 80 Zeichen pro Zeile
Pica-Breitschrift	mit bis zu 40 Zeichen Pro Zeile
Elite-Schrift	mit bis zu 96 Zeichen pro Zeile
Elite-Breitschrift	mit bis zu 48 Zeichen pro Zeile
Proportionalschrift	mit variabler Zeichenzahl pro Zeile (80 (m) bis 150 (i) Zeichen pro Zeile)
Proportional-Breitschrift	mit variabler Zeichenzahl pro Zeile (40 (m) bis 75 (i) Zeichen pro Zeile)

SCHRIFTPROBEN DES NEC

Aus der Branche - Produkte

Rockwell International bietet jetzt eine neue Bausteinfamilie an, die Intelligent Peripheral Controller der Typen R6541Q, R6500/41, R6500/42 und R6500/43. Es sind 1-Chip-Microprozessoren, die an den Daten- und Controllbus eines Host-Prozessors angeschlossen werden, um von diesem Steuerinformation und Daten zu übernehmen oder dorthin abzugeben und die eine eigene CPU mit erweitertem Befehlssatz und 23 bis 47 I/O-Pins haben. Dazu kommt ein 16-Bit Timer/Zähler und je nach Typ findet man ein maskenprogrammiertes ROM und ein kleines RAM oder auch nichts davon (R6500/43 und R6541Q). Die letztgenannten Versionen haben einen 4K Adreßbus und einen Datenbus, der auf die Pins herausgeführt ist, und die damit als Emulatoren für die Familie dienen können. Das Konzept dieser Bausteine muß als geschickt gewählt betrachtet werden. Sie sind kompatibel mit den Bussen von 6500, 6800, 8080 und Z80, weil das Interface zum Host asynchron arbeitet (unabhängige Clocks). Wesentliche Funktionen des IPCs werden durch Kontrollregister gesteuert, die der Host beschickt oder abfragt. Sie haben sieben Interruptarten. Die IPCs entsprechen damit der Entwicklung, die Peripherie der Computer immer selbständiger zu machen zwecks Entlastung der zentralen Einheit. Nach ihrer Architektur erlauben sie den Aufbau sehr leistungsfähiger Mehrprozessorsysteme. - Rockwell-Documents Nr. 29000D95 und 29651N38.

Models	R6541Q µP	R6500/41 IPC	R6500/42 IPC	R6500/43 IPC
• ROM (x8)	—	1536	1536	256
• RAM (x8)	64	64	64	64
• I/O Lines	23	23	47	23
• 16-Bit Counters	1x16	1x16	1x16	1x16
• Host/Slave Bus	65/80	65/80	65/80	65/80
• Expansion Bus	8K	4K	4K	8K/4K*
• Interrupts				
— External	5	4	4	5
— Internal	1	1	1	1
— Host	2	2	2	2
• Package	64-pin QUIP	40-pin DIP	64-pin QUIP	64-pin QUIP

*ROM OPTION

Ein 6805 Cross-Assembler für PDP 11-Systeme. Bei der Auswahl eines Mikroprozessors für kleine Steuerungs- und Meßsysteme stellt sich dem Entwickler die Frage, wie Software für das System entwickelt werden kann. Wenn nur kleine Stückzahlen erwartet werden, wird der Entscheidungs-

prozeß von den vorhandenen Möglichkeiten beeinflußt. Oft wird ein weniger geeigneter Prozessor ausgewählt, weil für ihn das Entwicklungssystem vorhanden ist.

Die Verwendung der EPROM-Typen der Serie 6805 bietet sich gerade im Bereich kleiner Stückzahlen an. Durch die Bootstrap Software zum Programmieren des EPROMs kann die Hardware des Entwicklungssystems auf ein Minimum beschränkt bleiben. Lediglich für die Softwareentwicklung ist eine möglichst komfortable Lösung zu finden. In Forschungseinrichtungen, Instituten und Ingenieurbüros ist oft eine PDP 11 vorhanden. Für sie wurde ein Cross-Assembler für den 6805 entwickelt. Quellprogramme können mit den sehr komfortablen PDP 11-Editoren geschrieben werden. Der Cross Assembler erzeugt aus ihnen ein Objectfile im Standard Object Format. Dieses Format ist weltweit Standard für Cross-Assembler. Es entspricht dem Paper Tape Format des AIM 65.

Mit den Makro- und If-Möglichkeiten kann die Entwicklung auch größerer Programme sehr bequem erfolgen. Die Adressierungsart der Befehle wurde als Teil der Mnemonics gewählt. So steht z.B. INCD für 'increment direct' (Adressierungsart 'direkt). So ist für jeden der 207 möglichen Opcodes ein eigenes Mnemonic definiert. Außerdem sind 17 Assembleranweisungen (Pseudo-Ops) vorhanden. Der Cross-Assembler, lauffähig unter RSX 11 M, ist beim Ingenieurbüro A. Spiegelberg zu beziehen (Dickhardstr. 46, 1000 Berlin 41, T. 030 - 851 7869). Zum Lieferumfang gehört die vollständige Benutzungsanweisung sowie ein FORTRAN-Programmvorschlag zum Brennen von EPROMs mit einer Digitalausgabe. Es können auch Versionen für andere Betriebssysteme und Prozessoren (6800, 6809, 6502) geliefert werden.

Das IHK-Bildungszentrum für München und Oberbayern führt ab Januar 1983 wieder verschiedene Seminare zur Mikroelektronik/EDV durch. Info: IHK-Bildungszentrum, von-Adrian-Str. 5, 8152 Feldkirchen-Westerham, T. 08063- 19 41.

Für ihr Mikrocomputersystem SK 65 in 19"-Technik legte die Firma System Kontakt einen umfangreichen und sauber gegliederten Datenkatalog vor. Das SK 65 kann sowohl als Subsystem oder als autarkes System eingesetzt werden (CPU 6502). Es hat gepufferte Busse, und es steht ein umfangreiches Kartenspektrum in industrieller Qualität zur Verfügung: A/D, D/A, Video, Bustreiber, I/O, 32 k PROM, PROM-Progr., 16 k RAM, FDC, Doppelfloppy, Relaiskarte, Optokoppler, CPU, IEC und ACIA sowie Stromversorgungen. Info: System Kontakt Ges. f. elektron. Bauelemente mbH, Siemensstraße 5, 7107 Bad Friedrichshall, T. 07131 - 7 10 94 (Herr Strobl).

BASIC-Compiler für CBM 8032. Heft 27 enthielt bereits den Hinweis, daß die Ges. f. math. Beratung & Software Dr. A. Quint mbH. in 6200 Wiesbaden, Kaiser-Friedrich-Ring 55, einen entsprechenden Compiler anbietet. Inzwischen liegt dazu ein sauber aufgemachtes ca. 150-seitiges Handbuch hier vor. Mit wenigen Einschränkungen können alle Programme, die unter Interpreter laufen, auch kompiliert werden. Es werden dokumentierte Source-Files für den Assembler erzeugt, die noch editiert werden können. Das Handbuch enthält für jeden Befehl die Eingangs- und Ausgangsvoraussetzungen sowie einen ausführlichen allgemeinen Teil zur Hantierung und darf damit als gelungen bezeichnet werden.

68000-Computer der Firma FORCE Computers GmbH. Wie aus verschiedener Werbung bekannt sein dürfte, bietet die Firma seit länger Zeit ein preiswertes 68000 Profi Kit 1 an, inzwischen auch ein Profi Kit 2 mit 8 MHz CPU. Beiden ist gemeinsam, daß man sie als abgeschlossene Kartensysteme betreiben kann, eine Erweiterung ist aber nur schwer möglich, weil die Bussignale nicht an den Kartenrand geleitet sind. Auf der ELECTRONICA war nun zu erfahren, daß ab etwa Januar 1983 ein drittes Board geliefert wird, das VME-Bus fähig ist, die Signale sind also auf entsprechende Federleisten herausgeführt. Damit steht dem Anwender der volle Adreßraum der 8 MHz CPU zur Verfügung. Je nach benutzten Bausteinen können on board jetzt 128-512 K Bytes benutzt werden und 16-64 K System-ROM und die gleiche Menge Anwender-ROM (EPROM). Es sind drei serielle Schnittstellen vorhanden und ein einfacher Debug-Monitor, line by line Assembler sowie die Möglichkeit des downloadings. Als paralleler Interfacebaustein wird bereits der neue PI/T 68230 eingesetzt mit 24 Bit Timer und Ports, die 8 oder 16 Bit breit benutzt werden können. Als Software-Optionen sind ein Editor, Assembler und BASIC vorgesehen. Für die Karte wurde ein Preis von etwa DM 3400,- genannt (+MWSt), so daß man auch hier von einem interessanten Angebot und auch recht zukunftssicheren Konzept sprechen kann (die noch 1983 und 1984 erwarteten CPU-Versionen wird man nicht einfach auf dieses board stecken können). Der Erweiterung dient bereits eine 512K DRAM-Karte 1 zum Anschluß an den VME-Bus. Sie hat Byte Parity Check, on board refresh und kann im Byte- oder Wortformat angesprochen werden. Ca. 3950,- DM + MWSt. Anschrift: FORCE Computers GmbH, Freischützstr. 92, 8000 München 81, T. 089 - 95 10 41–44.

Nichtflüchtige RAM-Module mit 64K Byte werden jetzt von der TecSys GmbH in 8 München 81, Arabellastr. 13 T. 089-91 39 34 angeboten, Hersteller ist Greeenwich Instruments. Es handelt sich um ein mit Lithiumbatterie gepuffertes CMOS-RAM-Modul, das vorteilhaft dort eingesetzt werden kann, wo es auf Datensicherheit ankommt. Die Signale sind auf eine 64-polige Buchsenleite herausgeführt.

Für den 68000 VME-Bus liefert die Fa. ELTEC jetzt eine 384K Byte RAM-Karte mit Error Detection und Correction. Die Karte kann in einem 256K Block und in einem 128 K Block im gesamten Adreßraum verschoben werden. Info: ELTEC Elektronik, Postfach 65, 6500 Mainz 42.

Editorial

Beim Abschluß dieses Heftes kommt die Nachricht, daß die Zahl der Arbeitslosen 2,038 Millionen erreicht hat. Die gleichzeitige Senkung des Diskontsatzes auf 5% ist zwar ein Trost, der die von den Wirtschaftsforschungsinstituten vorhergesagte Talsohle der Entwicklung etwas höher legt. Die jetzigen Schwierigkeiten sind nur zu einem Teil auf weltweite konjunkturelle Einflüsse und auf den hohen amerikanischen Zins zurückzuführen, sie beruhen auch auf Umstrukturierungen, die in der öffentlichen Diskussion zu wenig erwähnt wurden. So hat es bereits etwa 1977 ein Gutachten des angesehenen schweizer PROGNOS-Institutes gegeben, das u.a. wegen der geburtenstarken Jahrgänge auf hohe Arbeitslosigkeit (ca. 1,5 Mio.) Anfang der 80er Jahre hinwies.

Ein zweiter wichtiger Strukturfaktor ist wohl folgender: Seit Jahrzehnten liefern die Industrienationen hochwertige Maschinen und Know How in alle Welt. Es kann in der Folge eigentlich nicht wundern, wenn im Gefolge hierzulande arbeitsintensive Industrien wie die Textilindustrie in Schwierigkeiten geraten. Der Beschäftigungseffekt, den eine vom Maschinenbau gelieferte Maschne auf den Binnenmarkt ausübt, kann in kurzer zeitlicher Folge negativ sein, wenn die mit ihr hergestellten Erzeugnisse billiger auf unseren Markt zurückdrängen. Und man muß wohl hinzufügen: Viele Länder berücksichtigen die sozialen und allgemein gesellschaftlichen Kosten nicht, die jetzt und in Zukunft im Zusammenhang mit der Schonung der Umwelt und der Beschäftigten entstehen, also notwendige Kanalisationssysteme, Klärwerke, Reinhaltung der Luft, Sicherheit am Arbeitsplatz usw..

Und ein dritter struktureller Faktor ist ganz sicher der Computer. Man spricht davon, daß in etwa einem Jahrzehnt noch etwa 2,5 Mio. Arbeitsplätze durch ihn entbehrlich werden. Der Herausgeber kann die Zahl nicht prüfen, darf aber die Ansicht mitteilen, daß bei solchen Größenordnungen im Gegenzug kaum eine annähernd gleiche Menge von Arbeitsplätzen in der Halbleiterindustrie und bei ihren Zulieferern geschaffen wird.

Wir laufen damit auf einen Punkt zu, wo Arbeit ein viel zu knappes Gut wird, weil sie teilweise in internationaler Arbeitsteilung in anderen Ländern geleistet wird und teilweise von Robotern in ihren immer zunehmenden Erscheinungsformen.

Das erste Roboter- oder Computerzeitalter (Begriff des science fiction) hat wohl doch schon eingeläutet, ohne daß wir uns noch dagegen wehren könnten. Es ist nur anders gekommen, als es erträumt wurde, als die Befreiung des Menschen von harter Arbeit. Es kommt in einer wirtschaftlich schwierigen Phase als Beraubung vieler Menschen von jeglicher Arbeit für die Gesellschaft. Und eben besonders sind die geburtenstarken Jahrgänge im Alter um 20 betroffen, die gern mit den Idealen der Jugend in die Zukunft gehen würden und nicht sehen können, ob eine qualifizierte Ausbildung überhaupt eine Chance der beruflichen Betätigung bieten wird. Gestern kam die Meldung, daß die beim Staat bisher für sicher gehaltenen Berufe inzwischen auch bangen müssen: Baden-Württemberg will einige tausend Lehrer entlassen. Sicher kein hoffnungsvolles Fanal für die etwa eine Million Studierenden. Auf der anderen Seite werden 'ältere' Mitarbeiter mit 55 und 58 in den Ruhestand geschickt, wie hier im hamburger Raum in der Mineralölindustrie, weil Energie gespart wird..

Wir stehen also nicht nur vor einem Tal im Verlaufe konjunktureller Bewegungen, sondern vor weiter wirkenden Strukturproblemen, die die Arbeitsmöglichkeit schlechthin betreffen und die gesellschaftspolitisch angefaßt werden müssen. Die Frage ist doch wohl, wie die arbeitslosen und nicht anspruchsberechtigten Berufsanfänger und die Mittfünfziger aufgefangen werden. Was sollen sie denn überhaupt tun, wenn der Sinn ihres Lebens nicht mehr in der Einbindung in Arbeit und die damit verbundenen sozialen Kontakte und die Anerkennung des Wirkens liegen kann?

Investitionsprogramme, wie die Verkabelung der Bundesrepublik, können sicher Arbeitsplätze schaffen, auch solche die die Summe des Energieverbrauches senken und damit unser Land für die Zukunft besser vorbereiten. Die Verkabelung mit Kupferdraht ist dabei sicher unzweckmäßig, weil veraltet. Die aufkommende Glasfasertechnik läßt viel mehr Kommunikation und Nebenwirkung auf Investitionen zu. Nicht, daß unbedingt 20 Fernsehprogramme zur Auswahl gestellt werden sollten. Aber: Viele Vollzüge im gewerblichen und auch im privaten Bereich sind unwirtschaftlich und energiefressend, z.B. das Einholen von Auskünften und Aufträgen durch persönliche Fahrt oder Reise zum Gesprächspartner. Da fahren viele Einzelpersonen im Auto zur Post, zur Bank, zum Reisebüro, zum Kunden oder zur defekten Maschine, die in vielen Fällen mit einem Handgriff repariert werden kann, wenn man sie als Fachmann gesehen hat. Eine preiswerte Telekommunikation, ein bidirektionales Bildtelefon, wäre mit der Glasfaser möglich und würde viele dieser zeit- und benzinfressenden Bewegungen von Einzelpersonen im Pkw überflüssig machen, die auch noch die Umwelt mit Geräusch und Abgasen sowie mit dem Unfallrisiko belasten.

Zur Verkabelung der Bundesrepublik mit Glasfasern gehört also zugleich ein (fast kostenloses) Gesetz, das die Telekommunikation und die dadurch erhofften Investitionen dadurch animiert, daß das Fernmeldemonopol der Post gelockert wird und nicht als eine zusätzliche Einnahmequelle für den Staat mißbraucht werden kann. Bei mäßigen Gebühren greifen schließlich die gewerbliche Wirtschaft und der Privatmann immer nach dem günstigen Angebot und beleben damit die Wirtschaft. Telekommunikation wird also dann zu einem Investitionsstoß, wenn sie preiswert ist und nicht durch die Bereitstellung des x-ten Fernsehprogrammes. Telekommunkation hat damit Faktoren für Umweltschonung und für das Energiesparen. Und das Element der persönlichen Kontaktes, der bisher mit den erwähnten persönlichen Bewegungen verbunden war, brauchte nicht zu kurz zu kommen, denn viele Kontakte am breitbandigen und farbigen Bildtelefon dürften gleichwertig sein.

Von der 'Verrohrung' der Industrieländer wird im Gegensatz zur 'Verkabelung' weniger gesprochen, obwohl auch hier eine mögliche Investitionspritze läge. Gemeint ist einerseits die Wärme-Kraftkupplung in Heizkraftwerken und andererseits der Aufbau lokaler Rohrpostsysteme. Beides würde Energie sparen und die Umwelt entlasten.

Die Wärme-Kraftkupplung benutzt die bei der Elektrizitätserzeugung entstehende Abwärme, um mit ihr über ein Rohrsystem Gebäude (oder Prozesse) zu beheizen. In Hamburg werden seit Jahrzehnten ganze Stadtteile fernbeheizt. Es ist klar, daß in dezentralen Heizkraftwerken mehr für die Entgiftung des Rauchgases ('saurer Regen') getan werden kann, als in vielen tausend Einzelfeuerstellen. Und die Tatsache, daß man in gewissen Gebäudeverdichtungen Fernwärme erfolgreich verkaufen kann und daß man sich als Abnehmer für sie entschließt, spricht für die Energie-Wirtschaftlichkeit.

Die zweite Art der angesprochenen Verrohrung betrifft innerörtliche Rohrpostsysteme In Hamburg werden seit langen Jahren Transporte zwischen den großen Postämtern mit einer großformatigen Rohrpost vollzogen, weil die Verkehrsverhältnisse in der Innenstandt keine andere Lösung mehr zuließen. Ein Mehr an solchen Systemen, auch in Zusammenhang mit Telekommunikation, würde Investitionen und Arbeitsplätze bedeuten und könnte eine Vielzahl individueller Bewegungen mit dem Fahrzeug energie- und umweltschonend entbehrlich machen. Eine kleine Maßnahme dürfte in einer ernsthaften Zeitschrift eigentlich kaum erwähnt werden müssen, sie steht aber im Zusammenhang mit lokalen Transportsystemen: Wenn die Einwurfschlitze an den Briefkästen größer wären brauchten viele tausend Gewerbetreibende nicht täglich zur Auflieferung bereits freigemachter Sendungen zum Postamt zu fahren.

Es ist hier nicht der Ort und die Absicht, Wirtschaftspolitik machen zu wollen. Man soll nur eben nicht verkennen, daß es keine schnell wirkenden Heilmittel gibt und daß Maßnahmen sicher auch Folgeeinflüsse haben. Im wesentlichen kommt es darauf an, daß klare Zeichen gesetzt werden, wo der Weg weitergeht.

Im gezeichneten Umfeld darf man davon ausgehen, daß die Computerbranche in Hard- und Software weiter ihren Stand haben wird, mit unterschiedlichen Akzenten natürlich. Der Wert guter Entwicklungshilfsmittel (utilities, Betriebssysteme) und von Anwenderprogrammen steigt relativ zu den Preisen der Geräte. - In diesem Sinne ist es begrüßenswert, daß für dieses Heft wieder so viele gute Gedanken zusammengetragen werden konnten. Anläßlich des bevorstehenden Jahreswechsels dankt der Herausgeber den vielen engagierten freien Autoren für ihre Mitarbeit und den Lesern für die beständige Verbindung und wünscht ihnen beruflich und persönlich für 1983 alles Gute!

R.L.#

Assembler unter FORTH

R65C02-Assembler

Einbindung des erweiterten Befehlssatzes der CMOS-CPU und der IPCs in den vorhandenen 6502-Assembler + hochkomfortable Leistungserweiterungen (s.u.). Erzeugt wahlweise EPROM-fähigen Code für beliebige, auch virtuelle Speicherräume oder FORTH-Worte mit eingebundenem Code. Lieferbar ab Mitte Januar 1983. D000-EPROM für AIM 65, mit Dokumentation: DM 150,-- + MWSt.

CROSS–09

Cross-Assembler für MC6809 (Motorola). Tonbandcassette (6,5K), Diskette (CBM 3040/4040) oder 2 EPROMs, deren Inhalt nach hex 200-1B00 kopiert werden muß. DM 380,-- + MWSt.

CROSS–05

Cross-Assembler für MC6805/68705xx (Motorola), auf Cassette, Diskette oder EPROM für D000: DM 320,-- + MWSt.

Alle Assembler sind komfortabel ausgelegt, benutzen die Mnemonics des Herstellers und laufen unter FORTH des AIM 65 (für CBM-FORTH von Lowinski oder PHS ggfs. auf Anfrage). Sie erzeugen EPROM-fähigen Code für beliebige Speicherräume und enthalten alle üblichen Kontrollstrukturen wie BEGIN, WHILE, REPEAT, IF, ELSE, THEN usw. aber auch alle Branches (6809: Longbranches). Es kann mit externen Symbolen und Labels im Programmablauf gearbeitet werden. Zusätzlich sind viele Assembler-Anweisungen implementiert: Byte- Word- und Stringablage, .OPTLIS, Reserve Memory Bytes und binäre Argumente.

Mathe- ROM für 6502

Implementierung von P. Rix (s. Textteil) für Sockel D000. Fließkommaarithmetik und höhere mathem. Funktionen (wie in Microsoft-BASIC) für AIM 65-FORTH(komfortabler eigener Fließkommastapel) und für jedes 6502-Assemblerprogramm (dokumentierte Einsprungspunkte und Argumente). EPROM lieferbar ab Jan. 1983. DM 110,- + MWSt.

Roland Löhr, Hansdorfer Str. 4, D-2070 Ahrensburg, T. 04102 - 55 816.

Kleinanzeigen

6805 CROSS ASSEMBLER

Für PDP 11 Systeme unter RSX 11 M, Makros und bedingte Assemblierung, DM 420,- + 13% MWSt + Speichermedium, Versionen für 6800, 6809, 6502 sowie RT 11 auf Anfrage.
Ingenieurbüro Spiegelberg, Dickhardtstr. 46, 1000 Berlin 41, Tel.: 030 - 851 78 69.

"STARTING FORTH" von Leo Brodie, die beste und umfassendste Einführung in FORTH für DM 52,80. Dr. J. Schrenk, Postfach 904 , 7500 Karlsruhe 41.

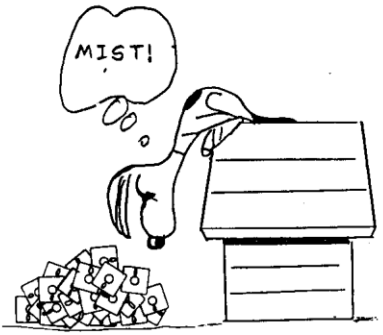
Wo bekommt man IEC-Bus-Kabel in 1,5 m Länge? KKB Bank, Postfach 10 10 48, Köln 1, Tel.: 0221 - 20 04-140 (Herr Schünemann).

CBM-2/3/4/8001 Speicherbelegung, 630 Adressen, 20 DM * Beschreibung von ROM-Routinen für reelle Arithm. Tape- & IEEE-Bus-I/O, Parameterübergaben an Maschinenprogramme etc. 25 DM * Zusammen 88 Seiten A4 für 35 DM * Alle ROM-Adressen als Label-Files auf Diskette 40 DM * Katalog kostenlos * I. M. Koch, Hessenring 79, 6367 Karben.

IBM-KK Schreibm./Drucker (Nixdorf-EDV), ü. Relais anzusteuern, o. Gehäuse, Preis VHS. Datamega-Microwaferset (gebr.) Kassetten = Scheckkartengröße, für AIM 65 geeignet. Preis VHS. Uher 5000 Tonband, voll steuerb. über Relais (24 V), Vorl./Rückl./Aufn./Wiederg. (AIM 65) Preis VHS. R. Pobloth, Fasanenweg 32, 2207 Kiebitzreihe T 04121-5530 (ab 18 h), Fa. 04121-50 001.

AIM/65-Screen-Editor, voll cursor- und bildschirmorientiert, Groß-/Kleinschreibung, Autorepeat, Scroll-up/down, Paging, Tabulator, Block-Move/-Copy/-Kill, Randausgleich, voll kompatibel mit AIM-Editor etc.. 2k-/4K-EPROM, DM 90,-/180,-. Handbuch DM 10,-. Info gegen Freiumschlag: Kurt Peter, Kölner Str. 6, 6053 Obertshausen, Tel.: 06104 - 71 789.

Kleinanzeigen kosten DM 10,-- für 2 Zeilen. Betrag bitte möglichst bei Auftragserteilung überw.



Wissen Sie noch...

auf welcher Diskette die Meßdaten vom 5. März 1981 stehen...

oder der Einspruch an das Finanzamt vom Juli 1982...?

Keine SORGE!

Wir finden alles wieder

D I S K E T T E N V E R W A L T U N G

Lauffähig auf allen PET/CBM - Systemen mit Basic 3.0 oder 4.0. 40 oder 80 Zeichen Bildschirm. 2040 / 4040 / 8050 Laufwerken unter DOS 1.0 / 2.0.

Die "DISKETTENVERWALTUNG" ist ein Hilfsprogramm für Commodore - Benutzer zur Verwaltung von bis zu 3.844 Disketten. Es wird geliefert auf Diskette und führt den Benutzer nach dem Systemstart im Dialog durch das Programm. Als Bedienungsanleitung erhalten Sie ein umfangreiches deutsches Handbuch mit vielen Beispielen.

F U N K T I O N S B E S C H R E I B U N G

- Anlegen eines Verzeichnisses aller Disketten mit ID, Diskname, Filenamen in alphabetischer Reihenfolge, Anzahl der freien Blöcke.
- Löschen einzelner Disketten aus dem Verzeichnis.
- Suchen einer bestimmten Diskette nach ID oder Disknamen.
- Suchen eines bestimmten Files nach Filenamen, fehlende Zeichen können durch * ersetzt werden. Der Filename wird mit der Diskette, auf der der File zu finden ist, angegeben.
- Ausdruck einer 62 x 62 Matrix mit graphischer Darstellung aller vergebenen Disk-IDs.
- Ausdruck aller vergebenen Disk-IDs in aufsteigender Reihenfolge.
- Ausdruck aller Disk-IDs mit dazugehörigem Disknamen in aufsteigender Reihenfolge.
- Ausdruck aller Disketten mit einer Mindest- bzw. Höchstzahl von freien Blöcken.

Das Programm verhindert die doppelte Vergabe von IDs und Disknamen durch eine Warnmeldung. Alle Funktionen sind auf dem Bildschirm oder durch Hardcopy darstellbar.

Der Preis: 339,00 DM incl. Umsatzsteuer und Handbuch. Auf Wunsch Handbuch vorab: 33,90 DM, wird auf Programmpreis angerechnet.

STECKER-SOFTWARE ... Sie werden sich wundern, was in ihr steckt!

STECKER

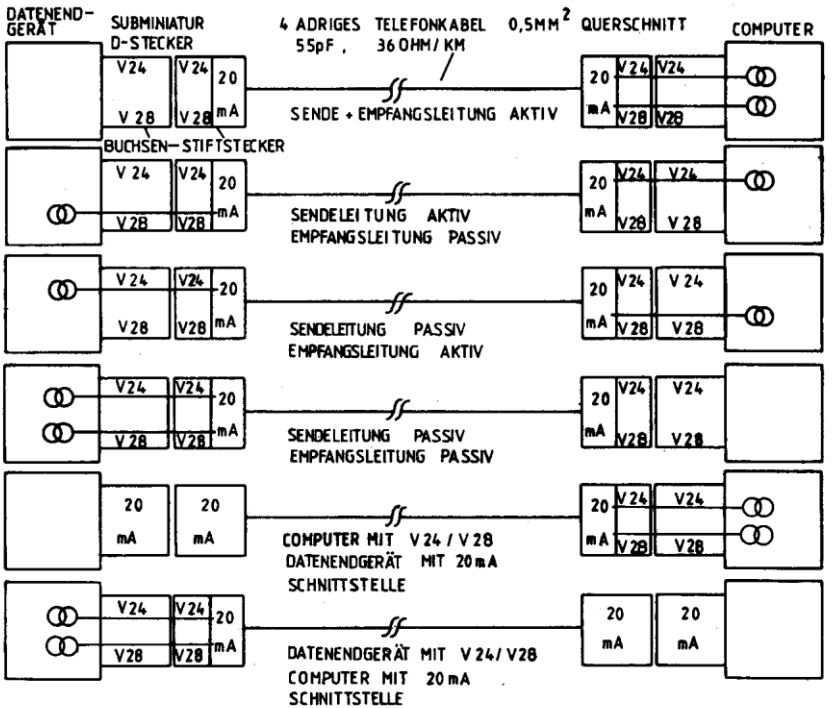
I N G E N I E U R B Ü R O

Postfach 60 07 66 5000 Köln 60 Neue Telefon-Nummer!! Tel.: (0221) 7 12 40 18

Die Stecker

von Stecker haben es in sich!

Die Wandlung der V.24/V.28 Spannungspegel in einen 20 mA Strompegel erfolgt durch eine Schaltung, die in die serienmäßigen Griffschalen der 25-poligen Subminiatur D-Stecker von AMP, Amphenol, Harting usw. eingebaut wird. Sie haben keine lästigen externen Geräte mehr herumstehen und müssen sich auch nicht mehr um deren Spannungsversorgung kümmern.



Der Stecker

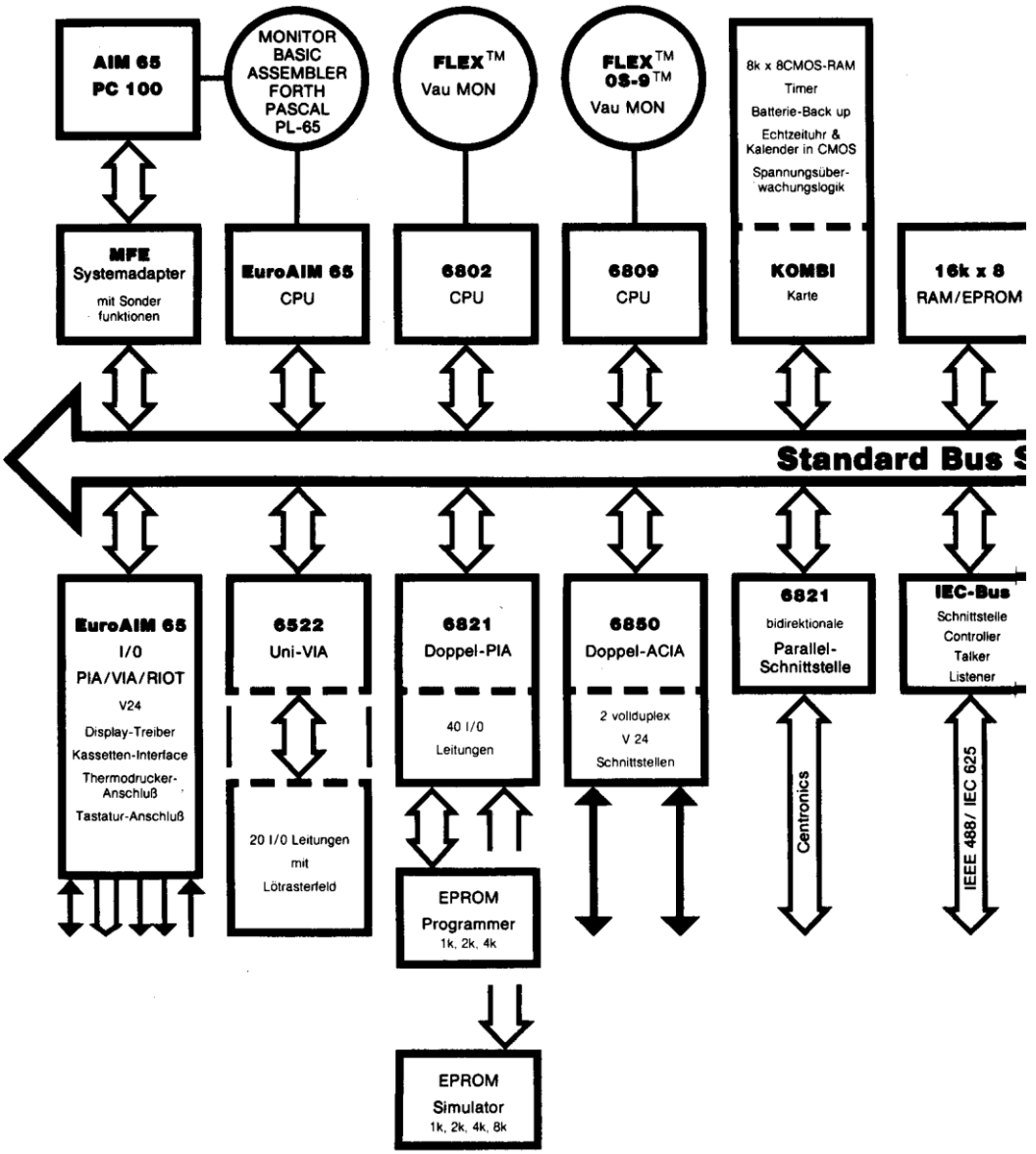
von Stecker wird einfach in die V.24 Buchsenleiste eingesteckt und schon können Sie mit der Datenübertragung bis zu 5 km je nach Widerstand und Kapazität der Leitung und der Baudrate ihre Daten zwischen Rechner und Datenendgerät übertragen.
Die Spannungsversorgung für

die Stecker

von Stecker erfolgt mit plus und minus 12 Volt über die in der DIN 66020 nicht genormten Steckkontakte 9 und 10.
Beide Spannungen werden nach CCITT Empfehlung für V.24 und V.28 typischerweise in Ihren Datenendgeräten und Computern bereitgehalten.
Alle Leitungen sind über Optokoppler galvanisch getrennt, so daß Einstreuungen über die Leitung nicht zur Zerstörung Ihrer Geräte und Anlagen führt. Die Datenübertragungsgeschwindigkeit beträgt max. 20 Kilobits/sec im vollduplex Mode.
Der Preis: 339,00/Stück incl. Umsatzsteuer. Nur gültig bis zum 31.12.1982.

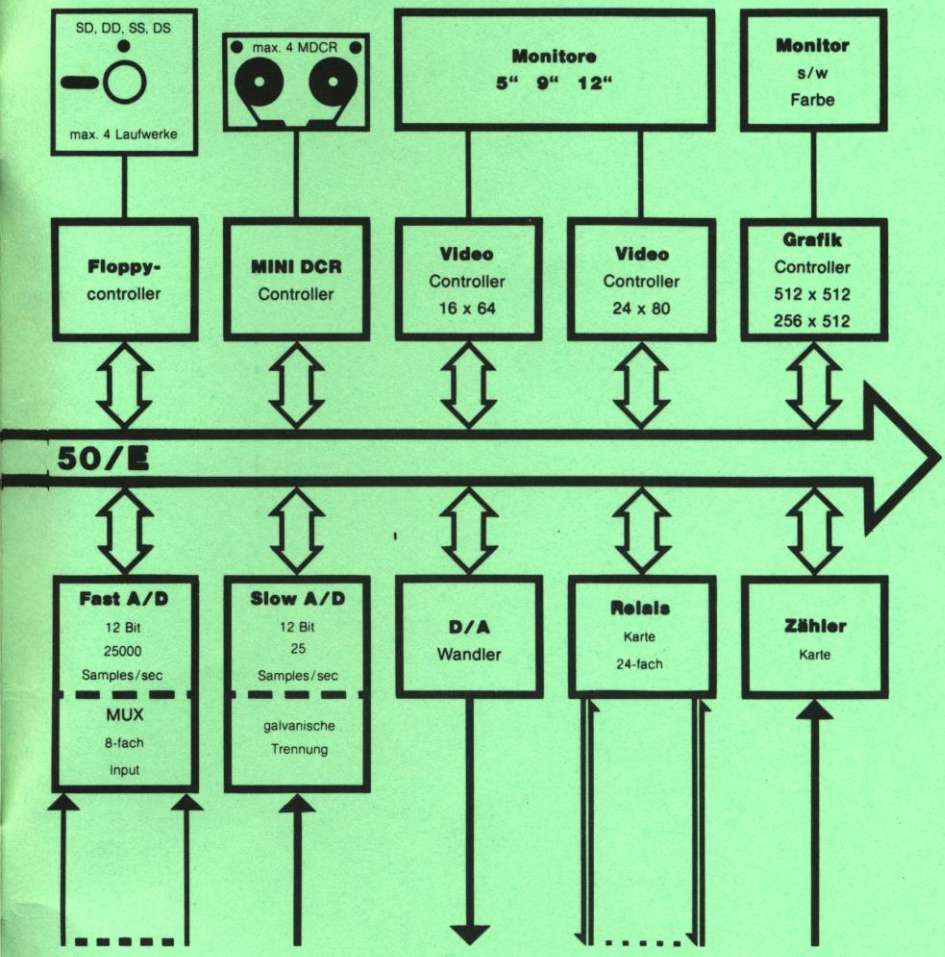
STECKER
INGENIEURBÜRO
Postfach 60 07 66 5000 Köln 60 Tel.: (0221) 7 12 40 18
Neue Telefon-Nummer!!

Dipl. Ing. (FH) Bernhard Heckl
Alte Wallensteinstraße 146
8500 Nürnberg 80
Telefon 0911/65 17 47



Und diese Vorteile bietet Ihnen das SS 50/E System

- Volle Kompatibilität aller Systemkarten untereinander.
- Jede Karte kann in jeden Busplatz gesteckt werden.
- Die Basisadresse kann bei allen Karten frei über den gesamten 64k-Bereich eingestellt werden.
- Alle Leitungen vom und zum Bus sind bipolar gepuffert.
- Elektrisch kompatibel zum SS 50 American-Standard.
- Adressierbar bis 1 M Byte.
- Voll DMA fähig.
- Einfach-Europaformat mit genormten Steckverbindern nach DIN 41612 Bauform C, a + c bestückt.
- Multiprozessorfähig.



Sie sehen hier die Basis eines Europakartensystems, welches Ihr Vertrauen verdient. Inovativ, zuverlässig und dennoch preiswert. Dieses System wird ständig erweitert und durch neue Technologien ergänzt. Daneben entwickeln und fertigen wir kundenspezifische Baugruppen und Systeme bedarfsgerecht.

65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

Herausgeber:
Dipl.-Volkswirt Roland Löhr
Hansdorfer Straße 4
D–2070 Ahrensburg
Tel.: 04 102 - 55 816

65xx MICRO MAG erscheint zweimonatlich, jeweils Mitte Februar, April usw..- Offsetdruck: Druckartist Gerhard M. Meier, Hamburg 70.

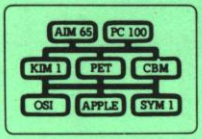
Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 54,-- (Inlandsendpreis). Ausland/foreign via surface mail DM 59,--, USA air 26 Dollar. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu zwei Wochen vor deren Ablauf.

Nachliefermöglichkeiten: Hefte 1-13 sollten als Buch nachbezogen werden (s. Anzeige unten). Solange Vorrat reicht, können auch noch einzelne Hefte der Nos. 7-13 geliefert werden. Hefte 14-27 sind unbeschränkt nachlieferbar zu DM 7,80/Stück + DM 2,50 je Sendung.

Private Besteller werden um Überweisung/Scheck (auch Auslandsschecks) zusammen mit der Bestellung gebeten. Konto Roland Löhr, Nr. 654 70-202 Postscheckamt Hamburg, BLZ 200 100 20.

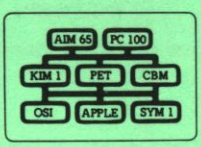
Leser-Service des Herausgebers

Das Buch 1-6 des 65.. MICRO MAG



220 Seiten, DM 26,--

Das Buch 7-13 des 65.. MICRO MAG



340 Seiten, DM 42,--

Thermokopf (Printerplatte) für AIM 65 und PC 100
mit Anleitung für den leichten Einbau. Auffrischung des Druckes DM 28,--

Thermopapier für AIM 65 ist von hier aus nicht mehr nachlieferbar.

FORTH User's Guide
Rockwell-Handbuch für das FIG-FORTH des AIM 65. Mit der Erläuterung des Befehlssatzes auch für andere FORTH-Betreiber geeeignet. Ca. 300 S. engl. DM 24,--

Diverse Assembler und Cross-Assembler für 6502, 6809, 6805, siehe im Heftinneren.

Alle Preise inkl. MWSt, zuzüglich DM 2,50/Sendung + ggfs. NN + DM 2,--