65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 7,80 Nr. 10 Dezember 1979

Am Jahreswechsel

befinden wir uns in einer tiefgreifenden wirtschaftlichen und politischen Herausforderung der großen alten Nationen, der diese hoffentlich mit civilen Mitteln begegnen können. - Diese Zeitschrift möchte mit ihren Mitteln der kompetenten Information weiterhin dazu beitragen, die Aufgaben der Zukunft in guter Vorbereitung anfassen zu können. Es wird ganz sicher noch mehr technische Intelligenz für unsere Produkte und für den Energieeinsatz geben müssen. Die gemeinsame Erörterung von weiterweisenden Ideen auf unserem Gebiet der Mikros gehört zu den notwendigen Komponenten.

Am Jahreswechsel gilt der Dank den Lesern und Autoren. Der Herausgeber wünscht ihnen ein glückliches und produktives 1980!

Inhaltsverzeichnis

SYNERTEC MDT1000 ist der Name eines im November vorgestellten weiteren Entwicklungssystemes (Micro Development Tool) mit Tastatur (54 Tasten) im Gehäuse und 31 cm Video-Monitor, Interface für 2 Cassetten, EPROM-Programmer, 4 kB statischem RAM, parallelem und seriellem Druckerinterface, Sockeln für 4 ROMs, ACIA für serielle Datenübertragung und einem Motherboard mit 4 Aufnahmen. Die Firmware umfaßt 12 kB, davon 4 kB Monitor und 8 kB Assembler/Editor, die mit numerierten Zeilen arbeiten. BASIC-ROMs in 8 kB zusätzlich als Option. Die Hardware wird als voll kompatibel zu Motorolas EXORciser-Bus (TM) herausgestellt. Preis in den USA $1495.

80 microcomputing ist der Titel einer neuen auf die 80er Mikros spezialisierten Zeitschrift, die ab 1980 im Verlage der Zeitschrift 'microcomputing', früher 'kilobaud', erscheinen wird. Info und Abo: Micro-Shop-Bodensee, Postfach 11 22, 7778 Markdorf.

VIDEO PLUS, die vom Micro-Shop-Bodensee vertriebene Videokarte, wurde in Heft 8 auf Seite 32 kurz vorgestellt. Kurz vor Redationsschluß konnte sie beim Herausgeber für Besprechungszwecke in Betrieb genommen werden, und zwar mit Betriebsprogramm in EPROM und Cassette. Bildformat 20 Zeilen mit je 80 Zeichen. Sie liefert ein sauberes Bild, eine Besprechung im einzelnen soll folgen. Als Benutzer würde man gerne eine klarere Betriebsanleitung sehen, möglichst in Deutsch.

Ein Ausbausystem für 65xx, inkl. Floppy Disk im IBM-Format, Controller und Betriebssystem, wird von der Firma F.J. Mertes, Hahnenstraße 16, 5030 Hürth-Efferen, angeboten. Tel.: 02233-680 28. Dazu gehören CPU- und RAM-Karten ebenso wie PROMMER. Ausbaubar bis zum kommerziellen Kleinrechnersystem mit Video-Terminal und Centronix-Drucker.

AIM-SPEZIAL (5)

Systeminitialisierung mit LOAD=T (Tape)

Das Programm TVINT (M. Helm, in dieser Ausgabe) zur direkten Initialisierung des DILINK (Ausgabe auf ein Bildschirmterminal) vom Magnetband her gab Anlaß zu weiteren Überlegungen und Versuchen.

Jedem AIM-Betreiber ist geläufig, daß während des Schreibens von Objectcode auf Magnetband (DUMP-Command) vor dem letzten Bandsatz die Frage MORE? erscheint. Antwortet man mit einem 'N', so wird ein letzter Satz geschrieben und man kehrt zum Monitor zurück. Antwortet man dagegen mit 'Y' (YES), so wird wieder die Frage nach dem FROM= und TO= gestellt. Man kann dann weitere Speicherauszüge auch aus ganz anderen Bereichen auf Band absetzen, eine Möglichkeit, die man im Anwenderhandbuch auf den Seiten 3-43 und F6/F7 zu schnell überliest, die sich aber sehr vorteilhaft zur Systeminitialisierung verwenden läßt.

Man kann sogar einzelne adressierte Bytes auf Magnetband absetzen, z.B. den DILINK-Vektor, Interruptvektoren, Printerflag, User Input- und Outputvektor, Belegung der Funktionstasten F1-F3 mit Sprungbefehlen, ferner in den Zellen A420-A426 die Registerinhalte für das Anwenderprogramm einschl. Programmzähler (Sternadresse).

Bei geschicktem Aufbau der Reihenfolge kann man ferner die Peripherie vollständig initialisieren: Datenrichtungsregister, Ausgabesignal an den Ports, Schaltung der Steuerleitungen, Inbetriebnahme der Timer, Interruptenable. Mit der Reihenfolge Interruptprogramm, Interruptvektor, Interruptenable, Starten eines Timers ganz am Schluß des Bandladens ist voraussichtlich ein Selbststart des Anwenderprogrammes möglich. Das kann sich als vorteilhaft erweisen, wenn wenig geschulte Bediener, z.B. in der Datenerfassung, das System mit wenigen einfachen Handgriffen (Laden) in Betrieb nehmen sollen. Alle verstreuten Vektoren, Flags und Programmteile stehen dann von Anfang an zur Verfügung, das Anwenderprogramm befindet sich wohlmöglich schon in Ausführung und riegelt über die geeignete Tastaturabfrage sogleich Fehlbedienungen ab.

Die Verwaltung des Interruptenable ist allerdings noch nicht ganz durchsichtig und möglicherweise auch nicht konsequent. In TAISET wird der Befehl SEI gegeben, in TIBY5 ein CLI. Am Schluß in DU13 steht statt des beabsichtigten CLI ein CLC.

Der für die Initialisierung bisher aufgezeigte Weg spart auch Speicherplatz, denn es entfällt ein Initialisierungsprogramm mit dem üblichen Transport von Parametern. Ein solches Programm wurde bisher vom Band ins RAM geladen oder vom Anwender in Festwertspeichern vorgehalten. Man kann sich weiterhin vorstellen, daß man während der Initialisierung auf ein schnelleres Bandleseformat nahtlos übergehen kann, z.B. auf den Hamming-Way, um größere Speicherbereiche beschleunigt zu laden. Hier liegen noch viele Ansatzpunkte für gestalterische Arbeit.

In die Initialisierungsüberlegungen sollte man für die tägliche Arbeit z.B. auch die Pointer für den Editor in 00DF-00E6 einbeziehen und wohlmöglich auch eine Standard-Symboltabelle im Editor-Quellentext, die von dem einen benutzten File als Objektcode gleich mitgeladen wird.

Kurze Versuche zeigten, daß man über diese Form der Initialisierung (durch das Bandladen) auch ein bereits initialisiertes BASIC übernehmen kann. Zu diesem gehören bekanntlich zahlreiche Pointer und auch Maschinenbefehle in der Nullseite (Zeropage) sowie initialisierte Bytes in

Speicherseite 2. Ab Zelle 0212 schließt sich der BASIC-Programmtext an. Er erstreckt sich bis zu einer Adresse, die in der Pointerzelle 0075/76 gelogt wird (dort um 1 erhöht, weil pointer to start of variables). Nach einem RUN werden dahinter die Variablen angelegt. Am Ende des Speichers hat man ggfs. selbständige oder mit BASIC verbundene Maschinenprogramme.

In den ersten Versuchen wurde der von BASIC initialisierte größte Teil der Nullseite auf Magnetband übernommen (ein bisher sicher noch zu großer Teil), ferner der Bereich ab Adresse 0200 bis zum Ende des Quellentextes (Adreßwort in 75/76 minus 1). Nach Ausschalten und Wiederladen des Objectfile konnte das abgespeicherte BASIC-Programm einwandfrei gelistet und betrieben werden (Wiedereintritt in BASIC mit Taste '6').

Das Laden von (hexadezimalem) Objectcode erfolgt wesentlich schneller als das LOAD unter BASIC. Wenn auch das Mitladen von Bestandteilen der Nullseite etwas Zeit benötigt, so wird man schon beim Laden eines kleineren lauffähigen BASIC-Programmes per Objectcode Zeitvorteile erreichen können (LOAD ist recht langweilig) und braucht nicht zu initialisieren. Diese Aussagen gelten auch für ein über die Pointer im Speicher verschobenes BASIC (s.u.).

Beispiel eines selbststartenden Programmes

Der bisher aufgezeichnete Initialisierungsweg über Magnetband hat seine Begrenzungen. Während des DUMP von Objectcode (D-Command) kann man nur solche Informationen abspeichern, die zum Zeitpunkt dessen tatsächlich in den Speicherzellen vorhanden ist, nicht aber eine vielleicht wünschenswerte andere Vorgabe. Diese Freiheit erreicht man nur mit einem vom Assembler mit OBJ-OUT=T erzeugten ladefähigen Programmmodul (wie im Programm TVINT), denn der Assembler teilt die Parametervorgaben nach dem Willen des Programmierers mit. Wir machen im nachstehenden Beispiel davon Gebrauch, und zwar hinsichtlich des Vektors (Adresseninhaltes) für das DILINK.

Dazu folgende Überlegungen: Das Betriebsprogramm erzeugt laufend Information für die Ausgabe (Printer, Terminal, Display). Während des Bandladens teilt es den Namen der aufgefundenen Files und die Blocknummern mit. Am Ende des Ladevorganges sendet es den Monitor-Prompt (<). An dieser Stelle haken wir ein: Mit einer letzten Zuweisung im Assemblerprogramm richten wir den Vektor für das DILINK auf ein kleines Beispielprogramm. Dieses kommt zur Ausführung, sobald der Prompt (<) über die Ausgaberoutinen läuft. Wir haben damit einen echten Selbststart erzielt, dessen Prinzip man für jedes Anwenderprogramm verwerten kann.

Man beachte: a) den DILINK-Vektor erst am Schluß des Quellenprogrammes auf das Anwenderprogramm richten, b) das Maschinenprogramm vom Assembler auf Magnetband generieren lassen, c) Magnetband wie üblich mit L=T laden, d) am Schluß der eigenen Routine den Stackpointer korrigieren und e) das DILINK in der benötigten Form für die normale Ausgabe wiederherstellen.

```
;SELBSTSTARTENDES PROGRAMM
;SYMBOLE
DILINK=$A406
OUTDIS=$EF05
*=$500
START
LDX #6          ZÄHLER UND ADDRESSER        0500 A2 LDX #06
LOOP TXA        X NACH A                    0502 8A TXA
STA 0,X         IN DIE ERSTEN ZELLEN        0503 95 STA 00,X
DEX                                         0505 CA DEX
```

```
BPL LOOP      SCHLEIFE 7×                    0506 10 BPL 0502
LDA #<OUTDIS  VEKTOR AUF NORMALE AUSGABE     0508 A9 LDA #05
STA DILINK    AUF DAS DISPLAY UMSETZEN       050A 8D STA A406
LDA #>OUTDIS                                 050D A9 LDA #EF
STA DILINK+1  FÜR HIGH                       050F 8D STA A407
LDX #$FF      STACKPOINTER AUF $FF           0512 A2 LDX #FF
TXS                                          0514 9A TXS
BRK           RÜCKGABE AN DEN MONITOR        0515 00 BRK

*=DILINK      ZUORDNEN DES VEKTORS VON START
.WOR START
.END
```

NACH DEM LADEN DIESES PROGRAMMES FINDEN WIR IN DEN ERSTEN ZELLEN ERWARTUNGSGEMÄSS: <M>=0000 00 01 02 03 04 05 06 ..

In Verfolgung des Selbststarter-Prinzips sollte man Versuche auch mit 'intelligenten' Programmmoduln auf Cassetten machen, die ein kleines Hauptprogramm, das den Ablauf und die Daten verwaltet, im Overlayverfahren unterstützen. Systeme mit kleinem Speicher können damit sehr lange Programme ausführen.

Ein Hinweis noch: In unserem Beispiel wurde der Vektor von OUTDIS erst am Schluß des Programmes nach DILINK gebracht. Es ist sicher zweckmäßiger, diesen Transport am Beginn des Anwenderprogrammes zu vollziehen, um die Kommunikationsmöglichkeit des Rechners zu gewährleisten.

Das AIM-Bandformat für Objectfiles

Die vorstehenden Aussagen beruhen auf einer weiteren Untersuchung des AIM-Bandformates. Dafür bot sich der Ein- und Ausgabepuffer für Magnetband TABUFF in den Speicherzellen ab 0116 an. Die Aufzeichnung von Daten in 'geblocktem Format' bedeutet: ein physischer Block von 82 Zeichen in TABUFF gliedert sich in eine größere Anzahl von Sätzen (records). Der erste Block einer Datei (FILE) beginnt mit der Block-Nr. 00, es folgt der Name in 5 ASCII-Bytes, ein 'CR' und ein ';' (hex 3B). Das folgende Byte enthält (in hex) die Menge der Datenbytes dieses Satzes. Die Startadresse HIGH und LOW des ersten enthaltenen Datenbytes schließen sich an. Es folgen jetzt die Datenbytes (bisher war es Steuerinformation) und in 2 Bytes die Checksum, die sich auch auf die Startadresse bezieht. Ein 'CR' (hex ØD) dient danach als Satztrenner. Der nächste Satz beginnt wieder mit der Sequenz ab ';'. Hierzu folgendes Schema:

```
BLOCK | NAME    | CR| ;| # OF |SAH|SAL| D A T A |CKSUM | CR|| ;.......
  #   | | | | | |   |  | BYTES|   |   |         |  |   |   ||
```

Es ist nun bemerkenswert, daß ein Datensatz nur von 1-24 Datenbytes enthalten kann. Ein mit FROM= und TO= vorgegebener großer Speicherbereich wird also in viele Sätze gegliedert. In der jedem Satz mitgegebenen Adressierung liegt der Schlüssel für die im vorigen Abschnitt aufgezeigte Systeminitialisierung vom Magnetband her.

Die satzmäßige Gliederung und Adressierung der Datenbytes mit ihrer Herkunfts-(=Ziel-)adresse bildet zusammen mit der Kontrollsumme sicher ein hohes Maß von Sicherheit. Andererseits erschwert sie die vollständige Rekonstruktion einer fehlerhaften Datei (Abbruch mit ERROR), weil nur bis zur schadhaften Stelle geladen werden kann. (Ein vollständiges - wenn auch fehlerhaftes Laden ermöglicht eine Programmprüfung ohne allzuviel Arbeit beim Nachtippen.) Es sollte allerdings möglich sein, eine Hilfsroutine für das Laden des Restes zu schreiben, wenn man den

Blockzähler BLK in 0115 erhöht und mit der Routine TIBY1 in ED53 fortfährt. Allerdings wäre vorher noch der Inhalt des TABUFF an seinen Bestimmungsplatz zu bringen.

Das Betriebsprogramm des KIM läßt ein Laden in neue Speicherbereiche zu. Wegen der satzweise mitgegebenen Adressen wird sich dergleichen beim AIM nur unter erschwerten Bedingungen vollziehen lassen, denn man wird jede von einem Bandblock kommende Startadresse prüfen und verändern müssen. Damit ist das Thema der Verschieblichkeit angesprochen.

Nach Ansicht des Herausgebers sollte man beim AIM 65/PC100 Verschieblichkeit dadurch gewährleisten, daß man seine Programme grundsätzlich als Quellenprogramme im Editor anlegt und als Textfile speichert. Durch einfaches Ändern der Sternadresse kann man sich dann das Programm für jeden anderen Speicherbereich assemblieren lassen. Die dabei entstehende Mühe ist sicher nicht größer als bei der Handhabung eines Verschiebeladers, zu dem konsequenterweise ja auch ein Umrechnungsprogramm für die Adressen gehört (RELOCATE). In diesem Zusammenhang sei auch auf die mögliche Lösung in Heft 7 hingewiesen: MOVE and RELOCATE.

Auch beim Programmaustausch untereinander sollte man Quellenprogramme verwenden, die vom Assembler generiert werden können. Nach aller Erfahrung möchte der Empfänger neue Bereichszuweisungen vornehmen oder die gebotenen Dienstleistungen mit anderen vorhandenen abstimmen oder ergänzen. Ein Quellenprogramm erspart dabei viel Eintipparbeit.

AIM-BASIC

In Heft 8, Seite 41, gingen wir bereits auf die Zeropage-Pointer und auf die Abspeicherung des BASIC-Programmtextes ab Zelle 0212 ein. Zahlreiche Leser möchten aus Gründen eines eigenen erweiterten Betriebssystemes BASIC auch in anderen RAM-Bereichen betreiben können. Das ist problemlos möglich. Unmittelbar nach der Initialisierung mit <5> findet man folgende Einstellung der Pointer:

ADRESSE	POINTERFUNKTION	NORMALE INITIALISIERUNG	BEISPIEL
$ 73/74	START DES PROGRAMMTEXTES	12 02	81 02
$ 75/76	START DER VARIABLEN	14 02	83 02
$ 77/78	START DER ARRAY-VARIABLEN	14 02	83 02
$ 79/7A	HÖCHSTE BENUTZTE SPEICHERSTELLE	14 02	83 02

Zugleich finden wir die Initialisierung <M>=0211 00 00 00. Wir brauchen jetzt nur die Pointer wie im Beispiel zu manipulieren, wenn wir BASIC-Texte z.B. ab Speicherstelle $0281 anlegen wollen. Dazu gehören drei Nullen in 0280-0282: <M>=0280 00 00 00, wobei die letzten beiden Nullen das Link (Bindeglied) mit der Bedeutung 'Ende des Textes' darstellen.

Nach den Beobachtungen stört sich der Interpreter nicht, wenn man zwischen BASIC-Text und Variablen einen unbenutzten Freiraum läßt.

Die Darstellung von BASIC-Variablen

Variable werden vom Interpreter-Programm erst nach RUN angelegt, und zwar hinter dem Programmtext (siehe o.a. Pointeradressen). Beim Anschluß von Maschinenprogrammen an BASIC wird man sich in vielen Fällen um die Informationsdarstellung in den Variablen kümmern müssen. Dazu die folgenden Abschnitte.

Folgende Variablenarten sind zu unterscheiden:

Ganzzahl-Variable (Integers)	z.B. A%, B4%
Gleitkommazahlen (Floating Point)	A, Z3
Stringvariable (alpanumerische Textketten)	A$, D1$

Array-Variable, mit DIM erklärt, stellen Datenfelder mit mehreren Variablen gleichen Typs dar. Sie können für die drei vorgenannten Variablenarten angelegt werden, also Ganzzahlen, Gleitkommazahlen und Strings.

Zunächst zu den einfachen Variablen: Sie werden im Programmtext erklärt und benutzt. Im Variablenspeicher werden sie grundsätzlich in 7 Bytes abgelegt. Die ersten 2 Bytes enthalten den Namen in ASCII-Darstellung (z.B. hex 41 31 für ASCII A1). Die Besetzung der höchsten Bits (Bit 7) in diesen beiden Bytes kennzeichnet zugleich die Variablenart:

VARIABLENART	BYTE 1	BYTE 2
GANZZAHL (INTEGER)	1. ZEICHEN + $80	2. ZEICHEN + $80 ODER NUR $80
GLEITKOMMA (FLOAT)	1. ZEICHEN	2. ZEICHEN ODER NUR $00
STRING	1. ZEICHEN	2. ZEICHEN + $80 ODER NUR $80

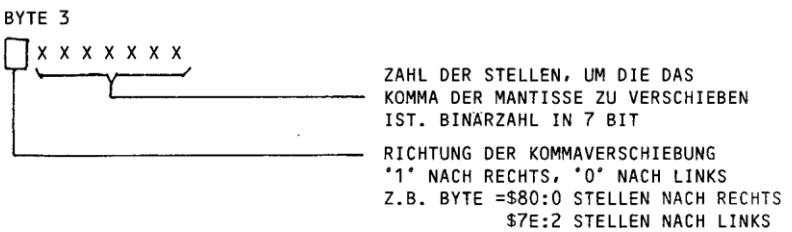
Nun zur Darstellung des Variableninhaltes: Für eine Integer-Variable - Erklärungsformat A% - nehmen die beiden folgenden Bytes den Variablenwert einschl. Vorzeichen auf (Zahlenbereich -32768 bis +32767), die Bytes 5-7 bleiben mit Inhalt 00 00 00 unbenutzt. Negative Zahlen sind in den Bytes 3 und 4 als Zweierkomplement dargestellt, also z.B. - 1 ≡ FFFF.

	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
GANZZAHL-VARIABLE	BINÄRZAHL				
	HIGH	LOW	00	00	00

Gleitkommazahlen decken den Wertebereich von $\pm$ 1,5 x 10^{38} bis $\pm$ 2,9 x10^{-39} ab. Ihr Ausdruck erfolgt im allgemeinen in exponentieller Schreibweise, außer im Bereich von 0,01 bis 999 999 999, für den Festkommazahlen ausgegeben werden. Für die Zahl selbst (Mantisse) und für das Vorzeichen des Exponenten muß BASIC das Vorzeichen verwalten. In die besondere Darstellungsform dieser Gleitkommazahlen unter BASIC muß man sich etwas vertiefen.

Auch sie sind als Binärzahlen dargestellt, allerdings in einem besonderen Format. Von der für den 6502 möglichen dezimalen Interpretation und Verarbeitung hat man also keinen Gebrauch gemacht. Wegen der bei Zahlenwandlungen häufig auftretenden Ungenauigkeit ist das sehr zu bedauern.

Byte 3 enthält den Exponenten der Mantisse zur Zahlenbasis 2. Ein hier gesetztes vorderstes Bit (Zelleninhalt ≥$80) gibt an, daß der Exponent positiv ist und umgekehrt. Die nachfolgenden 7 Bit bilden den Zahlenwert des Exponenten zur Zahlenbasis 2. Da im binären Zahlensystem die Multiplikation mit 2 eine Linksverschiebung einer Zahl um 1 Stelle und das Anhängen einer Null bedeutet, gibt der Exponent in Byte 3 an, um wieviele Stellen die in den Bytes 4-7 enthaltene Mantisse zu verschieben ist, um eine stellenrichtige Binärzahl abzubilden. Man muß genauer sagen: Der Exponent gibt an, um wieviele Stellen das Komma der Mantisse zu verschieben ist.

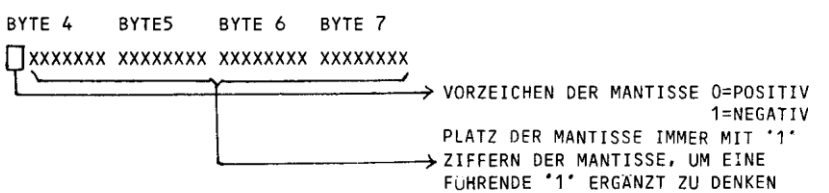


Das Exponentenbyte betrifft mithin die Kommaverwaltung für die Mantisse, die in den Bytes 4-7 als 'normalisierter' Zahlenwert abgebildet ist. Was ist nun 'normalisiert' und was ist die 'Mantisse'?

Normalisiert, d.h. man nimmt das Komma links von der Zahl an. Statt z.B. 133 schreibt man 0,133 x10^3 oder statt binär 101 schreibt man 0,101 x2^3. Was hinter dem Komma steht, ist die Mantisse. Der Exponent gibt an, um wieviele Stellen deren Komma zu verschieben ist. Der Begriff 'normalisiert' beinhaltet noch etwas mehr: Das Komma der Mantisse ist solange zu verschieben, bis vor dem Komma eine Null steht und bis zugleich auch in der ersten Nachkommastelle eine gültige Ziffer steht. Folgende Zahlen sind also noch nicht normalisiert: 1,33 x10^2 oder 0,0133 x10^4.

Warum nun wird eine Binärzahl oder auch eine Dezimalzahl normalisiert? Man möchte mit möglichst wenigen Bytes für die signifikanten (aussagefähigen) Ziffern auskommen. Lange Ketten angehängter Nullen stellen bei großen Zahlen eine Belastung dar, ebenso eine Reihe von Nullen zwischen dem Komma und den signifikanten Ziffern bei Bruchzahlen. Die Verschiebung der signifikanten Ziffern direkt hinter das Komma und die Beschneidung des Speicherraumes auf 4 Bytes je Zahl bringt zwar Speicherersparnis, bedingt aber besonders bei sehr großen Zahlen und bei Brüchen eine gewisse Ungenauigkeit.

Normalisiert heißt: Es gibt immer eine signifikante Ziffer hinter dem Komma, in binärer Notierung also eine '1'. Wenn man das so genau weiß, dann kann man wie Microsoft als Urheber des BASIC, sogleich an eine Speichersparmaßnahme herangehen und sagen: 'Diese 1 notieren wir gar nicht, wir denken sie uns nur immer hinzu. Das freiwerdende Bit verwenden wir für das Vorzeichen der Mantisse. Ist es 0, so ist die Mantisse positiv, ist es 1, so ist sie negativ'. Für die Darstellung von Binärzahlen in den Bytes 4-7 daher folgende Darstellung:

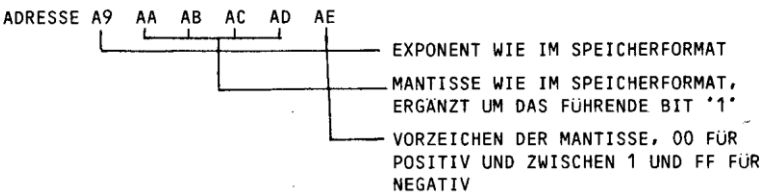


Hier nun einige kleine Beispiele für Gleitkommazahlen im Speicherformat, einschl. Exponentenbyte.

BYTE 3	4	5	6	7	DEZIMALZAHL
00	00	00	00	00	0
81	00	00	00	00	1
81	80	00	00	00	-1
82	00	00	00	00	2

BYTE 3	4	5	6	7	DEZIMALZAHL
82	80	00	00	00	-2
82	40	00	00	00	3
82	C0	00	00	00	-3
82	D0	00	00	00	-3,25 BINÄRMUSTER NACH KOMMAVERSCHIEBUNG 11.01 ENTSPRICHT 3 1/4
7F	00	00	00	00	1/4 = 0,25
7F	40	00	00	00	1/4 +1/8 = 0,375
7F	C0	00	00	00	-0,375

Solche Sparmaßnahmen bedingen Umformungen, die auf die Ausführungszeit gehen. Angenehm kann man in diesem Speicherformat für Gleitkommazahlen nicht rechnen. Wenn der Programmierer nun sagt 10 A=A+1, dann muß möglichst maschinengerecht eine Zahlenwandlung stattfinden. Die Variable wird aus dem Speicher in den sog. Gleitkomma-Akku gebracht (Zellen $00A9-AE) und in eine etwas andere Darstellung umgewandelt. Der Name gelangt dabei nicht in den Gleitkomma-Akku (FAC, Floating Point Accumulator), nur Exponent mit Vorzeichen (wie im Speicherformat) und Mantisse mit Vorzeichen. Der Name der zuletzt benutzten Variablen wird in 8D/8E festgehalten, der Speicherplatz ihres Zahlenwertes in 8F/90. Im Gleitkomma-Akku finden wir folgendes Format:



BEISPIELE

7F	C0	00	00	00	00	= 0,375
7F	C0	00	00	00	FF	= -0,375

Wer mit dem BASIC-Command USR(X) oder mit ATN(X) Variable mit einem Maschinenprogramm austauschen möchte, sollte sich um das Verständnis dieser Darstellungsformen für numerische Variable bemühen.

Stringvariable.

Hier gelten folgende Besonderheiten: Bei der Zuweisung eines Strings zu einer Variablen in einer Programmzeile wie z.B. 10 A$="AIM 65/PC100" wird sein Anfangswert als ASCII-Code in den Programmtext aufgenommen. Nach RUN wird eine Variable angelegt, die jedoch nicht den Zeichenstring selbst enthält, sondern seine Länge (in hex) sowie den Ort, an dem er aufgefunden werden kann. Das ist bei einer reinen Zuweisung der Quellentext.

BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
LÄNGE (HEX)	ADRESSE DES STRING-BEGINNES (HEX) LOW	HIGH	00	00

Nach Stringmanipulationen wie A$=A$+B$ wird die Länge und auch der ursprüngliche Inhalt ungültig. Der Interpreter kann den String jetzt also nicht mehr aus dem Quellentext des Programmes entnehmen. Der resultierende String

muß neu angelegt werden, und zwar am Ende des dem BASIC zur Verfügung gestellten Speicherraumes (top of memory). Von dort also wachsen die Strings dem Programm und seinen Variablen, bzw. dem Stringverzeichnis rückwärts entgegen. In die zuvor erwähnten Bytes 3-5 wird die neue Länge eingetragen sowie die neue Adresse des Strings am Ende des RAM.

Die Ablage der Strings am Ende des Speichers kann dort befindliche Anwenderprogramme zerstören, wenn bei der Initialisierung mit MEMORY SIZE? (Anlage des Pointers in 7F/80) nicht entprechende Einteilungen beachtet werden.

Array-Variable

Sie werden ab Pointeradresse in 77/78 gespeichert. Eine Kette von 7 Bytes beschreibt das Array (Descriptor) Die beiden ersten Bytes enthalten den Namen und über die Steuerbytes die Variablenart - wie bei den einfachen Variablen. Das 3. und 4. Byte geben in hex in der Reihenfolge Low High den Umfang des Arrays an:

UMFANG = (DIMENSIONIERUNG +1)×L +7

Die Dimensionierung richtet sich dabei nach der BASIC-Zuweisung DIM ..(..), bei der auch immer ein Null-Element angelegt wird. Daher +1 in der Klammer. 7 ist die Länge des Descriptors. L nimmt je nach Variablentyp verschiedene Werte an: L=2 für Ganzzahl-Variable, L=3 für String und L=5 für Gleitkomma-Arrays.

Byte 5 des Descriptors enthält 01, Byte 6 und 7 enthalten in der Reihenfolge High Low die durch DIM angewiesene Dimensionierung.

Numerische Variable werden direkt im Array gespeichert. Für Stringvariable wird - wie für einfache Strings - innerhalb des Arrays ein Zähler für die Länge geführt, sowie die Adresse verwaltet, unter der der jeweils gültige String aufgefunden werden kann, entweder im Quellentext oder - nach Veränderung der Stringvariablen - am Ende des RAM-Speichers.

Die Gliederung des BASIC-Speichers in 4 Abschnitte

Programmtext,
Einfache numerische und Stringvariable,
Arrays,
Strings bearbeiteter Stringvariabler

hat Auswirkungen auf die Aufführungszeit eines Programmes. Wenn man Arrays mit DIM ... früh im Programm erklärt hat, so muß jede hinzukommende einfache Variable das Array unter Umrechnung nach hinten schieben. Es ist daher vorteilhaft, einfache Variable vor DIM-Statements zunächst erklärt zu haben, ihre Benutzung kann dann beliebig erfolgen. Bezüglich der Ausführungszeit noch einige Hinweise:

Man sollte den Namen ausgedienter Variabler in anderen Funktionen weiterverwenden. Das spart Speicherplatz und ermöglicht ein schnelleres Auffinden. Den Namen der am häufigsten benutzten Variablen deklariere man ganz am Anfang eines Programmes. Diese Empfehlung betrifft ebenso die in Programmschleifen häufig verwendeten Laufvariablen I, J, K usw.. Auch hier verwende man möglichst wenige Namen.

Bei eindeutiger Laufvariabler, z.B. I, führt ein Programm in Schleifen (FOR I= ... TO ...) schneller aus, wenn man einfach nur NEXT schreibt und nicht NEXT I. Auch die Verwendung mehrerer Anweisungen in einer Programmzeile, jeweils durch Doppelpunkt getrennt, führt zu Speicherplatzersparnis und schnellerer Ausführung, wie z.B. 10 FOR I=1 TO 10: PRINT I: NEXT.

R. L.

MTEST - SPEICHERPRÜFUNG

Ing. grad. Horst Steder, Talstraße 10 b, 6000 Frankfurt 56

E: AIM Utility to detect Softerrors in RAM memory. A random pattern is written and compared continuously. Some errors will only be detected after a break and return to monitor and a restart.

Der 'RAM-Test with Random Patterns' in Heft 3 dieser Zeitschrift für den KIM-1 veröffentlicht,wurde in MTEST für den AIM 65 adaptiert und erweitert. Wie dort erwähnt, sollte man eine Speicherprüfung nicht nur mit regelmäßigen Bitmustern durchführen, sondern vor allem auch mit zufälligen. Weiterhin sollte das geschriebene Muster nicht in Regelmäßigkeiten der Speicherarchtitektur einrasten, daher wurden zufällige Bitmuster gewählt, die jeweils 11 Byte umfassen und die aneinandergereiht werden. Ein Testprogramm muß weiterhin in endloser Schleife gefahren werden können, um ggfs. unter Erwärmung eintretende Speicherfehler ('softerrors') erfassen zu können.

Zur Heimtücke möglicher Speicherfehler sei aus den Erfahrungen des Verfassers folgendes mitgeteilt: Ein schadhaftes 2114-RAM zeigte mit der bisher aufgezeigten Zufallszahlenprüfung auch nach stundenlangem Test keinen Fehler. Auch das Einfügen einer Zeitverzögerung von einigen Sekunden zwischen Beschreiben und Vergleichen sowie das Umspeichern der Wertepaare vor dem Vergleich brachte nichts zutage. Erst das Anhalten des Programmes mit BREAK und ein Neustart zeigten den Fehler. Daher wurde in Zeile 261 das Unterprogramm KBSCAN aus dem Monitorprogramm eingefügt, und zwar für das bequeme Anhalten nach einem kompletten Schreibzyklus. Der den Fehler anzeigende Wiederstart wird mit '=267' und'G' bewirkt. Dabei erscheint das Sternchen wieder auf dem Display. - Die Ursache für das erwähnte dynamische Phänomen konnte aus Zeitgründen noch nicht eingekreist werden.*

Zur Programmbedienung: Es wird wie üblich gestartet und reagiert mit den interaktiven Prompt 'FROM=' und 'TO=' für die Eingabe der Start- und Endadresse des zu testendem RAM-Bereiches. Dabei wird nur der höherwertige Adreßteil verwertet, das heißt es werden immer volle Speicherseiten geprüft. Es können aber auch kleinere Bereiche innerhalb einer page getestet werde-, wenn man das Adreßbyte LOW in $A41D nach Y überträgt.

```
02E1          PASS 1
02E1          PASS 2
0000
0000          ;         ************************************
0000          ;         *                                  *
0000          ;         *    PROGRAM  < M T E S T >        *
0000          ;         *                                  *
0000          ;         *                     H.STEDER 10/79 *
0000          ;         ************************************
0000
0000
0000          ;THIS PROGRAM IS A MODIFIED <AIM 65> VERSION OF R.LOEHR'S
0000          ;RAM TEST. IT TESTS THE RAM IN AN ENDLESS LOOP USING
0000          ;RANDOM PATTERNS IN 11 BYTES TO AVOID 'LOCK-IN' ON THE 8/16
0000          ;BIT STRUCTURE. IF AN ERROR OCCURS, THE AIM-65 DISPLAY WILL
0000          ;SHOW 'MEM FAIL', THE ADDRESS, THE VALUE IN THE DEFECTIVE
0000          ;CELL AND THE CORRECT VALUE WHICH WAS WRITTEN INTO IT.
0000          ;
0000          ;EXAMPLE:   <MEM FAIL 1123 D4 B4
0000
0000
```

```
0000                    *=$0000
0000 TABLE              *=*+11
000B POINTL             *=*+1
000C POINTH             *=*+1
000D COUNT              *=*+1
000E STRTPG             *=*+1
000F ENDPGE             *=*+1
0010 SAVX               *=*+1
0011 SAVY               *=*+1
0012 RNDVAL             *=*+3
0015 ADDR               =$A41C
0015 AIM65              =$E182
0015 FROM               =$E7A3
0015 TO                 =$E7A7
0015 KEP                =$E7AF
0015 BLANK              =$E83E
0015 KBSCAN             =$E90C
0015 DSPLAY             =$E97A
0015 NUMA               =$EA46
0015
0015
0015                    *=$0200
0200
0200           20A3E7 JSR FROM        ;GET START-PAGE
0203           AD1DA4 LDA ADDR+1
0206           850E   STA STRTPG      ;STORE IN INTERMEDIATE
0208           203EE8 JSR BLANK
020B           20A7E7 JSR TO          ;GET END PAGE
020E           AD1DA4 LDA ADDR+1
0211           850F   STA ENDPGE      ;STORE AS END VALUE
0213           8512   STA RNDVAL     ;START VALUE FOR RANDOM ROUTINE
0215           203EE8 JSR BLANK
0218 RUN       A92A   LDA #'*'        ;'*' INDICATES THAT PROGRAM
021A           207AE9 JSR DSPLAY      ;IS RUNNING
021D           A921   LDA #33         ;FIRST FOR 33 PASSES
021F           850D   STA COUNT       ;WITH CHANGING 00-FF PATTERNS
0221           A000   LDY #0          ;ADDRESS-LOW POINTER
0223           8400   STY TABLE       ;FIRST PATTERN = 00
0225
0225 FCHECK    203602 JSR MAC         ;MAIN TEST SUBROUTINE
0228           C60D   DEC COUNT
022A           D0F9   BNE FCHECK      ;ALL PASSES ?
022C
022C RCHECK    20BB02 JSR RAND        ;FORM A RANDOM NUMBER

022F           A8     TAY             ;SAVE IN Y
0230           203602 JSR MAC
0233           4C2C02 JMP RCHECK      ;ENDLESS LOOP, END OF MAIN PROGRAM
0236
0236           ;  ***  SUBROUTINES  ***
0236
0236 MAC       20D502 JSR SHIFT
0239           204302 JSR WRICOM
023C           20C902 JSR INVERT
023F           204302 JSR WRICOM
0242           60     RTS
0243
0243 WRICOM    A50E   LDA STRTPG
0245           850C   STA POINTH
0247           A000   LDY #0           ;SET UP AS INDIR. ADDR. POINTER
0249           840B   STY POINTL
024B           A20A   LDX #10          ;SET UP AS COUNTER FOR 11
```

```
024D WRITE   B500    LDA TABLE,X
024F         910B    STA (POINTL),Y
0251         CA      DEX
0252         1002    BPL INCY        ;SKIP IF NOT THROUGH
0254         A20A    LDX #10         ;RESTART COUNTER
0256 INCY    C8      INY             ;INCREMENT ADDRESS POINTER
0257         D0F4    BNE WRITE       ;NO PAGE CROSSING
0259         E60C    INC POINTH      ;INCREMENT TO NEXT PAGE
025B         A50F    LDA ENDPGE      ;COMPARE FOR LAST PAGE
025D         C50C    CMP POINTH
025F         B0EC    BCS WRITE       ;NOT YET, CONTINUE
0261
0261         200CE9  JSR KBSCAN      ;EXIT AFTER ANY COMPLETED WRITE
0264                                 ;CYCLE USING THE <ESCAPE> KEY
0264         4C6D02  JMP MOV1
0267         A902    LDA #>RUN       ;ADJUST STACK TO CONTINUE...
0269         48      PHA             ;AT *=$0267 CORRECTLY WITH RTS
026A         A917    LDA #<RUN-1     ;ALSO SHOW '*' AGAIN
026C         48      PHA
026D MOV1    A50E    LDA STRTPG      ;RESET TO START PAGE
026F         850C    STA POINTH
0271         A000    LDY #0
0273         840B    STY POINTL
0275         A20A    LDX #10
0277 COMPAR  B10B    LDA (POINTL),Y
0279         D500    CMP TABLE,X     ;COMPARE MEMORY TO TABLE
027B         D011    BNE ERROR       ;IF NOT EQUAL --> ERROR MESSAGE
027D
027D MOVEON  CA      DEX             ;SAME SEQUENCE AS ABOVE
027E         1002    BPL *+4
0280         A20A    LDX #10
0282         C8      INY
0283         D0F2    BNE COMPAR
0285         E60C    INC POINTH
0287         A50F    LDA ENDPGE
0289         C50C    CMP POINTH
028B         B0EA    BCS COMPAR
028D         60      RTS
028E
028E ERROR   8610    STX SAVX        ;EXIT WITH MESSAGE ON ERROR
0290         8411    STY SAVY        ;SAVE X AND Y
0292         203EE8  JSR BLANK
0295         A031    LDY #$31
0297         20AFE7  JSR KEP         ;SHOW 'MEM FAIL' ON AIM-65 DISPLAY
029A         A50C    LDA POINTH      ;WITH ADDRESS HIGH...
029C         2046EA  JSR NUMA
029F         A411    LDY SAVY        ;AND ADDRESS LOW...
02A1         2046EA  JSR NUMA
02A4         203EE8  JSR BLANK

02A7         A411    LDY SAVY
02A9         B10B    LDA (POINTL),Y   ;AND VALUE OF DEFECTIVE CELL
02AB         2046EA  JSR NUMA
02AE         203EE8  JSR BLANK
02B1         A610    LDX SAVX
02B3         B500    LDA TABLE,X     ;ALSO CORRECT VALUE
02B5         2046EA  JSR NUMA
02B8         4C82E1  JMP AIM65       ;JUMP BACK TO MONITOR
02BB
02BB RAND    A512    LDA RNDVAL      ;RANDOM NUMBER ROUTINE
02BD         0A      ASL A
02BE         2613    ROL RNDVAL+1
02C0         2614    ROL RNDVAL+2
02C2         9002    BCC NXT
```

```
02C4            4987    EOR #$87
02C6 NXT        8512    STA RNDVAL
02C8            60      RTS
02C9
02C9 INVERT     A20A    LDX #10         ;COUNTER FOR 11
02CB VERT       B500    LDA TABLE,X     ;INVERT TABLE CONTENTS
02CD            49FF    EOR #$FF
02CF            9500    STA TABLE,X
02D1            CA      DEX
02D2            10F7    BPL VERT
02D4            60      RTS
02D5
02D5 SHIFT      A209    LDX #9          ;COUNTER FOR 10
02D7 SHIFT1     B500    LDA TABLE,X
02D9            9501    STA TABLE+1,X   ;SHIFT ONE PLACE HIGHER
02DB            CA      DEX
02DC            10F9    BPL SHIFT1
02DE            8400    STY TABLE       ;INSERT NEW CHAR TO
02E0            60      RTS             ;LOWEST POSITION
02E1
02E1
02E1
02E1                    .END
02E1             ERRORS= 0000
```

```
                        **  SYMBOL TABLE  **

ADDR    A41C    AIM65   E182    BLANK   E83E    COMPAR  0277
COUNT   000D    DSPLAY  E97A    ENDPGE  000F    ERROR   028E
FCHECK  0225    FROM    E7A3    INCY    0256    INVERT  02C9
KBSCAN  E90C    KEP     E7AF    MAC     0236    MOV1    026D
MOVEON  027D    NUMA    EA46    NXT     02C6    POINTH  000C
POINTL  000B    RAND    02BB    RCHECK  022C    RNDVAL  0012
RUN     0218    SAVX    0010    SAVY    0011    SHIFT   02D5
SHIFT1  02D7    STRTPG  000E    TABLE   0000    TO      E7A7
VERT    02CB    WRICOM  0243    WRITE   024D
```

****

VIDEO-NEWS. Die tägliche Arbeit am Bildschirm kann eine wahre Freude sein, wenn man dafür das richtige Gerät hat. Ein solches ist das SANYO DATA DISPLAY, Modell DM5912CX mit 18 MHz Bandbreite, 31 cm Bildschirmdiagonale und dem angenehm grünen Leuchtphosphor P31. Es fiel dem Herausgeber ertmals bei einer Vorführung im Hamburger Computer Club auf, als es am PET betrieben wurde und ein hervorragend scharfes und vor allem ruhiges Bild lieferte. Daraufhin wird nun auch hier seit zwei Monaten ein solches Display am AIM 65 betrieben und von Besuchern und Workshopteilnehmern anerkennend begutachtet. Im Vergleich mag man gar nicht mehr an den Betrieb mit modifizierten Fernsehern zurückdenken. Jeder Bildpunkt wird konturenscharf wie eine kleine Perle abgebildet, das Bild steht absolut ruhig, wie man es von den Terminals der großen EDV gewohnt ist. Hochfrequente Störungen beeinflussen es nicht. Die Front des Monitors wird von einer dunkel getönten Plexiglashaube umschlossen, so daß die hellen Zeichen gegen einen tiefdunklen Hintergrund gesehen werden (die hier bevorzugte Betriebsart; invers grüngrundig, dunkle Buchstaben). Zu den Spezifikationen: Maximale Auflösung 1920 Zeichen (80 Zeichen, 24 Zeilen), Eingabesignal Composite Video, Europäische Zeilen- und Bildfrequenz von 15,75 KHz bzw. 50 Hz, 220 Volt Wechselstrom. Info und Bezug: SANYO VIDEO Vertrieb GmbH & Co, Lange Reihe 29, 2000 Hamburg 1, Tel.: 040-24 62 66.

BINÄRDISPLAY-PROGRAMM

Michael Zimmermann, Eberstädter Str. 170, 6102 Pfungstadt

E: The contents of any memory cell is written to the display in binary format, together with address and hex notation.

Dieses Programm bringt den Inhalt beliebiger Speicherzellen in hexadezimaler und in binärer Notierung auf das Display, zusammen mit der ausgewählten Speicheradresse. Die drei Anwenderfunktionstasten werden wie folgt belegt: F2 erlaubt die Anwahl einer Startadresse. Mit F3 geht man zur nächsthöheren über, F1 führt zur nächsttieferen Adresse. Nach jeder Anzeige kehrt man automatisch auf die Kommando-Ebene des Monitors zurück.

```
*=$10C
JMP DOWN                                    4C3502
JMP SHOW                                    4C0002
JMP UP                                      4C2702

;ADRESSEN DER MONITORROUTINEN
ADDIN=$EAAE
BLANK=$E83E
LDAY=$EB58
NUMA=$EA46
OUTDP=$EEFC
NXTADD=$E2CD
WRITAZ=$E2DB
ADDR=$A41C

;ANZEIGEN SPEICHERSTELLE
*=$200
SHOW JSR ADDIN  GET ADDRESS                 20AEEA
SHØ5 JSR BLANK  WRITE A SPACE               203EE8
LDY #0                                      A000
LDA #$1C                                    A91C
JSR LDAY        GET CHAR. UNDER ADDRESS     2058EB
PHA             SAVE CHAR                   48
JSR NUMA        WRITE CHAR IN HEX           2046EA
JSR BLANK       A SPACE                     203EE8
LDX #7          COUNTER FOR 8               A207
SH10 PLA        GET CHAR. BACK              68
ASL A           SHIFT TO GET BIT            0A
PHA             SAVE THE REST               48
LDA #'0'                                    A930
BCC SH20        SKIP WITH '0'               9002
LDA #'1'        TO WRITE '1'                A931
SH20 JSR OUTDP  WRITE BIT                   20FCEE
DEX                                         CA
BPL SH10        UNTIL 8 TIMES DONE          10F1
PLA             ADJUST STACK                68
RTS             RETURN TO MONITOR           60

;ERHOEHEN ADRESSE UND ANZEIGEN
UP JSR BLANK    A SPACE                     203EE8
LDY #1                                      A001
JSR NXTADD      INCREMENT ADDR              20CDE2
JSR WRITAZ      WRITE CURRENT ADDRESS       20DBE2
JMP SHØ5        SHOW THE REST               4C0302
```

```
;VERMINDERN ADRESSE UND ANZEIGEN
DOWN JSR BLANK                              203EE8
LDA ADDR          DECREMENT LAST ADDRESS    AD1CA4
BNE DO1Ø          NO PAGE BORDER            D003
DEC ADDR+1        DECREMENT PAGE-#          CE1DA4
DO1Ø DEC ADDR     ADDRESS LOW               CE1CA4
JSR WRITAZ        WRITE                     20DBE2
JMP SHØ5                                    4C0302
.END
```

****

Michael Zimmermann

Listung der Assembler-Symboltafel (AIM)

E: This utility lists the symbol table of the AIM assembler as in the example at the bottom.

Der Benutzer weist dem Assemblerprogramm einen Speicherbereich für die Symboltafel mit FROM= und TO= zu. Auch nach dem Ende der Assemblierung stehen die benutzten Symbole dort noch zur Verfügung, und zwar mit 6 Bytes für ihren Namen und mit 2 Bytes für ihren Wert. Dieses Dienstprogramm bereitet diese Information für den Druck auf und erstellt einen Abdruck entsprechend dem Beispiel am Schluß des Programms.

```
;ASSEMBLER-VARIABLE
PTRA=$3A
CNTA=$0B

;WORK-AREA
CNT=$36
PTR=$38

;MONITOR LINKS
RSET=$E0BF
BLANK2=$E83B
BLANK=$E83E
NUMA=$EA46
OUTPRI=$EEFC

;PROGRAMM
*=$200 COPY ASSEMBLER VARIABLES TO WORKAREA
LDX #1                                      0200 A2 LDX #02
LØ5 LDA PTRA,X                              0202 B5 LDA 3A,X
STA PTR,X                                   0204 95 STA 38,X
LDA CNTA,X                                  0206 B5 LDA 0B,X
STA CNT,X                                   0208 95 STA 36,X
DEX                                         020A CA DEX
BPL LØ5                                     020B 10 BPL 0202
L10 LDA #$0D   NEW LINE                     020D A9 LDA #0D
JSR OUTPRI                                  020F 20 JSR EEFC
JSR BLANK2                                  0212 20 JSR E83B

;GET SYMBOL TABLE & PRINT IT
LDY #0                                      0215 A0 LDY #00
L15 LDA (PTR),Y  CHAR. FROM NAME            0217 B1 LDA (38),Y
JSR OUTPRI       & COPY                     0219 20 JSR EEFC
INY                                         021C C8 INY
CPY #6           6 CHARACTERS?              021D C0 CPY #06
```

```
BNE L15    NEXT CHAR                          021F D0 BNE 0217
JSR BLANK2 2 SPACES                           0221 20 JSR E83B

;GET ADDRESS & PRINT
LDA (PTR),Y CONVERT TO HEX                    0224 B1 LDA (38),Y
JSR NUMA                                      0226 20 JSR EA46
JSR BLANK                                     0229 20 JSR E83E
INY                                           022C C8 INY
LDA (PTR),Y                                   022D B1 LDA (38),Y
JSR NUMA                                      022F 20 JSR EA46
CLC     ADDITIONSVORBEREITUNG                 0232 18 CLC
LDA #8                                        0233 A9 LDA #08
ADC PTR AUF NAECHSTES SYMBOL EINSTELLEN       0235 65 ADC 38
STA PTR                                       0237 85 STA 38
BCC L20                                       0239 90 BCC 023C
INC PTR+1                                     023B E6 INC 39
L20 DEC CNT+1 RESTZAHL DER SYMBOLE            023D C6 DEC 37
BPL L10       NAECHSTES SYMBOL                023F 10 BPL 020D
DEC CNT                                       0241 C6 DEC 36
BPL L10       NAECHSTES SYMBOL                0243 10 BPL 020D
JMP RSET      ZURUECK ZUM MONITOR             0245 4C JMP E0BF
.END
```

```
<*>=200
<G>/
```

```
  PTRA    00  3A
  CNTA    00  0B
  CNT     00  36
  PTR     00  38
  RSET    E0  BF
  BLANK2  E8  3B
  BLANK   E8  3E
  NUMA    EA  46
  OUTPRI  EE  FC
  L05     02  02
  L10     02  0D
  L15     02  17
  L20     02  3D

  ROCKWELL AIM 65
```

****

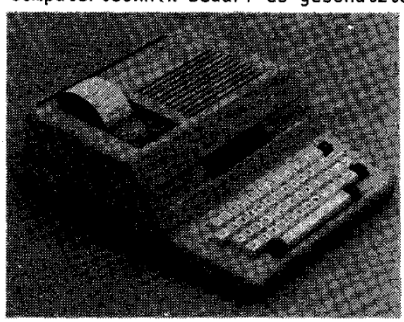
BUCHBESPRECHUNG

Anwendungsbeipiele für den Mikroprozessor 6502 - von Herwig Feichtinger, 96 S. in der RPB-Buchreihe (Nr. 173) des Franzis-Verlages, München, ISBN 3-7723-1731-6. Als FUNKSCHAU-Redakteur für Mikroprozessoren hat Herr Feichtinger viele eigene Arbeiten für die 6502-Systeme veröffentlicht und deren Betrieb damit befördert. Das Buch kommt damit aus kompetenter Feder und darf mit seiner Zielsetzung als gelungen bezeichnet werden. Es geht besonders auf den KIM-1 ein, stellt in knappen Abschnitten die Hardware vor und bringt in etwa 3/4 des Inhaltes Software, Programme in Maschinensprache. Es handelt sich dabei um 'nützliche' Anwendungen, um den Betrieb von Peripherie: Funktionsgenerator, NF-Zähler, Drucker, Oszilloskop, Baudot- und ASCII-Terminals usw.. Die Programme werden grundsätzlich als disassemblierter Ausdruck aus der Maschine oder als Dump präsentiert. Die Kommentare sind dabei knapp gehalten, es steht die lauffähige Problemlösung mehr im Vordergrund als die Unterrichtung in Programmierungstechniken.

Vorstellung des Siemens PC 100

Mit dem PC 100 bietet das Haus Siemens seit dem Frühjahr 1979 einen Komplettcomputer auf der Basis der 6502-CPU an. Komplettgerät heißt dabei: Alle Bestandteile des Systems sind in einem ansprechenden hellen Gehäuse untergebracht. Es ist unmittelbar betriebsbereit, man braucht es nur an die Steckdose anzuschließen.

Der PC 100 ist damit das ideale Gerät für Betreiber, die sich nicht erst lange um den Aufbau oder die Beschaffung eines Netzteiles, eines Gehäuses und um den Zusammenbau der Komponenten kümmern wollen und können, die stattdessen bei Ankunft des Computers und seiner deutschsprachigen Dokumentation gleich mit der Arbeit beginnen wollen oder die ein fertiges System, das etwa wie eine Reiseschreibmaschine aussieht, in die Hände von Mitarbeitern legen wollen. In diesen Fällen und z.B. auch bei der Ausbildung in Schulen, Hochschulen und Betrieben für Mikroprozessorprogrammierung und Computertechnik bedarf es geschützter Geräte.



Unter 'betriebsbereit' versteht das Haus Siemens zugleich auch die Ausrüstung des PC 100 mit einem BASIC-Interpreter in 8 kB Festwertspeicher. In der gut dokumentierten und leicht erlernbaren höheren Programmiersprache BASIC, die zugleich benutzerfreundlich und dialogfähig ist, kann der Anwender seine Aufgaben unmittelbar anfassen, ohne den Umweg über die abstraktere Maschinensprache gehen zu müssen.

Mit diesem Konzept für den PC 100 beschreitet das Haus Siemens einen eigenen unabhängigen Weg zum Anwender, zu einem rasch wachsenden Markt im Bereich zwischen einfachen Mikrocomputersystemen und den Systemen der mittleren Datentechnik. Der PC 100 wird durch weitere Ausbaumodule ergänzt werden.

Nach den Beobachtungen des Herausgebers hat sich das Konzept 'betriebsbereiter Computer mit BASIC' bereits mehrfach in branchenspezifischen Lösungen niedergeschlagen, für die man eine größere Zahl von Geräten bestellte, mit Anwenderprogrammen versah und bei Betreibern aufstellte. Das Konzept überzeugte kürzlich auch bei einem Workshop. Eine Viertelstunde vor seinem Beginn wurden etwa 20 PC 100 ausgepackt und für die Teilnehmer aufgestellt, sie waren alle sofort und pünktlich funktionsbereit.

Zu den Spezifikationen des PC 100: BASIC-Tischrechner in geschlossenem Gehäuse mit Netzteil, alpanumerischer Tastatur (mit 22 Sonderzeichen und 9 Kontroll- und 3 anwenderspezifischen Funktionen), alphanumerischer LED-Leuchtanzeige (20 Zeichen breit, Darstellung der ASCII-Zeichen in 16 Segmenten) und einem alphanumerischen Thermodrucker mit ebenfalls 20 Zeichen pro Zeile und einem Durchsatz von 120 Zeilen je Minute. Zum Netzteil ist zu bemerken, daß es neben den unmittelbar benötigten 5 und 24 Volt auch +12 V und -12 V mit je 0,2 A stabilisiert abgibt, als Vorhaltung für Systemerweiterungen.

Die Tastatur ist robust gebaut, für den Benutzer bedienungsfreundlich schräg angebracht und arbeitet prellfrei. Die System-RESET-Taste ist gegen unbeabsichtigte Betätigung geschützt. Eine Umschalttaste erlaubt den Betrieb mit einem Terminal, eine weitere erlaubt die Programmerprobung im Einzelschritt-Modus.

Die Speicher des PC 100 umfassen 4 kByte statisches RAM (Schreib-/Lesespei-

cher), ein Betriebssystem in 8 kB ROM für die zahlreichen auf der Computerplatine enthaltenen Interfaces (Tastatur, Drucker, LED-Anzeige, Cassettenrecorderanschluß,Betrieb eines TTY-Terminals) sowie in 8 kB ROM das Interpreterprogramm für BASIC. Es bleibt ein freier Sockel für 4 kB ROM/EPROM für die Aufnahme von Anwenderprogrammen oder für das auch lieferbare Assembler ROM.

Der Computer enthält drei Interfacebausteine, von denen zwei für die systemeigenen o.a. Interfaces belegt sind. Das dritte steht mit 2x8 Portleitungen (jede per Programm wahlweise als Ein- oder Ausgang schaltbar) sowie mit 4 Steuerleitungen dem Anwender ungeteilt zur Verfügung. Die Steuerleitungen können dabei per Programm auf die Erkennung aufsteigender oder fallender Impulsflanken geschaltet werden oder dem Handshake-Verkehr mit anderen peripheren Einheiten dienen. Der freie Interfacebaustein enthält ferner zwei Timer/Zähler, die für vielfältige Messungen und auch für die Impulserzeugung mit definierter Länge einsetzbar sind. Diese Interfaceleitungen wie auch die drei Prozessorbusse und die Stromversorgung sind auf Kontaktleisten an der Rückseite herausgeführt und eignen daher für den bequemen Anschluß von Peripherie oder für die Systemerweiterung.

Die Anschlußmöglichkeit für zwei Cassettenrecorder erweist sich als besonders nützlich. Mit handelsüblichen Geräten lassen sich Programme und Daten unter Namensetikett abspeichern und gezielt zurückladen. Die Laufwerksfunktion Start/Stop kann dabei direkt vom Computer gesteuert werden.

Das Betriebsprogramm des PC 100 unterstützt nicht nur die genannten Interfaces samt Magnetbandbetrieb, es ermöglicht selbstverständlich auch das Aufschlagen und Verändern von Speicherinhalten. Die Funktionsauslösung erfolgt dabei mit einem einzigen Tastendruck (Kommandoebene des Betriebsprogrammes). Der Anwender kann außerdem drei Funktionstasten mit eigenen Programmen belegen, die dann ebenso mit einem einzigen Tastendruck zur Ausführung kommen.

Als besonders wertvoll erweist sich der im Betriebssystem enthaltene Text-Editor. Nach Anweisung des Benutzers wird ein Textspeicher angelegt, der der allgemeinen Textbearbeitung, Auskunftssystemen und dergl. dienen kann, vor allem aber der Niederlegung und Korrektur von Programmtexten. Durch Such- und Veränderungsbefehle kann man jede Zeile gezielt ansteuern, Einfügungen und Auslassungen vornehmen sowie irgendwo enthaltene Textketten, z.B. Variable, unter ihrem Namen auffinden und gezielt abändern. Der Inhalt des Textspeichers kann selbstverständlich auch auf Magnetband abgespeichert, von dort zurückgerufen oder von weiteren Magnetbandfiles her gezielt ergänzt werden.

Die Bemühungen des Hauses Siemens, diesem Rechner eine saubere Dokumentation in deutscher Sprache beizustellen, sind sehr zu loben. Für das System stehen bisher folgende Schriften zur Verfügung: Bedienungsanleitung mit Schaltplan, BASIC-Handbuch und Assembler-Handbuch, alle als vorläufige Ausgaben bezeichnet, gleichwohl umfangreich, detailliert und mit klarem Druck gut lesbar. Diese Bücher beschreiben in Hard- und Software alle Aspekte des Rechners, seiner Befehle, Kommandos und sein Betriebssystem.

Wie alle 65xx, so wird auch der PC 100 von dieser Zeitschrift softwaremäßig unterstützt. Zusammen mit den Handbüchern gehört er damit zu den Systemen, für die man als Anwender ständige Anregungen für die eigene Arbeit findet. Er zielt auf Anwendungen in der Meß-, Steuer- und Prüftechnik und auf das Ausbildungswesen, er wendet sich ebenso an den interessierten privaten Betreiber. Sein Vertriebsweg führt für OEM-Kunden über die europäischen Geschäftsstellen des Hauses Siemens, für die privaten Einzelkäufer über die Computershops in den großen Städten. Zentrale Info Siemens AG, Bereich Bauelemente Vertrieb, Postfach 80 17 01, D-8000 München 8o, Tel.: 089-4133-4390.

R. L.

C B M - V I E W

Peter Trübger, Melissenweg 37, 2000 Hamburg 65

Am Beispiel des bekannten VIEW-Programmes soll gezeigt werden, wie man die CHR-GET-Routine modifizieren kann, um den BASIC-Befehlssatz zu erweitern. Die CHR-GET-Routine steht im RAM. Sie inkrementiert den Pointer in den BASIC-Text, liest den Inhalt in den Akku und untersucht auf ':' und SPACE. Der Interpreter springt diese Routine mit JSR an.

So wie in VIEW der IRQ-Vektor zunächst auf das eigene Maschinenprogramm gerichtet wird, an die sich später die KEY-Routine anschließt, kann man einen Keil auch in die CHR-Get-Routine setzen, so daß bestimmte Eingaben abgefragt werden können. Das VIEW sei hier nochmals als besonders einfaches Beispiel vorgeführt.

Es ist zweckmäßig, für viele neue BASIC-Befehle eine Vortaste wie z.B. das '>' zu wählen, damit sich im RUN-Modus nicht die erweiterte CHR-GET-Routine negativ auf die Rechenzeit auswirkt. Mit den vorgeschlagenen Umstellungen könnte man Programme wie z.B. VIEW, RENUMBER, TRACE, REPEAT usw. symbolisch mit 'V', 'N10', 'T', 'R' aufrufen.

```
VIEW 2
033A 78          1 VIEW SEI          DIESE BEFEHLSSEQUENZ SCHALTET DEN
033B A5 90       2 LDA $90           IRQ-VEKTOR VON $E62E AUF $0349 UM
033D 49 67       3 EOR #$67          UND UMGEKEHRT.
033F 85 90       4 STA $90
0341 A5 91       5 LDA $91
0343 49 E5       6 EOR #$E5
0345 85 91       7 STA $91
0347 58          8 CLI
0348 60          9 RTS

0349 A2 00      10 LDX #$00          SCHREIBT VOR DEM TASTENINTERRUPT
034B BD 00 01   11 LOOP LDA $0100,X EINE PAGE AUF DEN BILDSCHIRM
034E 9D 00 80   12 STA $8000,X       (ADRESSEN AB $8000)
0351 E8         13 INX
0352 D0 F7      14 BNE LOOP
0354 4C 2E E6   15 JMP $E62E         WEITER IM BETRIEBSSYSTEM

0357 A9 4C      16 KEIL LDA #$4C     BRINGE ZUNÄCHST EINEN JMP-BEFEHL
0359 85 79      17 STA $79           NACH KØ5 AN DEN ANFANG DER CHR-GET-
035B A9 64      18 LDA #$64          ROUTINE
035D 85 7A      19 STA $7A
035F A9 03      20 LDA #$03
0361 85 7B      21 STA $7B
0363 60         22 END RTS

0364 C9 3E      23 KØ5 CMP #62       IST ES EIN '>"?
0366 D0 05      24 BNE END2          NEIN
0368 CD 00 02   25 CMP 512           STEHT ES IN DER ERSTEN SPALTE?
036B F007       26 BEQ PRG           JA
036D C9 3A      27 END2 CMP #58      DOPPELPUNKT? ANSCHLUSS AN DIE DURCH JMP
036F B0 F2      28 BCS END           VERÄNDERTE CHR-GET-ROUTINE
0371 4C 7D 00   29 JMP 125           DORT FORTFAHREN

0374 20 3A 03   30 JSR VIEW          SCHALTET VIEW AN UND AB
0377 4C 89 C3   31 JMP $C389         READY-MELDUNG
037A            32 .END
```

INITIALISIEREN:SYS 855, D.H. MIT DER ROUTINE KEIL, DANN >, RETURN.
VERÄNDERN DER AUFGESCHLAGENEN SEITE MIT POKE 845,X.

DAS PROGRAMM IN DATA-STATEMENTS:

```
 5 FOR J=826 TO 889:READ X:POKE J,X:NEXT:END
10 DATA 120,165,144,73,103,133,144,165,145,73,229,133,145,88,96,162,0,189,0
20 DATA 1,157,0,128,232,208,247,76,46,230,169,76,133,121,169,100,133,122
30 DATA 169,3,133,123,96,201,62,208,5,205,0,2,240,7,201,58,176,242,76,125,0
40 DATA 32,58,3,76,137,195
```

(Anmerkung des Herausgebers: Für die in Heft 9 veröffentlichten Programme TRACE und VIEW gibt es zwar Vorbilder in amerikanischen Zeitschriften, die abgedruckten Programme stellen jedoch eigenständige Bearbeitungen und Erweiterungen durch die Autoren dar.)

****

BERECHNETES GOTO FÜR CBM-COMPUTER

Johann Leitner, Kurfürstenstraße 2, 6000 Frankfurt 90

Das Programm RESTORE LINE NUMBER von Herrn Kornnagel (Heft 9) gab Anlaß, analoges für das GOTO zu versuchen (s.u.). Außer als Ersatz des Befehles ON X GOTO fallen dem Autor hierfür allerdings keine plausiblen Anwendungen ein. Vielleicht gibt es hierzu Vorschläge. Ähnliches müßte auch mit dem Befehl GOSUB möglich sein.

```
;BERECHNETES GOTO FÜR CBM-COMPUTER
;SYNTAX DES AUFRUFES:SYS 826X
;X IST KONSTANTE, VARIABLE ODER FORMEL
;ADRESSEN DES BETRIEBSSYSTEMS:

FRMNUM =$CC8B        ;$CCA4 BEIM PET
GETADR =$D6D2        ;$D6D0 BEIM PET
GTO    =$C7AD        ;50333 BEIM PET
SUCHZE =$C530        ;$C526 BEIM PET
TXTTAB =$28          ;$7A   BEIM PET
UERROR =$C7EB        ;$C7DB BEIM PET

*=826
826 $033A 20 8B CC  START  JSR FRMNUM       ;BERECHNE X
829  033D 20 D2 D6         JSR GETADR       ;WANDLE UM
832  0340 A5 28            LDA TXTTAB
834  0342 A6 29            LDX TXTTAB+1
836  0344 20 30 C5         JSR SUCHZE       ;SUCHE ZEILE
839  0347 90 03            BCC ERROR        ;NICHT GEFUNDEN
841  0349 4C B0 C7         JMP GTO+3
844  034C 4C EB C7  ERROR  JMP UERROR       ;UNDEFD STATEMENT
847  034F                  .END
```

****

APPLE USER GROUP EUROPE - JAHRESTREFFEN AM 26./27. JAN. 1980

Vereinssitzung am 26.1. sowie Besprechung der einzelnen Arbeitsgruppen (PASCAL, Amateurfunk, Schulunterricht, formale Sprachen). 27.1.1980 Gedanken- und Softwareaustausch. Anmeldungen/Info: Vors. W. Dederichs, Postfach 4068, 4320 Hattingen, Tel. 02324-67412.

P R I N T (AIM)

Mathias Helm, Adolf-Ey-Straße 2a, 3392 Clausthal-Zellerfeld

E: The AIM printer writes 60 characters per line from the text buffer. This is enabled by a trick: Three paper straps are produced which may be glued together at their sides to render these long lines.

Dieses Programm dient zum Ausdrucken aus dem Textspeicher des Editors in 3 Streifen, die nebeneinander geklebt das Bild eines 60-spaltigen Druckes ergeben. Insbesondere bei Quellenprogrammen erhält man auf diesem Wege eine wesentlich bessere Übersicht.

Zur Programmbedienung: Im Editor gibt man wie üblich das LIST-Kommando. Sobald der AIM mit OUT= nach der gewünschten Ausgabeeinheit fragt, betätigt man die Tasten ESCAPE und F1 (Anwenderfunktion).

```
;SYMBOLE
OUTALL=$E9BC
INALL=$E993
SAVNOW=$F934
CRLOW=$EA13
ENDERR=$FA5C
RESNOW=$F8D0
OUTPRI=$F000
DONE=$E790
ATBOT=$F8E9
UP1=$F713
PRIFLG=$A411
COUNT=$A419
NOWLN=$DF

*=$70
MES
.BYT 'PAPIERSPAREN J/N?'                    <M>=0070 50 41 50 49
SAVE *=*+1 SAVEEX *=*+1                     < > 0074 45 52 53 50
SAVFLG *=*+1                                < > 0078 41 52 45 4E
SAVCNT *=*+1                                < > 007C 20 4A 2F 4E
TAB1 *=*+1                                  < > 0080 3F
TAB2 *=*+1
*=$10C ;USERF. 1
JMP START                                   010C 4C JMP 0F28

*=$F28
START LDX #0                                0F28 A2 LDX #00
M1 LDA MESG,X                               0F2A B5 LDA 60,X
JSR OUTALL                                  0F2C 20 JSR E9BC
INX                                         0F2F E8 INX
CPX #17                                     0F30 E0 CPX #11
BNE M1                                      0F32 D0 BNE 0F2A
LDA #0 ;SAVE=0 ENTSPRICHT                   0F34 A9 LDA #00
STA SAVE ;"NICHT PAPIER SPAREN"             0F36 85 STA 71
STA TAB1 ;INITIALISIEREN DES LINKEN         0F38 85 STA 75
M2 JSR INALL ;TABULATORS                    0F3A 20 JSR E993
CMP #'N'                                    0F3D C9 CMP #4E
BEQ M4                                      0F3F F0 BEQ 0F4C
CMP #'J'                                    0F41 C9 CMP #4A
BEQ M3                                      0F43 F0 BEQ 0F48
JMP M2                                      0F45 4C JMP 0F3A
```

```
M3 LDA #1                                  0F48 A9 LDA #01
STA SAVE ;"PAPIER SPAREN"                  0F4A 85 STA 71
M4 JSR SAVNOW ;NOWLN RETTEN                0F4C 20 JSR F934
LDA COUNT                                  0F4F AD LDA A419
STA SAVCNT ;COUNT RETTEN                   0F52 85 STA 74
JSR CRLOW ;CR/LF DISPLAY/PRINTER           0F54 20 JSR EA13
LDA PRIFLG                                 0F57 AD LDA A411
STA SAVFLG ;PRIFLG RETTEN                  0F5A 85 STA 73
LDA #$80                                   0F5C A9 LDA #80
STA PRIFLG ;DRUCKER EIN                    0F5E 8D STA A411
LDA #20                                    0F61 A9 LDA #14
STA TAB2 ;TABULATOR RECHTS SETZEN          0F63 85 STA 76
JSR PRPASS                                 0F65 20 JSR 0F76
JSR PRPASS                                 0F68 20 JSR 0F76
JSR PRPASS                                 0F6B 20 JSR 0F76
LDA SAVFLG                                 0F6E A5 LDA 73
STA PRIFLG ;PRINTERFLAG WIEDERHERSTELLEN   0F70 8D STA A411
JMP ENDERR ;RUECKSPRUNG IN DEN EDITOR      0F73 4C JMP FA5C

PRPASS JSR RESNOW ;NOWLN=STARTZEILE        0F76 20 JSR F8D0
LDA SAVCNT                                 0F79 A5 LDA 74
STA COUNT ;ZAHL D. AUSZUDRUCKENDEN ZEILEN  0F7B 8D STA A419
LDA #0                                     0F7E A9 LDA #00
STA SAVEEX ;BISHER JEDE ZEILE KUERZER      0F80 85 STA 72
M5 LDY #0                                  0F82 A0 LDY #00
LDA TAB1                                   0F84 A5 LDA 75
BEQ M7 ;WIR SIND IM LINKEN STREIFEN        0F86 F0 BEQ 0F97
M6 LDA (NOWLN),Y ;LADEN EINES ZEICHENS     0F88 B1 LDA (DF),Y
BEQ M12 ;00= EDITORENDE                    0F8A F0 BEQ 0FCE
CMP #$0D ;ZEILENENDE?                      0F8C C9 CMP #0D
BEQ M10                                    0F8E F0 BEQ 0FC0
INY                                        0F90 C8 INY
CPY TAB1 ;LINKER TABULATOR ERREICHT?       0F91 C4 CPY 75
BNE M6                                     0F93 D0 BNE 0F88
LDY TAB1                                   0F95 A4 LDY 75
M7 LDA (NOWLN),Y                           0F97 B1 LDA (DF),Y
BEQ M12                                    0F99 F0 BEQ 0FCE
CMP #$0D                                   0F9B C9 CMP #0D
BEQ M8                                     0F9D F0 BEQ 0FAB
JSR OUTPRI ;LADEN IN PRIBUF                0F9F 20 JSR F000
LDA #1 ;IN DIESEM STREIFEN AB              0FA2 A9 LDA #01
STA SAVEEX ;JETZT ZU DRUCKENDE ZEICHEN     0FA4 85 STA 72
INY                                        0FA6 C8 INY
CPY TAB2 ;RECHTER TABULATOR                0FA7 C4 CPY 76
BNE M7                                     0FA9 D0 BNE 0F97
M8 JSR CRLOW ;AUSDRUCK VON PRIBUF          0FAB 20 JSR EA13
M9 JSR DONE ;GENUEGEND ZEILEN?             0FAE 20 JSR E790
BEQ M12                                    0FB1 F0 BEQ 0FCE
LDY #0 ;ERSATZROUTINE FÜR "UPNO"           0FB3 A0 LDY #00
JSR ATBOT                                  0FB5 20 JSR F8E9
BCS M12                                    0FB8 B0 BCS 0FCE
JSR UP1                                    0FBA 20 JSR F713
JMP M5 ;NAECHSTE ZEILE                     0FBD 4C JMP 0F82
M10 LDA SAVE ;SOLL GESPART WERDEN?         0FC0 LDA 71
BNE M11                                    0FC2 D0 BNE 0FC7
JMP M8                                     0FC4 4C JMP 0FAB
```

```
M11 LDA SAVEEX ;IN DIESEM STREIFEN SCHON   OFC7 A5 LDA 72
BEQ M9 ;ZU DRUCKENDE ZEICHEN GEWESEN?      OFC9 F0 BEQ OFAE
JMP M8                                     OFCB 4C JMP OFAB
M12 LDA TAB2 ;TABULATOREN FUER             OFCE A5 LDA 76
STA TAB1     ;NAECHSTEN STREIFEN SETZEN    OFD0 85 STA 75
CLC                                        OFD2 18 CLC
ADC #20                                    OFD3 69 ADC #14
STA TAB2                                   OFD5 85 STA 76
RTS                                        OFD7 60 RTS
.END
*************************************************************
```

Interrupt-Demonstrationsprogramme für das VIA 6522

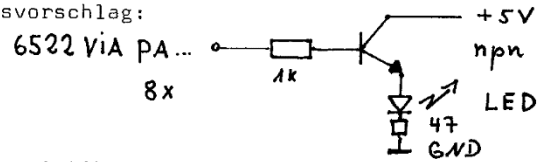
E: Two examples show the mechanism of interrupt programming, a foreground program with initializing included and a background interrupt program which is only activated when the interrupt occurs. The first simple example enables interrupt from Timer T1, the second asynchronously outputs data to an ASCII device in the handshake mode.

In Heft 7 (Das VIA 6522) haben wir die vielfältigen Einsatzmöglichkeiten des Versatile Interface Adapter vorgestellt. Zur Bequemlichkeit der PET-Betreiber sind die Registeradressen ihres Interfacebausteines in hexadezimaler und in dezimaler Notierung (für PEEK und POKE in BASIC) nachzutragen:

HEXADEZIMAL	DEZIMAL	FUNKTION AUF DEM VIA DES PET
E840	59456	ORB, E/A-REGISTER PORT B, SYSTEMPORT
E841	59457	ORA, E/A-REGISTER PORT A, USER PORT
E842	59458	DDRB, DATENRICHTUNGSREGISTER PORT B
E843	59459	DDRA, DATENRICHTUNGSREGISTER PORT A
E844	59460	T1C-L TIMER 1 COUNTER LOW, LESEN ZÄHLER LOW, SCHREIBEN VORSPEICHER LOW
E845	59461	T1C-H TIMER 1 COUNTER HIGH, STARTEN TIMER DURCH SCHREIBEN, T1L-L WIRD NACH T1C-L ÜBERTRAGEN. LESEN T1C-H
E846	59462	T1L-L SCHREIBEN ODER LESEN DES VORSPEICHERS LOW
E847	59463	T1L-H SCHREIBEN ODER LESEN DES VORSPEICHERS HIGH
E848	59464	T2C-L LESEN ZÄHLER LOW, SCHREIBEN VORSPEICHER LOW
E849	59465	T2C-H LESEN UND SCHREIBEN ZÄHLER HIGH, TIMERSTART
E84A	59466	SCHIEBEREGISTER SR
E84B	59467	HILFSREGISTER ACR FÜR SCHALTUNG DER PORTS, SONDERFUNKT.
E84C	59468	PCR, PERIPHERAL CONTROL REGISTER FÜR STEUERLEITUNGEN
E84D	59469	IFR, INTERRUPT-ANZEIGEREGISTER
E84E	59470	IER, INTERRUPT ENABLE REGISTER, INTERRUPTZULASSUNG
E84F	59471	ORA, REGISTER PORT A OHNE EINFLUSS AUF HANDSHAKE

Wir erinnern uns, daß die digitalen Funktionen des Interfacebausteines durch initialisierendes Einschreiben in eines oder mehrere seiner 16 Register geschaltet werden. Unsere Programme benutzen das Anwender-VIA des AIM 65/PC100. Durch Ändern der Registeradressen und Interruptvektoren können sie für jedes andere System mit einem VIA adaptiert werden. Man achte aber darauf, welche Funktionen bereits evtl. für den Systembetrieb belegt sind.

Im ersten Beispiel werden einige oder alle Ausgangspins des User-Ports A, nämlich PAØ bis PA7, für Demonstrationszwecke auf die Basis eines Transistors gelegt, der eine Leuchtdiode mit etwa 40 mA antreibt. Das Programm erzeugt durch Inkrementieren des Portinhaltes UDRA in Adresse A001 ein Lauflicht. Schaltungsvorschlag:



In früheren Artikeln wurde bereits darauf hingewiesen, daß Interruptprogrammierung drei Programmteile erfordert,

das Initialisieren des Systems für Interruptbetrieb,

ein Vordergrundprogramm, das nützliche Tätigkeiten ausführt oder nur in einer Warteschleife verharrt und

ein selbständiges Interruptprogramm, das die Antwort des Prozessors auf das Interrupt-Ereignis enthält.

Es geht also nicht an, wie es einige Anfänger schon versuchten, vom Vordergrundprogramm direkt in das Interrupt-Programm überzuleiten. Durch dessen Schlußbefehl RTI wird der Programmablauf in die Wüste geschickt, weil auf dem Stack noch keine definierte Rückkehradresse abgelegt war.

Unser Initialisierungsprogramm schaltet Port A auf Ausgang, indem das Datenrichtungsregister mit 'FF' beschrieben wird. Der Portinhalt wird dann zu '00' gesetzt, für Timer 1 wird im Interrupt Enable Register UIER Interrupt ermöglicht. Timer 1 wird im ACR auf free running geschaltet, nach jedem Nulldurchgang erfolgt also ein Nachladen aus den Vorspeicher-Latches UT1L-L und UT1L-H. Der Interruptvektor ist auf die eigene Interrupt-Routine gerichtet.

Das anschließende Vordergrundprogramm fragt die Tastatur in einer endlosen Schleife ab. Nur die Tasten 'A' und 'B' sind zur Steuerung zugelassen. A erhöht die Zeit zwischen den Interrupts, B erniedrigt sie.

Das Interruptprogramm erhöht nur den Inhalt des Ports A und löscht durch Lesen des Timers das Interrupt-Flag, so daß später ein neuer Interrupt eintreten kann. Es scheint aber der Hinweis wichtig, daß dieser Programmteil keinen Gebrauch von den Indexregistern X und Y macht, so daß dort auch keine Maßnahmen zu deren Rettung und Wiederherstellung getroffen werden müssen. Eine solche Maßnahme erfolgt dort nur für den Akku mit PHA und PLA.

```
UDDRA=$A003    PORT
UDRA=$A001     DATENRICHTUNG
UIER=$A00E     INTERRUPT ENABLE
UACR=$A00B     HILFSREGISTER
```

```
UT1CL=$A004 T1 COUNTER LOW
UT1LL=$A006 TIMER LATCH LOW
UT1CH=$A005 COUNTER 1 HIGH
IRQV4=$A400 INTERR.-VEKTOR
GETKEY=$EC40 TASTATURABFRAGE
UT1LH=$A007 T1 LATCH HIGH
```

```
*=A00                                   ==0A00
;INITIALISIERUNG
LDA #$FF        DATENRICHTUNG           A9FF
STA UDDRA       AUSGANG                 8D03A0
LDA #0          PORT DEFINIERT '0'      A900
STA UDRA                                8D0FA0
LDA #$C0        ENABLE INTERRUPT        A9C0
STA UIER        FUER T1                 8D0EA0
LDA #$40        FREEE RUNNING MODE F. T1 A940
                                        ==0A11
STA UACR        HILFSREGISTER           8D0BA0
LDA #$FF        TIMERVORGABE UND        A9FF
STA UT1LL                               8D06A0
STA UT1CH       TIMERSTART              8D05A0
LDA #<INTER     IRQ-VEKTOR SETZEN       A949
STA IRQV4                               8D00A4
LDA #>INTER                             A90A
STA IRQV4+1                             8D01A4
CLI             CLEAR INTEERUPT FLAG    58
VORDER                                  ==0A27
JSR GETKEY      TASTATURABFRAGE         2040EC
CMP #'A'        TASTE 'A'?              C941
BEQ ERH         ERHOEHEN                F007
CMP #'B'        B?                      C942
BEQ ERNIE       ERNIEDRIGEN             F00D
JMP VORDER      ABFRAGESCHLEIFE         4C270A
ERH                                     ==0A35
LDX UT1LH       LATCH 1 ERHOEHEN        AE07A0
INX                                     E8
STX UT1LH                               8E07A0
JMP VORDER      SCHLEIFE                4C270A
ERNIE                                   ==0A3F
LDX UT1LH       WIE VOR                 AE07A0
DEX             ./. 1                   CA
STX UT1LH                               8E07A0
JMP VORDER      SCHLEIFE                4C270A
INTER                                   ==0A49
PHA             RETTE A                 48
LDA UT1CL       INTERRUPT FLAG LOESCHEN AD04A0
INC UDRA        PORTINHALT ERHOHEN      EE0FA0
PLA             RESTORE A               68
RTI             RÜCKEHR VOM INTERRUPT   40
.END
```

Ein zweites Interrupt-Demonstrationsprogramm dient der asynchronen Datenübertragung vom AIM zu einer ASCII-Ausgabeeinheit mit Handshake. In der Initialisierungsphase wird wie folgt eingerichtet: Datenrichtungsregister A für Ausgabe, die Steuerleitung CA2 wird im PCR (Peripheral Control Register) auf Handshake-Betrieb geschaltet. Ihr Signal geht auf '0', sobald neue Daten in den Port geschrieben worden sind und kehrt mit einer aktiven Flanke an CA1 auf '1' zurück. CA1 nimmt das Quittungssignal des Empfängers entgegen und reagiert auf abfallende Impulsflanke. Dieses Signal an CA1 darf asynchron eintreffen. In einem Versuchsaufbau kann man es durch ein handgetaktetes Flip-Flop erzeugen. Wie im vorigen Beispiel mag man

das an Port A erzeugte parallele Ausgangssignal auf ein LED-Interface geben.

Zur Initialisierung gehört die Einrichtung des Interruptvektors ebenso wie das Interrupt-Enable für den Pin CA1. Das Vordergrundprogramm startet die Datenübertragung, indem es schon das erste Zeichen aus dem Text sendet.

Im Interruptprogramm werden A und X gerettet und hangelt sich durch den zu sendenden Text, den es auf Port A zur Ausgabe bringt. Sobald ein Semikolon (;) angetroffen wird, ist die Übertragung abzuschließen (man kann beliebige andere Steuerzeichen geben). Nach Justierung des Stackpointers und Ausschaltung der Interruptmöglichkeit für CA1 kehrt man mit BRK zum Monitorprogramm zurück.

```
;SYMBOLE
PORTA=$A001    ANWENDER-VIA
PORTAD=$A003   DATENRICHTUNG
ACR=$A00B      HILFSREGISTER
PCR=$A00C      STEUERREGISTER
IER=$A00E      INTERRUPT ENABLE REG
IRQV4=$A400    INTERRUPT VEKTOR

*=$900
;INITIALISIERUNG
LDA #$FF                                  0900 A9 LDA #FF
STA PORTAD     DATENRICHTUNGSREGISTER     0902 8D STA A003
LDA #%1000     CA2-HANDSHAKE MODE         0905 A9 LDA #08
STA ACR        HILFSREGISTER              0907 8D STA A00C
LDA #%10000010 CA1 INTERR. ENABLE         090A A9 LDA #82
STA IER                                   090C 8D STA A00E
LDA #0                                    090F A9 LDA #00
STA MEM        HILSZELLE INITIALISIEREN   0911 8D STA 0928
LDA TEXT       1. ZEICHEN SENDEN          0914 AD LDA 0929
STA PORTA                                 0917 8D STA A001
LDA #<INTER    INTERRUPT-VEKTOR SETZEN    091A A9 LDA #31
STA IRQV4                                 091C 8D STA A400
LDA #>INTER                               091F A9 LDA #09
STA IRQV4+1                               0921 8D STA A401
CLI            ENABLE INTERRUPT           0924 58 CLI
SCHLEI JMP SCHLEI  WARTESCHLEIFE          0925 4C JMP 0925

MEM .BYT 00    HILFSZELLE                 <M>=0928 00
TEXT .BYT 'MESSAGE'                       <M>=0929 4D 45 53 53 41 47 45

;INTERRUPTPROGRAMM
INTER PHA      RETTE A UND X              0931 48  PHA
TXA                                       0932 8A  TXA
PHA                                       0933 48  PHA
INC MEM        HOCHZAEHLEN                0934 EE INC 0928
LDX MEM        FUER INDIZIERUNG LADEN     0937 AE LDX 0928
LDA TEXT,X     NAECHSTES ZEICHEN LADEN    093A BD LDA 0929,X
CMP #';'       SEMIKOLON?                 093D C9 CMP #3B
BEQ AUS        JA, ABBRECHEN              093F F0 BEQ 0948
STA PORTA      SENDEN AM PORT             0941 8D STA A001
PLA            A UND X WIEDERHERSTELLEN   0944 68 PLA
TAX                                       0945 AA TAX
PLA            AKKU                       0946 68 PLA
RTI            ENDE INTERRUPT             0947 40 RTI
```

```
AUS SAUBERES SYSTEM UEBERGEBEN          ==0948
PLA          2MAL FÜR X UND A,          0948 68 PLA
PLA                                     0949 68 PLA
PLA          3MAL WEGEN INTERRUPT       094A 68 PLA
PLA                                     094B 68 PLA
PLA                                     094C 68 PLA
LDA #2       CA1-DISABLE                094D A9 LDA #02
STA IER      IM INTERR. ENABLE REG.     094F 8D STA A00E
LDA #0       CB2-HANDSHAKE ZURUECKNEHMEN 0952 A9 LDA #00
STA PCR                                 0954 8D STA A00C
STA PORTA    AUSGANGSSIGNAL =0          0957 8D STA A001
BRK          RÜCKGABE AN MONITOR        095A 00 BRK
.END
```

R.L.

****

Buchbesprechungen

Taschenrechner + Mikrocomputer Jahrbuch 1980. Herausgeber Dr.-Ing. Harald Schumny, Vieweg Verlag Braunschweig/Wiesbaden 1979, 258 S.,ISBN 3-528-04154-4, DM 22,80. Dem Herausgeber, den weiteren 23 Autoren und dem Verlag ist ein wirklich nützliches Buch gelungen. Etwa drei Viertel des Inhaltes (Fachteil) bestehen aus abgeschlossenen Aufsätzen zu Taschenrechnern, Mikrocomputern und Speichern. Am Schlusse des Buches findet man eine umfangreiche und sehr detaillierte Produkt-, Leistungs- und Ausstattungsübersicht für Taschenrechner und Mikros/Minis sowie einen großen Anschriftenteil, der einen hohen Eigenwert hat. Der besonders interessierende Fachteil Mikroprozessoren enthält allein siebzehn Aufsätze und ist thematisch außerordentlich vielseitig. Er geht auf die Hardware und Software zahlreicher verbreiteter Mikroprozessoren ein und stellt vor allem Schnittstellen und Signalbelegungen heraus - eine sehr nützliche Referenz. Dem Herausgeber ist für die besondere Mühe zu danken, mit der er viele der von den Autoren benutzten Fachausdrücke aus dem Text herausgezogen und in einem grafisch abgesetzten Kästchen mit Erklärungen versehen hat. Er erreicht dadurch eine auch für den Anfänger verständliche Darstellung.

The Best of MICRO, Volume 2. Herausgegeben von Robert M. Tripp, Chelmsford, USA 1979, 224 S. DM 32,-. Das Buch ist eine Zusammenfassung der Hefte 7-12 von 'MICRO', der amerikanischen Fachzeitschrift für 6502. Der Inhalt ist in vier Abschnitte etwa gleichen Umfanges gegliedert worden: AIM/SYM/KIM, APPLE II, PET, General und bildet damit eine kompakte Fundgrube für jeden Systembetreiber. Es ist vorwiegend ein Softwarebuch, frei von Anzeigen (und damit angenehm zu lesen) und mit Schwerpunkt bei den nützlichen Systemprogrammen und bei der Erklärung der Systeme. Im ersten Abschnitt fällt auf, daß SYM-1 und KIM-1 wesentlich stärker bedacht sind als der AIM 65 (Uhrenprogramm und Anwenderbericht), der auf dieser Seite des Ozeans offensichtlich mehr beachtet wird. Der Herausgeber ist sicher, daß er dieses Buch noch oft für seine Arbeiten heranziehen wird. Bezug: Micro-Shop-Bodensee, M. und R. Nedela, Postfach 1122, 7778 Markdorf 1, Tel.07544-3575.

****

Kleinanzeigen der Leser

DOPPEL-FLOPPY FÜR PET ODER CBM, 2×80 KB, BETRIEBEN ÜBER IEC-BUS, INKL. DIVERSER PROGRAMME ZU DM 2.500,- ABZUGEBEN. PETER TRÜBGER, MELISSENWEG 37, 2000 HAMBURG 65, TEL.: 040-601 50 69.

T V I N T (AIM)

Mathias Helm, Adolf-Ey-Straße 2a, 3392 Clausthal-Zellerfeld

E: The DILINK is fully initialized by an assembler-generated tape for a TV-display (monitor). The DELETE key is recognized for the display too if used within the 2o first characters within a line.

```
;PROGRAMM DIENT ZUM BETREIBEN EINES BILDSCHIRMTERMINALS
;BEI AKTIVER AIM-TASTATUR. INNERHALB DER ERSTEN 20 SPALTEN
;ERKENNT DAS PROGRAMM 'DEL' UND GIBT AN DAS TERMINAL EIN 'BS'.
;ES KANN NUR AUF BAND ASSEMBLIERT WERDEN UND VON DORT ALS
;MASCHINENPROGRAMM GELADEN WERDEN.
;DAS PROGRAMM WURDE ENTWICKLET AUS DER ROUTINE
;IN 65XX MICRO MAG NR. 7, SEITE 35

*=$FD8                                  0FD8 29 AND #7F
START AND #$7F                          0FDA C9 CMP #0D
CMP #$0D                                0FDC F0 BEQ 0FE5
BEQ M5                                  0FDE C9 CMP #20
CMP #$20 ;IN ERSTEN 20 SPALTEN NACH 'SP'
BEQ M6                                  0FE0 F0 BEQ 0FEF
JMP $EEA8                               0FE2 4C JMP EEA8
M5 LDA #$0A                             0FE5 A9 LDA #0A
JSR $EEA8                               0FE7 20 JSR EEA8
LDA #&0D                                0FEA A9 LDA #0D
JMP $EEA8                               0FEC 4C JMP EEA8

M6 LDA $A482                            0FEF AD LDA A482
CMP #$FD ;IN $A482 (DRB2) BEI DEL       0FF2 C9 CMP #FD
BEQ M7                                  0FF4 F0 BEQ 0FFB
LDA #$20 ;ES WAR NUR 'SP'               0FF6 A9 LDA #20
JMP $EEA8                               0FF8 4C JMP EEA8
M7 LDA #$08 ;'BS'                       0FFB A9 LDA #08
JMP $EEA8                               0FFD 4C JMP EEA8

*=$A406 ;AB HIER VEKTOREN DER INITIALISIERUNG
.WOR START                              <M>=A406 D8 0F
*=$A417
.DBY $0147 ;DIE FUER MEIN TERMINAL HOECHSTMOEGLICHE  <M>=A417 01 47
.END       ;BAUDRATE - ANWENDERHANDBUCH SEITE 9-29 BEACHTEN !!!
```

Anmerkungen des Herausgebers: Dieses von Herrn Helm dankenswerterweise in Assembler geschriebene Programm initialisiert das DILINK, den Sprungvektor für Ausgabe auf dem Bildschirm bei weiterhin aktiver AIM-Tastatur, auf eine neue Weise. Es war bisher kaum bekannt, daß ein vom Assembler mit OBJ-OUT=T direkt auf Magnetband erzeugtes Maschinenprogramm beim Laden die diversen mit Sternadresse zugewiesenen Konstanten ebenso erzeugt, wie der Assembler selbst. Auf diesem Wege läßt sich der AIM für viele Zwecke vom Tonband direkt initialisieren, ohne daß man ein Initialisierungsprogramm extra starten muß. Der AIM 65/PC100 wird damit auch ohne zusätzlichen Festwertspeicher benutzerfreundlicher und kann nach kurzem 'start up' vom Magnetband weniger geübten Benutzern zum Betrieb überlassen werden.

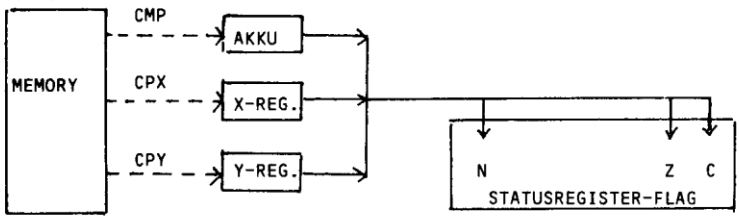
Mit der Erkennung der DELETE-Taste auch für den Bildschirm sind weitere Möglichkeiten für die Cursor-Steuerung aufgezeigt. Es ist nur eine Frage der Zeit und des Fleißes, in der Zwischenroutine der Ausgabe beliebige Sonderzeichen zu erkennen und zur Ausführung auf dem Bildschirm gelangen zu lassen.

Ein Leitfaden für die Programmierung (Teil 5)

E: This chapter covers the compare-instrucions. The mechanism is explained of how to get decisions like in FORTRAN when working on data fields and to filter off unallowed ranges of incoming data. Execution of user commands is vectored by a jump table.

7.6 Vergleichsbefehle

Der Inhalt der drei CPU-Register Akkumulator, Indexregister X und Y kann mit dem Inhalt beliebiger Speicherzellen verglichen werden. Die zur Verfügung stehenden Maschinenbefehle heißen CMP, CPX und CPY.



Die Vergleichsbefehle arbeiten immer rein binär, sie sind also nicht von der etwa eingeschalteten dezimalen Betriebsweise des Prozessors abhängig. Als Ausdruck des Vergleichsergebnisses setzen sie die in der Grafik bezeichneten Flags im Status. Dieser wird beeinflußt, als wenn eine Subtraktion Register minus Speicherzelleninhalt stattgefunden hätte. Weder der Inhalt des herangezogenen Registers noch der Speicherzelleninhalt werden verändert. - Ein Vergleich ist also nur eine 'gedachte' Subtraktion.

Auf einen Vergleich läßt man einen bedingten Verzweigungsbefehl folgen, ein BRANCH. Die Darstellungen im Programmierhandbuch enthalten etwas verwirrenden Ballast. Nach einem Vergleich interessieren die statusmäßigen Ergebnisse nur wie folgt:

Registerinhalt ≥ Memory oder Direktoperand:	Carry Set	R .GE M
Registerinhalt = Memory oder Direktoperand:	Z-Flag Set	R .EQ M
Registerinhalt < Memory oder Direktoperand:	Carry Clear	R .LT M

Der Status des N-Flags braucht den Programmierer also überhaupt nicht zu bekümmern.

Für die Bedingungsabfrage und Verzweigung 'on condition' (wenn die Bedingung erfüllt ist) stehen uns zwei Befehlspaare zur Verfügung - mit jeweils antivalenter Ausführung, nämlich BCS (Branch on Carry Set), BCC (Branch on Carry Clear), BEQ (Branch on Equal) und BNE (Branch on Not Equal). Mit diesen Befehlen wird der normale Programmablauf verlassen und an anderer Stelle fortgesetzt, wenn nach dem Vergleich die abgefragte Bedingung vorliegt.

Erinnern wir uns kurz der in Abschnitt 7.3 (Heft 5) dargestellten Auswirkung der Subtraktion auf den Status. Carry Set bedeutet 'es ist nicht geborgt worden, weil der Registerinhalt größer oder gleich Inhalt der Speicherzelle war', Carry Clear heißt 'es ist geborgt worden, weil er kleiner war'. Das Z-Flag gibt auch dort die zusätzliche Information, daß das Subtraktionsergebnis Null ist.

Die Arbeitstabelle auf der folgenden Seite ermöglicht eine schnelle Ausdeutung des Statusbytes.

BEDEUTUNG DER STATUSANZEIGE

	1. NIBBLE			2. NIBBLE			
Wert	Negativ	Overflow ...	Break	Dezimal	Interrupt	Zero	Carry
0							
1			B				C
2						Z	
3			B			Z	C
4		O			I		
5		O	B		I		C
6		O			I	Z	
7		O	B		I	Z	C
8	N			D			
9	N		B	D			C
A	N			D		Z	
B	N		B	D		Z	C
C	N	O		D	I		
D	N	O	B	D	I		C
E	N	O		D	I	Z	
F	N	O	B	D	I	Z	C

Die Bitposition 5 ist nicht besetzt (future expansion)

****

UNTER DEN 16-BIT-MIKROS hat der MC 68000 von Motorola ein zukunftsweisendes Design, das die bevorzugten Eigenschaften der 65xx und des 6809 vervollkommnet und bis an die Maschinen der großen EDV heranführt. Rockwell wird diese CPU als second source herstellen und vertreiben, es wird also die echte 16-BIT-CPU der Prozessorfamilie. Der Herausgeber hat im Dezember einen 68000-Lehrgang besucht und wird diese Maschine vorstellen und sie künftig softwaremäßig berücksichtigen.

Bei den statusmäßigen Ergebnissen haben wir aus Gründen der Anschaulichkeit auch Vokabeln aus FORTRAN notiert: .GE für Greater or Equal, .EQ für Equal, .LT für Less Than (weniger als). Wir sehen, daß die unmittelbaren Verzweigungsmöglichkeiten eigentlich gering sind. In bald jedem Programm möchte man auch dann bequem verzweigen können, wenn eine der folgenden Bedingungen vorliegt:

Registerinhalt > größer aber nicht gleich Speicherzelle .GT Greater Than

Registerinhalt ≤ kleiner oder gleich Speicherzelle .LE Less Equal

Die Verzweigungen für das Greater Than und das Less Equal der höheren Programmiersprachen muß man auf Maschinenebene durch zwei aneinandergereihte 'branches on condition' verwirklichen, wobei für das .GT die Abfolge unbedingt einzuhalten ist:

Beispiel 7.6.1 Feststellung auf Register > Memory (Greater Than).

```
        LDA ...     1. OPERAND
        CMP ...     VERGLEICHSBEFEHL
        BEQ LESSEQ  SCHEIDE DEN FALL 'GLEICH' AUS
        BCS GTHAN   VERZWEIGE NUR FÜR 'GRÖSSER'
LESSEQ  ...         VERARBEITUNG SETZT FORT MIT 'KLEINER ODER GLEICH'
        ...
GTHAN   ...         SPÄTERE VERARBEITUNG FÜR 'GRÖSSER ALS'.
```

Wenn man für die vorstehende Feststellung das BCS als erstes programmierte, so würde auch der Fall 'gleich' auf den Ast GTHAN gerissen, das nachfolgende BEQ würde nie mehr zu einer Verzweigung führen.

Beispiel 7.6.2. Abfrage auf Register ≤ Memory (Less Equal).
Vom überlegungsmäßigen Ansatz handelt es sich um die Antivalenz zum Greater Than:

```
        CPX ...     VERGLEICH REGISTER X MIT MEMORY ODER DIREKTOPERAND
        BCC LESSEQ  ES IST GEBORGT WORDEN
        BEQ LESSEQ  FÜR GLEICHHEITSBEDINGUNG
GTHAN   ...         VERARBEITUNG FÜR X > MEMORY
        ...
LESSEQ  ...         VERARBEITUNG FÜR X ≤ MEMORY
```

In diesem Beipiel ist die Reihenfolge der Verzweigungsbefehle unerheblich.

Man merke: Für die vorstehenden Feststellungen brauchen wir uns um den Status des N-Flags und um die Verzweigungsbefehle BPL und BMI überhaupt nicht zu kümmern. Weiterhin: Für die Compare-Befehle muß das Carry-Flag in keiner Weise im Voraus definiert gesetzt zu werden (Unterschied zur Subtraktion!).

Die Vergleichsbefehle CMP, CPX und CPY stehen wie folgt zur Verfügung (Implementierung):

mit Direktoperand,

für Zeropage-Adressierung

für absolute Adressierung.

Der den Akku betreffende Befehl CMP ist dabei der mächtigste, denn es gibt in in folgenden zusätzlichen Adressierungen:

indiziert mit X, Zeropage oder absolut

indiziert mit Y, absolut

indirekt indiziert mit X oder mit Y

Der Programmierer halte sich vor Augen, daß Vergleichsbefehle in zahlreichen Anwendungen überhaupt nicht erforderlich sind, weil der durch eine vorausgegangene Operation gesetzte Status bereits eine ausreichende Information enthält. In Abschnitt 7.1 (Heft 5, S. 32) wiesen wir bereits darauf hin, daß Ladebefehle den Status ebenfalls setzen. Wenn wir feststellen wollen, ob der Inhalt einer Zelle z.B. '0' ist, so programmieren wir

```
        LDA ZELLE       SETZE STATUS
        BEQ NULL        VERZWEIGE, WENN '0'
```

oder umgekehrt:

```
        LDX ZELLE
        BNE UNGLEI      VERZWEIGE, WENN NICHT '0'.
```

Das Gleiche gilt für die in 7.2 besprochenen Befehle INCREMENT x oder Y (INX, INY) und DECREMENT X oder Y (DEX, DEY) sowie für die noch zu besprechenden verwandten Befehle INCREMENT oder DECREMENT IN MEMORY (INC und DEC). Diese Befehle beeinflussen das N-Flag (Status des Bit 7 betreffend) und das Z-Flag (Status des Ergebnisses =0 oder ≠ 0). Nach einer Erhöhung oder Erniedrigung um 1 kann man auf jeden Fall, dh. ohne besonderen Compare-Befehl, verzweigen wenn der Inhalt des Registers oder der Speicherzelle zu Null geworden ist oder z.B. von positiv nach negativ umklappte, z.B. beim Herunterzählen. Wir machen von dieser automatischen Veränderung des Prozessorstatus in allen Zählschleifen (Loops) mit Indexregister X oder Y Gebrauch; fast jedes Programm zeugt davon.

Das Prozessorkonzept empfiehlt, in Programmschleifen von einem Vorgabewert in X oder in Y mit DEX oder mit DEY herunterzuzählen, bis der Inhalt des Registers entweder zu Null geworden ist (BNE LOOP, solange Register noch nicht NULL) oder bis sein Inhalt von 'positiv Null' nach negativ 'FF' umklappt (BPL LOOP durchlaufe die Schleife, solange der Registerinhalt positiv ist). Wir sollten diesen Programmaufbau immer zu verwirklichen suchen, weil wir dann auf Compare-Befehle verzichten können, die Befehle DEX und DEY liefern uns automatisch den benötigten Status.

Man beachte im vorigen Absatz die antivalente Formulierung der Abfrage 'durchlaufe die Schleife solange das Register noch nicht Null ist - BNE - oder solange noch nicht negativ - BPL'.- Das Rückwärtsschreiten mit DEX, DEY oder DEC (in Memory) kann man anschaulich auch mit Krebsgang bezeichnen.

Ein Aufwärtszählen ist nicht immer zu vermeiden. Hier wird man mit den Vergleichsbefehlen prüfen müssen, ob das Maschinenregister schon einen Grenzwert errreicht hat. Man wählt meistens einen Direktoperanden im Befehl (Adressierung immediate).

Beispiel 7.6.3: Aufwärtszählen mit Schleifenausführung, solange X im Wertebereich 1-4 ist.

```
        LDX #$01        SCHLEIFE INITIALISIEREN
SCHLEI  ...             AUSFÜHRUNG
        INX             AUFWÄRTSZÄHLEN
        CPX #$05        ABFRAGE DER ENDBEDINGUNG+1
        BCC SCHLEI      RÜCKWÄRTSVERZWEIGUNG BIS EINSCHLIESSLICH X=4
```

Der Verzweigungsbefehl wird auf Grund des 'Borgens' gegen den Operanden 05 solange ausgeführt, wie 1 ≤ X ≤ 4. Statt des Befehles BCC hätte man hier auch BNE wählen können.

Beispiel 7.6.4: Im Akku z.B. aus einer Tastaturabfrage übergebene ASCII-Bytes sollen darauf überprüft werden, ob sie in den für Hexadezimalzahlen erlaubten Wertebereich von 0-9 und von A-F fallen. Es handelt sich also um ein Filter für erlaubte ASCII-Codes mit zwei Durchlässen.

```
KEY     JSR GETKEY    HOLE ZEICHEN VON DER TASTATUR
        CMP #$30      ABFRAGE ASCII '0'
        BCC KEY       ZURÜCKWEISUNG, WENN ALS STEUERZEICHEN KLEINER
        CMP #$3A      ABFRAGE < ':', DEM AUF '9' FOLGENDEN ZEICHEN
        BCC HEX       ZIFFER ZWISCHEN 0-9
        CMP #$41      ABFRAGE ASCII 'A'
        BCC KEY       SONDERZEICHEN AUS DEM ZWISCHENBEREICH
        CMP #$47      VERGLEICH MIT 'G' ALS DEM FOLGEZEICHEN VON 'F'
        BCS KEY       ZURÜCKWEISUNG, WEIL ZEICHEN ≥ 'G'
HEX     ...           VERARBEITUNG EINES ZULÄSSIGEN HEX-ZEICHENS.
```

Aus diesem Beispiel und dem vorigen entnehme man, daß man die obere Grenze mit einem Direktoperanden abfragt, der um 1 größer als der höchste noch zulässige Wert ist.

Beispiel 7.6.5: Wir wandeln das vorhergehende Beispiel der Tastaturabfrage ab. Auf einer vom Programmierer geschaffenen Kommando-Ebene sind 5 Tasten mit besonderen Funktionen belegt, z.B. nach der Sinnfälligkeit des Buchstabens mit A, F, K, Q und X belegt, nicht jedoch unter dem Gesichtspunkt eines zusammenhängenden aufsteigenden Wertebereiches (wie im Filter-Beispiel). Für diese Anwendung empfiehlt sich eine indizierte Programmschleife, die eine Tabelle mit den zulässigen Werten (wiederum in ASCII-Code hinterlegt) abfragt. Also: Filtermethode für aufeinanderfolgende Werte, Tabellenmethode für mehr zufällig gestreute Werte.

```
KEY     JSR GETKEY    HOLE ASCII-ZEICHEN
        LDX #$04      ZÄHLER FÜR 5
KEY1    CMP TAB,X     TABELLE ABRASTERN
        BEQ FUNKT     AUSFÜHRUNG DER ZULÄSSIGEN FUNKTION
        DEX           KREBSGANG
        BPL KEY1      NÄCHSTEN TABELLENWERT VERGLEICHEN
        BMI KEY       BRANCH ALWAYS, KEINE KORRESPONDENZ ZUR TABELLE, NEUE
                      TASTATURABFRAGE ODER AUCH FEHLERMELDUNG

FUNKT   ...           SIEHE FOLGENDE AUSFÜHRUNGEN

TAB     .BYT $41, $46, $4B, $51, $58      ENTSPRICHT ASCII A, F, K, Q, X.
```

Nach diesem Beispiel fragt sich nun, wie man im Programmabschnitt FUNKT gezielt auf die zu den zulässigen Tastenbetätigungen gehörenden ausführenden Programme stoßen soll. Erinnern wir uns: Wenn wir mit BEQ FUNKT bei Entsprechung mit einem Wert in Tabelle TAB aus der Schleife KEY entfernen, dann hat das Register X noch einen definierten und unzerstörten Wert, der irgendwo zwischen 04 und 00 liegt.

Mit diesem X könnte man in eine zweite Tabelle mit Sprungvektoren (Sprungadressen) indizieren. Hier wäre ein Befehl JMP (indirekt Tabelle),X nützlich. Der 6502 hat ihn nicht, es gibt ihn in der späteren Generation. Es kommt noch eine weitere Überlegung hinzu: Der Zählerstand in X wird in lückenloser Zahlenfolge abgeliefert. Ein Sprungvektor hat aber 2 Bytes. Man sollte eine Sprungtabelle in 2-er Schritten abrastern können oder 2 Tabellen haben, die eine für das Adreßbyte LOW, die andere für das Byte HIGH.

Es gibt aber zunächst eine 'kleine' Lösung, z.B. von Stephen Wozniak in seinem 'SWEET 16' verwirklicht (Heft 1, S. 24). Wenn alle ausführenden Routinen in einer Speicherseite (page) beginnen, dann braucht man nur

das Adreßbyte LOW indiziert aus einer Sprungtabelle zu entnehmen und in einen Sprungbefehl einzusetzen. Diese Methode hat natürlich ihre Risiken. Ihre Empfindlichkeit gegen Programmverschiebungen zeigte sich auch promt beim ersten Abdruck des Programmes in BYTE 11/1977, S. 150 ff.

Die universellere Methode lädt einen vollständigen Adreßvektor von 2 Bytes in einen JMP-Befehl, Opcode $4C. Das ist das Verfahren der Programm-Modifikation. Folgende Alternative ist empfehlenswerter, sie ist dem indirekten Sprungbefehl mit Opcode $6C angepaßt: Man hinterlegt den Sprungvektor in 2 Bytes, die sich nicht im Programm, sondern im Arbeitsspeicher befinden. Dieser Programmierstil ist in allen unseren Maschinen benutzt worden, besonders für die Interruptvektoren. Man springt die ausführende Routine mit dem JMP (indirekt Vektor) an.

Aus dem Beispiel 7.6.5 verbleibt die Frage, wie man mit einem einzigen X- oder Y-Wert zwei Tabellenbytes laden und hinterlegen kann. Nichts ist einfacher: Man multipliziere X mit 2 durch Linksverschiebung.

```
FUNKT   TXA             X NACH AKKU ÜBERTRAGEN
        ASL  A          MULTIPLIKATION MIT 2
        TAX             RÜCKÜBERTRAGUNG ZUR INDIZIERUNG
        LDA TAB2,X      HOLE NIEDRIGEN SPRUNGVEKTOR
        STA VEKT,X      ABLAGE
        INX             STEIGE IN DER TABELLE AUF
        LDA TAB2,X      SPRUNGVEKTOR HIGH
        STA VEKT,X      ABLAGE
        JMP (VEKT)      INDIREKTER SPRUNG IN DIE AUSFÜHRENDE ROUTINE

TAB2    .WORD ... ...SPRUNGVEKTOREN
```

Im Zusammenhang mit den Vergleichsbefehlen verbleibt für den Anfänger ein größeres Problem. In vielen Fällen ist die zu vergleichende Information in mehreren Bytes hinterlegt, in einem Datenfeld. Man möchte zwei Felder auf die bekannten Bedingungen überprüfen.

Zur anschaulichen gleichmäßigen Darstellung gehen wir von zwei 3-byte-langen Datenfeldern aus, FELD1 und FELD2, in denen die Information wie bei normaler Schreibweise für Zahlen von links nach rechts mit abfallender Ziffernwertigkeit eingeschrieben ist.

Beispiel 7.6.6: Prüfung von Datenfeldern auf Gleichheit.

```
        LDX #$02        ZÄHLER UND ADDRESSER FÜR 3
VGL     LDA FELD1,X     HOLE 1. OPERANDEN
        CMP FELD2,X     VERGLEICHE MIT DEM ZWEITEN
        BNE UNGLEI      AUSSPRUNG BEIM ERSTEN AUFTAUCHENDEN 'UNGLEICH'
        DEX             KREBSGANG
        BPL VGL         NÄCHSTE PRÜFUNG
GLEI    ...             VERARBEITUNG FÜR 'GLEICH'
        ..
UNGLEI  ...             VERARBEITUNG FÜR 'UNGLEICH'
```

Beispiel 7.6.7: Prüfung von Datenfeldern auf größer/kleiner.
Wir erinnern uns der Ähnlichkeit der Vergleichsbefehle mit der Subtraktion. Wir wenden daher auf das niederwertigste Byte den CMP-Befehl an. Die beiden nachfolgenden SBC-Befehle verwalten das Carry-Flag wie bei der Subtraktion.

```
        LDX #$02        ZÄHLER FÜR 3
        LDA FELD1,X     1. OPERAND, NIEDERWERTIGSTES BYTE
        CMP FELD2,X     ERSTE BEEINFLUSSUNG DES CARRY-FLAGS
        DEX             KREBSGANG
```

```
VGL     LDA FELD1,X  2. UND 3. BYTE PRÜFEN
        SBC FELD2,X  BEEINFLUSSUNG DES CARRY-FLAGS IN DER HÖHEREN STELLE
        DEX          KREBS
        BPL VGL      NÄCHSTES BYTE
        BCC KLEIN    BEARBEITUNG FÜR FELD1 KLEINER ALS FELD2
GROES   ...          BEARBEITUNG FÜR 'GROESSER/GLEICH'
```

Die beiden letzten Instruktionsfolgen in 7.6.6 und 7.6.7 sind recht lang und sehen wenig glücklich aus. Mit ihnen wäre Schlimmes zu befürchten, wenn man kombinierte Abfragen wie FELD1 > FELD2 ODER FELD1 ≤ FELD2 durchführen will.

Die Abfrage des Carry Set oder Clear nach der Bearbeitung eines Datenfeldes leuchtet nach der Darstellung der Subtraktion in Abschnitt 7.3 unmittelbar ein. daneben wird ein Merker benötigt, der auch Zwischenergebnisse für GLEICH/UNGLEICH festhält. Herr Zimmermann hat in seinem Advanced Subroutine Package, Abschnitt 1.5 (Heft 3, S. 17) die zu programmierende Lösung aufgezeigt: Für den Vergleich verzichtet er vollkommen auf den CMP-Befehl und erhält damit einen gleichmäßigen und klaren Schleifenaufbau. Das Carry-Flag wird zu Beginn gesetzt - wie für die Subtraktion erforderlich. Sein Status zeigt im Carry-Flag zum Schluß das GROESSER/GLEICH bzw. das KLEINER an. Eine anfangs auf Null gesetzte Merk-Hilfszelle STAT zeigt zum Schluß daneben das GLEICH/UNGLEICH an.

Beispiel 7.6.8: Kombinierter Vergleich von Datenfeldern.

```
        SEC          SUBTRAKTIONSVORBEREITUNG
        LDA #$00     STATUSMERKER AUF '0'
        STA STAT
        LDX #$02     ZÄHLER UND ADDRESSER FÜR 3, ENDE DER INITIALISIERUNG
V10     LDA FELD1,X  VERGLEICH DER FELDER
        SBC FELD2,X
        BEQ V20      STATUSMERKER NICHT VERÄNDERN
        STA STAT     AKKUINHALT MIT SICHERHEIT UNGLEICH NULL
V20     DEX          KREBS
        BPL V10      SCHLEIFE
V30     ...
```

Eine Programmverzweigung ab V30 würde wie folgt aussehen:

```
FÜR 'GLEICH'
        LDA STAT     STATUS IM LADEVORGANG HOLEN
        BEQ GLEICH   VERZWEIGUNG, ENTSPRICHT FORTRAN .EQ

FÜR GROESSER/GLEICH
        LDA STAT
        BEQ GG       ENTSPRICHT FORTRAN .GE
        BCS GG

FÜR 'GROESSER'
        LDA STAT
        BEQ V40      ABSONDERUNG DES FALLES 'GLEICH'
        BCS GROESS   VERZWEIGUNG, FORTRAN .GT ENTSPRECHEND
V40     ...          FORTSETZUNG FÜR 'KLEINER' UND 'GLEICH'

FÜR KLEINER
        BCC KLEIN    ENTSPRICHT FORTRAN .LT

FÜR KLEINER/GLEICH
        LDA STAT
        BEQ KLEIGL   ENTSPRICHT FORTRAN .LE
        BCC KLEIGL   DITO FÜR 'KLEINER'
```

65xx-Befehle und ihre Adressierungsarten

MSB \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRK	ORA (ø,X)				ORA ø	ASL ø		PHP	ORA #	ASL A			ORA !	ASL !	
1	BPL R	ORA (ø),Y				ORA ø,X	ASL ø,X		CLC	ORA !,Y				ORA !,X	ASL !,X	
2	JSR (!)	AND (ø,X)			BIT ø	AND ø	ROL ø		PLP	AND #	ROL A		BIT !	AND !	ROL !	
3	BMI R	AND (ø),Y				AND ø,X	ROL ø,X		SEC	AND !,Y				AND !,X	ROL !,X	
4	RTI	EOR (ø,X)				EOR ø	LSR ø		PHA	EOR #	LSR A		JMP !	EOR !	LSR !	
5	BVC R	EOR (ø),Y				EOR ø,X	LSR ø,X		CLI	EOR !,Y				EOR !,X	LSR !,X	
6	RTS	ADC (ø,X)				ADC ø	ROR ø		PLA	ADC #	ROR A		JMP (!)	ADC !	ROR !	
7	BVS R	ADC (ø),Y				ADC ø,X	ROR ø,X		SEI	ADC !,Y				ADC !,X	ROR !,X	
8		STA (ø,X)			STY ø	STA ø	STX ø		DEY		TXA		STY !	STA !	STX !	
9	BCC R	STA (ø),Y			STY ø,X	STA ø,X	STX ø,Y		TYA	STA !,Y	TXS			STA !,X		
A	LDY #	LDA (ø,X)	LDX #		LDY ø	LDY ø	LDX ø		TAY	LDA #	TAX		LDY !	LDA !	LDX !	
B	BCS R	LDA (ø),Y			LDY ø,X	LDA ø,X	LDX ø,Y		CLV	LDA !,Y	TSX		LDY !,X	LDA !,X	LDX !,Y	
C	CPY #	CMP (ø,X)			CPY ø	CMP ø	DEC ø		INY	CMP #	DEX		CPY !	CMP !	DEC !	
D	BNE R	CMP (ø),Y				CMP ø,X	DEC ø,X		CLD	CMP !,Y				CMP !,X	DEC !,X	
E	CPX #	SBC (ø,X)			CPX ø	SBC ø	INC ø		INX	SBC #	NOP		CPX !	SBC !	INC !	
F	BEQ R	SBC (ø),Y				SBC ø,X	INC ø,X		SED	SBC !,Y				SBC !,X	INC !,X	

LEGENDE: R=RELATIV, ø=ZEROPAGE, !=ABSOLUT, X= INDIZIERT, Y=INDIZIERT, (.)=INDIREKT, #=DIREKTOPERAND
A=AKKU, BEFEHLE OHNE LEGENDE: IMPLIED

Der vorstehende Abschnitt möge aufgezeigt haben, daß die Vergleichsbefehle einen hohen Stellenwert in der Programmablaufsteuerung besitzen. Ihre Handhabung für die Filterung von zulässigen Wertebereichen, für die Steuerung von Funktionsaufrufen etc. und für den Vergleich von Datenfeldern folgt einfachen Regeln. Mit wenigen Maschineninstruktionen erhält man eine Entscheidungsfähigkeit wie z.B. unter FORTRAN.

Im Vorgriff auf weitere Besprechungen folgt eine Tabelle mit dem 65xx-Befehlssatz und den Adressierungsarten, der regelmäßige Strukturen im Opcode zu entnehmen sind.

(wird fortgesetzt) R. L.

****

Erzeugung eines 'kalten Resets' beim AIM 65

ohne Zerstörung der Speicherinhalte

Manfred Reichert, Werderstraße 3, 7500 Karlsruhe 1

Viele AIM 65-Besitzer haben sich sicher schon oft geärgert, wenn sie beim Arbeiten mit dem DILINK-Vektor (in A406/A407) oder bei anderen Gelegenheiten das System ins 'Jenseits' beförderten und ein Zurückrufen ins 'Diesseits' nur durch Abschalten der Spannung möglich war. Nicht gerade sehr angenehm, wenn man zuvor erst mühevoll die RAMs mit Daten versehen hatte. Ohne Zerstörung des Speicherinhaltes konnte nur derjenige sein System wieder zum Leben erwecken, der Besitzer eines zweiten Terminals bzw. eines ASCII-Fernschreibers war. Dabei konnte nach Umschalten des Schalters KB/TTY die Änderung des Vektors von außen her erfolgen.

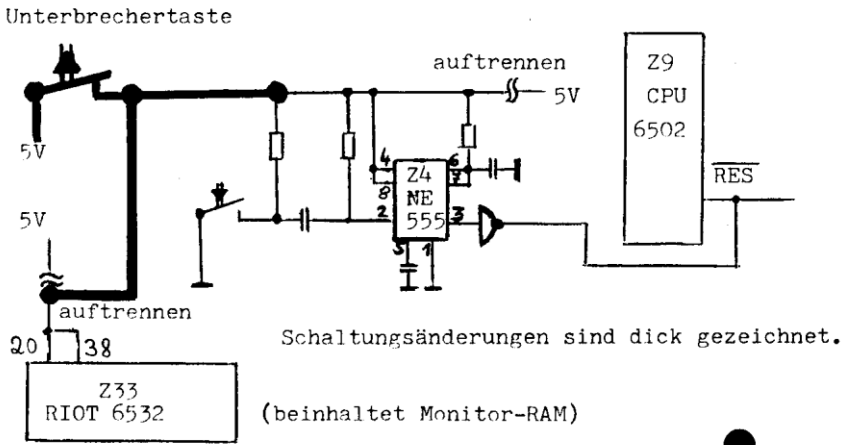
Wie kommt es nun dazu? Beim AIM 65 wird zwischen zwei RESET-Arten unterschieden:

1. Der sogenannte 'cold reset'. Durch ihn erfolgt eine Initialisierung der Monitorparameter sowie der durch den Anwender veränderbaren Vektoren im Monitor-RAM beim Anlagen der Betriebsspannung an das 'kalte' System.
2. Der sogenannte 'warm reset' (warmer Reset). Hierunter versteht man das Zurücksetzen des Systems im bereits eingeschalteten (warmen) Zustand mit Hilfe der RESET-Taste. Dabei werden allerdings nur die Monitor-Steuerparameter initialisiert, nicht jedoch die vom Anwender veränderbaren.

Da der DILINK-Vektor (display linkage = Zeiger zur in EF05 beginnenden Display-Subroutine) anwenderveränderbar ist, eignet er hervorragend in irgend eine vom Anwender definierte Subroutine hineinzuspringen, statt in die Display-Routine. Dabei ist es absolut unmöglich, den Vektor vom AIM-Keyboard her zu ändern (Monitorbefehle 'M' bzw. '/')! Dies kann nur durch ein Programm erfolgen. Genau so gefährlich ist es, wenn die eigene Subroutine noch nicht einwandfrei arbeitet und der obligate RTS-Befehl nicht erreicht wird.

In beiden Fällen hängt sich das System auf und kann auch durch die RESET-Taste nicht mehr zurückgesetzt werden - nur noch durch einen kalten Reset. Es wird daher folgende einfache Schaltungsänderung auf der AIM-Platine empfohlen:

Abtrennen der Versorgungsspannung (5V) vom Monitor-RAM (RIOT 6532) und von der Timerschaltung (NE555V) und Einfügen eines Unterbrecherschalters. Mit dieser Schaltungsänderung wird nach kurzer Betätigung der Unterbrechertaste ein 'kalter Reset' erzeugt, ohne daß die Netzspannung dabei abgeschaltet werden muß. Die Speicherinhalte bleiben damit erhalten und ein Experimentieren mit dem DILINK-Vektor ist nicht mehr so nervenaufreibend.



Schaltungsänderungen sind dick gezeichnet.

Buchbesprechung

Wolfgang Schneider: Programmieren von Heimcomputern, Band 1: Einführung in BASIC. Vieweg-Verlag Braunschweig 1979, 140 S. DM 24,-, ISBN 3 528-04160 9. Es handelt sich um ein wirkliches Lehrbuch für BASIC, es hebt sich damit von anderen Veröffentlichungen ab, die nur die Sprachelemente beschreiben. Autor und Verlag haben sich um die Darstellung vieler Beispiele, um reiche Kommentierung und um die grafische Heraushebung wichtiger Merksätze und Regeln für das Vorgehen bemüht. Jedes Sprachelement ist mit Beispielen und Übungsaufgaben belegt. Programmablaufpläne und weitere Grafiken unterstützen den Text. 10 vollständig programmierte und kommentierte Beispiele am Schluß des Buches zeigen auf etwa 40 Seiten, wie man das Wissen aus den einzelnen Kapiteln anwendet, um vollständige Programme zu schreiben. Diese Beipiele umfassen kaufmännische Anwendungen ebenso wie statistische und technische. Es ist ein Buch zum Selbststudium, es sollte auf dem Arbeitstisch des Anfängers seinen Platz neben dem Computer haben.

****

Kleinanzeigen der Leser

Verkaufe neuwertigen KIM-1 Mikrocomputer zum Preis von DM 330,- oder tausche diesen gegen AIM 65-BASIC-ROM. Knut Haase, Klosterstraße 72, 4000 Düsseldorf.

Suche 8k MICROSOFT-BASIC für KIM-1 auf Kassette mit originalen SOURCE-Listings. Angebote an G. Siegismund, Kellergasse 46, 8435 Dietfurt.

BASIC-Erweiterung für AIM 65/PC 100 durch die GWK

Kurz vor Redaktionschluß erhielt der Herausgeber das GWK-EPROM mit der seit einiger Zeit angebotenen BASIC-Erweiterung zum Zwecke der Besprechung. Eine Bedienungsanleitung sowie das ebenfalls von dort angebotene BASIC-Programm-Listing waren beigefügt.

Die BASIC-Erweiterung: Sie wird nach Bestellerwunsch für verschiedene Adreßbereiche und als EPROM oder als Tonbandcassette geliefert. Adreßbereiche entweder D000-D7FF oder 9000-97FF, also 2 kByte lang. Der erste Bereich ist mit dem des Assembler-ROMs identisch. Die EPROM-Version kann also direkt auf die Platine in die Steckfassung des Assemblers gebracht werden - ein Vorteil für kleine, nicht erweiterte Systeme, die vorwiegend mit BASIC betrieben werden sollen. Die für den Adressenbereich ab 9000 angebotene Version sollte man wählen, wenn der AIM erweitert ist und wenn das Assembler-ROM weiterhin benutzbar bleiben soll (von Umschaltern einmal abgesehen). In diesem Falle kann man auch die vom Magnetband zu ladende RAM-Version wählen.

Es handelt sich um ein Maschinenprogramm, das dem BASIC zusätzliche Statements vor allem für die Ein- und Ausgabeseite verleiht und das die Editierung und Ausprüfung von Programmen erleichtert. Weil ROM-resident, ist das AIM-BASIC ein weitgehend in sich geschlossenes Interpreterprogramm, es gibt also kaum offene Stellen im RAM-Bereich, in die der Benutzer eingreifen kann. Wie im Programm CBM-VIEW in diesem Heft demonstriert, könnte man in der Zeropage einen Keil in die CHR-GET-Routine treiben. Bei GWK wählte man eine Aussprungstelle, die nach END durchlaufen wird. Die neuen Statements haben daher den 'Dialekt' END:...

Enthalten sind das OPEN und CLOSE von Eingabe- und Ausgabedateien (files), für die folgende Einheiten erklärt werden können: Tastatur, Display/Printer, Magnetband, Lochstreifen, Anwender. Bezüglich der Handhabung im einzelnen würde man in der Dokumentation gerne einige ausgeführte Beispiele sehen.

Die weiteren Dienstleistungen umfassen den direkten Aussprung zu Maschinenprogrammen, wie im SYS-Befehl, implementiert als END:#XXXX, einen RENUMBER, der Zeilenabstände in 10er Abständen Vergibt, einen TRACE (zeilenweises Abarbeiten in BASIC) und eine Neu-Editierung des Programmtextes (ohne Tonband oder Text-Editor). Hierbei kann man eine Zeile unter ihrer Nummer auffinden und - wie beim Change-Command des Editors - eine Zeichenkette gezielt abändern, ohne die gesamte Zeile neu eintippen zu müssen, eine nützliche Einrichtung. Als allgemeine von Monitor und BASIC aufrufbare Dienstleistung ist ein Zahlenrechner und -konverter hex-dez-hex implementiert, der für Statements wie DATA, PEEK und POKE oft erwünscht wird. Zwei weitere Befehle betreffen die Rücknahme einer RETURN-Adresse vom Stack und das Zurücksetzen des Stackpointers. - Die Zeit war zu kurz, das gesamte neue Instrumentarium schon in der täglichen Praxis zu erproben.

Das BASIC-Programm-Listing ist eine Dokumentation auf etwa 70 Seiten. Sie enthält eine kurze Strukturbeschreibung für Variable und den BASIC-Stack, eine Erklärung der in der Zeropage benutzten Zellen, den Hexcode für die reservierten BASIC-Worte sowie auf etwa 65 Seiten eine Disassembler-Liste im AIM-Format mit beigefügten nachvollzogenen Kommentaren in deutscher Sprache. Diese Kommentare bezeichnen alle erkannten Eintrittspunkte und Erledigungsblöcke (Tätigkeitsbeschreibung) des Interpreterprogrammes und beziehen sich in vielen Fällen sogar auf den einzelnen Maschinenbefehl und seine Ausführung.

Eine solche aufbereitete Dokumentation ist für jede eigene Vertiefung in BASIC lehrreich und nützlich, sie erspart langes Suchen.

T A P E - C A T A L O G C B M 3 0 0 1

Ing. (grad.) Uwe Kornnagel, Lahnstraße 6, 6096 Raunheim/Main

```
10 REM KATALOGISIERUNG EINER AUF TAPE-DRIVE #1 EINGELEGTEN CASSETTE
25 SYS826:PRINT"¤ C A T A L O G   T A P E # 1"  :REM ¤=CLEAR SCREEN
40 PRINT " TYPE  NAME           LAENGE"
50 PRINT:SYS 837
55 PRINT:PRINT"***CBM CATALOG END ***":END
```

MASCHINENPROGRAMM:

```
033a    LDA   # 01
033c    STA   D4              Laufwerk # 1 ansprechen.
033e    JSR   F812            Auf "PLAY" mit PRESS... warten.
0341    JSR   FDD0            Print CRT
0344    RTS

        Catalog - Routine

0345    JSR   F855            Lies 192 Byte von 027A - 0339
0348    LDA   027A
034b    CMP   # 05            End of Tape Label gefunden ?
034d    BNE   0350            nein
034f    RTS                   ja
0350    CMP   # 04            Beginn of Data Label gefunden ?
0352    BEQ   0379            ja
0354    CMP   # 01            Beginn of Programm Label gefunden ?
0356    BNE   0345            nein
0358    LDA   # CE            ja
035a    JSR   03BC            Print   "PRGM"
035d    JSR   03AB            Print den Filenamen.
0360    SEC
0361    LDA   027d            Programmlänge aus Header errechnen.
0364    SBC   027b
0367    TAX
0368    LDA   027e
036b    SBC   027c
036e    JSR   DCD9            Programmlänge dezimal ausdrucken.
0371    LDA   # DA
0373    JSR   03BC            Print    "BYTE" und CRT
0376    CLC
0377    BCC   0345            suche neuen File
0379    LDA   # D4
037b    JSR   03BC            Print      "DATA"
037E    JSR   03AB            PRINT FILENAME
0381    LDA   # ØØ
0381    STA   03ff            Blockzähler auf 0 setzen.
0386    JSR   F855            Lies nächsten Block
0389    INC   03ff
```

```
038c  LDA  027A
038f  CMP  # 05          End of Tape Label gefunden ?
0391  BEQ  039B          ja
0393  CMP  # 04          Beginn of Data Label gefunden ?
0395  BEQ  039B          ja
0357  CMP  # 01          Beginn of Programm Label gefunden ?
0399  BNE  0386          nein

039b  LDX  03ff
039d  LDA  # ØØ
03a0  JSR  DCD9          Blockzähler dezimal ausdrucken
03a3  LDA  # E1
03a5  JSR  03BC          Print   "BLOKS"  und CRT
03a8  CLC
03a9  BCC  0348          Header auswerten

      Filenamen drucken

03ab  LDA  # 00
03ad  STA  0290          Markiere End of TEXT
03b0  LDA  # 20
03b2  JSR  FFD2          Print Space
03b5  LDA  # 7F
03b7  LDY  # 02
03b9  JMP  CA1C          Drucke den Text

      Interne Textausgabe

03bc  LDY  # 03
03be  JMP  CA1C

      Internes Textfeld

      * = 03ce
03ce  .BYTE 20,50,52,47,4d,00          ;PRGM
03d4  .BYTE 20,44,41,54,41,00          ;DATA
03da  .BYTE 20,42,59,54,45,0d,00       ;BYTE und CRT
03e1  .BYTE 20,42,4c,4f,43,4b,53,0d,00 ;BLOCKS und CRT
```

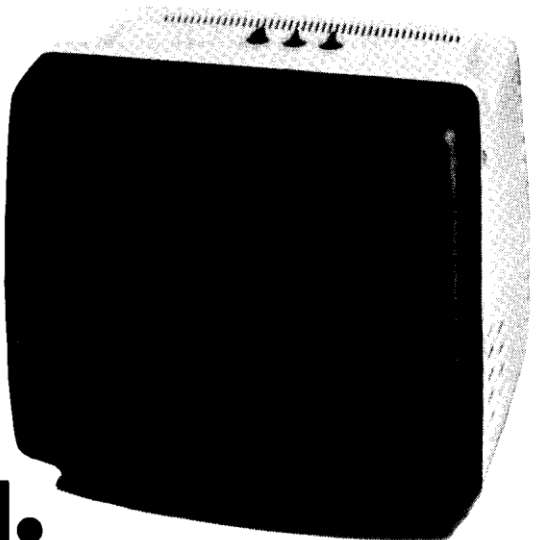
Alle Adressen und Daten sind Hex. angegeben.

++

DIE NÄCHSTE AUSGABE wird eine Besprechung des OHIO-Challenger Superboard II enthalten.

DAS VON DER IMPULSTECHNIK WEGNER VORGESTELLTE FARBGRAFIK-TERMINAL überzeugte durch seine Vielseitigkeit. Es kann an jeden handelsüblichen Farbfernseher angeschlossen werden. Nach Auffassung des Herausgebers eignet es besonders für die auffällige Kommunikation mit großen 'Bannerschriften' in Farbe, z.B. in Firmen und an Plätzen an denen viele Personen passieren, auch für werbliche Zwecke.

Wer beruflich »in die Röhre schaut« sieht bei uns grün. Und dies nicht nur scharf, sondern vielmehr auch ruhig.



Für alle, die mit Bildschirmdaten leben, entwickelte Sanyo den augenfreundlichen Typ DM5912CX. Denn EDV-Informationen lesen, kann weitaus deutlicher und erholsamer sein. So schrieb uns ein anwendungserfahrener Experte: „Der von Ihnen gelieferte Monitor ist ein wirklich angenehmes Display für den Mikroprozessor. Das grüne Bild steht scharf und ruhig und beansprucht mein leider krankes Augenlicht in keiner Weise.".
Das alles wurde von uns in ein formschönes, elfenbeinfarbiges Kunststoffgehäuse mit Rauchglas-Frontblende eingebaut.

Technische Hauptmerkmale:

Bildschirmdiagonale:	12" (31 cm)
Videobandbreite:	18 MHz
Geometriefehler:	kleiner als 1%
Displayformat:	80 Charakter, 24 Zeilen
Eingangssignal:	Standard Video 1 V_{ss}

SANYO
VIDEO-SYSTEME

Ausführliche Informationen mit Fachhändler-Nachweis:
ANYO Video Vertrieb GmbH. & Co., Lange Reihe 29, 2000 Hamburg 1, Telefon (040) 24 62 66

MATRIXDRUCKER FÜR AIM 65

● Sehr kontrastreich und klar arbeitend,

Friktionsantrieb, für normales Rollenpapier (preiswertes Telexpapier).

Zeichenerzeugung: Nadelmatrix-Druckkopf mit 7 Nadeln vertikal,

Druckgeschwindigkeit: 100 Zeichen/Sek.

Zeichenhöhe 2,75 mm, Zeilenabstand 4,23 mm.

Zeichenvorrat: Beliebig; normal: 64 Standard ASCII-Zeichen.

Preis: DM 745,-

● INTERFACEKARTE:

Beinhaltet Treiber,

Schutzschaltung für Druckkopf,

Motorsteuerung und Stromversorgung (ohne Trafo:

2x 24 V, 1 A AC).

Preis: DM 73,- Preise zuzüglich 13% MWSt.

● SOFTWARE (Source-Listing):

Standard: Zeichenerzeugung mit AIM-Zeichengenerator,

80 Zeichen/Zeile,

Zeilenpuffer vom Anwender definierbar,

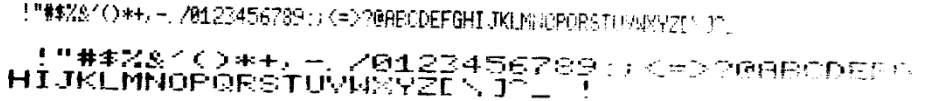
im Kaufpreis enthalten.

● SOFTWARE-OPTION:

Vorwärts-/Rückwärtsdruck, Fettdruck, Cursivdruck

Beliebiger Zeichensatz und Hardcopy eines Video-RAMs.

(A5 Februar 1980) Eine verkleinerte Druckprobe:



I. DOHMANN

ELEKTRONISCHE BAUGRUPPEN
IM WIEHAGEN 15
4830 GÜTERSLOH 12
TEL.: 05241 - 67 480

WEGNER-IMPULSTECHNIK

FARBGRAPHIKTERMINAL

Das von mir entwickelte preisgünstige Farbgraphikterminal besteht aus einem Basisgerät und Einschubkassetten in verschiedenen Leistungsstufen. Als Basisgerät dient die von verschiedenen Firmen für Telespiele angebotene F8-Platine.

Bei allen Ausführungen sind 8192 Bildpunkte adressierbar. Hintergrund- und Vordergrundfarben sind für Zeile und Bildpunkt getrennt wählbar. Eine 15-polige Buchsenleiste verbindet das Farbterminal mit dem anzuschliessenden Prozessorsystem über eine parallele Schnittstelle. Die Ansteuerung für alle bekannten Standardterminalfunktionen im ASCII-Code. Farb- und Sonderfunktionen werden durch Kontrollzeichen generiert.

Die Serie beginnt mit dem Gerät STEWE I:

5x7 Bit-Matrix zur Darstellung von Zahlen, Buchstaben und Sonderzeichen (64). Volle Cursor-Kontrolle (zeichenweise, bzw. bitweise) Punkt malen/nicht malen oder löschen. Verschiedene Strichstärken. Frei wählbare Farben für Vorder- und Hintergrund, Positionierung wie beim Cursor, auch für Sonderzeichen und Gruppen. Display löschen, Zeile löschen, pulsierender Cursor ja/nein.

Dieses preisgünstige Gerät der Serie wird nur komplett geliefert, es ist voll ausbau- und erweiterungsfähig. Preis ab DM 748,-inkl. MWSt.

Die Terminalcassette STEWE II bietet darüber hinaus:

Wahlweise Umschaltung auf 7x9 und 5x5 Bit-Matrizen (96 Zeichen bei 7x9). Groß- und Kleinschreibung, Definition von Zeilen- und Zeichenabstand. Vollständige Anwenderprogramme können dauerhaft gespeichert werden (EEPROM).

Ausführung STEWE III:

Zusätzlich mit Tabulator, unabhängigem Laufschriftgenerator. Möglichkeit der Ein- und Ausgabe von selbstdefinierten Zeichen.

Sonderausführungen sind in vielen Variationen erhältlich. Die Cassetten enthalten Mikrocomputersysteme der 6502-Familie. Spannungsversorgung über das Basisgerät.

Das Basisgerät ist ferner ausbaubar als

DIGITAL-MULTIMETER mit farbiger Anzeige der Meßwerte auf dem Bildschirm und als

LOGIKANALYSATOR für 8-48 Kanäle mit binärer, hexadezimaler oder auch gemischter Darstellung. Es können bis zu 9 Logikzustände gespeichert werden. Er ist zudem extern triggerbar.

BITTE VERLANGEN SIE MEINEN SONDERDRUCK!

WEGNER-IMPULSTECHNIK 6395 WEILROD 11
PFINGSTBERGWEG 10

GWK

GESELLSCHAFT FÜR TECHNISCHE ELEKTRONIK mbH.

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

NEU! NEU! NEU! NEU! NEU!

BASIC ERWEITERUNG

Erweitert den Befehlssatz des AIM und PC100 Basic um folgende zusätzliche Befehle:
OPEN und CLOSE von INPUT und OUTPUT FILES.
Als INPUT/OUTPUT DEVICE können definiert werden:
Keyboard/Display/Printer,Tape,Paper Tape,User,Memory/Dummy
SYS Befehl.Aussprung zu Unterprogrammen in Maschinensprache.
ATN.
MEMORY SIZE kann geändert werden,ohne Programm zu zerstören.
RENUMBER.Nummeriert das Programm in 10ner Schritten neu durch.
Rechnet sämtliche Sprungadressen um.
TRACE.Erlaubt zeilenweises Debugging.
CHANGE.Ermöglicht änderungen innerhalb einer Programmzeile.
ADRESS CALCULATOR.Wandelt Hexadezimale und Dezimale Zahlen um und rechnet mit diesen auch im gemischten Modus.

Lieferbar für Adressbereich D000 bis D7FF und 9000 bis 97FF
Wahlweise im EPROM oder auf Cassette.

BASIC LISTING

Reichlich kommentiertes Disassembler Listing des AIM 65 BASIC V1.1.Umfang ca 70 Seiten DIN A4.

PROTOTYP BOARD

Neue Platine des GWK EURO BOARD EXPANSION SYSTEM.Experimentierplatine mit sehr gut durchdachter Aufteilung.Über 1800 Rasterbohrungen.Geeignet zum Wrappen,Fädeln und für Handverdrahtung.

PREISE entnehmen Sie bitte unserer nebenstehenden Anzeige.

GWK EURO BOARD EXPANSION SYSTEM

Microcomputer AIM 65,1 K Byte RAM		875,--	988,75 DM
Microcomputer AIM 65,4 K Byte RAM		1o5o,--	1186,5o DM
Resident ASSEMBLER,4K,in ROM		23o,--	259,9o DM
BASIC,Interpreter,8K,in ROM's		28o,--	316,4o DM
BASIC ERWEITERUNG,2K,auf Cassette		145,--	163,85 DM
BASIC ERWEITERUNG,2K,in EPROM,steckbar in Z 24		28o,--	316,4o DM
BASIC LISTING,ca 70 Seiten DIN A4,gut kommentiert		45,--	5o,85 DM
AIM 65 ANWENDERHANDBUCH,deutsch		35,--	39,55 DM
AIM 65 MANUALS,englisch	je	15,--	16,95 DM
AIM 65 SCHALTPLANPOSTER		5,--	5,65 DM
THERMOPAPIER,approved,schwarzdruckend,lagerfähig,5Om		6,5o	7,35 DM
STECKERLEISTEN,44 Pole,nach MIL-C-21 097		12,5o	14,13 DM
MOTHER BOARD,12 Steckplätze f.Expansion,3 f.Appl. direkt an AIM steckbar,oder Einbau in 19" Gehäuse.Voll gepuffert.		485,--	548,o5 DM
ADAPTER BOARD,Busplatine ohne Pufferung,9 Steckplätze		295,--	333,35 DM
RAM BOARD, 3 x 4 K Byte statisches RAM,unabhängig voneinander beliebig adressierbar,write protectable.		945,--	1o67,85 DM
EPROM BOARD,12 K Byte,für die 1K EPROM's 2708 oder 2758.		395,--	446,35 DM
EPROM PROGRAMMER BOARD,mit res.Betriebsprogramm,Softwareumschaltung für 1K,2K und 4K Eprom's mit einer Versorgungsspannung.		795,--	898,35 DM
VIA/PIA BOARD,32 I/O Kanäle,beliebig adressierbar.		335,--	378,55 DM
PROTOTYP BOARD, Experimentierplatine zum Wrappen oder Fädeln.Durchdachte Aufteilung,über 1800 Rasterbohrungen.		75,--	84,75 DM
PET INTERFACE,ermöglicht den Anschluss der GWK BOARD's an den PET.Pufferung,Rückgewinnung der oberen Adressbits.		425,--	48o,25 DM
POWER SUPPLY BOARD,hochwertiges Schaltnetzteil incl.Trafo.			
5V/3A;26V/o,7A		315,--	355,95 DM
5V/3A,Notstromversorgung;26V/o,7A;+/- 12V/o,7A		45o,--	5o8,5o DM
5V/6A,Notstromversorgung;26V/o,7A;+/- 12V/o,7A		595,--	672,35 DM
GEHÄUSE,für AIM,Kunststoff		147,--	166,11 DM
GEHÄUSE,für AIM,Metall mit Platz für Netzteil u.kl.Erweiter.		345,--	389,85 DM
PULTGEHÄUSE,für AIM und System,Metall,mit 19" Baugruppenträger.		595,--	672,35 DM
Diverse LSI Chips:65oo Famile,EPROM's,RAM's,usw		bitte anfragen.	
Alle Preise zuzüglich Versandspesen.		zzgl MWST	incl.MWST

GWK GESELLSCHAFT FÜR TECHNISCHE ELEKTRONIK mbH.

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

65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

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

65xx MICRO MAG erscheint zweimonatlich. Offsetdruck: Gerhard M. Meier, Hamburg 70.

BEZUGSBEDINGUNGEN: Abonnement ab laufender Ausgabe für 6 Hefte DM 45,- (Inlandsendpreis). Ausland/Foreign via surface mail DM 50,-. USA air $30. Die früher erschienenen Ausgaben dieser Zeitschrift sind nachlieferbar. Einzelpreis für Nos. 1-6 DM 7,- /St., Nos. 7-9 DM 7.80/St. Nachlieferung bis Frühjahr 1980, danach Sammelband für die Ausgaben 1-6. Richten Sie bitte Ihre Überweisungen/Schecks an Roland Löhr, Konto 20/01121 bei der Vereins- und Westbank in Ahrensburg, BLZ 200 300 00. Zuschreibung auch über das Postscheckkonto der Vereinsbank, PSchA Hamburg 2244-207.