65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 7,- Nr. 6 April 1979

Aktuelle Programme

AIM 65 (Rockwell) wird seit Dezember 1978 in bemerkenswerten Stückzahlen ausgeliefert und dürfte inzwischen schon von gut einem Drittel der Leser betrieben werden. Das ist ein wichtiger Grund, das vorliegende Heft auf dieses leistungsfähige Gerät auszurichten und bereits einen zweiten Satz von Systemprogrammen aktuell zu veröffentlichen - während andere Zeitschriften im In- und Ausland noch zu keiner eingehenden Anwenderbesprechung vorgedrungen sind (Stand Anfang April). Die ebenfalls abgedruckte AIM Monitor Cross Reference List macht das Betriebssystem transparenter und wird sich für die Programmentwicklung als sehr nützlich erweisen.

HYPERTAPE ist nun auch auf SYM-1 lesbar. PET-Benutzer mögen sich über den Einstieg in die Maschinenprogrammierung durch den Artikel PET-6502-Assembler freuen. PET-Besitzer waren bisher durch den Mangel an Dokumentation benachteiligt. Die Verhältnisse dürften sich inzwischen durch das PET USER MANUAL und durch die Veröffentlichungen des PET User Club wesentlich gebessert haben. Damit gewinnt die Programmierung von Anwender-Peripherie im Laufe der Zeit zunehmend an Interesse.

Mit dem Bezug der ersten 6 Hefte dieser Fachzeitschrift läuft für viele Leser das Erstabonnement ab. Falls zutreffend, finden Sie anbei ein Hinweisschreiben.

Inhaltsverzeichnis

AIM SPEZIAL

Das Assembler-ROM wird bereits seit Ende Februar ausgeliefert. Wir werden auf seinen Gebrauch möglichst in der nächsten Nummer eingehen. Die BASIC-ROMs werden für Anfang April erwartet. - In Stückzahlen wird man den AIM 65 auch ohne Tastatur beziehen können (dedicated controller).

In Kooperation mit EEM Semi in Phoenix, Arizona, wird Rockwell als second source ab Herbst 1979 auch 8 k statische RAMs liefern.

Auf der gut besuchten HOBBYTRONIK, Anfang März in Dortmund, stellte die Firma Feltron (Zeissler), Postfach 1169 in 5210 Troisdorf-Spich ein nicht eben billiges Metallgehäuse für den AIM vor, das nach dem Eindruck des Herausgebers allerdings nicht ganz glücklich konzipiert ist. Der RESET-knopf und die Umschalter KB/TTY, STEP/RUN sind nicht herausgeführt, Zusatzplatinen sollten das AIM-Format haben, um in die Schübe (2) zu passen. - Der MIRO-Shop-Bodensee (Nedela) will in Kürze mit einem Kunststoffgehäuse herauskommen, DM 165,- (inkl.MWSt).

AIM-Betreiber mögen folgende Besonderheiten beachten: Die Befehlseingabe für die Adressierungsart (indirekt),Y mit dem I-Command ist im Handbuch und in den Kärtchen fehlerhaft abgedruckt. Man muß richtig eingeben: LDA (Pointer),Y, wenn man nicht fälschlich in der indirekten Adressierung mit X landen will.

Beim Bandschreiben wird, wie üblich, in TIOSET, $EE27, die Interruptmöglichkeit ausgeschlossen. Nach Ansicht des Herausgebers wird sie nach dem Dump aber nicht wiederhergestellt, denn in $E523 steht ein CLC statt des dort eigentlich erwarteten CLI.

Die Befehlseingabe unter I-Command für LDX absolut,Y (Opcode BE) führt zu Fehlern. BE wird als eine 1-byte-lange Operation ins Memory eingetragen (der Zuordnungszähler rückt nur um eine Adresse weiter). Das Display zeigt zwar, was man eingetippt hat, der Printer zeigt einen 3-byte-langen Befehl - aber mit einer Adressierung, die dem zufälligen Speicherinhalt an der Eintragungsstelle entspricht. Es ist noch nicht bekannt, was hiergegen unternommen werden soll.

Von bisher drei Geräten ist bekannt, daß sie den EOR-Befehl unter dynamischen Bedingungen nicht immer einwandfrei ausführen wollen.

Fortsetzung auf Seite 48

****

HANNOVER-MESSE:

DIE FIRMA GWK TECHNISCHE ELEKTRONIK AUS ÜBACH-PALENBERG (HERREN KLAPPA UND GRAWE) STELLT AUS IN DER CEBIT-HALLE 1-C, STAND 6206: AIM 65 UND DAS ERWEITERUNGSSYSTEM FÜR AIM UND KIM. DIESER STAND KANN ALS MÖGLICHER ANLAUF- UND VERABREDUNGSPUNKT FÜR 65XX SYSTEMBETREIBER DIENEN, NACHDEM ROCKWELL KEINEN DACHPAVILLON MEHR HAT.

DER HERAUSGEBER IST AM 19.4.79 (DONNERSTAG) BESTIMMT IM CEBIT, WAHRSCHEINLICH AUCH AM FREITAG (20.4.) ODER AM VORLETZTEN MESSETAG.

COMMODORE STELLT SEINE IN DER FACHPRESSE BEREITS VORGESTELLTE ERWEITERTE GERÄTELINIE AUS.

DER ELEKTOR VERLAG DÜRFTE EIN KIM-ÄHNLICHES SELBSTBAUSYSTEM ZEIGEN, DAS AB HERBST BESPROCHEN UND IM HANDEL ALS BAUSATZ ANGEBOTEN WERDEN SOLL.

THE HAMMING-WAY

AIM TAPE DUMP & LOAD WITH 3150 BAUD - SELFCORRECTING CODE

E: This fast tape format uses the AIM audio interface and the 6522 VIA for equal bit representation. Each 8 bit data byte is transmitted as a 16 bit Hamming byte. This enables single bit repair during data recovery. Thus the format is made very safe for data that are not allowed to get lost in any case. The effective data transmission rate is about 185 bytes/sec. Dump and load will even work satisfactory at double speed without hamming. Velocity may not be the limit.

Datenspeicherung auf Magnetbandcassette ist für unsere kleinen Systeme noch immer das preiswerteste Verfahren. Es besteht daher ein dringendes Interesse, zu schnelleren und sehr sicheren Aufzeichnungen zu kommen, immerhin können AIM 65 und PET mit zwei Cassettenlaufwerken betrieben werden und eignen sich daher von Haus aus für Auskunftssysteme und für ein file processing, die fortlaufende Datenverarbeitung von einem Magnetband in der Eingabe zu einem zweiten in der Ausgabe. Mit den hier vorgelegten und in zahlreichen Experimenten mühsam erarbeiteten Programmen ist die Entwicklung nicht abgeschlossen, denn der Autor hat auf einem normalen DM-350-Laufwerk und über das AIM-Interface versuchsweise schon 8333 Baud gefahren. Hierüber wird später berichtet werden. Man darf aber jetzt schon darauf hinweisen, daß an mehreren Stellen daran gearbeitet wird, dem Anwender Cassettenlaufwerke zur Verfügung zu stellen, die in allen Betriebsfunktionen elektrisch fernsteuerbar sind. Damit werden unsere Systeme noch in diesem Jahr wesentlich leistungsfähiger.

Das hier vorgelegte Programmpaar für Magnetbandaufzeichnung und für das Wiederlesen unter ausschließlicher Verwendung des AIM Audio-Interface arbeitet mit etwa 3150 Baud, was knapp 400 Zeichen in der Sekunde entsprechen würde. Es kann nach den bisherigen Beobachtungen auch mit diesem Tempo sicher betrieben werden, wenn man die Einsprungspunkte in die Unterprogramme entsprechend ändert und dafür sorgt, daß der Softwareschalter D1 immer ≠ 00 ist. - Der Autor würde sich sehr über Anwenderberichte dazu freuen, auch über die mögliche Steigerung des Tempos durch Verkleinerung der Zeitkonstanten. Bei der hier gewählten Bit-Darstellung dürfte die Grenze bei etwa 4500 Baud liegen.

Bit representation:  = 1 = 0

Die vorstehende Zeichnung erklärt die gewählte Aufzeichnungsform. Jedes Bit beginnt mit einem 'burst' von 127 µsek Dauer, darauf folgt für die '1' eine Halbwelle mit gleicher und für eine '0' eine Halbwelle mit doppelter Dauer. Der 'burst' ermöglicht beim Wiederlesen die Zwangssynchronisation und macht das Verfahren auch zuverlässiger, als die hier auch schon ausprobierte Wechselphasenkodierung (bi-phase encoded), bei der der Signalpegel keinerlei Bedeutung mehr hat, sondern nur noch die Zeit zwischen den Phasenwechseln. So angewandt für den KIM-1 mit dem ZIPTAPE von Lew Edwards. ZIPTAPE erfordert ein besonderes Interface, einige Leser hatten Schwierigkeiten damit (liest nicht), Herr Helmut Eckhardt in Köln 91 ist nach Beseitigung von hochfrequenten Störquellen sehr zufrieden.

Der Autor würde von den Lesern gerne erfahren, ob sie den ZIPTAPE ggfs. auch schon am VIA des AIM 65 haben betreiben können.

Die Programmentwicklung erfolgte nicht nur unter dem Gesichtspunkt der Geschwindigkeit, sondern vor allem auch unter dem der Sicherheit. Es wurde daher ein selbstkorrigierender Hamming-Kode eingesetzt, der die durchschnittliche Datenübertragungsrate auf etwa 185 nutzbare Zeichen in der Sekunde herabsetzt, wenn man die notwendigen Etiketten berücksichtigt. Es kann hier nicht auf die Hamming-Kodes eingegangen werden. Sie sind nach dem amerikanischen Informationstheoretiker R. W. Hamming benannt ('Error detecting and error correcting codes', 1950 veröffentlicht in Bell Systems Technical Journal 29, S. 147). Ein informativer Aufsatz zum hier verwandten Kode findet sich in BYTE 2/1979, S. 180: Michael Wimble, Hamming Error Correcting Code. Statt der 8 Bits eines Bytes werden effektiv 16 Bit übertragen. Die Generierung erfolgt erst im Augenblick der Transmission und belastet den Speicher nicht. Im Zuge des Datenempfanges werden unerlaubte Kombinationen erkannt, aus 16 Bit wird wieder 1 Byte komprimiert. Sofern nur ein einzelnes Bit umgeklappt ist, kann der Fehler automatisch repariert werden. Mehr-Bit-Fehler werden erkannt. Sie sind nicht reparierbar und führen zum Programmabbruch (Rückkehr zum Monitor mit ERROR). - Der hier gefundene kompakte Algorithmus für Sendung und Empfang der Hamming-Bytes kann natürlich auch in andere Arten der Datenübertragung übernommen werden.

Das Schreibprogramm läßt kurze Interrupts zu, wodurch es sich vom entsprechenden Monitor-Programm unterscheidet. Diese Möglichkeit konnte eingeräumt werden, weil die schönen Timer des R6522 VIA Vorspeicher-Latches haben. Man kann also, während der Timer an der alten Phase noch herunterzählt, bereits bestimmen, welche Dauer die folgende Halbwelle an PB7 haben soll (free running mode). (Beim SYM-1 hängt das Audio-Interface leider nicht mit PB7 zusammen.) Das Schreibprogramm ist daher geeignet, eine laufende Dokumentation von seriell an PB6 einkommenden Daten vorzunehmen (Meßwesen!). Am VIA gibt es bekanntlich Betriebsarten (Modes 010 und 011), bei denen jedes 8. Bit einen Interrupt auslöst. Das Clocking für das Schieberegister kann dabei wahlweise vom Timer 2 her oder von externer Clock erfolgen (an CB1). Zur Interruptbearbeitung wird dann die kurze Sicherung der Register A, X und Y gehören. Nach der Auslegung des Dump-Programmes können im Interruptbetrieb bis zu etwa 1400 Baud an PB6 seriell empfangen und dokumentiert werden. Man beachte aber die Zeitbedürfnisse genau, auch die des Vordergrundprogrammes, das man allerdings noch linearisieren und beschleunigen kann.

Zur Inbetriebnahme: Die Magnetköpfe sollen natürlich sauber sein. Fernerhin ist die Vergleichsspannung am LM311 (Bauteil Z8) mittels Potentiometer VR1 sorgfältig für das Bandlesen einzustellen. Man nehme ggfs. ein Oszilloskop um festzustellen, daß die '0' und '1' unterscheidende Phase wirklich im Verhältnis 2:1 steht. Und man kann ein Versuchsband in endloser Schleife mit SYNCs ($16) oder zunächst mit $AA beschreiben, in Zelle $0364 des Leseprogrammes ein BRK (00) einsetzen und die Einstellarbeiten an VR1 so ausrichten, daß zunächst einmal nur die SYNCs erkannt werden. Danach wird man Bandsätze mit z.B. 256 SYNCs schreiben und versuchen, je Satz 254 bis 255 SYNCs wiederzugewinnen. Wenn das gelungen ist, dann wird man auch ganze Programme laden können.

Und man halte sich vor Augen, daß die Durchzugsgeschwindigkeit des Laufwerkes von der Umgebungstemperatur und von der Betriebszeit während des Tages abhängig veränderlich sein mag. Einwirkungen aus dieser Quelle schaltet man dadurch aus, daß man die Konstante für den Zeitvergleich in $0404 um $\pm$ 1 bis 2 verändert.

Das Schreibeprogramm wird man als Anwenderfunktion auf eine der Tasten F1, F2 oder F3 legen, dafür den Sprungbefehl 4C 00 02 in die Adressen 010C ff. bringen. Der Benutzer wird interaktiv geführt. Handhabung der Endadresse wie beim KIM: + 1. Das Leseprogramm ist ebenfalls interaktiv. Man hinterlegt seinen Adreßvektor als User Input Function in Adresse 0108 und tritt mit Tastendruck 'L' und 'U' in die Ausführung. Wie bei LOKIM und QREAD in Heft 5 lädt man mit UIN=0 an alte Stelle, UIN=1 an neue Adresse 'TO' und mit UIN=3 und 'TO' hat man wieder die RALOAD-Option zur Programmverschiebung (s. Heft 1). Kleine Zwischenräume zwischen den Programmabschnitten erleichtern eigene Bearbeitungen.

```
<M>=010C 4C 00 02

0200  D8          HAMMD   CLD             BINARY MODE
0201  20 A3 E7     H1      JSR FROM        GET FROM ADDRESS
0204  B0 FB                BCS H1          IN CASE OF ERROR
S206  20 3B E8             JSR BLANK2
0209  AD 1C A4             LDA ADDR        DEPOSIT AS A POINTER ADDRESS
020C  85 D3                STA PNTL        TO MEMORY FOR DUMP
020E  AD 1D A4             LDA ADDR+1
0211  85 D4                STA PNTH
0213  20 A7 E7     H2      JSR TO          GET 'TO' ADDRESS
0216  B0 FB                BCS H2
0218  38                  SEC             CALCULATE # OF BYTES
0219  E5 D3                SBC PNTL
021B  85 D5                STA NBL
021D  AD 1D A4             LDA ADDR+1
0220  E5 D4                SBC PNTH
0222  85 D6                STA NBH
0224  20 13 EA             JSR CRLOW       NEUE ZEILE
0227  20 A2 E8             JSR FNAM        GET NAME ETC.
022A  20 4D EB             JSR CLRCK       CHECKSUM = 00
022D  AD 34 A4             LDA TAPIN       WHICH TAPE?
0230  20 1C EE             JSR TIOSET      START TAPE
0233  58                  CLI             RE-ENABLE INTERRUPT

0234  A9 EC                LDA #$EC        OPEN CHANNELS FOR AIM AUDIO IN-
0236  8D 0C A8             STA PCR         TERFACE AS IN TAOSET. CA2= LOW
0239  A9 C0                LDA #$C0
023B  8D 0B A8             STA ACR         P87 FREE RUNNING MODE

023E  A9 01                LDA #$01        HAMMING SWITCH ≠ 00
0240  85 D1                STA HASWI

0242  A9 00                LDA #$00        START TIMER BY WRITING COUNTER
0244  8D 05 A8             STA T1C-H       HIGH
0247  8D 07 A8             STA T1L-H       PRE-LOAD LATCH
024A  20 BC FF             JSR PATC25      WAIT FOR FIRST PHASE-OUT

024D  A2 20                LDX #$20        COUNTER FOR 32 SYNCS
024F  A9 16        H3      LDA #$16        SYNC CHAR.
0251  20 CC 02             JSR SANS        SEND WITHOUT HAMMING
0254  CA                  DEX             COUNT DOWN
0255  D0 F8                BNE H3
0257  A9 2A                LDA #$2A        '*' CHAR.
0259  20 CC 02             JSR SANS        SEND

025C  BD 2E A4     H4      LDA NAME,X      SEND FILENAME
025F  20 C0 02             JSR AVEC        SEND WITH HAMMING
```

Magnetisierung beim DUMP etwa 7/10. Beim Lesen soll die Nadel in den roten Bereich kommen.

```
0262  E8                    INX             5 BYTES TO TRANSMIT
0263  E0 05                 CPX #$05
0265  D0 F5                 BNE H4
0267  A2 00                 LDX #$00        NOW FOLLOW SAL, SAH, NBL, NBH
0269  B5 D3        H5       LDA PNTL,X
026B  20 C0 02              JSR AVEC
026E  E8                    INX
026F  E0 04                 CPX #$04        4 BYTES
0271  D0 F6                 BNE H5

0273  B1 D3        H6       LDA (PNTL),Y    TRANSMIT DATA FROM MEMORY,
0275  AA                    TAX             MAIN LOOP    - SAVE IN X
0276  18                    CLC
0277  6D 1E A4              ADC CKSUM       ADD TO CHECKSUM WITHOUT GOING JSR
027A  8D 1E A4              STA CKSUM
027D  90 03                 BCC H7          SKIP ON NO CARRY
027F  EE 1F A4              INC CKSUM+1
027A  8A          H7        TXA             RESTORE BYTE
0283  20 C0 02              JSR AVEC        SEND IT WITH HAMMING
0286  E6 E3                 INC PNTL        INC POINTER TO MEMORY
0288  D0 02                 BNE H8          SKIP ON NO CARRY
028A  E6 D4                 INC PNTH
028C  A5 D3        H8       LDA PNTL        COMPARE FOR END
028E  CD 1C A4              CMP ADDR
0291  A5 D4                 LDA PNTH
0293  ED 1D A4              SBC ADDR+1
0296  90 DB                 BCC H6          DO IT AGAIN

0298  A9 2F                 LDA #$2F        '/' CHAR, TERMINATOR
029A  20 CC 02              JSR SANS        SEND
029D  AD 1E A4              LDA CKSUM       SEND CHECKSUM WITH HAMMING
02A0  20 C0 02              JSR AVEC
02A3  AD 1F A4              LDA CKSUM+1
02A6  20 C0 02              JSR AVEC
02A9  A9 04                 LDA #$04        EOT-CHAR.
02AB  20 CC 02              JSR SANS        WITHOUT HAMMING
02AE  20 BC FF              JSR PATC25      WAIT FOR PHASE-OUT
02B1  8C 0B A8              STY ACR         DISABLE PB7 FREE RUNNING
02B4  68 68                 PLA, PLA        ADJUST SP TO RETURN TO MONITOR
02B6  00                    BRK             EXIT TO MONITOR

02C0  84 D1        AVEC     STY HASWI       SUBROUTINE TO DUMP WITH HAMMING, D1=00
02C2  85 D2                 STA TEMPA       SAVE FOR LOWER NYBBLE
02C4  4A 4A                 LSR, LSR        ISOLATE UPPER NYBBLE
02C6  4A 4A                 LSR, LSR
02C8  A8                    TAY             ADDRESSER TO HAMMING TABLE
02C9  B9 00 03              LDA TABH,Y      GET HAMMING BYTE

02CC  86 D7        SANS     STX TEMPX        SAVE X
02CE  A0 08        H9       LDY #$08        COUNTER FOR 8 BIT
02D0  AA                    TAX             SAVE BYTE
02D1  A9 7F        H11      LDA #$7F        GENERATE A SHORT BURST
02D3  8D 06 A8              STA T1L-L       LATCH
02D6  20 BC FF              JSR PATC25      WAIT TILL TIME IS OUT
02D9  8A                    TXA             GET BYTE OR IT'S REST
02DA  0A                    ASL A           SHIFT INTO CARRY POSITION
02DB  AA                    TAX             SAVE THE REST
02DC  A9 7F                 LDA #$7F        THAT WOULD BE FOR A '1'
```

```
02DE  B0 02              BCS H10         SKIP IF BIT WAS A '1'
02E0  A9 FE              LDA #$FE        THAT'S FOR A '0'
02E2  8D 06 A8    H10    STA T1L-1       TO LATCH FOR TIMER RECHARGE
02E5  20 BC FF           JSR PATC25      WAIT FOR TIME OUT
02E8  88                 DEY             ALL BITS DONE?
02E9  D0 E6              BNE H11

02EB  A5 D1              LDA HASWI       DECIDE FOR THE HAMMING-WAY
02ED  D0 0D              BNE H12         NUPE, EXIT
02EF  C6 D1              DEC HASWI       NOW = FF
02F1  A5 D2              LDA TEMPA       ISOLATE THE LOWER NYBBLE
02F3  29 0F              AND #$0F
02F5  A8                 TAY             ADDRESSER TO HAMMING TABLE
02F6  B9 00 03           LDA TABH,Y      GET HAMMING BYTE
02F9  4C CE 02           JMP H9          AND SEND

02FC  A6 D7       H12    LDX TEMPX       RESTORE
02FE  60                 RTS             AND EXIT
02FF  87                 LOP             LEMGTH OPERATOR FOR TABLES

<M>= 0300  00 E1 72 93   TABH            TO GENERATE HAMMING BYTE
< >  0304  B4 55 C6 27                   FROM NYBBLE
< >  0308  D8 39 AA 4B
< >  030C  6C 8D 1E FF

DATA RECOVERY ROUTINE FROM TAPE

<M>=0108  20 03                          USER INPUT FUNCTION

<M>=0310  10 80 40 08    TABC            TABLE FOR SINGLE BIT REPAIR
< > 0314  20 04 02 01

<M>=0318  87 06 05 03    TABD            TABLE TO DETECT TRANSMISSION
< > 031C  00 84 82 81                    ERRORS ON HAMMING BYTES

0320  D8          HAMMR  CLD             BINARY MODE
0321  A0 2A              LDY #$2A        DISPLAY 'IN=' AND WAIT FOR ENTRY
0323  20 70 E9           JSR KEPR        OF FUNCTION # FROM KEYBOARD
0326  6A 6A 6A           ROR, ROR, ROR   ROLL KEY IN THRU CARRY
0329  85 D8              STA FLAG        SWITCH FOR RELOCATION
032B  A2 00              LDX #$00
032D  20 A2 E8           JSR FNAM        GET FILENAME TO BE SEARCHED FOR
0330  20 A7 E7    H20    JSR TO          TAPE # AND 'TO'
0333  B0 FB              BCS H20         ON ERROR
0335  85 D9              STA NSAL        NEW START ADDRESS
0337  AD 1D A4           LDA ADDR+1      GET NSAH
033A  85 DA              STA NASH
033C  A0 66              LDY #$66
033E  20 96 E3           JSR CHER1
0341  A8                 TAY             Y=00

0342  A9 37              LDA #$37        OPEN CHANNELS FOR INPUT FROM
0344  8D 02 A8           STA DDRB        AUDIO INTERFACE
0347  AD 34 A4           LDA TAPIN       WHICH DEVICE?
034A  20 1C EE           JSR TIOSET      START THE ENGINE
034D  A9 EE              LDA #$EE        CA2 TO DATA IN
034F  8D 0C A8           STA PCR
0352  C8          H21    INY             Y=01
0353  84 D1              STY HASWI       HAMMING SWITCH   TO '1'
```

```
0355  A2 0 0          LDX #$10        COMPARE FOR A STREAM OF 17 SYNCS
0357  20 F2 03        JSR OHNE+2      START WITH SINGLE BIT RECOVERY
035A  C9 16     H22   CMP #$16        A SYNC?
035C  D0 F4           BNE H21         TRY WITH MORE BITS
035E  20 F0 03        JSR OHNE        NOW FULL BYTES
0361  CA              DEX             COUNT DOWN
0362  10 F6           BPL H22
0364  C9 2A           CMP #$2A        '*' ?
0366  D0 F2           BNE H22         MUST BE A SYNC IF NOT '*'

0368  A2 00           LDX #$00        COMPARE FOR FILENAME, 5 BYTES
036A  20 EC 03        JSR MIT         GET BYTE THE HAMMING-WAY
036D  DD 2E A4  H23   CMP NAME,X
0370  D0 E0           BNE H21         TRY A NEW BLOCK ON NO MATCH
0372  E8              INX
0373  E0 05           CPX #$05
0375  D0 F3           BNE H23         CONTINUE FILENAME
0377  20 EC 03  H24   JSR MIT         NOW GET OSAL, OSAH, NBL, NBH
037A  95 CE           STA $CE,X       X ≥ 05 !  DEPOSIT
037C  E8              INX
037D  E0 09           CPX #$09        4 BYTES DONE?
037F  D0 F6           BNE H24

0381  24 D8           BIT FLAG        DECIDE IF LOAD TO NEW LOCATION
0383  50 08           BVC H25         NO
0385  A5 D9           LDA NSAL        INSERT NEW START ADDRESS
0387  85 D3           STA PNTL        INTO POINTER TO MEMORY
0389  A5 DA           LDA NSAH
038B  85 D4           STA PNTH
038D  18        H25   CLC             CALCULATE ENDADDRESS+1
038E  A5 D3           LDA PNTL
0390  65 D5           ADC NBL
0392  85 D5           STA NBL         GIVING EOPL
0394  A5 D4           LDA PNTH
0396  65 D6           ADC NBH
0398  85 D6           STA NBH         GIVING EOPH
039A  20 4D EB        JSR CLRCK       CHECKSUM = 00

039D  20 EC 03  H26   JSR MIT         DATA RECOVERY MAIN LOOP W. HAMMING
03A0  91 D3           STA (PNTL),Y    STORE BYTE
03A2  20 4E E5        JSR CHEKA       ADD TO CHECKSUM
03A5  E6 D3           INC PNTL        INCR. POINTER TO MEMORY
03A7  D0 02           BNE H27
03A9  E6 D4           INC PNTH
03AB  A5 D3     H27   LDA PNTL        COMPARE FOR END
03AD  C5 D5           CMP NBL         = EOPL
03AF  A5 D4           LDA PNTH
03B1  E5 D6           SBC NBH         = EOPH
03B3  90 E8           BCC H26         DO IT AGAIN

03B5  C8              INY             Y = 01
03B6  20 EE 03        JSR MIT+2       FIRST SET D1 TO '01'
03B9  C9 2F           CMP #$2F        A '/' ? TERMINATOR
03BB  F0 03           BEQ H29         ALL WENT OKAY
03BD  4C 85 E3  H28   JMP CKERR       ERROR EXIT TO MONITOR
03C0  20 EC 03  H29   JSR MIT         COMPARE CHECKSUM
03C3  CD 1E A4        CMP CKSUM
03C6  D0 F5           BNE H28         ERROR
```

```
03C8  20 EC 03          JSR MIT
03CB  CD 1F A4          CMP CKSUM+1
03CE  D0 ED             BNE H28          ERROR
03D0  4C F8 E3          JMP $E3F8        RETURN TO MONITOR, END OF MAIN.

03EC  84 D0       MIT   STY HAMM         SUBROUTINE ENTRY WITH HAMMING
03EE  84 D1             STY HASWI
03F0  A0 08       OHNE  LDY #$08         ENTRY WITHOUT  HAMMING
03F2  2C 00 A8    H30   BIT DRB          WAIT TILL AFTER THE BURST
03F5  10 FB             BPL H30
03F7  A9 FF             LDA #$FF         START TIMER FROM 'FF'
03F9  8D 95 A4          STA DIV8
03FC  2C 00 A8    H31   BIT DRB          WAIT FOR NEXT BURST
03FF  30 FB             BMI H31
0401  AD 96 A4          LDA DIV64        & GET TIME  ELAPSED
0404  C9 E5             CMP #$E5         TIMING COMPARATOR IN THE RANGE
                                         $E3-E7 FOR GENERATING A CARRY ON '1'
0406  90 07             BCC H32          SKIP ON '00'-BIT
0408  A5 D0             LDA HAMM         COPY IN CHECK BITS FROM TABLE
040A  59 17 03          EOR TABD-1,Y
040D  85 D0             STA HAMM
040F  26 DB       H32   ROL FBYT         ROLL IN CARRY TO FORM A BYTE
0411  88                DEY              COUNT DOWN
0412  D0 DE             BNE H30          GET MORE BITS

0414  A5 D1             LDA HASWI        DECIDE FOR THE HAMMING-WAY
0416  F0 05             BEQ H33          WHEN COMING WITH THE FIRST NYBBLE
0418  30 03             BMI H33          WHEN COMING WITH THE SECOND NYBBLE
041A  A5 DB             LDA FBYT         TAKE THIS FOR FINAL BYTE
041C  60                RTS              OUT ON NO HAMMING

041D  08          H33   PHP              TO DECIDE LATER
041E  A5 D0             LDA HAMM         GET A STATUS FOR THE CHECK BITS
0420  84 D0             STY HAMM         RENEW WITH '00'
0422  F0 11             BEQ H35          ALL WENT OKAY, BUT 2ND TURN NECESSARY?
0424  30 05             BMI H34          SKIP FOR SINGLE BIT ERROR
0426  68 68             PLA, PLA         MORE THAN 2 BITS WENT WRONG, NOT
0428  4C 85 E3          JMP CHERR        CORRECTABLE, ERROR EXIT TO MONITOR

042B  29 07       H34   AND #$07         CUT OFF PARITY BIT
042D  A8                TAY              TAKE REST AS AN ADDRESSER
042E  A5 DB             LDA FBYT         SINGLE BIT REPAIR OF HAMMING BYTE
0430  59 10 03          EOR TABC,Y       FLIP THE BIT
0433  85 DB             STA FBYT         DEPOSIT CORRECTED BYTE

0435  A5 DB       H35   LDA FBYT
0437  28                PLP              DECIDE:FIRST OR 2ND NYBBLE?
0438  D0 0B             BNE H36          IT'S THE 2ND
043A  C6 D1             DEC HASWI        HAMMING SWITCH TO 'FF'
043C  0A 0A             ASL, ASL         ADJUST NYBBLE LEFTMOST AND DEPOSIT
043E  0A 0A             ASL, ASL
0440  85 DC             STA TEMPB
0442  4C F0 03          JMP OHNE         GET 2ND HAMMING BYTE

0445  29 0F       H36   AND #$0F         ISOLATE LSD
0447  05 DC             ORA TEMPB        COPY-IN FIRST NYBBLE, BYTE COMPLETE
0449  A0 00             LDY #$00         ADJUST Y
044B  60                RTS              OFF AND AWAY
044C  EA                NOP              RALOAD BYTE
```

Fortsetzung auf Seite 40

AIMPLOT - MESSWERTE PLOTTEN

E: This utility plots the results of measurements on the AIM printer at a speed of 9 dots per sec.. VALDOT converts a parameter in A into a dot position (hex 00≤A≤63), AIGRA does the printout.

Mit den hier vorgelegten Unterprogrammen wird der Drucker des AIM 65 zum Meßwertplotter, der etwa 9 Meßwerte/Sek. ausgibt. Das Unterprogramm VALDOT konfektioniert einen im Akku übergebenen Meßwert (im hexadezimalen Wertebereich 00-63 entspr. dezimal 0-99) auf die entsprechende Meßpunktposition. AIGRA besorgt den Ausdruck dieses Punktes. Der Anwender braucht seinen Meßwert also nur auf den darstellbaren hexadezimalen Wertebereich 00-63 umzurechnen.

Zur Arbeitsweise des Printers orientiere man sich im AIM USER's GUIDE auf den Seiten 7-19 ff. Dort wird insbesondere auch vor Manipulationen am timing des Druckers gewarnt. In dieser Hinsicht braucht der Anwender keine Befürchtungen zu hegen, denn dem Autor gelang ein neuer Einstieg in die Originalroutinen des Monitors mit seinen unveränderten Zeitkonstanten. Hinsichtlich der Kommentierung wird weitgehend auf das MONITOR PROGRAM LISTING hingewiesen.

```
0200  2C 11 A4   AIGRA   BIT PRIFLAG     ROUTINE ENTSPRICHT IPST IN $F045
0203  10 2A               BPL OUT        FÜR AUSGABE EINER ZEILE.
0205  20 CB F0            JSR PINT       INITIALISIERE
0208  20 66 02            JSR NIPSU
020B  A9 C1               LDA #$C1
020D  8D 0C A8            STA PCR
0210  20 A0 FF            JSR PAT23
0213  D0 08               BNE NIPO2
0215  20 A0 FF            JSR PAT23
0218  D0 03               BNE NIPO2
021A  4C 79 F0            JMP PRIERR
021D  20 30 02   NIPO2    JSR NPDOT
0220  20 30 02            JSR NPDOT
0223  AD 77 A4            LDA IDOT
0226  C9 0A               CMP #$0A       NUR 1 ZEILE
0228  90 F3               BCC NIPO2
022A  A9 E1               LDA #$E1
022C  8D 0C A8            STA PCR        MOTOR AUS
022F  60         OUT      RTS

0230  A9 00      NPDOT    LDA #$00       ROUTINE
0232  8D 01 A8             STA DRAH     ENTSPRICHT
0235  AD 0D A8   NDOTO    LDA IFR        PRNDOT IN
0238  29 02               AND #$02       $F087
023A  F0 F9               BEQ NDOTO
023C  AD 0C A8            LDA PCR
023F  49 01               EOR #$01
0241  8D 0C A8            STA PCR
0244  EE 77 A4            INC IDOT
0247  AD 79 A4            LDA IOUTU
024A  0D 00 A8            ORA DRB
024D  8D 00 A8            STA DRB
0250  AD 78 A4            LDA IOUTL
0253  8D 01 A8            STA DRAH
0256  A9 A4               LDA #$A4
```

Probeausdruck:

Von 0-99 ansteigende Meßwerte.

```
0258  8D 08 A8          STA T2L
025B  A9 06             LDA #$06
025D  8D 09 A8          STA T2H
0260  20 66 02          JSR NIPSU
0263  4C BA F0          JMP $F0BA       FÜHRE DEN REST DER ROUTINE
                                        PRNDOT AUS.

0266  A2 00       NIPSU LDX #$00        ROUTINE ENTSPRICHT IPSU IN $F0E3
0268  20 21 F1          JSR INCP
026B  BD 60 A4    NIPS1 LDA IBUFM,X
026E  CD 77 A4          CMP IDOT
0271  D0 16             BNE NIPS3
0273  AD 7A A4          LDA IBITL
0276  F0 08             BEQ NIPS2
0278  0D 78 A4          ORA IOUTL
027B  8D 78 A4          STA IOUTL
027E  D0 09             BNE NIPS3
0280  AD 7B A4    NIPS2 LDA IBITU
0283  0D 79 A4          ORA IOUTU
0286  8D 79 A4          STA IOUTU
0289  0E 7A A4    NIPS3 ASL IBITL
028C  2E 7B A4          ROL IBITU
028F  CA CA             DEX, DEX
0291  10 D8             BPL NIPS1
0293  4C 18 F1          JMP $F118       ZUM REST DER ROUTINE IPSU

CALCULATE DOT POSITION FROM VALUE IN A

0296  48          VALDOT PHA             RETTE PARAMETER
0297  A2 00             LDX #$00        PRINT-BUFFER KOMPLETT LÖSCHEN
0299  20 38 F0          JSR OUTPR
029C  A2 00             LDX #$00        X ALS ADDRESSER
029E  68                PLA             HOLE WERT ZURÜCK
029F  C9 05       DIVA  CMP #$05
02A1  90 05             BCC FEIN        REST < 5
02A3  E9 05             SBC #$05        DIVIDIERE DURCH 5, BIS REST < 5
02A5  E8                INX             ADDRESSER + 1
02A6  D0 F7             BNE DIVA        SPRINGE IMMER
02A8  18          FEIN  CLC             ADDITIONSVORBEREITUNG
02A9  2C 82 EF          BIT #$04        OPERAND AUS DEM FESTWERTSPEICHER
02AC  08                PHP             RETTE STATUS
02AD  49 03             EOR #$03        INVERTIERE 2 BITS
02AF  69 01             ADC #$01        RECHNUNG IM 4-ER KOMPLEMENT
02B1  28                PLP             STATUS ZURÜCK
02B2  F0 02             BEQ SPEI        ÜBERSPRINGE
02B4  29 03             AND #$03        GGFS. BIT 2 MASKIEREN
02B6  9D 60 A4    SPEI  STA IBUFM,X     DRUCKSPEICHER
02B9  8A                TXA             IST X ...
02BA  2C 97 F0          BIT #$01        GERADE ODER UNGERADE? DIREKTOPERAND
02BD  D0 08             BNE ZUR         WENN UNGERADE
02BF  BD 60 A4          LDA IBUFM,X
02C2  69 05             ADC #$05        ADDIERE ZUR DOT-POSITION
02C4  9D 60 A4          STA IBUFM,X
02C7  60           ZUR  RTS
```

Testprogramm:

```
A9 00           LDA #$00   ANFANGSWERT
85 00           STA $00    ZÄHLER
20 96 02   T1   JSR VALDOT RECHNE
20 00 02        JSR AIGRA  DRUCKE
E6 00           INC $00    ZÄHLER
A5 00           LDA $00    ZÄHLER
C9 64           CMP #$64   SCHON 100?
D0 F2           BNE T1     NEIN
00              BRK        ENDE
```

Das im Kasten abgedruckte Erprobungsprogramm plottet von 0-99 (dez.) aufsteigende Meßwerte, die im Akkumulator übergeben werden.

R.L.

A I M G R A P H - GRAPHICS CAPABILITY FOR THE AIM PRINTER

E: This program lends 63 graphics characters to the AIM printer. You may even create other character fonts like Arab or Chinese by only altering the contents of the table.

Beim Studium des AIM MONITOR PROGRAM LISTING stellt man fest, daß das ROM ab Zelle F2E1 auch ein character generator ROM ist. Die Punktmatrix ist in 5 Tabellenabschnitten für die Spalten enthalten. Dabei wird die Tabelle mit dem hexadezimalen Wert des abzudruckenden Zeichens als Index angesteuert. Das ist wiederum eine fast klassische Lösung, wie man Hardware durch Software ersetzen kann. Unser Programm verfolgt diese Linie weiter und düpiert den Programmablauf an der Stelle, wo der Monitor vom Unter-Programm INCP zurückkommt. Der in $A47D und $A47E aufgebaute Pointer für das zu verwendende Punktmuster wird auf die entsprechende Stelle unserer ab Zelle 0300 niedergelegten Tabelle manipuliert.

Nach dieser Methode kann man selbstverständlich beliebige andere Zeichensätze erzeugen, sogar auch Mehrfachsätze im Direktzugriff. Dem Autor mangelte es an Zeit, mit diesen Möglichkeiten zu spielen, also fehlt auch die übliche Grafik-Ausgabe eines hübschen Mädchens. Die Leser werden das sicher bald besorgen und sicher auch Anstrengungen unternehmen, Spiele wie z.B. LIFE auf den Drucker zu bringen.

AIMGRAPH kann auf eine fast identisch gleiche Subroutine AIGRA zurückgreifen wie das Programm AIMPLOT in diesem Heft. Nur der Befehl für die Zeilenzählung wird wie folgt geändert:

```
0226  C9 5A              CMP #$5A        FÜR 90 DOTS
```

Das aufgerufene Unterprogramm NIPSU ist durch das nachfolgende NIPSU2 zu ersetzen. Wer AIMPLOT und AIMGRAPH zusammen betreiben möchte, kann einen Softwareschalter in AIGRA vor der Punktzählung und entsprechend auch in den sich sehr ähnelnden Unterprogrammen NIPSU/NIPSU2 befragen.

```
0266  A2 00       NIPSU2  LDX #$00        ENTSPRICHT ETWA IPSU IN $F0E3
0268  20 21 F1            JSR INCP
026B  BD 60 A4    NIPS1   LDA IBUFM,X
026E  29 3F               AND #$3F        BESCHNEIDE ALS ADDRESSER
0270  A8                  TAY
0271  18                  CLC             ADDITIONSVORBEREITUNG
0272  A9 1F               LDA #$1F        UMRECHNEN AUF NEUE TABELLENBASIS
0274  6D 7D A4            ADC JUMP        VON INCP ERRECHNETE ADRESSE
0277  85 00               STA PNTL        MACHE $00/01 ZUM TABELLENPOINTER
0279  A9 10               LDA #$10        DITO FÜR HOHE ADRESSE
027B  6D 7E A4            ADC JUM+1
027E  85 01               STA PNTL+1
0280  B1 00               LDA (PNTL),Y    HOLE DOT-MUSTER AUS TABELLE
0282  2C 7C A4            BIT IMASK       DOT GESETZT?
0285  F0 16               BEQ NIPS2       ... WIE IM ABSCHNITT IPSU
0287  AD 7A A4            LDA IBITL
028A  F0 08               BEQ NIPS3
028C  0D 78 A4            ORA IOUTL
028F  8D 78 A4            STA IOUTL
0292  D0 09               BNE NIPS2
0294  AD 7B A4    NIPS3   LDA IBITU
0297  0D 79 A4            ORA IOUTU
029A  8D 79 A4            STA IOUTU
```

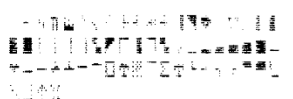
```
029D  0E 7A A4     NIPS2   ASL IBITL
02A0  2B 7B A4             ROL IBITU
02A3  CA CA                DEX,DEX
02A5  10 C4                BPL NIPS1
02A7  4C 18 F1             JMP $F118        VERWENDE REST VON IPSU
```

```
<M>=0300 00 80 C0 E0 F0 F8 FC 40 0C 20 10 10 10 08 04 FE   CHARACTER
< > 0310 18 AA 02 C6 1C 00 10 10 00 0E 1E FE 02 80 18 82   GENERATOR
< > 0320 00 10 00 04 F4 00 10 10 00 10 28 10 00 06 1C 80   TABLE
< > 0330 1C FE FE FE FE FE 00 00 00 00 0E FE FE 00 02 0E   FOR A
< > 0340 00 80 C0 E0 F0 F8 FC 40 1C 20 10 38 10 08 04 82   GRAPHICS
< > 0350 8C 54 02 AA 88 00 10 10 00 0E 1E 80 0C 80 04 CC   FONT
< > 0360 00 10 1C FC C0 00 20 08 00 10 10 10 00 0E 3C 60
< > 0370 44 00 FE FE FE FE FE 00 00 00 0E 3E 02 00 02 0E   BUIT UP AND
< > 0380 C0 80 C0 E0 F0 F8 FC 40 38 20 10 FE 1E 08 04 82   IN SUCCESSION
< > 0390 FE AA 02 92 FE 1E 1E F0 F0 0E 1E 80 01 80 FE 01   AS TABLES
< > 03A0 00 F0 00 04 E8 06 C0 06 FE FE 7C FE 00 1E EE 01   COLØ THRU COL4
< > 03B0 82 00 00 FE FE FE 00 FE 00 00 FE 1E 02 FE 02 FE   IN MONITOR
< > 03C0 20 80 C0 E0 F0 F8 FC 40 1C 20 10 38 10 08 04 82   PROGRAM
< > 03D0 8C 54 02 82 88 10 00 00 10 0E 1E 80 60 80 04 CC
< > 03E0 00 10 70 FC C0 08 00 00 10 00 10 10 FE 3E 3C 0C   INVERSE
< > 03F0 44 00 00 00 FE FE 00 00 FE 00 E0 0E 02 FE 02 E0   REPRESENTATION
< > 0400 00 80 C0 E0 F0 F8 FC 40 0C 20 10 10 10 08 04 FE   POSSIBLE BY
< > 0410 18 AA 02 82 1C 10 00 00 10 0E 1E 80 80 FE 18 82   EXOR-ING
< > 0420 00 10 00 04 F4 10 00 00 10 00 28 10 FE FE 1C 02   TABLE CONTENTS
< > 0430 1C 00 00 00 00 FE 00 00 00 FE E0 06 02 FE FE E0
```

Wie aus der Instruktion in $026B *ersichtlich, versieht das Programm die im Printer Buffer ab* $A460 *stehende Information mit einer graphischen Bedeutung. Diese Information per Programm dorthin zu bringen, hat keinerlei Schwierigkeiten. Dagegen wurde die Frage noch nicht gelöst, wie man vom EDITOR aus direkt und interaktiv mit einer USER OUTPUT FUNCTION zum graphischen Abdruck der offenen Textzeile gelangt. Hierzu sind Vorschläge sehr willkommen.*

Zur Erprobung des AIMGRAPH folgendes Programm zum Abdruck der ersten 20 ASCII-Zeichen ($20-$33 *entsprechend Zwischenraum bis 3). Durch Ändern des Anfangswertes im Akkumulator gelangt man zum Abdruck des gesamten Zeichensatzes.*

```
0500  A2 LDX #00          ADDRESSER          Für ASCII $20-$5F
0502  A9 LDA #20          ASCII = BLANK      ergibt sich folgender
0504  9D STA A460,X       IBUFM,X            Zeichensatz:
0507  38 SEC              ADDIERE 1
0508  69 ADC #00
050A  E8 INX
050B  E0 CPX #14          20 ZEICHEN?
050D  D0 BNE 0504
050F  20 JSR 0200         DRUCKE
0512  00 BRK              ZUM MONITOR ZUR.
```



R.L.

```
XX22  A5 E5               LDA $E5          NOW FOR THE OFFSET OF DATA
  24  65 E2               ADC $E2
  26  91 E3               STA (E3),Y       AND STORE
  28  A5 E6               LDA $E6
  2A  69 00               ADC #$00         ADD A POSSIBLE CARRY
  2C  C8                  INY
  2D  91 E3               STA (E3),Y
  2F  18                  CLC              FLAG 'INSIDE TABLE'
  30  60                  RTS
```

Fortsetzung von S. 44
ASP

ABSIND-BOTLN

NAME	WERT	DEFIN.	REFERENZEN							
ABSIND	FC5C	3739	3729							
ABSOL	FCA6	3773	3732							
ABSOL1	FCB2	3778	3775							
ABSX	FC72	3749	3741							
ABSY	FC63	3742								
ABSY1	FC6E	3747	3744							
ACCUM	FC23	3715								
ACR	A80B	216	876	2640	2696					
ADDA	F92A	3416	3143	3323						
ADDA1	F933	3421	3419							
ADDBK1	EDBD	2093	666	2059	4161					
ADDBLK	EDBA	2092								
ADDIN	EAAE	1664	529	969	1025	1149	4009			
ADDNE	EAB1	1665	1222							
ADDN1	EAB7	1668	1675							
ADDN2	EAC7	1676	1670	1672						
ADDN3	EADC	1686	1679							
ADDN4	EAE8	1695	1699							
ADDN5	EAF7	1702	1691							
ADDN6	EAFD	1706	1701	1711						
ADDN7	EB0D	1714	1724							
ADDN8	EB2B	1729	1716	1721						
ADDR	A41C	124	519	522	524	534	594	595	597	603
			604	618	620	713	715	803	807	949
			951	972	974	1030	1032	1681	1683	1696
			1707	1714	1718	1722	1775	1778	3044	3060
			3062	3064	3071	3083	3094	3356	3358	3400
			3401	3424	3426	3544	3547	3564	3566	3655
			3657	3875	3903	3905	4047			
ADDRS1	F910	3400	780	3453						
ADDS1	E55D	908	832	947						
ADDS1A	F916	3402	3397							
ADD1	E565	911	909							
ADFLD	0133	86	3687	3693	3707	3751	3765	3829	3833	3936
			3940							
AD1	F928	3415	3122	3479	3516	3617				
ASSEM	0000	46	505							
ATBOT	F8E9	3373	3129	3134	3176	3250				
ATEND	F8F9	3385	3525							
ATTOP	F8DB	3363	3110	3117						
AT01	F8F7	3381	3365	3368	3376	3378	3388	3391		
AT02	F8F5	3379	3389	3392						
BACKWD	FDD9	3923	3921							
BASIEN	B000	47	509							
BASIRE	B003	48	509							
BKCKSM	F1E7	2663	2073	2623						
BKCK1	F1F1	2667	2674							
BKCK2	F20F	2680	2617	2677						
BKCK3	F21A	2686	2683							
BKERR	E62F	1021	1018							
BKFLG	A410	113	351	772	775	784	790	849	1125	
BKOK	E634	1023	1020							
BKO2	E64C	1034	1027							
BKS	0100	65	126	1031	1033	1040	1041	1141	1182	1186
BLANK	E83E	1298	518	533	570	572	578	662	779	1005
			1039	1055	1297	1396	3661	3669	3870	3878
			4128							
BLANK2	E83B	1297	3050	3663	3664	3863	3864			
BLK	0115	150	646	2061	2067	2092	2103			
BLKO	0168	152	922	2642	4160					
BOTLN	00E1	55	3046	3350	3351	3375	3377	3385	3386	3474
			3477	3487	3490	3492	3521	3522	3524	3528

A I M 6 5
MONITOR CROSS REFERENCE LIST
Dietmar Fay, Geleitsstr. 10/1008
6000 Frankfurt 70.

BOTLN–CLRLUP

NAME	WERT	DEFIN.	REFERENZEN							
BOTLN			3530							
BRCOMP	FD86	3889	3825							
BRKA	E61B	1012	506							
BRKK	E6E5	1125	509							
BRK1	E620	1014	1022							
BRK2	E6F3	1132	1136							
BRK3	E6F1	1131	1099	1107	1144					
BRK4	E6FA	1135	1100	1108						
BRNCHC	FD0F	3825	3809							
BT	F721	3147	3609							
BYTESM	A42F	88	3813	3814	3844	3855	3873	3884	3897	3930
CBUFF1	F1E2	2657	2651	2653						
CDO2	FA8F	3593	3596							
CFLG	F8B2	3338	3311							
CFND1	FAA0	3600	3594							
CGA	E5EE	983	505							
CGALL	E5FC	996	980	984	988	992	1002			
CGPC	E5D4	969	505							
CGPC0	E5D7	970	1151							
CGPC1	E5DD	972	970	3860	4107					
CGPS	E5EA	979	506							
CGS	E5FA	995	506							
CGX	E5F2	987	506							
CGY	E5F6	991	506							
CH	0130	85	3962	3965	3972					
CHAR1	F5AD	2991	2991	2947						
CHAR2	F5B3	2993	2993	2949						
CHEKA	E54E	898	751	887						
CHEKAR	E54B	897	615	617	619	712	714			
CHNG	F876	3311	3609							
CHNGG	E2A0	570	505							
CHNG1	E2A6	572	586							
CHN1	F87C	3313	3318							
CHN2	F88C	3319	3315							
CHN3	F8A9	3332	3334							
CHN4	F8AF	3335	3331							
CH2	E2B8	581	574							
CH3	E2C5	588	576							
CH4	E2C0	584	579	582						
CKB	E76B	1181	353							
CKBUFF	F1D2	2649	2628	2667	2681					
CKB1	E780	1191	1185	1188						
CKB2	E76D	1182	1192							
CKERR	E385	685	631	634	724	731	734			
CKERO	E38E	688	685	2088						
CKERO0	E394	690	1729	3788						
CKER1	E396	691	680	696						
CKER2	E3A3	697	693							
CKFREQ	EE75	2190	2161	2162	2168	2184	2186	2187	4174	
CKF1	EE7A	2192	2193							
CKF2	EE81	2195	2214							
CKF3	EE99	2207	2202							
CKF3A	EE9D	2209	4068							
CKF4	EEA1	2212	2191	2213						
CKSUM	A41E	125	630	633	730	733	836	838	900	901
			903	957	959	1682	1726	1750	1751	3040
			3054							
CLR	EB44	1743	2436	3191	3261	3324	3570	3580	3859	4040
			4127	4129						
CLRBK	E6FE	1139	507							
CLRCK	EB4D	1749	614	700	802	939				
CLRLUP	FBE9	3687	3689							

NAME	WERT	DEFIN.	REFERENZEN							
CMPBR1	FDBB	3911	3909							
CNTH30	A417	120	421	1861	1870	1875				
CNTL30	A418	121	1859	1872	4018					
CODFLG	A437	92	3660	3700	3868					
COL0	F2E1	2790	2790	2790	2787					
COL1	F321	2800	2800	2800	2787					
COL2	F361	2810	2810	2810	2787					
COL3	F3A1	2820	2820	2820	2787					
COL4	F3E1	2829	2829	2829	2787					
COM	FA78	3582	3587							
COMB	E1C4	501	501	501	501	501	501	476		
COMCN1	000B	3605	3592							
COMIN	E1A1	482	687	1451	1740	2089	2440	3307		
COMM	FA88	3590	3586							
COMPBR	FD9E	3899	3891							
COMTBL	FAAC	3607	3607	3607	3607	3593				
CONVRT	FD12	3829	3822	3839	3894	3901				
CORR	FB00	3631	3631	3631	3800					
COUNT	A419	122	822	833	1200	1205	1212	3215	3341	3459
			3493							
CPIY	A42A	139	141	171	172	1651	1657	1677	1700	1704
			1758	1764	1777	1780	1784	1786	1791	1797
			1803	1809	1837	1844	1845	1846	1854	2124
			2125	2126	2151	2152	2153	2710	2740	3486
			3489	3616	3620					
CR	000D	258	589	1078	1087	1090	1243	1345	1360	1381
			1399	1407	1471	1546	1585	1669	2101	2269
			2390	3120	3140	3155	3164	3198	3276	3292
			3314	3435	4057					
CRA	AC01	247								
CRB	AC03	249								
CRCK	EA24	1576	517	550	689	776	1154	1166	1373	1730
			1735	2852	3103	3167	3217	3224	3283	3598
			3658	3867	4134					
CRCK1	EA2C	1580	1578							
CRCK2	EA3B	1587	1584							
CRLF	E9F0	1546	545	799	862	1567	3166	4074	4099	4153
			4154							
CRLOW	EA13	1563	425	514	555	783	873	971	1038	1054
			1353	1581	3034	3037	3077	3233	3242	3306
			3576							
CR2J	EA23	1571	1549	1552	1554	1556				
CUREAD	FE83	4021	1372	1455	1470					
CURPOS	A416	119	1283	1289	1583	1745	2377	2383	2384	2393
CURPO2	A415	118	649	678	831	1257	1258	1263	1270	1665
			1744	1972	1975	2095	2272	2281	2305	3210
			3259	3264	3286	3287	3296	3321	4023	4035
DATIN	000E	223	2118							
DATOUT	000C	224	2693							
DDRA	A803	208								
DDRA2	A481	177								
DDRB	A802	207	2115							
DDRB2	A483	179								
DEBKEY	ED2A	2018	1889	1993						
DEBK1	ED2C	2019	1924							
DEBTIM	1388	236	2021	2023						
DEHALF	EC23	1870	1841	1851						
DELAY	ECOF	1859	1431	1840	1847	1850	2222	2227	2238	2245
DE1	EC18	1862	2024							
DE2	EC1B	1863	1865	1878	2464					
DIBUFF	A438	154	1285	1706	2096	2098	2107	2109	2280	2291
			3158	3202	3443					

DILINK-ENDE2

NAME	WERT	DEFIN.	REFERENZEN							
DILINK	A406	105	2262							
DISASM	F46C	2854	554	1159	2850	3866				
DISFLG	A40F	112	552	1117						
DISPLY	FD6E	3875	3881							
DIV1	A494	196								
DIV64	A496	198	2164	2167	2172	2175				
DIV8	A495	197								
DI1024	A497	199	4164							
DLNE	F74C	3171	3608							
DNNO	F6D8	3110	3149	3221	3573					
DNPA7	A484	183								
DONE	E790	1205	356	1157	3246					
DON1	E7A0	1214	1207							
DOWN	F724	3149	3608							
DOW1	F6E3	3114	3111							
DOW2	F6E8	3116	3121							
DPPA7	A485	184								
DRA	A80F	220								
DRAH	A801	206	2446	2458	2466					
DRA2	A480	176	376	1892	2003					
DRB	A800	205	370	412	417	878	1095	1097	1103	1105
			1303	1429	1838	1842	2055	2057	2143	2190
			2192	2212	2223	2225	2236	2244	2455	2456
			2467	2469	2633	2635				
DRB2	A482	178	1893	1902	1987	2006				
DUK2	E5A4	946	952							
DUMP	E43B	772	505							
DUMPKI	E587	933	796							
DUMPTA	E56F	920	1337							
DUMPT1	E57B	925	929							
DU0	E447	777	778							
DU1	E444	776	847							
DU1A	E46D	792	785							
DU1B	E452	781	782							
DU10	E4DB	843	812							
DU10A	E4F8	856	861							
DU11	E50A	864	4097							
DU12	E511	867	872							
DU13	E520	873	641	737	866	869	966	1092	4145	
DU14	E529	877	4092							
DU2	E47D	799	841							
DU6	E49F	816	809							
DU7	E4A0	817	815							
DU8	E4A2	819	814							
DU9	E4B9	829	834							
EDI	F6B6	3086	3057	3068						
EDIT	F639	3034	504							
EDI0	F644	3038	3039							
EDI1	F653	3044	3049							
EDI2	F663	3051	3052							
EDI2B	F6CC	3097	3091							
EDI3	F673	3057	3058							
EDI4	F680	3062	3055							
EDI5	F68D	3068	3069							
EDI6	F69B	3074	3081							
EDI7	F6AA	3080	3072							
EDI8	F6AE	3082	3073							
EMSG1	E06C	324	324	3035						
EMSG2	E072	325	3131	3571						
END	00E5	57	3063	3065	3070	3080	3387	3390		
ENDERR	FA5C	3570	3136	3254	3291	3531				
ENDE2	FA6F	3577	3575							

ENPA7-GET1

NAME	WERT	DEFIN.	REFERENZEN							
ENPA7	A486	185								
ENTRY	FA8D	3592	3079							
EPPA7	A487	186								
EQS	00BD	262	305	306	312	313	317	318	321	
EQUAL	E7D8	1249	996	1664	4052					
ERR	F495	2877	2862	2864						
ERRFLG	FD2B	3841	3831	3836						
ERRJMP	FDD6	3922	3895	3902	3925	3928				
ERROR	FA72	3578	3282							
ERRORM	FCC5	3788	3738	3740	3761	3762	3767	3770	3798	3823
			3922	3995						
ERRO	FA78	3580	3113	3186	3222	3577	3582	3599		
ESCAPE	001B	260	1420	1433	1448					
EVAL	FC0E	3706	3701							
FCH	F81E	3267	3258							
FCHAR	F80C	3258	3312	3609						
FCHA1	F80F	3259	3317							
FC1	F823	3269	3281							
FC2	F82E	3274	3293	3316						
FC3	F834	3276	3271	3273						
FC4	F843	3283	3277							
FC5	F849	3285	3275							
FC6	F84E	3287	3297							
FC7	F853	3289	3301							
FC8	F85A	3292	3290							
FC9	F868	3298	3295							
FNAM	E8A2	1356	1314	1319	1336	1341				
FORMA	0116	76	2881	2931	2945					
FORMDS	FD45	3859	3849							
FORMD1	FD58	3866	3862							
FORMD2	FD69	3873	3865							
FORM1	FD7D	3884	3869							
FORWRD	FDE0	3926	3919							
FROM	E7A3	1217	777	3038						
GAP	A409	107	2699							
GCNT	E785	1197	544	1153	3241					
GCN1	E78C	1200	1198							
GETA1	EE2B	2150	2155							
GETFMT	F499	2879	2876							
GETID	E425	760	716	937						
GETKD0	EC38	1881	1984							
GETKEY	EC40	1888	1445	1978	1982					
GETKY	EC43	1889	1908	1921	1922					
GETK0	EC55	1898	1907							
GETK00	EC67	1909	1904							
GETK1	EC71	1915	1895							
GETK1B	EC80	1922	1917							
GETK10	ECEC	1984	1966							
GETK11	ECC9	1968	1964							
GETK12	ECD2	1972	1977							
GETK13	ECE1	1979	1969							
GETK14	ECEB	1983	1980							
GETK2	EC82	1924	1419	1920						
GETK3	EC8D	1931	1934							
GETK4	EC93	1935	1932							
GETK5	ECA4	1944	1940							
GETK6	ECB9	1957	1953							
GETK7	ECBE	1960	1938	1947	1951					
GETK8	ECBF	1963	1943	1956	1959					
GETTAP	EE29	2149	702	720	1495	2043	2050	2130		
GETTTY	EBDB	1834	1432	1443	1505					
GET1	EBE2	1838	1839							

GET3-IPS3

NAME	WERT	DEFIN.	REFERENZEN
GET3	EBED	1842	1849
GID1	E427	761	764
GO	E261	543	504
GOBK	E26D	547	359
GOBKO	E278	551	548
GOBK1	E286	556	546 553
GOERR	E608	1001	998
GOGO	FA4A	3560	3437 3445 3471 3542
GOGO1	FA5B	3568	3561
GOTIT	FE5F	3998	3992
HATCH	FCB6	3781	3726
HATCJ	FC3D	3726	3709
HEX	EA7D	1632	1715 3830
HIST	A42E	149	347 349 1056 1058
HISTM	A42E	87	88 89 90 91 92
HISTP	A414	117	345 1053 1071
IBITL	A47A	164	2526 2534 2595
IBITU	A47B	165	2531 2535 2593
IBUFM	A460	157	1286 2392 2404 2519
ICOL	A475	159	2476 2556 2559 2567 2573 2576 2581 2597
IDIR	A474	158	2474 2553 2570 2585
IDOT	A477	161	1882 2429 2453 2481
IER	A80E	219	
IEVEN	F486	2867	2860
IFR	A80D	218	1863 2447 2724 4167 4179
IMASK	A47C	166	2479 2524 2588
IMMED1	FCC1	3786	3783
IN	F764	3184	3608
INALL	E993	1489	611 640 744 745 747 1077 1080 1083 1086 1089 3192 4114 4141
INCP	F121	2553	2517
INCS2	E566	913	840 851
INDX	FC81	3756	3753
INFLG	A412	115	741 1236 1310 1400 1489 1576 2072 2080 2649 3230 4089
INL	F76D	3187	3184 3234
INLOW	E8F8	1399	1404
INLUP	FE35	3974	3978
INPU	F7CB	3229	3608
INPU1	F7D8	3234	3232 3236
INTAB1	E743	1171	1171 1171 369
INTAB2	E752	1175	1175 375
INTAB3	E756	1177	1177 1177 380 383 389
INO2	F77A	3192	3197 3205 3211 3328 4117
INO2A	F785	3196	4113
INO3	F7A8	3212	3199
INO3A	F7B9	3219	3105
INO3B	F799	3206	3201
INO5	F7C5	3223	3214 3216
IOFFST	A476	160	2478 2562 2565 2579 2582 2605
IOUTL	A478	162	2457 2528 2529 2591
IOUTU	A479	163	2454 2532 2533 2540 2542 2592
IPO0	F050	2415	1883
IPO2	F066	2426	2418 2420 2431
IPO3	F073	2432	4045
IPO4	F078	2434	2412
IPST	F045	2411	2387 4126
IPSU	F0E3	2516	2414 2463
IPS0	F04A	2413	1981
IPS1	F0E8	2519	2538
IPS2	F10E	2534	2525 2530
IPS3	F105	2531	2527

IRQV1-MEM3

NAME	WERT	DEFIN.	REFERENZEN							
IRQV1	E078	329	4187							
IRQV2	A404	102	329							
IRQV3	E154	439	1177							
IRQV4	A400	100	445							
IRQ1	E163	448	443							
IRQ2	E17F	463	355							
ISX	FE03	3946	3942							
JD1	E723	1156	1167							
JD2	E72B	1159	1155							
JD3	E73C	1166	1164							
JD4	E742	1168	1158							
JMPR	E1C1	497	495							
JTBL	FAB8	3608	3608	3608	3608	3608	3608	3601	4103	
JUMP	A47D	167	492	494	497	2522	2601	2603		
KDISA	E70A	1147	508	1150						
KEP	E7AF	1225	516	656	664	1013	1132	1221	1231	1378
			1467	1738	2439	3036	3132	3326	3572	
KEPR	E970	1467	844	1309	1330	1359				
KEYF1	010C	70	511							
KEYF2	010F	71	511							
KEYF3	0112	72	511							
KIFLG	F8B6	3340	3171	3187						
KI2	F8B8	3341	3339							
KMASK	A42A	171	1913	1995	2007	2020				
LDAY	EB58	1758	535	884	2523	3469	3540	3876	4048	
LDIY	A42A	141	1761	1763	1766	1768	1769			
LENGTH	00EA	60	1162	2883	2929	2967	3161	3179	3212	3431
			3456	3457	3495	3497	3503			
LF	000A	259	1557	2248	3196					
LL	E8FE	1404	483	688	879	3578	3584	4125		
LMNEM	0117	77	2908	2915						
LOAD	E2E6	609	505							
LOADKI	E3A4	700	1320							
LOADK1	E3A7	701	706	718						
LOADK2	E3AA	702	707							
LOADK3	E3B7	708	704							
LOADK5	E3D1	719	728							
LOADK6	E3D3	720	726							
LOADK7	E3E8	729	722							
LOADTA	E32F	645	651	671	1315					
LOADT2	E364	669	674							
LOAD1	E2E9	611	613	635						
LOAD1A	E349	657	661							
LOAD2	E306	624	627							
LOAD4	E321	636	622							
LOAD5	E323	637	639							
LST	F7E1	3240	3609							
LST01	F7F0	3245	3251							
LST02	F7F8	3248	3244							
LST3	F803	3252	3247							
LT10	EA5A	1610	1608							
MATCH	FE51	3990	3986							
MATCH1	FE5C	3995	3989							
MCM2	E196	476	479							
MCM3	E1AC	488	477							
MCNT	0020	500	475							
MEIN	E24D	531	1009							
MEM	E248	529	504							
MEMERR	EB33	1735	583	755	3084	3567				
MEM1	E24F	532	526							
MEM2	E251	533	539							
MEM3	E260	540	530							

NAME	WERT	DEFIN.	REFERENZEN
MNEENT	FB9E	3654	508
MNEM	FE06	3951	3665
MNEML	F5B9	2995	2995 2995 2907 3985
MNEMR	F5F9	3005	3005 3005 2909 3991
MNNDX1	F4AF	2891	2900
MNNDX2	F4B3	2894	2897
MNNDX3	F4BA	2899	2890 2892
MODE	F55B	2978	2978 2978 2869
MODEM	FBC1	3666	4004
MODE2	F59F	2989	2989 2989 2880
MOFF	00E0	228	1173 2118 2432 2693
MON	00C0	227	2415
MONCOM	E1E5	504	504 504 504 504 491 493
MONRAM	A400	98	1127 1130 1134 1760 1762
MOVAD	0126	81	83 84 3904 3908 3910 3913 3916 3955
			3975 3976 3984 3990
MPRST	0010	230	4168
MREAD	FAD0	3612	1498
MR11A	F512	2941	2938
MSP12	0002	231	2448
MTBL	F2D7	2787	2787 2787 2787 2787 2600 2602
MT2	0020	232	1864
M1	E000	305	305 515 655 663 679 690 691 843
			1012 1131 1135 1220 1225 1308 1329 1358
			1377 1737 2438 3035 3131 3325 3571
M10	E02D	313	313 1329
M11	E031	314	314 314 314 1737
M12	E03B	315	315 315 315 315 2438
M3	E005	306	1220 3325
M4	E008	307	307 307 307 307 307 307 515
M5	E01C	308	308 843
M6	E021	309	1135
M7	E024	310	1131
M8	E027	311	1012
M9	E02A	312	1308
NAME	A42E	144	149 670 761 765 767 925 1385 1393
NAM0	E8CF	1377	1357
NAM01	E8D6	1380	1388
NAM02	E8E9	1390	1382 1384
NAM03	E8EB	1391	1395
NAM04	E8F5	1396	1392
NEWCOL	F163	2590	2560 2568 2577 2583
NEWROW	F160	2588	2571
NHIS	E688	1066	350 1060
NH1	E690	1071	1069
NMIV1	E075	328	4187
NMIV2	A402	101	328 381 384 390
NMIV3	E07B	332	1177
NMI4	E0B1	355	357
NMI5	E0B4	356	352 354
NOUT	EA51	1605	1602
NOWLN	00DF	54	3076 3119 3137 3153 3289 3346 3347 3357
			3359 3363 3366 3373 3374 3395 3396 3407
			3408 3411 3417 3418 3420 3423 3425 3436
			3444 3470 3513 3535 3536 3538 3541 3560
			3563 3565 3615
NOWS1	F909	3395	3532
NPUL	A40A	109	2718 2774
NULLC	00FF	257	1559 2250
NUMA	EA46	1597	536 895 958 960 1057 1059 1593 2104
			2903 2942 3877
NXTADD	E2CD	592	588 757 798 1007 1736 3885

NXTA1-OUTT1

NAME	WERT	DEFIN.	REFERENZEN							
NXTA1	E2DA	598	596							
NXT5	E60D	1005	506							
OK	FDE7	3929	3924	3927						
OLDLEN	00E9	59	3162	3189	3320	3432	3454	3505	3506	
ONEBYT	FD3E	3854	3807							
ONEKEY	ED05	1999	1915							
ONEK1	ED09	2001	2015							
ONEK2	ED0B	2003	1901	2000						
ONEK3	ED1C	2009	2012							
ONEK4	ED29	2016	2010							
OPCMP1	FCD5	3796	3793							
OPCODE	A434	91	3801	3802	3837	3846	3912	3914	3915	3917
			3923	3926	3952	3953	3954	4003		
OPCOMP	FCCB	3792	3682	3716	3723	3725	3735	3737	3746	3748
			3750	3757	3764	3772	3777	3779	3785	3787
OP03	F144	2573	2554							
OP04	F130	2562	2557							
OP05	F150	2579	2575							
OP06	F15D	2585	2580							
OP07	F13F	2570	2563							
OUTALL	E9BC	1510	1547	1558	1560	1610	2920	2948	2951	2966
			3157							
OUTA1	E9C8	1518	1514							
OUTA2	E9D0	1523	1519							
OUTA3	E9E2	1534	1524							
OUTA4	E9EA	1539	1535							
OUTCK	E538	887	823	825	827	942	944			
OUTCKS	E531	882	829	946						
OUTCK1	E53B	888	837	839	855	857	859	938		
OUTCK2	E547	894	891							
OUTDD1	EF7B	2331	431	2293	2301	2311	4027	4039		
OUTDD2	EF87	2340	2342							
OUTDD3	EF8B	2343	2339							
OUTDIS	EF05	2267	1177	1244	2252					
OUTDP	EEFC	2258	1486							
OUTDP1	EF02	2262	1262	1480	1588					
OUTD1	EF14	2274	2270							
OUTD1A	EF17	2275	4042							
OUTD2	EF20	2280	2276							
OUTD2A	EF2F	2287	1269	2284						
OUTD3	EF33	2290	2297							
OUTD4	EF48	2299	2283							
OUTD5	EF56	2307	2273	2314						
OUTD7	EF76	2325	2285	2304	2308	2315	4041			
OUTFLG	A413	116	792	864	889	1331	1408	1511	1550	1564
			1569	2100	2102	2105	2650	3163	4094	
OUTLOW	E901	1407	1566							
OUTLUP	FE30	3972	3980							
OUTL1	E906	1409								
OUTPR	FD38	2401	1291	2385	2397					
OUTPRI	FD00	2373	1346	1529	1586	2258				
OUTPR1	FD3A	2402	2406							
OUTPR2	FD44	2407	2403							
OUTPUT	E97A	1475	469	473	658	694	1148	1228	1242	4008
			4054	4059	4061	4150				
OUTTAP	F24A	2710	935	955	963	964	1521	2626	2629	2701
			2702	2703	2704					
OUTTA1	F290	2745	2714							
OUTTA2	F294	2747	2763							
OUTTA3	F2B2	2759	2754	2761						
OUTTTY	EEA8	2219	1484							
OUTT1	EECB	2232	2241							

OUTT2-PRADR4

NAME	WERT	DEFIN.	REFERENZEN
OUTT2	EEFB	2253	2249 2251
OUT01	F00F	2381	2376
OUT04	F025	2392	2379
OUT05	F033	2398	2391 2396
OUT1	E97B	1476	1540
OUT1A	E986	1481	1478
OUT2	E98F	1485	1482
PACK	EA84	1638	723 746 748 766 768 1623 1626 1720 3835
PAK1	EA96	1647	1643
PAK2	EA9F	1653	1656
PAREN	FC76	3751	3711
PATCH1	FE7C	4016	424
PATCH4	FE9C	4035	2274
PATCH5	FEB1	4044	2437
PATCH6	FEB7	4047	3087
PATCH8	FEBC	4050	3591
PATCH9	FED8	4065	
PATC10	FEE9	4074	590
PATC11	FEEF	4077	1238
PATC12	FEF8	4082	1267 3193 4119
PATC13	FF03	4089	3218
PATC14	FF0D	4094	3253
PATC15	FF17	4099	3600
PATC16	FF24	4107	3886 4011
PATC17	FF2A	4110	3195
PATC18	FF3D	4119	1028 1450
PATC24	FFB5	4174	2188
PATC25	FFBC	4179	2752 2759 4180
PATC8C	FED1	4060	4058
PAT17A	FF33	4114	4112 4116
PAT17B	FF3A	4117	4110
PAT18A	FF48	4125	4122
PAT19	FF59	4133	467
PAT2A	FE91	4028	4025
PAT20	FF60	4137	1084
PAT21	FF72	4147	426
PAT21A	FF74	4148	4152
PAT21B	FF7F	4153	4149
PAT22	FF9A	4160	2641
PAT23	FFA0	4163	2417 2419
PAT23A	FFA5	4165	4169
PAT23B	FFB2	4171	4166
PAT4A	FEAE	4042	4037
PAT9A	FEE2	4069	4066
PAT9B	FEE8	4072	4077 4079 4091 4096
PCADJ3	F54D	2969	2955
PCADJ4	F554	2973	2971
PCLLD	EB56	1757	2857 2902 2933
PCR	A80C	217	2119 2416 2433 2450 2452 2694
PHXY	EB9E	1791	1256 1356 1440 2031 2220 2268 2374 2615
PINT	F0CB	2473	2413
PLNE	F727	3152	3112 3128 3175 3177 3181 3219 3248 3262 3335 3608
PLXY	EBAC	1803	1292 1370 1446 2039 2246 2277 2325 2398 2644
PNTLUP	FBD0	3672	3675
POMSG	FF88	4157	4157 4157 4157 4157 4157 4148
PRADR1	F4F7	2927	2953
PRADR2	F4FF	2931	2944
PRADR3	F519	2945	2928 2930
PRADR4	F52C	2952	2946 2950

PRBL2-REGQ

NAME	WERT	DEFIN.	REFERENZEN							
PRBL2	F545	2965	2854	2904	2923					
PRDOTO	F08C	2447	2449							
PRIERR	F079	2436	2421							
PRIFLG	A411	114	654	665	676	681	1121	1476	1526	1531
			2093	2110	2411	3208	3209	4044	4083	4085
PRITR	E6E1	1121	1970							
PRMN1	F4D7	2912	2922							
PRMN2	F4DB	2914	2918							
PRNDOT	F087	2445	2426	2427						
PRNTYX	F538	2960	2958							
PROMPT	E7BD	1236	3190	3268						
PROMP1	E7C5	1239	4080							
PRPC	F53C	2962	2855	3871						
PRST	0000	225	2415	2432						
PRTIME	06A4	235	2459	2461						
PR1	E7CC	1242	1247	1250	1295	1299				
PR2	E7CF	1243	1240							
PSLOC	E81E	1283	1277	1281						
PSLS	E7DC	1254	1454							
PSLO	E7FB	1267	1260							
PSLOA	E814	1278	1274							
PSLOB	E81C	1282	1272							
PSLOD	E823	1285	1290							
PSLOO	E802	1270	1264							
PSL1	E837	1294	543	1014	1152	1255	3240			
P00	F749	3167	3165							
P01	F73B	3161	3154	3156						
P02	F729	3153	3160							
P03	F73F	3163	3252							
QM	E7D4	1246	481	1001	1021	1368	3579	3597		
RA	AC00	246	405	2349	2356	2359	2361			
RB	AC02	248	406	2353						
RBYTE	E3FD	741	624	629	632	637	708	729	732	897
RBYT1	E407	745	743							
RB2	E95C	1454	1462							
RCHEK	E907	1413	358	551	800	1156	3245			
RCHTTY	E926	1428	1414							
RCHT1	E93B	1437	1428	1449	1472					
RCHT2	E928	1429	1430	1436						
RCH2	E91F	1424	1426							
RCH3	E925	1427	1417	1423						
RDADDR	FBE5	3685	3680	3692						
RDBIT	EE3B	2159	2123	2150						
RDBIT1	EE43	2162	2163							
RDBIT2	EE51	2168	2169							
RDBIT4	EE67	2184	2160	2185						
RDLUP	FE14	3956	3960	3968						
RDRUB	E95F	1455	1380	1464	1504	1668	3269	3690	3697	3956
RDR1	E96A	1461	1457							
RD1	EA70	1623	1620							
RD2	EA5D	1614	573	997	1197					
READ	E93C	1440	3313	4030	4032	4050				
READ1	E94A	1445	1442							
READ2	E94D	1446	1444							
REA1	E956	1450	1421	1434						
REDOUT	E973	1470	1015	1614	1625					
RED1	FE96	4032	470							
RED2	E976	1471	1459	4033						
REENTR	F6CF	3103	504							
REG	E227	514	504							
REGF	A40E	111	547	1112	2849					
REGQ	F461	2849	463							

REGT-SAVE

NAME	WERT	DEFIN.	REFERENZEN						
REGT	E6D9	1112	507						
REG1	E232	518	549	2851					
RELADR	F530	2955	2934						
REPLAC	F93F	3431	3180	3223					
REP2	F93E	3427	3442						
RESNOW	F8D0	3356	3484	3556					
RINT	A485	190	4165						
RMNEM	0118	78	2910	2914					
ROLLFL	A47F	170	1990	2008					
ROONEK	ECEF	1987	1415	1424	1888	1992			
ROO1	ED00	1994	1989						
ROUT	F286	2737	2732						
ROUT1	F28B	2739	2764						
ROW1	F421	2839	2839	2839	1935				
ROW2	F429	2840	2840	2840					
ROW3	F431	2841	2841	2841					
ROW4	F439	2842	2842	2842					
ROW5	F441	2843	2843	2843					
ROW6	F449	2844	2844	2844					
ROW7	F451	2845	2845	2845					
ROW8	F459	2846	2846	2846					
RQP	F977	3462	3458	3460					
RSET	E0BF	362	4187						
RSPAC	EA7B	1628	1616	1618	1622	1624	1639	1641	1645
RS1	E0C9	369	372						
RS2	E0D4	375	378						
RS20	E702	1141	1143						
RS3	E0F3	389	387	393					
RS3A	E0F1	388	382	385					
RS3B	E11A	408	404						
RS4	E11D	412	414						
RS5	E129	417	418						
RS6	E13E	425	436						
RS7	E144	427	413						
RS8	E146	428	435						
RTMODE	F491	2875	2870						
RTS1	F55A	2976	2974						
RUB	0008	261	1456						
R10	F9C7	3497	3494						
R100	F9CF	3503	3449						
R101	F9DA	3509	3507						
R102	F9E3	3513	3517						
R103	F9FA	3525	3523						
R104	FA17	3539	3537	3555					
R105	FA31	3549	3545						
R1051	FA41	3555	3553						
R106	FA44	3556	3548						
R107	FA0A	3532	3526						
R108	F9EF	3518	3515						
R11	F9CC	3500	3496						
R2W	F95F	3449	3433						
R5	F99D	3479	3475						
R55	F9A8	3483	3481						
R6	F984	3468	3466	3483					
R7	F9AB	3484	3478						
R8	F947	3435	3500						
R87	F94E	3440	3434	3498					
R88	F953	3443	3447						
R9	F9BE	3493	3491	3557					
SADDR	EB78	1774	581	753	3090	3093	3847		
SAVA	A421	130	332	439	444	566	709	717	
SAVE	00E7	58	3352	3353	3527	3529			

NAME	WERT	DEFIN.	REFERENZEN
SAVNOW	F934	3423	3056 3066 3452 3510
SAVPC	A425	134	339 341 346 348 456 459 519 560
			562 973 975 1160 1163 1165 1184 1187
			1757 2962 2963 2969 2973 3654 3656
SAVPS	A420	129	334 449 521 523 564 999
SAVS	A424	133	343 366 461 484 556
SAVX	A422	131	336 450 559
SAVY	A423	132	337 451 558
SEMI	E9BA	1508	820 852
SETBOT	F8C5	3350	3147 3511
SETREG	E113	405	397 400 403
SETSPD	F2C0	2769	2690
SETSP1	F2CA	2774	2772
SETSP2	F2D3	2778	2776
SETZ	F282	2735	2731
SHIS	E665	1051	507
SHOW	E64D	1037	507
SH1	E652	1039	1046
SH11	E66A	1053	1062
SIZEM	FB0F	3633	3633 3633 3805
SP12	0001	226	2415 2432
SR	A80A	215	
SRCHLP	FE44	3984	3994
SRCHM	FE47	3985	3988
START	E182	467	486 496 4155
STARTM	FBAA	3658	3789 4108
STASH	FD2C	3844	3824 3856 3898 3931
STA1	E185	468	4075 4135
STBKEY	A42B	172	1910 1918 1930 2004 2013
STBYTE	E413	751	625 727
STCODE	FB1E	3636	3636 3636 4002
STIY	A427	138	1052 1061 1634 1654 1658 1776 1779 1782
			1788 2221 2226 2229 2230 2231 2232 2233
			2235 2243 2290 2295 2296 2309 2312 2313
			2343 2346 2712 2717 2730 2737 2746 2753
			2926 2937 2939 3284 3300 3319
STLO	FE6E	4007	4010
STLOAD	FE73	4009	3958
STOP	F870	3306	3609
STORCH	FBF6	3693	3703
STOR1	FC0A	3702	3699
STRING	00EB	61	3278 3294
STSHLP	FD30	3846	3852
SUB	F91D	3407	3115 3116 3332 3549
SUB1	F927	3412	3410
SWSTAK	EBBA	1813	1796 1804
SWST1	EBBD	1815	1825
SYNC	EDFF	2123	2128 2132
SYNC1	EE11	2130	2134
S1	A41A	123	805 808 824 826 882 908 910 941
			943 948 950 3047 3402 3403 3464 3465
			3467 3468 3473 3476 3480 3482 3539 3543
			3546 3550 3551 3554 3602 3603 4104
S2	0106	126	788 789 856 858 913 915
TABUFF	0116	151	650 657 669 2036 2051 2063 2068 2070
			2075 2078 2658 2664 2665 2669 2670 2672
			2676 2679 2684
TABUF2	00AD	153	2655 2686
TABY2	F1A7	2628	2632
TABY3	F1CE	2644	2621
TAISET	EDEA	2114	701 2042
TAOSET	F21D	2690	933 2624

TAOS1-T1LH

NAME	WERT	DEFIN.	REFERENZEN
TAOS1	F238	2700	2706
TAPIN	A434	145	1371 2116
TAPOUT	A435	146	2691
TAPTR	A436	147	675 2032 2038 2066
TAPTR2	A437	148	867 923 2616 2619 2638
TAP1	E8B3	1363	1361
TAP2	E8BC	1368	1365
TAP3	E8C2	1370	1367
TEMPA	A433	90	3810 3815
TEMPX	A431	89	3706 3712 3728 3889
TEXT	00E3	56	3045 3344 3345 3364 3367
TIBYTE	ED3B	2031	1492
TIBY1	ED53	2042	647 2035 2047
TIBY3	ED56	2043	2048
TIBY4	ED63	2049	2045
TIBY5	ED65	2050	2054
TIBY5A	ED88	2065	2062
TIBY6	EDAF	2083	2076
TIBY7	EDB0	2084	2064 2079
TIB1	ED48	2036	2034
TIMG	A40B	110	2720 2778
TIOSET	EE1C	2139	2117 2692
TIOS1	EE22	2142	2139
TIOS2	EE24	2143	2141
TMASK1	0126	83	3667 3672 3794
TMASK2	0127	84	3668 3673 3797
TMSG0	E048	316	316 655
TMSG1	E04D	317	1377
TMSG2	E050	318	1358
TMSG3	E052	319	319 690
TMSG5	E05F	321	
TMSG6	E061	322	322 663
TMSG7	E066	323	323 679
TO	E7A7	1220	781 3051
TOBYTE	F18B	2615	871 893 924 926 1516 2643
TOGL	E6E7	1127	1113 1118 1122
TOGL1	E6F6	1133	1128
TOGTA1	E6BD	1095	508
TOGTA2	E6CB	1103	508
TOPNO	F8BC	3344	3053 3074 3104
TO1	E7A9	1221	1218
TP	F6D2	3104	3609
TPO1	F8C0	3346	3354
TRACE	E6DD	1117	507
TRY	F258	2716	2738
TRYINY	FC85	3758	3755
TRYJMP	FC94	3765	3759
TRYZP	FC28	3717	3714
TRY34	FC40	3727	3718
TRY56	FC5A	3738	3696
TSPEED	A408	106	2159 2201 2204 2208 2713 2769 4067 4069
TTYTST	E842	1302	1239 1254 1413 1441 1481 1548 1580 3574 3861 4121
TYPE	012E	82	3670 3678 3719 3743 3768 3773 3781 3999
TYPTB	FB5E	3644	3644 3644 3998
TYPTR1	FAE2	3627	3627 3627 3792
TYPTR2	FAF1	3629	3629 3629 3796
T1CH	A805	210	2698
T1FR	00C0	239	2695
T1I	0000	238	875 1172 2639
T1L	A804	209	2726 4181
T1LH	A807	212	2723 2751 2758

T1LL-ZPY

NAME	WERT	DEFIN.	REFERENZEN						
T1LL	A806	211	2721	2749	2756				
T2H	A809	214	416	419	1862	1877	2195	2200	2462
T2I	0000	237							
T2L	A808	213	422	1860	1874	2022	2121	2197	2460
UACR	A00B	40							
UDDRA	A003	32							
UDDRB	A002	31							
UDRA	A00F	44							
UDRAH	A001	30							
UDRB	A000	29							
UIER	A00E	43							
UIFR	A00D	42							
UIN	0108	66	1324	1502					
UOUT	010A	67	1351	1537					
UP	F6F9	3127	3185	3608					
UPCR	A00C	41							
UPNO	F709	3133	3127	3220	3235	3249	3274		
UP1	F713	3137	3135	3141					
UP4	F720	3144	3118	3130	3138				
USR	A00A	39							
UT1CH	A005	34							
UT1L	A004	33							
UT1LH	A007	36							
UT1LL	A006	35							
UT2H	A009	38							
UT2L	A008	37							
VALID	FCDD	3799	3795						
VECKSM	E694	1076	508						
VECK1	E69E	1080	1082	4144					
VECK2	E6AC	1086	1079	1088	1091				
VECK4	FF6F	4145	4137						
VECK5	FF66	4141	4143						
WHEREI	E848	1308	609	1076	3229				
WHEREO	E871	1329	786	3243					
WHE1	E85C	1316	1312						
WHE2	E868	1321	1317						
WHE3	E870	1325	1322						
WHICHT	E8A8	1358	1369						
WHRO1	E885	1338	1334						
WHRO2	E88E	1343	1339						
WHRO3	E897	1348	1344						
WHRO4	E89F	1353	1349						
WRAX	EA42	1593	605	1042	2961	2964			
WRITAD	E2DD	603	520						
WRITAZ	E2DB	602	571	686	1008	1739	3042	3061	3662
XORY	FDEF	3935	3739	3760					
XORYRT	FED2	3944	3947						
XORYZ	FDF1	3936	3731						
XORY1	FDFC	3941	3938						
ZON	F25D	2718	2734	2736					
ZON1	F261	2720	2728						
ZON2	F26C	2724	2725						
ZPAGE	FC38	3724	3721						
ZPX	FC55	3736	3733						
ZPY	FC50	3734							

P E T - 6 5 0 2 - A S S E M B L E R

Von Karlheinz Lehner, Zeppelinstraße 29, 6236 Eschborn 1

E: This 6502 assembler is designed to run on a PET 2001-8K. It accepts one line of source code, processes it and deposits it into a reserved Kbyte of memory. It follows the rules and regulations as stated in the 'KIM Assembler Manual Preliminary' with a few exceptions (simpler syntax for addresses in the parse routine and use of blanks as delimiters in .BYTE or .WORD directives). Provisions are made for 60 defined labels and 30 forward references. Error messages are printed. Code can be executed immediately or the monitor can be loaded.

Dieser 6502 Assembler richtet sich grundsätzlich nach den Regeln und Richtlinien des "KIM Assembler Manual Preliminary". Er übernimmt eine Zeile nach der anderen, speichert die Labels, falls vorhanden, sucht Opcode und Adresse und er speichert sie im letzten Kbyte des 8Kbyte-Speichers ab. Darüber hinaus verarbeitet er Pseudobefehle (assembler directives) wie z.B =, .BYTE, oder .END. Bei Verzweigungsbefehlen werden unbekannte Befehle gesondert gespeichert und zum Schluß bearbeitet.

Es handelt sich um einen Ein-Pass-Assembler in dem Sinne, daß jede Zeile nur einmal eingegeben werden muß. Die Speicherung des Objektkodes mittels POKE erlaubt entweder die direkte Ausführung nach dem Assemblieren oder man kann den Monitor laden, mit dem Maschinenprogramm arbeiten und es abspeichern. Vorteilhaft ist es auch, daß der Objektkode nicht durch eine Programmänderung oder durch RUN gelöscht werden kann. Der Speicherraum von 1 Kbyte wird in den meisten Fällen wahrscheinlich ausreichen. Man kann sich aber auch so einrichten, daß in dem 1 Kbyte nur das ausführende Programm liegt, während sich Daten in dem Speicherraum befinden, der normalerweise vom Assembler belegt wird. Und man könnte auch den Puffer des zweiten Kassettenrekorders und den Bildschirmspeicher benutzen.

Wegen des begrenzten Speicherraumes wurde die Syntax der Adressen in der parse routine etwas vereinfacht. Danach ist eine Adresse entweder
a) eine Zahl zur Basis 2, 8, 10 oder 16,
b) ein ASCII-Zeichen (immediate mode),
c) ein label oder
d) ein label, ein arithmetischer Operator (+, -, x, /) und a) oder c).

Das deckt die meisten der in der Praxis vorkommenden Fälle ab.

Assembler directives sind: =, .END, welches unbekannte labels verarbeitet, .BYTE und .WORD, wobei die Daten durch *Leerzeichen* zu trennen sind (nicht Kommas).

Die Höchstzahl der definierten labels ist 60, die Höchstzahl der labels für Vorwärtsverzweigungen ist 30. Diese Zahlen können ggfs. vergrößert oder verkleinert werden - je nach Bedarf an Speicherraum oder labels. Hierfür sind die Zeilen 1 und 150 entsprechend abzuändern.

In Zeile 1010 erfolgt ein einfaches INPUT. Diese Zeile kann leicht geändert werden, um Quellenprogramme vom Magnetband einzulesen. In der davorliegenden Arbeitsphase kann man sein Programm durch einen Text-Editor zurechtmachen, es abspeichern und dann in den Assembler eingeben. Es ist dann jedoch zu beachten, daß entweder ein gedrucktes Anführungszeichen oder ein GET# -Befehl genommen wird, weil ein Komma in der Adresse die Eingabe terminiert.

Eine einfache Dump-Routine beginnt ab Zeile 60000. Sie listet labels, Vorwärtslabels und den Speicherinhalt ab 7168. - Daten für die Op odes sind in den DATA-Statements gespeichert. Sie sind dadurch gegen zufälliges Löschen gesichert. Andererseits wurde auf ein Einlesen in ein Array verzichtet, weil der Speicherraum dadurch doppelt belegt worden wäre. So hat man sichere Daten mit minimalem Speicherraumbedarf.

Unter folgenden Bedingungen wird die Übersetzung abgebrochen und eine Fehlermeldung angezeigt: Wenn versucht wird, einen label zweimal zu definieren (LABEL PREVIOUSLY DEFINED ERROR), wenn der Programmzählerstand nicht definiert wird (PC = 0 ERROR), wenn der Abstand zwischen Verzweigung und label zu groß wird (LABEL OUT OF RANGE ERROR), wenn "A" als label gebraucht wird (ILLEGAL LABEL ERROR), wenn ein unbekannter Opcode auftritt (ILLEGAL OPCODE ERROR), wenn auch nach .END ein label unbekannt bleibt (UNKNOWN LABEL ERROR) oder wenn ein unbekanntes label in .BYTE, .WORD oder in einer Adresse (außer einer Verzweigung) auftritt (ILLEGAL FORWARD REFERENCE ERROR). Dagegen wird nicht auf eine undefinierte Kombination von Opcode und Adressierungsart der Maschinensprache geachtet (z.B. wird LDX (LOOP),Y ohne Fehlermeldung sinnlos verarbeitet).

Das Programm ist unter dem Gesichtspunkt der leichten Lesbarkeit und der Übersichtlichkeit geschrieben worden, und zwar durch zahlreiche REM-Statements und durch gezielte Numerierung der Funktionsblöcke mit zusammenhängenden Zahlen. So geben die Zeilennummern bereits eine Zuordnung zur Funktion.

VARIABLES:

MAIN ROUTINE:INPUT - A$ - ONE LINE OF SOURCE CODE
OUTPUT - MACHINE PROGRAM, TABLES

N	LABELS COUNTER	F	FLAG (=1 IF FORWARD REFERENCE FOUND)
LB$(N)	LABELS		
LB(N)	VALUE OF LABELS	BG	BEGIN FLAG (=0 IF NEW START)
M	UNKNOWN LABELS COUNTER	AD$	ADDRESS STRING
UL$(M)	UNKNOWN LABELS	AD	ADDRESS CODE
UL%(M)	VALUE OF PC WHERE ENCOUNTERED [WHEN POS.-BRANCH;NEG.- JMP/JSR]	K	LOCAL DUMMY
		T$	CURRENT CHAR. OF A$
PC	PROGRAM COUNTER, NEXT FREE POS.	T	POINTER TO CURRENT CHAR. OF A$
A$	LINE OF ASSEMBLY CODE		
L	DUMMY	CB$	CHARACTER BLOCK STRING
		F$	ASSEMBLER DIRECTIVE
		OP$	OPCODE STRING

ADDRESS CODE DETERMINATION ROUTINE: INPUT AD$ AND OP$
OUTPUT MACHINE PROGRAM

R$,R	READ DUMMIES	K	DUMMY
		DI	DIFFERENCE

PARSE ROUTINE: INPUT P$ PARSE STRING
OUTPUT V VALUE OF PARSE STRING

R	POINTER IN P$	L$	LABEL STRING BUILT FROM P$
BA	BASE OF NUMBER	P1	ASC OF P1$
P1$	CURRENT CHAR. IN P$	K	DUMMY
AR	ARITHM. OPERATOR CODE/INDICATOR	F	FLAG (=1 IF LABEL UNKNOWN)
W	TEMP. STORAGE OF V IN ARITHMETIC OPERATIONS		

Achtung: Aus Gründen der Übersichtlichkeit wurden in den Zeilen einige Zwischenräume gelassen.

```
1000 REM
1001 REM GET LINE
1002 REM
1010 INPUT A$:RESTORE:R$=""
1020 IF BG=0 AND LEFT$(A$,2)><"*=" THEN PRINT "PC = 0 ERROR":END
1030 BG=1
2000 REM
2001 REM PREP CHECK
2002 REM
2010 T=1:GOSUB9000:T$=LEFT$(CB$,1)
2020 IF T$=";"THEN1000
2030 IF T$="." THEN7000
2040 IF T$="*" THEN P$=MID$(CB$,3):GOSUB 20000:PC=V:GOTO1000
3000 REM
3001 REM LABEL?
3002 REM
3010 IF LEN(CB$)><3 THEN3050
3020 READ R$:IF R$="152" THEN RESTORE:R$="":GOTO3060
3030 IF CB$=R$ THEN OP$=CB$:GOTO5000
3040 GOTO3020
3050 IFCB$="A"THENPRINT"ILLEGAL LABEL ERROR":END
3060 K=0
3070 IFLB$(K)=CB$THENPRINT"LABEL PREVIOUSLY DEFINED ERROR":END
3080 K=K+1:IFK<NTHEN3070
3090 LB$(N)=CB$:LB(N)=PC:N=N+1
4000 REM
4001 REM REG/PC LINE?
4002 REM
4010 GOSUB9000:IFLEFT$(CB$,1)><"*"THEN4030
4020 P$=MID$(CB$,5):GOSUB20000:PC=PC+V:GOTO1000
4030 IFLEFT$(CB$,1)><"="THEN4050
4040 P$=MID$(CB$,2):GOSUB20000:LB(N-1)=V:GOTO1000
4050 IFLEFT$(CB$,1)="."THEN7000
4060 OP$=CB$
5000 REM
5001 REM PROCESS ADDRESS
5002 REM
5010 GOSUB9000:AD$=CB$
5020 IFLEFT$(AD$,1)="#"THEN5700
5025 IFLEFT$(AD$,1)=";"THEN AD$=""
5030 IF AD$="A"ORAD$="" THENAD=0:GOSUB5600:GOTO1000
5040 IFRIGHT$(AD$,3)=",X"THENAD=3:P$=MID$(AD$,2, LEN(AD$)-4):GOTO5900
5050 IFRIGHT$(AD$,3)=",Y"THENAD=4:P$=MID$(AD$,2,LEN(AD$)-4):GOTO5900
5060 IF RIGHT$(AD$,1)=")"THENAD=0:P$=MID$(AD$,2,LEN(AD$)-2):GOTO5920
5070 IFRIGHT$(AD$,2)=",X"THEN5200
5080 IFRIGHT$(AD$,2)=",Y"THEN5300
5090 P$=AD$:GOSUB20000:IFLEFT$(OP$,1)="B"AND
     OP$><"BIT"AND OP$><"BRK"THEN5400
5100 IFLEFT$(OP$,1)="J"AND F=1 THEN5500
5110 IF V<256THENAD=2:GOTO5910
5120 AD=1:GOTO5930
5200 P$=LEFT$(AD$,LEN(AD$)-2):GOSUB20000:IFV<256 THENAD=5:GOTO5910
5210 AD=6:GOTO5930
5300 P$=LEFT$(AD$,LEN(AD$)-2):GOSUB20000:IF V<256 THENAD=6:GOTO5910
5310 AD=7:GOTO5930
5400 AD=0:GOSUB5600:IF F=1 THEN UL$(M)=P$:UL%(M)=PC:M=M+1:PC=PC+1:GOTO1000
5405 D1=V-PC-1:IF ABS(D1)>127THENPRINT"LABEL OUT OF RANGE ERROR:";
     AD$:END
```

Das folgende Anwendungsbeispiel summiert die Zahlen von 1 bis N:

```
        *=8000
N       *=*+1
SUML    *=*+1
SUMH    *=*+1
        *=7168
        LDA #0          ANWENDUNG Z.B. FÜR N = 100:
        STA SUML
        STA SUMH        POKE 8000,100:SYS(7168):PRINT
LOOP    CLC
        LDA N           PEEK (8001)+256 *PEEK(8002)
        ADC SUML
        STA SUML        ERGEBNIS:5050
        BCC NEWN
        INC SUMH
NEWN    DEC N
        BNE LOOP
        RTS
        .END
```

```
  1 POKE 134,0:POKE 135,28:REM LIMIT BASIC MEMORY TO 7K
  4 REM
  5 REM 6502 ASSEMBLER BY KARL LEHNER
  6 REM
 10 PRINT"c$";TAB(13);"6502 ASSEMBLER":PRINT
 20 PRINT"SOURCE CODE ZEILENWEISE EINGEBEN. FREIEN"
 30 PRINT"SPEICHERRAUM 7168-8191 BENUTZEN. KOMMA-"
 40 PRINT"GEBRAUCH BENOETIGT ANFUEHRUNGSZCHN.!":PRINT
100 DEF FNA(X)=INT(X/256):DEF FNB(Y)=Y-256*FNA(Y):
    REM HIGH AND LOW ORDER BYTE
150 DIM LB$(59), LB(59), UL$(29), UL%(29)
500 DATA ADC,105,109,101,97,113,117,125,121,AND,41,45,37,33,49,53,
    61,57,ASL,10,14
510 DATA6,,,22,30,BCC,144,BCS,176,BEQ,240,BIT,,44,
    36,BMI,48,BNE,208,BPL,16,BRK,0
520 DATA BVC,80,BVS,112,CLC,24,CLC,24,CLD,216,CLI,88,CLV,184,
    CMP,201,205,197,193,209,213
530 DATA 221,217,CPX,224,236,228,CPY,192,204,196,DEC,,
    206,198,,,214,222,DEX,202
540 DATA DEY,136,EOR,73,77,69,65,81,85,93,89,INC,,238,
    230,,,246,254,INX,232,INY
550 DATA 200,JMP,108,76,JSR,,32,LDA,169,173,165,161,
    177,181,189,185,LDX,162,174
560 DATA 166,,,,182,190,LDY,160,172,164,,,180,188,
    LSR,74,78,70,,,86,94,NOP,234,ORA
570 DATA 9,13,5,1,17,21,29,25,PHA,72,PHP,8,PLA,104,PLP,
    40,ROL,42,46,38,,,54,62,ROR
580 DATA 106,110,102,,,118,126,RTI,64,RTS,96,SBC,233,237,
    229,225,241,245,253,249
590 DATA SEC,56,SED,248,SEI,120,STA,,141,133,129,145,
    149,157,153,STX,,142,134,,,,
600 DATA 150,STY,,140,132,,,148,TAX,170,TAY,168,
    TSX,186,TXA,138,TXS,154,TYA,152
```

```
5410 IF DI<0 THENDI=DI+256
5420 POKEPC,D :PC=PC+1:GOTO1000
5500 AD=1:GOSUB5600:UL$(M)=P$:UL%(M)=-PC:M=M+1:PC=PC+2:GOTO1000
5600 REM
5601 REM SEARCH OPCODE TABLE
5602 REM
5610 IF R$="152" THEN PRINT "ILLEGAL OPCODE ERROR":END
5620 IF R$><OP$ THEN READR$:GOTO5610
5630 FORK=0 TO AD:READR:NEXT:POKEPC,R: PC=PC+1:RETURN
5700 AD=0: GOSUB5600
5710 IF MID$(AD$,2,1)=">" THEN GOSUB5750:POKE PC,FNA(V):GOTO5740
5720 IFMID$(AD$,2,1)="<" THEN GOSUB5750:POKEPC,FNB(V):GOTO5740
5730 P$=MID$(AD$,2):GOSUB20000:POKEPC,V: IF F=1THEN7515
5740 PC=PC+1:GOTO1000
5750 P$=MID$(AD$,3):GOSUB 20000: IF F=1 THEN 7515
5760 RETURN
5900 GOSUB 20000:REM 1 BYTE ADDRESS
5910 IF F><1 THEN GOSUB5600:POKEPC,V:PC=PC+1:GOTO 1000
5915 GOTO 7515
5920 GOSUB 20000:REM 2 BYTE ADDRESS
5930 IF F><1 THEN GOSUB5600:POKE PC, FNB(V):PC=PC+1:POKE PC,FNA(V):
     PC=PC+1:GOTO1000
5935 GOTO7515
7000 REM
7001 REM PROCESS ASSEMBLER DIRECTIVES
7002 REM
7010 F$=MID$(CB$,2,3):IF F$="END" THEN 8000
7020 IF F$= "BYT" THEN 7500
7030 IF F$="WOR" THEN7600
7500 GOSUB 9000:IF LEFT$(CB$,1)><"'"THENP$=CB$:GOSUB 20000:POKE PC,V:
     GOTO7515
7510 FOR K=2 TO LEN(CB$)-1: POKE PC,ASC(MID$(CB$,K,1)):PC=PC+1:NEXT
7515 IF F=1 THENPRINT"ILLEGAL FORWARD REFERENCE ERROR:";P$:END
7520 IF T>LEN(A$) THEN 1000
7530 GOTO7500
7600 GOSUB9000:P$=CB$:GOSUB20000:POKE PC,FNB(V):PC=PC+1:POKEPC,FNA(V):
     PC=PC+1
7605 IF F=1 THEN7515
7610 IF T>LEN(A$)THEN1000
7620 GOTO7600
8000 REM
8001 REM PROCESS UNKNOWN LABELS
8002 REM
8010 IFM=0 THEN8100
8015 FOR K=0TOM-1:L=0
8020 IF L>=N THENPRINT "UNKNOWN LABEL ERROR:";UL$(K):END
8030 IF LB$(L)><UL$(K) THENL=L+1:GOTO8020
8035 IF UL%(K)<0 THEN POKE-UL%(K),FNB(LB(L)):POKE-UL%(K)+1,FNA (LB(L)):
     GOTO8055
8040 DI=LB(L)-UL%(K)-1:IF ABS(DI)>127 THEN PRINT"LABEL OUT OF RANGE
     ERROR:";UL$(K):END
8050 POKE UL%(K),DI
8055 NEXT
8100 PRINT:PRINT"END OF ASSEMBLY":PRINT
8110 PRINT "NUMBER OF LABELS:";TAB(24);N
8120 PRINT"NUMBER OF FORWARD REF.:";TAB(24);M:END
```

```
 9000 REM
 9001 REM GET CHAR BLOCK
 9002 REM
 9010 CB$=""
 9015 T$=MID$(A$,T,1):IF T$=" "THEN T=T+1:GOTO9015
 9020 IF T>LEN(A$) THEN RETURN
 9030 T$=MID$(A$,T,1):IF T$=" " THEN RETURN
 9040 CB$=CB$+T$:T=T+1:GOTO9020
20000 REM
20001 REM PARSE ROUTINE
20002 REM
20005 R=1:W=0:F=0
20010 L$="": V=0:BA=0
20020 P1$=MID$(P$,R,1):IF P1$>="0" AND P1$<="9"
      THEN BA=10:GOTO25010
20030 IF P1$="$" THEN BA=16: GOTO25000
20040 IF P1$="@" THEN BA=8:GOTO 25000
20050 IF P1$="%" THEN BA=2:GOTO25000
20060 IF P1$="'" THEN V=ASC(MID$(P$,R+1,1)):RETURN
20070 L$=L$+P1$:R=R+1:P1$=MID$(P$,R,1):IF R>LEN(P$) THEN26000
20080 IF P1$="+"THEN AR=1:GOTO20200
20090 IF P1$="-"THEN AR=2:GOTO20200
20100 IF P1$="*"THENAR=3:GOTO20200
20110 IF P1$="/" THEN AR=4:GOTO 20200
21120 GOTO20070
20200 GOSUB 26000:W=V:R=R+1:GOTO 20010
25000 R=R+1:P1$=MID$(P$,R,1)
25010 IF R>LEN(P$) THEN 28000
25020 P1=ASC(P1$):V=V*BA-(P1<60)*(P1-48)-(P1>60)*(P1-55):GOTO25000
26000 K=0
26010 IF K=N THEN F=1:RETURN
26015 IF L$><LB$(K) THEN K=K+1:GOTO26010
26020 V=LB(K)
28000 IF AR=0 THEN RETURN
28010 ON AR GOTO28100, 28110,28120,28130
28100 V=W+V
28110 V=W-V
28120 V=W*V
28130 V=W/V

60000 FOR K=ØTON:PRINTLB$(K):NEXT:PRINT:
      FOR K=ØTOM:PRINT UL$(K),UL%(K):NEXT:PRINT
60005 FOR K=7168 TO8191:PRINTK,PEEK(K):NEXT
```

****

SYM-1 HYPERTAPE LOADER

Anton Zoegal, Lindenstraße 5, D-8411 Waldetzenberg

E: *This program makes HYPERTAPE readable on the SYM-1*

Bedienungsanleitung:

.1 Laden des Programms an beliebige Speicherstelle mit dem Verschiebelader (RALOAD, Heft 1)
.2 Setzen des URVEC auf die Startadresse des HYPERTAPE-LOADER
.3 Eingabe Funktion UØ ID. Es erscheint wie bei der normalen Laderoutine ein L in der Anzeige
.4 Starten des Recorders
.5 Wenn richtig geladen, dann erscheint wieder der '.' (Punkt) des Monitors.

Ein Beispiel: Startadresse des HYPLOAD = 4000, gewünschtes ID = 20

```
.SD 4000,A66D
.UØ 2o
.
```

Fehlermeldungen wie im Original-Ladeprogramm. Verschiebung oder ID = 00 werden nicht unterstützt, sind aber leicht einbaubar.

```
: SYM-1 HYPERTAPE LOADER
:
: EINSPRUNG UEBER MONITOR (URCVEC)
: STARTADRESSE DIESER ROUTINE MIT
: FUNKTION  .SD 4000,A66D EINGEBEN
:
: AUFRUF DER FUNKTION
: .UØ ID
:
STAR  .OR  4000                    FESTLEGEN DER START-ADRESSE
LOCN  INSTR       LABEL  OPER  OPERANDEN      KOMMENTAR

4000  C9 14       STAR   CMP   14             UØ FUNKTION ?
4002  FO 02              BEQ   LADH           JA
4004  38         BAD.   SEC                  FEHLER-KZ
4005  60                RTS                  ZUM MONITOR RETURN
                 : LADE ROUTINE VORPROGRAMM
4006  20 08 82   LADH   JSR   PSHV           PARAMETER AUSRICHTEN
4009  20 08 82         JSR   PSHV
400C  AO 00            LDY   00             KIM-FORMAT KZ
400E  20 B6 8D         JSR   STRT           IN START ROUTINE BAND
4011  AD 02 AO         LDA   DDRI           DATEN RICHTUNGS-REG
4014  29 BF            AND   OBF            SETZEN PB6 = INPUT
4016  8D 02 AO         STA   DDRI
4019  A9 00            LDA   00             AUX CONTROL
401B  8D 0B AO         STA   VACR           REGISTER LADEN
401E  A9 AE            LDA   OAE            WERT FUER COUNTER LATCH
4020  8D 04 AO         STA   LATL           LOW SETZEN
4023  20 84 40   LOT2   JSR   SYNC           SYNC ZEICHEN WARTEN
4026  20 AD 40   LOT4   JSR   RDCX           LESE BYTE VOM BAND
4029  C9 2A            CMP   2A             STERN 1. Zeichen nach
402B  FO 06            BEQ   LO11           SYNC (X'16')
402D  C9 16            CMP   16             SYNC-ZEICHEN
402F  DO F2            BNE   LOT2           RE-SYNC
4031  FO F3            BEQ   LOT4           WARTE AUF STERN
                 : SYNCHRONISIERUNG OK
                 : SUCHE ID
4033  A5 FD      LO11   LDA   MODE           SCHALTER AUSSERHALB
4035  29 BF            AND   OBF            SYNC AUS
4037  85 FD            STA   MODE           UND RESTORE
4039  20 DF 40         JSR   RDBX           LESE EIN BYTE (2 ASCII)
403C  CD 4E A6         CMP   ID             IST ID GLEICH VORGABE
403F  DO  E2           BNE   LOT2           NEIN WEITERSUCHEN
4041  20 DF 40         JSR   RDBX           LESE START-ADRESSE
4044  20 78 8E         JSR   CHKT           CHECKSUM-ROUTINE
4047  8D FE 00         STA   SAL            LOW-
404A  20 DF 40         JSR   RDBX
404D  20 78 8E         JSR   CHKT
4050  8D FF 00         STA   SAH            HIGH-STARTADR.
```

```
                        : LADE DATEN VOM BAND IN SPEICHER (SAL)
4053  20 DF 40  LOT7    JSR   RDBX          LESE BYTE (2 ASCII)
4056  B0 28             BCS   NHER          UNG. ZEICHEN
4058  20 78 8E          JSR   CHKT          CHECKSUM RECHNEN
405B  A0 00             LDY   00            INDEX = 0
405D  91 FE             STA   (SAL),Y       BYTE IN SPEICHER
405F  EE FE 00          INC   SAL           POINTER + 1
4062  D0 EF             BNE   LOT7          KEIN UEBERLAUF
4064  EE FF 00          INC   SAH           HIGH + 1
4067  4C 53 40          JMP   LOT7          NAECHSTES BYTE LADEN

                        : DATENENDE CHECKSUM PRUEFUNG
406A  20 DF 40  LOT8    JSR   RDBX          LESE 1 HEX BYTE
406D  CD 36 A6          CMP   CHKL          CHECKSUM LOW GLEICH ?
4070  D0 0A             BNE   CKER          NEIN- FEHLER
4072  20 DF 40          JSR   RDBX
4075  CD 37 A6          CMP   CHKH          CHECKSUM HIGH ?
4078  D0 02             BNE   CKER          NEIN - FEHLER

                        : DATEN RICHTIG GELADEN
                        : ZURUECK ZUM SUPERMON
407A  18                CLC                 FEHLERSCHALTER AUS
407B  60                RTS                 ZURUECK

                        : DATEN FEHLERHAFT CHECKSUM UNGLEICH
407C  A9 CC       CKER  LDA   0CC           LADE CC
407E  38          NGEX  SEC                 FEHLERSCHALTER AN
407F  60                RTS                 BRINGT .ER CC

                        : DATEN UNGUELTIG (NICHT HEX)
4080  A9 FF       NHER  LDA   0FF           FF IN ACCU
4082  D0 FA             BNE   NGEX          BRINGT .ER FF

                        : SYNCHRONISIER-ROUTINE (SYNC-ZEICHEN SUCHEN)
4084  A9 6D       SYNC  LDA   6D            L - IN ANZEIGE
4086  8D 00 A4          STA   DIG           ERSTE STELLE
4089  A5 FD             LDA   MODE          TEST MODE-SWITCH
408B  09 40             ORA   40            SYNC-BIT AUS
408D  85 FD             STA   MODE
408F  20 AA 40  SYN5    JSR   SYNB          LESE SYNC-BIT
4092  66 FC             ROR   CHAR          SHIFT IN CHAR
4094  A5 FC             LDA   CHAR          LADE CHAR IN AC
4096  C9 16             CMP   16            IST DIES SYNC
4098  D0 F5             BNE   SYN5          NEIN
409A  A2 0A       SY10  LDX   0A            INIT FUER 10 SYNC
409C  20 AD 40  SY11    JSR   RDCX          LESE BYTE HEX
409F  C9 16             CMP   16            SYNC - ZEICHEN ?
40A1  D0 EC             BNE   SYN5          NEIN RE-SYNC
40A3  CA                DEX                 X - 1
40A4  D0 F6             BNE   SY11          LOOP 10 X
40A6  8E 00 A4          STX   DIG           L-AUS ANZEIGE LOESCHE
40A9  60                RTS                 RETURN

40AA  4C C4 40  SYNB    JMP   RBIT          BIT-LESEN

                        : LESE EIN HEX-BYTE VOM BAND
40AD  8E FA 40  RDCX    STX   TMPX          SICHERN X-REG
40B0  A9 FF             LDA   0FF           8 BITS an
40B2  48        KBIS    PHA                 SICHERN IN STACK
40B3  20 C4 40          JSR   RBIT          LESE EIN BIT
40B6  66 FC             ROR   CHAR          SHIFT IN CHAR
```

```
40B8  68          PLA                 COUNTER ZUR. V. STACK
40B9  0A          ASL   A             SHIFT 1 BIT LINKS
40BA  D0 F6       BNE   KBIS          UNGLEICH 0 = WEITER
                : ALLE 8 BITS GELESEN UND IN CHAR GESCHOBEN
40BC  A5 FC       LDA   CHAR          CHAR IN ACCU
40BE  2A          ROL                 PARITY-BIT RAUS
40BF  4A          LSR
40C0  AE FA 40    LDX   TMPX          RESTORE X-REGISTER
40C3  60          RTS                 RETURN

                : EIN BIT VOM BAND LESEN
40C4  A2 02     RBIT  LDX   02        2 HALBWELLEN
40C6  20 C9 8D        JSR   GETR      RESYNC
40C9  20 C9 8D  WALO  JSR   GETR
40CC  90 FB           BCC   WALO      WARTE AUF HOHE FREQ.
40CE  CA              DEX             X - 1
40CF  D0 F8           BNE   WALO      MUSS SICHERHEITSHALB.
                                      2 X KOMMEN

                : ZAEHLEN DER HALBWELLEN MIT HOHER FREQ
40D1  E8        HFCT  INX             X + 1
40D2  20 C9 8D        JSR   GETR      FREQUENZ-TEST
40D5  B0 FA           BCS   HFCT      HOHE FREQ ZAEHLEN
40D7  E0 04           CPX   04        1=3 CYC,0=6 CYC
40D9  B0 02           BCS   CYZO      WENN GROESSER 3
40DB  38        CYON  SEC             EINE LOG 1 GEFUNDEN
40DC  60        RTRN  RTS             ZURUECK MIT CARRY SET
40DD  18        CYZO  CLC             EINE LOG 0 GEFUNDEN
40DE  60              RTS             RETURN,BIT IN CARRY

                : LESE EIN BYTE AUS 2 ASCII ZEICHEN
                : WIRD EIN HEX BYTE ZUSAMMENGESTELLT
40DF  20 AD 40  RDBX  JSR   RDCX      LESE EIN BYTE (ASCII)
40E2  C9 2F           CMP   '/'       ENDE DER DATEN
40E4  F0 0E           BEQ   RDND      JA
40E6  20 3E 8E        JSR   PAKT      ASCII - HEX
40E9  B0 F1           BCS   RTRN      NICHT HEX -FEHLER-
40EB  AA              TAX             SICHERN ACCU in X
40EC  20 AD 40        JSR   RDCX      LESE 2. BYTE (ASCII)
40EF  86 FC           STX   CHAR      GESICHERTES BYTE
40F1  4C 3E 8E        JMP   PAKT      2. HALBBYTE VERARB.

                : DATEN ENDE
40F4  68        RDND  PLA             GESICHERTE RETURN ADR
40F5  68              PLA             IGNORIEREN
40F6  4C 6A 40  JMPE  JMP   LOT8      ZUR ENDROUTINE

40F9  13        LOP   .HS   13        LOP FUER VERSCHIEBE-
40FA  00        TMPX  .HS   0000      Lader
40FC  EA              NOP             NOP FUER RELOC
```

KLEINANZEIGEN DER LESER

MICROSOFT-BASIC. SUCHE ORIGINAL SOURCE LISTINGS MIT KOMMENTAREN.
H. GRAWE, POSTFACH 1806, 5100 AACHEN. TEL.: 0241 - 80 77 18.

NOTVERKAUF: AIM 65 - 1 K RAM UND 4 K ASSEMBLER ROM. NEUPREIS DM 1240,-.
NOCH NICHT BENUTZT: VK DM 1050,-. AUSSERDEM: "SC/MP" NACH ELEKTOR, HEXA IN-/
OUTPUT, CPU-, MUX/INTERRUPT- UND BUSPLATINE, MONITOR +LIT., VK 500,- DM.
MICHAEL OSSEN, JOHANNESGASSE 6, 85 NÜRNBERG, TEL.: 0911 - 22 53 40, 18-19 UHR.

PET.DER HERAUSGEBER S. GÜNSTIG AUCH GEBR. GERÄT. TEL.: 04102 - 55 816.

EPROM-PROGRAMMIERER KIM-1: 2708

Ingo Dohmann, Im Wiehagen 15, 4830 Gütersloh 12

E: *Inexpensive interface and program to get 2708's burnt in.*

Das von Herrn Dohmann entworfene und seit längerem betriebene Interface dürfte nur wenige Mark kosten. Hier sein - wie immer - knapp kommentiertes Programm:

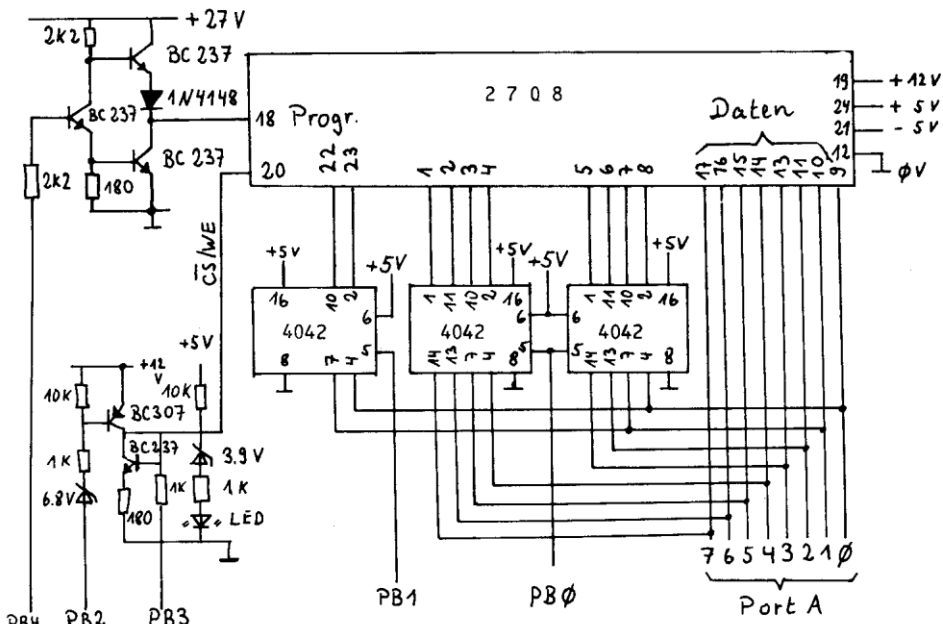
```
0000                ZAEHLS  *=*+1          1702    PB      *=*+1
0001                ZAEHLL  *=*+1          1703    PBD     *=*+1
0002                ZAEHLH  *=*+1          1704    T1000   =$1707
0003                POINTL  =$FA           1704    SHOW    =$1DAC
0003                POINTH  =$FB
0003                        *=$1700
1700                PA      *=*+1
1701                PAD     *=*+1

0200  D8            INIT    CLD
0201  A9 00                  LDA #$00
0203  85 01                  STA ZAEHLL
0205  85 02                  STA ZAEHLH
0207  8D 00 17                STA PA-PORT A  AUSGABE LÖSCHEN
020A  A9 FF                  LDA #$FF
020C  8D 01 17                STA PAD  PORT A  Data direction
020F  8D 03 17                STA PBD  "   B   Data Direction
0212  A9 17                  LDA #$17
0214  8D 02 17                STA PB-Data Reg.  PORT VORBESETZEN   0 1 2 3 4
0217  A9 14                  LDA #$14
0219  8D 02 17                STA PB - Data Reg.  SPEICHER GELÖSCHT
021C  00                     BRK
021D  EA                     NOP
021E  20 B5 02    PRUEF   JSR AUSGAB       PRÜFEN OB GELÖSCHT
0221  A9 00                  LDA #$00
0223  8D 01 17                STA PAD          ALS INPUT
0228  0D 02 17                ORA PB           0226  A9 08    LDA #$08
022B  8D 02 17                STA PB           CS EIN
022E  AD 00 17                LDA PA           INHALT LESEN
0231  49 FF                  EOR #$FF         INVERTIEREN
0233  A8                     TAY
0234  A9 F7                  LDA #$F7         CS AUS
0236  2D 02 17                AND PB
0239  8D 02 17                STA PB
023C  A9 FF                  LDA #$FF
023E  8D 01 17                STA PAD          ALS AUSGABE
0241  98                     TYA
0242  F0 03                  BEQ OK
0244  4C F8 02                JMP STOER
0247  20 DA 02    OK      JSR ERHOE
024A  D0 D2                  BNE PRUEF        WENN FERTIG:STOP
024C  00                     BRK
024D  EA                     NOP
024E  A9 10                  LDA #$10 - WIE   PROGRAMMIEREN
0250  8D 02 17                STA PB           SPEICHER TAKTEN
0253  A9 64                  LDA #$64         ≙ DEZ. 100
0255  85 00                  STA ZAEHLS       SCHLEIFENZÄHLER
0257  A0 00                  LDY #$00
```

```
0259  20 B5 02    PROGR   JSR AUSGAB      Ausgabe der Adresse
025C  B1 01                LDA (ZAEHLL),Y
025E  8D 00 17             STA PA          Ausgabe des Datums.
0261  98                   TYA
0262  21 01                AND (ZAEHLL,X)  ZEIT ZUM EINPENDELN
0264  21 01                AND (ZAEHLL,X)
0266  21 01                AND (ZAEHLL,X)
0268  98                   TYA
0269  8D 02 17             STA PB
026C  20 ED 02             JSR ZEIT
026F  A9 10                LDA #$10        BRENNIMPULS ...
0271  8D 02 17             STA PB          ABSCHALTEN
0274  20 DA 02             JSR ERHOE
0277  D0 E0                BNE PROGR
0279  C6 00                DEC ZAEHLS
027B  D0 DC                BNE PROGR
027D  A9 14                LDA #$14
027F  8D 02 17             STA PB          ALLES ABSCHALTEN
0282  00                   BRK
0283  EA                   NOP
0284  20 B5 02    VERGL   JSR AUSGAB      VERGLEICHEN VON EPROM UND
0287  A9 00                LDA #$00        SPEICHER
0289  8D 01 17             STA PAD         ALS INPUT
028C  A9 08                LDA #$08                  028E  0D 02 17  ORA PB
0291  8D 02 17             STA PB          CS EIN
0294  AE 00 17             LDX PA
0297  A9 F7                LDA #$F7
0299  2D 02 17             AND PB
029C  8D 02 17             STA PB          CS AUS
029F  A0 00                LDY #$00
02A1  A9 FF                LDA #$FF
02A3  8D 01 17             STA PAD         ALS AUSGABE
02A6  8A                   TXA
02A7  D1 01                CMP (ZAEHLL),Y
02A9  F0 03                BEQ WEITER
02AB  4C F8 02             JMP STOER
02AE  20 DA 02    WEITER  JSR ERHOE
02B1  D0 D1                BNE VERGL
02B3  00                   BRK
02B4  EA                   NOP

02B5  A5 01       AUSGAB  LDA ZAEHLL      UNTERPROGRAMM Z. ADRESSENAUSGABE
02B7  8D 00 17             STA PA  A Data Reg.
02BA  EE 02 17             INC PB  B-Data Reg -1
02BD  EA EA                NOP,NOP
02BF  CE 02 17             DEC PB  B-Data Reg +1
02C2  A5 02                LDA ZAEHLH
02C4  8D 00 17             STA PA  A-Data Reg.
02C7  A9 02                LDA #$02
02C9  0D 02 17             ORA PB          SPEICHERTAKT
02CC  8D 02 17             STA PB Data Reg B
02CF  EA EA                NOP,NOP
02D1  A9 FD                LDA #$FD
02D3  2D 02 17             AND PB
02D6  8D 02 17             STA PB
02D9  60                   RTS
02DA  E6 01       ERHOE   INC ZAEHLL      UPRO:ADRESSE UM 1 ERHÖHEN
02DC  D0 0E                BNE WEIT        KEIN ÜBERTRAG
```

```
02DE  E6 02              INC ZAEHLH
02E0  A9 24              LDA #$24
02E2  C5 02              CMP ZAEHLH      ENDE ERREICHT?
02E4  D0 06              BNE WEIT
02E6  A9 20              LDA #$20        NEU LADEN
02E8  85 02              STA ZAEHLH
02EA  A9 00              LDA #$00        FLAG SETZEN
02EC  60          WEIT   RTS
02ED  A9 01       ZEIT   LDA #$01        ERZEUGE PROGRAMMIERIMPULSZEIT
02EF  8D 07 17            STA T1000
02F2  2C 07 17    LP      BIT T1000
02F5  10 FB               BPL LP
02F7  60          RTS
02F8  A5 01       STOER   LDA ZAEHLL     STOERUNG MIT ADRESSE ANZEIGEN
02FA  85 FA               STA POINTL
02FC  A5 02               LDA ZAEHLH
02FE  85 FB               STA POINTH
0300  4C AC ID            JMP SHOW
```



INTERFACE FÜR DIE EPROM-PROGRAMMIERUNG

FORTSETZUNG:

Für die Inbetriebnahme des HAMMING-WAY sind folgende Informationen nachzutragen: Man verwende Bandcassetten mit hohen Gleichlaufeigenschaften, wie BASF SM. CRO2-Band ist nicht notwendig. Wenn man es doch verwendet, dann sollte beim Bandlesen die CRO2-Taste nicht gedrückt sein. Wenn das Leseprogramm häufig mit ERROR strandet, dann verarbeitet Ihr Gerät den EOR-Befehl in 040A wahrscheinlich nicht richtig. In diesem Falle lesen Sie bitte unter AIM SPEZIAL in diesem Heft nach, ehe Sie sich die Zähne unnötig ausbeißen.

R. L.

A S P - Advanced Subroutine Package (Teil 5)

Michael Zimmermann, Eberstädter Str. 170, 6102 Pfungstadt

E: *These new subroutines cover packing of ASCII data and searching in tables as well as data recovery from tables.*

Inhaltsverzeichnis für den 5. Teil

* * *

Hinweise für die Benutzer des AIM 65

Alle bis jetzt vorgelegten ASP-Unterprogramme sind ohne Einschränkung auch für den AIM geeignet, mit anderen Maschinen konnte der Verfasser bisher keine Erfahrung sammeln. Bei der Anordnung des Arbeitsbereiches in der Page Zero des AIM ist aber zu beachten, daß dort von AD-FF der Ausgabepuffer angesiedelt wurde.

Um hier eine Sicherheit, auch hinsichtlich eines möglichen größeren Puffers für einen Mini-Floppy-Controller zu erreichen, ist eine Zuordnung des Arbeitsbereiches an Adressen unterhalb $80 empfehlenswert. Bei einem 4 kAIM ist es ohne große probleme möglich, sämtliche ASP-Routinen speicherresident zu halten. Diese sollten sinnvoll im oberen Teil des Speichers stehen, wobei die von den ASP-Routinen selbst aufgerufenen Unterprogramme, wie VORB, LEN usw., an den allerhöchsten Adressen stehen sollten, damit beim eventuellen Fortlassen entbehrlicher Routinen deren Adressierung in den Aufrufen nicht geändert werden muß.

3.2 Packen von alphanumerischen Daten (PACK)

Nachdem am AIM alle numerischen Daten generell im ASCII-Format eingegeben werden, soll an dieser Stelle das in der letzten Folge angekündigte Unterprogramm zum Packen von Daten veröffentlicht werden.

Der Aufruf hat folgende Struktur:

```
      JSR    PACK
      .BYTE  Länge des Zielfeldes in den oberen 4 Bits verschlüsselt,
             Länge des Quellfeldes in den unteren 4 Bits verschlüsselt.
      .WORD  Adresse der gepackten Zahlen (Zielfeld)
      .WORD  Adresse des ASCII-Daten enthaltenden Quellfeldes.
```

Nach Obernahme der Parameter wird die Länge der Datenfelder berechnet und der Arbeitsbereich gelöscht. Es folgt Lesen und Prüfen der Zeichen aus dem Quellfeld. Liegt ein Minuszeichen vor, so wird das Negativ-Flag in F0 gesetzt. Ist das eingelesene Zeichen außerhalb des zulässigen Wertebereiches von 0 bis 9, so wird zum Fehlerausgang mit gesetztem Carry-Flag verzweigt.

Ein gültiges Zeichen wird im Akkumulator zunächst um 4 Bitpositionen nach links verschoben und anschließend mit dem gesamten Arbeitsbereich um

weitere 4 Bitstellen geshiftet, so daß das zuletzt bearbeitete Zeichen an der niederwertigsten Stelle des Arbeitsbereiches steht. Es folgt die Bearbeitung des nächsten Zeichens, bis das Quellfeld bearbeitet ist. Die jetzt im Arbeitsbereich enthaltene gepackte Information wird jetzt in das Zielfeld übertragen. - Sollen neben dem Minuszeichen noch weitere nichtnumerische hexadezimale Kombinationen bearbeitet werden, so sind die entsprechenden Abfragen und Verarbeitungen an der betreffenden Stelle einzubauen.

```
XX00  20 XX XX    PACK    JSR VORB        INITIALIZE
  03  20 XX XX            JSR LEN         GET LENGTH
  06  20 XX XX            CLR             CLEAR WORKA
  09  A0 00               LDY #$00        FOR INDIRECT ADDRESSING
  0B  B1 E5       P10     LDA (E5),Y      LOAD ASCII CHAR.
  0D  C9 2D               CMP #$2D        A "-", NEGATIVE ?
  0F  D0 04               BNE *+4         SKIP
  11  E6 F1               INC $F1         SET NEG FLAG
  13  D0 1D               BNE P50         BRANCH ALWAYS
  15  C9 30       P20     CMP #$30        CHECK VALID RANGE
  17  90 28               BCC PE          BELOW
  19  C9 3A               CMP #$3A
  1B  B0 24               BCS             BEYOND
  1D  A2 04               LDX #$04        INITIALIZE ROL
  1F  86 F6               STX $FC
  21  A2 0F               LDX #$0F
  23  2A 2A 2A            ROL,ROL,ROL
  26  2A                  ROL
  27  2A          P30     ROL
  28  3E D0 00    P40     ROL $D0,X       MEMORY D0-DF
  2B  CA                  DEX
  2C  10 FA               BPL P40         INNER LOOP
  2E  C6 F6               DEC $F6
  30  D0 F5               BNE P30         OUTER LOOP
  32  C8          P50     INY             INCR. CHARACTER COUNT
  33  C4 FA               CPY $FA
  35  D0 D1               BNE P10         NEXT CHARACTER
  37  A4 F9               LDY $F9         MOVE PACKED DATA OUT OF
  39  A2 DF               LDX #$DF        ZERO PAGE BUFFER
  3B  20 XX XX            JSR MNACH+2
  3E  18                  CLC             FLAG 'OKAY'
  3F  90 01               BCC *+1         SKIP ALWAYS
  41  38          PE      SEC             FLAG'ERROR'
  42  60                  RTS
```

5. Tabellenverarbeitung

Die Verarbeitung von Tabellen hat große Ähnlichkeit mit der Vektorbearbeitung. Ohne die Grundlagen im einzelnen zu beschreiben, soll an dieser Stelle erklärt werden, daß bei der Tabellenverarbeitung das Suchen in strukturierten Daten im Vordergrund steht, wohingegen bei der Vektorbearbeitung Schleifen den Schwerpunkt bilden. Beiden ist gemeinsam, daß in den Unterprogrammen die Adresse des eigentlichen Befehls modifiziert wird.

5.0 Übernahme der Parameter für die Tabellen (PARM)

Da bei der Verarbeitung mehr Informationen benötigt werden, als in den

5 Bytes bei dem normalen Aufruf unserer Unterprogramme unterzubringen sind, müssen diese Daten vor dem eigentlichen die Tabelle verarbeitenden Unterprogramm übergeben werden. Diese Daten werden vom verarbeitenden Unterprogramm in den Speicherstellen E7 ff. erwartet.

Das Unterprogramm PARM wird in folgender Form aufgerufen und übergibt die nachfolgend beschriebenen Angaben:

```
JSR    PARM
.BYTE  Länge des Suchargumentes
.BYTE  Gesamtzahl der Raster in der Tabelle
.BYTE  Länge eines Rasters
.WORD  Adresse des Suchargumentes bzw. der Nummer des zu isolierenden Rasters
```

Aus den 1 Byte breiten Parametern ergibt sich, daß eine Tabelle maximal 256 Raster haben kann und das ein einzelnes Raster ebenfalls nicht länger als 256 Bytes sein kann.

```
XX00  20 XX XX   PARM   JSR VORB      GET PARMS FROM CALLER
  03  A2 04             LDX #$04      LOAD COUNT
  05  B5 E2     P10     LDA $E2,X     AND MOVE
  07  95 E7             STA $E7,X     TO NEW DESTINATION
  09  CA                DEX
  0A  10 F9             BPL P10
  0C  60                RTS
```

5.1 Suchen in einer Tabelle (SEARCH)

Mit diesem Unterprogramm ist es möglich, ein Suchargument gegen Suchbegriffe in einer Tabelle zu vergleichen. Bei Gleichheit wird eine Adresse modifiziert, und es ist möglich, aus dem entsprechenden Raster einen Wert zu bearbeiten.

Der Aufruf hat folgenden Aufbau:

```
JSR    SEARCH
.BYTE  Offset, Stelle an der der zu bearbeitende Wert relativ zum
       Suchargument im Raster steht,
.WORD  Adresse, die modifiziert werden soll
.WORD  Basisadresse der Tabelle.
```

Das Unterprogramm übernimmt zuerst die Parameter. Darauf wird die Länge des Suchargumentes, die mit der Länge des Suchbegriffes identisch ist, in das Y-Register geladen und Suchargument und Suchbegriff werden verglichen. Bei Ungleichheit wird im nächsten Raster weiter verglichen. Wurden sämtliche Raster bis zum Ende der Tabelle abgearbeitet, ohne daß Gleichheit eintrat, so wird das Carry-Flag gesetzt.

Bei Gleichheit wird die Adresse des zu bearbeitenden Begriffes (Feldes) ermittelt und an der vereinbarten Stelle abgespeichert, so daß eine weitere Verarbeitung stattfinden kann. Nach Ablauf des Unterprogrammes sollte das Carry-Flag abgefragt werden, um sich des erfolgreichen Suchens zu vergewissern.

```
XX00  20 XX XX   SEARCH  JSR VORB      GET PARMS
  03  A4 E7       S10    LDY $E7       SET INITIAL CONDITIONS
  05  38 D8              SEC,SED
  07  B1 EA       S20    LDA (EA),Y    COMPARE TABLE
  09  F1 E5              SBC (E5),Y    AND PARAMETER
  0B  D0 16              BNE S30       NOT FOUND
  0D  88                 DEY           ADDRESS NEXT CHAR.,COUNT DOWN
  0E  10 F7              BPL S20
```

```
XX10  18              CLC             FOUND
  11  A0 00           LDY #$00        MODIFY ADDRESS ...
  13  A5 E5           LDA $E5         AND STORE IT INDIRECT AT $E3
  15  65 E2           ADC $E2
  17  91 E3           STA (E3),Y
  19  C8              INY
  1A  A5 E6           LDA $E6         DITTO HIGH
  1C  69 00           ADC #$00        IF THERE WAS A CARRY
  1E  91 E3           STA (E3),Y
  20  18              CLC             FLAG 'FOUND'
  21  90 12           BCC S90         BRANCH TO RETURN
  23  18        S30   CLC
  24  A5 E5           LDA $E5         INCREMENT POINTER TO TABLE
  26  65 E9           ADC $E9
  28  85 E5           STA $E5         = NEW POINTER LOW
  2A  90 02           BCC *+2         SKIP ON NO CARRY
  2C  E6 E6           INC $E6
  2E  C6 E8           DEC $E8         DECREMENT RASTER COUNT
  30  D0 D1           BNE S10         REPEAT ON POSITIVE
  32  38              SEC             FLAG 'NOT FOUND'
  33  60        S90   RTS
```

5.2 Holen des Wertes aus einer Tabelle (GETTAP)

In vielen Fällen ist das Tabellenraster, in dem die benötigte Information steht, bekannt. In diesem Falle reicht die Angabe der Rasternummer in hexadezimaler Form, und ein Suchvorgang ist nicht erforderlich (direkte Zuordnung).

Der Aufruf hat folgende Form:

```
    JSR    GETTAB
.BYTE  Offset, Stellung des zu bearbeitenden Wertes relativ z.Rasteranfang
.WORD  Adresse, die modifiziert werden soll
.WORD  Adresse der Tabelle
```

Das Unterprogramm übernimmt die Parameter und vergleicht, ob das vorgegebene Raster innerhalb der Tabelle liegt. Wenn nicht, wird das Carry-Flag gesetzt und das Unterprogramm abgebrochen. Wenn ja, so wird mit der Rasternummer und der Rasterlänge auf dieses Raster eingestellt, die gefundene Adresse mit der Stelle modifiziert und diese Adresse zur eigentlichen weiteren Bearbeitung abgespeichert.

```
XX00  20 XX XX    GETTAB  JSR VORB
  03  A2 00              LDX #$00
  05  A5 E8              LDA $E8
  07  C1 EA              CMP (EA,X)      ARE WE INSIDE TABLE?
  09  B0 25              BCS G90         NO
  0B  A1 EA              LDA (EA,X)
  0D  85 E8              STA $E8
  0F  18 D8              CLC,CLD
  11  A5 E5        G10   LDA $E5         ADDING LOOP TO ADVANCE
  13  65 E9              ADC $E9         TO ADDRESS OF REQUIRED
  15  85 E5              STA $E5         ELEMENT
  17  90 02              BCC *+2         SKIP ON NO CARRY
  19  E6 E6              INC $E6
  1B  C6 E8              DEC $E8         DID WE ARRIVE AT THE FOOT OF
  1D  D0 F2              BNE G10         REQUIRED ELEMENT?
  1F  A0 00              LDY #$00
  21  18                 CLC
```

Fortsetzung auf Seite 13

Ein Leitfaden für die Programmierung (Teil 3)

Im noch laufenden Abschnitt 7.3 werden die logischen Befehle besprochen. Der Leser wird bemerkt haben, daß es in Heft 5, S. 43 bei PROF1 richtig heißen muß: LDX #$08. Besprochen wurde bisher vor allem der AND-Befehl.

Im Akkumulator wird eine Maske vorgehalten, die auf einen Direktoperanden oder auf eine Speicherzelle (die auch ein Peripherieregister sein darf) angewandt wird. Das Ergebnis der bitweisen Verknüpfung gelangt in den Akku, womit die Maske zerstört wird. Nach dem AND findet sich eine logische '1' im Ergebnis nur an den Bitstellen, an denen sich sowohl in der Maske als auch im Operanden eine '1' befunden hatte.

In messenden und steuernden Anwendungen kann man also das logische 'sowohl als auch' feststellen. Wie die Beispiele 7.4.4 bis 7.4.8 zeigen, programmiert man - wegen der Zerstörung der Maske im Akku - in vielen Fällen ökonomischer mit dem BIT-Befehl (der ja auch ein logisches UND bewirkt, ohne den Akkuinhalt zu zerstören), in Zusammenarbeit auch mit den Verschiebebefehlen.

In arithmetischen Anwendungen verwendet man das 'Zurechtschneiden' mit AND, um mit einem Operanden eine Tabellenadressierung vorzunehmen, wie z.B. in den Programmabschnitten AVEC und H10 im Programm HAMMING-WAY in dieser Ausgabe.

Beispiel 7.4.9 Maskieren eines Bytes zum Zwecke indizierter Adressierung. Indiziert wird mit den niedrigsten 3 Bit von WERT.

```
LDA WERT
AND #$07      BITMUSTER DES DIREKTOPERANDEN '0000 0111'
TAY           ERGEBNIS IN 3 BIT ALS ADDRESSER NACH Y
LDA TAB,Y     LADEN EINES TABELLENWERTES.
```

Beispiel 7.4.10 Maskieren eines Bytes zum Zwecke indizierter Adressierung. Es soll mit den Bits 3 , 4 und 5 indiziert werden.

```
LDA WERT
LSR           DIE ADRESSIERENDEN BITS RECHTSBÜNDIG AUSRICHTEN
LSR
LSR
AND #$07      ABSCHNEIDEN UNERWÜNSCHTER BITS
TAX           ERGEBNIS ALS ADDRESSER NACH X
STA TAB,X     BEARBEITUNG AM INDIZIERTEN SPEICHERPLATZ
```

Beispiel 7.4.11 Maskieren eines Bytes zum Zwecke indizierter Adressierung. Es soll mit den Bits 3 , 4 und 5 auf jeden zweiten Platz einer Tabelle indiziert werden.

```
LDA WERT
AND #$38      MASKE MIT '0011 1000'
LSR           DAS ADRESSIERENDE ERGEBNIS BLEIBT MIT '2'
LSR           MULTIPLIZIERT
TAY           ALS ADDRESSER
CMP TAB,Y     ARBEIT MIT DER ADRESSIERTEN STELLE IN DER TABELLE.
```

Der ORA-Befehl gestattet das 'Hineinkopieren' eines Bitmusters in eine im Akkumulator vorgehaltene Maske. Im Ergebnis wird an allen Bitstellen eine logische '1' stehen, an denen im Akkumulator oder im Operanden oder in beiden eine '1' vorhanden war.

Beispiel 7.4.12 Am Port eines Peripheriebausteines soll eine als Ausgang programmierte Leitung PB3 mit Sicherheit auf 'high' gesetzt werden, ohne daß die bisher ausgehenden Signale unterbrochen oder gestört werden.

```
LDA DRB     LADE BISHERIGEN INHALT DES PORTS
ORA #$08    KOPIERE '0000 1000' HINEIN
STA DRB     NEUER INHALT DES PORTS. PIN PB3 WIRD AUCH DANN
            AUF 'HIGH' GESCHALTET, WENN ES VORHER SCHON 'HIGH'
            HATTE. ES IST JA KEINE PRÜFUNG DES BISHERIGEN
            AUSGANGSSIGNALS ERFOLGT.
```

Beispiel 7.4.13 Ein Byte ist in zwei ASCII-Zeichen empfangen worden. Das vordere dekodierte Halbbyte stehe bereits linksbündig in Zelle 0200, das hintere rechtsbündig im Akku.

```
ORA $0200   VOLLSTÄNDIGES BYTE JETZT IM AKKU
```

Der ORA-Befehl eignet sich nach vorstehenden Beispielen zum 'Hinzuschalten' von Bits. Für die Abfrage von Bitmustern ist er wenig geeignet:

Beispiel 7.4.14 Feststellen, ob PB3 logisch '1' führt.

```
LDA DRB     LADE BISHERIGEN INHALT DES PORTS
AND #$80    MASKIERE MIT '0000 1000'
ORA #$00    MASKIERE MIT '0000 0000'
BNE TU      VERANLASSE ETWAS, WENN PB3=1
```

In der Praxis nimmt man die kürzere Befehlsfolge nach Beispiel 7.4.3 mit AND, weil das Statusregister nach AND bereits die benötigte Information enthält.

Vielseitige Möglichkeiten bietet der logische Befehl EOR. Auch er arbeitet bitweise Maske im Akkumulator gegen Operand. Seine Wirkung ist gleich wie beim ODER, soweit es sich nicht (exklusive der) Kombination '1' gegen '1' handelt, bei der das Bitergebnis '0' wird.

```
1 0 1 0 1 0 1 0     BITMUSTER OPERAND
0 1 1 0 1 0 0 1     MASKE IM AKKU
-----------------
1 1 0 0 0 0 1 1     ERGEBNIS IM AKKU NACH EOR
  x   x             x 'EXCLUSIVE-OR'
```

M.a.W.: Das Ergebnis hat in all' denen Bitstellen den entgegengesetzten Wert wie der Operand, in denen die Maske mit einer '1' vorbesetzt war.

Der Leser überzeugt sich sofort, daß die nachstehenden Instruktionen zu einer völligen Umkehr, zur 'Verneinung', des Operanden führen:

```
LDA #$FF    DIREKTOPERAND '1111 1111'
EOR OPERAND NEGATION, NOT IM AKKU
```

Damit ist dann eine vierte logische Operation, das NOT, ausgeführt, die zur arithmetischen Komplementbildung in verschiedenen Zahlensystemen angewandt werden kann (s. Programm AIMPLOT, Abschnitt VALDOT).

Beispiel 7.4.15 Am Port DRB ist PB5 einzuschalten, wenn es vorher 'low' führte, es ist auszuschalten, wenn es vorher 'high' war (to flip). Andere Pins dürften nicht gestört werden.

```
LDA DRB     HOLE BISHERIGES BITMUSTER DES PORTS
EOR #$20    MASKE '0010 0000' FÜR PB5
STA DRB     UMSCHALTUNG VON PB5.
```

Die volle Leistungsfähigkeit entfaltet der EOR-Befehl in Zusammenarbeit mit AND und CMP.

Beispiel 7.4.16 Wenn an den Eingangspins PB4 und PB5 zugleich 'high' level herrscht, soll der Pegel an Ausgangspin PB0 invertiert (geflipt) werden. Wenn nur PB4 oder PB5 'high' ist, soll PB1 invertiert werden. Bei PB4=PB5=low soll gewartet werden.

```
WAIT   LDA DRB      HOLE INHALT DES PORTS
       TAX          SCHNELLE ZWISCHENSPEICHERUNG
       AND #$30     MASKE '0011 0000' FÜR PB4, PB5
       BEQ WAIT     BEIDE PINS FÜHREN 'LOW'
       CMP #$30     SIND BEIDE 'HIGH'?
       BEQ BEIDE
EINER  TXA          HOLE PORTINHALT ZURÜCK
       EOR #$02     BITMUSTER FÜR PB1 ZUR INVERTIERUNG
       STA DRB      ZURÜCKSCHREIBEN IN DEN PORT
       ...
BEIDE  TXA          WIE OBEN
       EOR #$01     INVERTIERUNG FÜR PBØ
       STA DRB      ZURÜCKSCHREIBEN
```

Beispiel 7.4.17 Gleiche Aufgabenstellung wie in 7.4.16. Lösung mit einer kleinen Tabelle ab Speicherzelle 00E0 wie in Beispiel 7.4.10.

```
       LDA DRB      PORTINHALT
       AND #$30     MASKIEREN FÜR PB4, PB5
       LSR, LSR     BILDE AUS DEM VERBLEIBENDEN BITMUSTER
       LSR, LSR     EINEN ADDRESSER FÜR TABELLE
       TAX          ZUR INDIZIERUNG
       LDA $E0,X    HOLE OPERANDEN FÜR EOR
       EOR DRB      INVERTIERUNG
       STA DRB      ZURÜCKSCHREIBEN
00E0   00           TABELLE MIT OPERANDEN AB ADRESSE 00E0
00E1   02
00E2   02
00E3   01
```

Es ist leicht einzusehen, daß man bei komplizierten logischen Entscheidungen übersichtlicher und bequemer mit der Tabellenmethode verfährt, als mit einem länglichen Abfragegerüst.

WIRD AUSFÜHRLICH FORTGESETZT

R. L.

****

BUCHBESPRECHUNG: *Microprocessor Interface Techniken, von Austin Lesea und Rodnay Zaks, Markdorf 1979, Verlag Micro-Shop-Bodensee, 414 S. DM 39,-*

Dem Verleger ist für die Initiative zu danken, das bewährte SYBEX-Buch in deutscher Übersetzung vorzulegen. Es ist ein Hardware-Buch, das in acht großen Kapiteln mit zahllosen Schemazeichnungen, Schaltungen und Fotos alle Fragen behandelt, die mit der Datenein- und -ausgabe, mit dem Anschluß von Peripherieeinheiten ('von der Tastatur bis zum Floppy-Disk') zusammenhängen. Übersichten über Formatierungen und Signalstandards runden die Darstellungen ab. Ein eigentlich unentbehrliches Buch für jeden, der E/A-schaltungen entwirft oder verstehen will. Wegen des Fehlens eines eigenen deutschen Vokabulars für das Gebiet muß die Übersetzungsarbeit für Bernd Pol schwierig gewesen sein. Ihm ist eine klar verständliche Darstellung gelungen.

FORTSETZUNG VON SEITE 2: AIM SPEZIAL

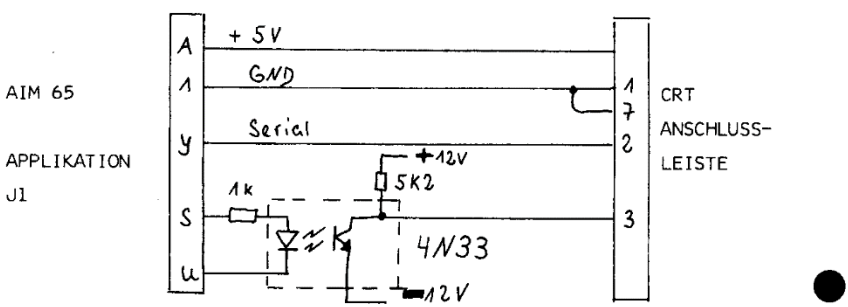
Klassisches Beispiel hierfür ist der Befehl in O4OA des Programmes HAMMING-WAY in diesem Heft. Nach den Untersuchungen von Herrn I. Dohmann können Einflüsse der Spannungsversorgung nicht ausgeschlossen werden, beim Herausgeber half eine geerdete Alu-Platte unter dem AIM, die aber zu einer gelegentlichen Verabschiedung des Monitors aus seiner Warteschleife führte. Alle Hochfrequenzgeräte waren abgeschaltet.

Die USER OUTPUT FUNCTION mit Vektor in O1OA konnte hier noch nicht eingehend studiert werden. Sie sei nicht so einfach zu verwirklichen wie die Eingabefunktion. Vielleicht gibt es elegante Lösungsvorschläge hierzu?

Ein AIM- und KIM-Ausbausystem wird nicht nur von der GWK in Übach-Palenberg (s. Anzeige), sondern mit gepuffertem S-64E-Bus auch von Pro Comp mit vielfältiger Kartenauswahl angeboten. Nähere Angaben durch den Herausgeber.

Die Nachschubfrage für Thermopapier in einer preiswerten Spitzenqualität ist gelöst (s. Anzeige auf der Rückseite). Für das Plotten von Meßwerten (siehe Programm AIMPLOT in diesem Heft) könnte längsliniertes Registrierpapier angefertigt werden, wenn sich größere laufende Bedarfsfälle ergeben. - Unbefriedigend ist auch in der Großstadt Hamburg die Beschaffung von Eckverbindungssteckern mit 44 und z.B. auch 30 Pins (f. Tastaturen) in Kleinstückzahlen. Bei gleichen Beobachtungen an anderen Orten könnte die Beschaffungsfrage bequem auf dem Postwege geregelt werden.

Für den Anschluß eines CRT-Terminals mit Standard-Anschlußstecker ging dem Herausgeber folgender Vorschlag zu, in dem ein preiswerter Opto-Koppler verwendet wird.



VERLÄNGERN SIE BITTE RECHTZEITIG IHR ABONNEMENT!

Mit diesem 6. Heft erhalten viele Abonnenten den schriftlichen Hinweis, daß ihr Erstabonnement abgelaufen ist. Geben Sie daher bitte möglichst umgehend mit dem Antwortcoupon Nachricht, daß Sie das 65xx MICRO MAG weiter beziehen wollen, damit die Lieferung ab Juni 1979 regelmäßig weiterläuft und veranlassen Sie bitte dann Scheckzahlung und Überweisung.

MICRO - SHOP - BODENSEE HARDWARE ANGEBOT

MEMORY - PLUS

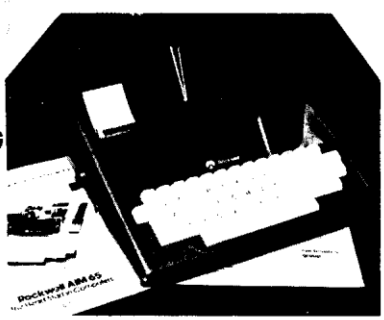
8K STATIC RAM
8K EPROM ADRESSLOGIK F. 2716/2516
EPROM PROGRAMMIERER
VIA 6522 , 2 TIMER +2x8BIT-PORT
+SERIELL/PARALLEL SHIFT REGISTER
ALLE IC AUF SOCKEL
AIM-65/SYM-1/KIM-1 KOMPATIBEL
FERTIG GETESTET,BURNED IN
PREIS Incl. 12% Mwst. DM 748,-
GEHAUESE fuer AIM-65 DM 165,-
STROMVERSORGUNG fuer AIM-65
5V/6AMP und 24V/1,5AMP DM 295,-
WIR BIETEN EINE GARANTIERTE SERVICE LEISTUNG
INNERHALB 2 JAHREN (DEFEKT,FALSCHE BEDIENUNG)
FUER DEN BETRAG VON DM 50,- ZUSAETZLICH ZUM
KAUFPREIS FUER DIE PRODUKTE : KIM;SYM;AIM; KTM-2 UND
MEMORY-PLUS UND IN KUERZE VIDEO-PLUS
MICRO-SHOP-BODENSEE LIEFERT DEN AIM AB LAGER!!!!
IN UNSEREM BUCHANGEBOT FINDEN SIE JETZT AUCH:
BASIC and the Personal Computer DM 46,-
The little Book of BASIC Style DM 29,-
UND UNSEREN B E S T S E L L E R ;:
Microprocessor I N T E R F A C E Techniken
Deutsche Aussabe DM 39,-

Wir sind 6502 SPEZIALISIERT : HARD- SOFT- und BOOKWARE
MICROCOMPUTER
KIM-1 DM 475,-
SYM-1 DM 695,-

KTM-2 DM 895,-
UPPER und lower ASCII TASTATUR
mit GRAPHIC (128)und VIDEO AUSGANG

AIM-65 (4K RAM) DM1050,-
AIM-Assembler DM 230,-
AIM- BASIC DM 280,-

HARDWARE PREISE + 12 % Mwst.



BASIC-ROMs (2) für SYM-1 jetzt lieferbar!

Vertrieb für kilobaud MICROCOMPUTING

MSB

M.&R. Nedela
Postfach 1122
Marktstraße 3
D-7778 Markdorf

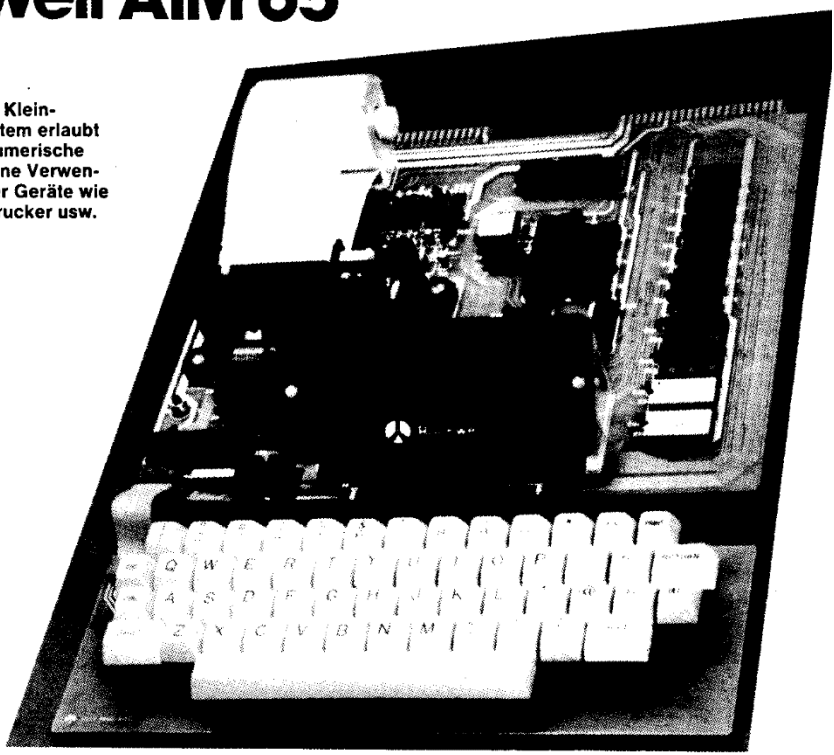
Micro-Shop-Bodensee

6502 Hardware - Software - Verkauf - Service.

For learning, designing, work or just plain fun...
The Head-Start in Computers

Rockwell AIM 65

Das vollständige Klein-Entwicklungssystem erlaubt erstmals alphanumerische Ein-/Ausgabe ohne Verwendung zusätzlicher Geräte wie TTY, CRT, TV, Drucker usw.



Rockwell

- **Tastatur im «Terminal style»**
- **20 charakter alphanumerische Anzeige**
- **20 charakter alphanumerischer Thermodrucker**
- **R 6502 CPU**
- **bis 4K RAM «on board», extern erweiterbar**
- **Interfaces für zwei Kassettengeräte und TTY**
- **Assembler und Basic in ROM**
- **Lieferbar**

PREISE:

AIM 65 mit 1K Byte RAM	875,--DM
AIM 65 mit 4K Byte RAM	1050,--DM
Resident Assembler ROM 4K	230,--DM
BASIC Interpreter ROM 8K	280,--DM
On Board RAM Erweiter. 3K	175,--DM

Alle Preise zuzüglich MWST und Versandspesen.

D 5132 Uebach Palenberg
Aachener Str. 56
Tel.: 02451 / 41911

GWK
GESELLSCHAFT
FÜR TECHNISCHE ELEKTRONIK mbH.

GWK EURO BOARD EXPANSION SYSTEM

SYSTEMERWEITERUNG FÜR AIM 65 KIM 1 SYM

Das GWK EURO BOARD EXPANSION SYSTEM bietet sowohl dem professionellen Anwender,als auch dem Hobbyisten,die Möglichkeit,seinen AIM 65,KIM 1,oder SYM optimal zu erweitern.
Dies wird erreicht durch die folgenden grundlegenden Konzepte:
Die Adressdecodierung der einzelnen Komponenten erfolgt auf jeder Karte.Das heißt,jede Komponente kann beliebig adressiert werden. Es wird kein Adressraum verschenkt.
Der GWK SYSTEM BUS und die Steckverbinder der Einzelkomponenten sind pinkompatibel zum Expansionconnector von AIM,KIM undSYM.
Hierdurch wird erreicht,daß einzelne BOARD'S auch ohne Motherboard direkt am entsprechenden Microcomputer laufen.

Bisher sind folgende Komponenten lieferbar:

MOTHER BOARD — 485,--DM

Einbau in Gehäuse 19 Zoll oder direkt an AIM,KIM,SYM steckbar. Daten-,Adress- undKontrollbus voll gepuffert und dynamisch abgeschlossen.9 Steckplätze für Expansion,3 für Application. Genügend Platz für ein Netzteil.

RAM BOARD — 895,--DM

3 x 4 K Byte statisches RAM.Unabhängig voneinander beliebig adressierbar.

EPROM BOARD — 395,--DM

12 K Byte für die 1K Eproms 2708 oder 2758

EPROM PROGRAMMER BOARD — 795,--DM

Mit residentem Betriebsprogramm.Softwareumschaltung — 745,--DM
für 1K-,2K-,4K Byte EPROM'S 2708/58,xx16,xx32.Zwei Ausführungen: Für Eproms mit einer bzw.drei Betriebsspannungen.

VIA – PIA BOARD — 285,--DM

32 I/O Kanäle.Beliebig Adressierbar.Wahlweise mit VIA oder PIA bestückbar. Zwei Obergabesteckverbinder.

In Vorbereitung: VIDEO INTERFACE,ANALOG I/O,CPU-BOARD,POWER SUPPLY, UNIVERSAL MEMORY BOARD.

Alle Preise zuzüglich Mehrwertsteuer und Versandspesen.

GWK TECHNISCHE ELEKTRONIK GmbH
HARDWARE SOFTWARE SYSTEMENTWICKLUNG
Aachener Str.56 · D 5132 Uebach Palenberg
Tel.: 02451 / 41911

65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

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

65xx MICRO MAG erscheint zweimonatlich. Beiträge, die nicht besonders gekennzeichnet sind, stammen vom Herausgeber.

BEZUGSBEDINGUNGEN: Abonnement für 6 Hefte im Inland DM 40,- (Endpreis). Ausland/Foreign: DM 46,- (surface mail).
Firmen erhalten Rechnung, Rechnungserteilung sonst nur auf Wunsch. Richten sie bitte Ihre Überweisung/Euroscheck an:

Roland Löhr, Konto 20/01121, Vereins- und Westbank, BLZ 200 300 00.

Alle früher erschienenen Hefte dieser Zeitschrift sind weiterhin lieferbar. Einzelne Hefte können zu DM 7,- inkl. Porto nachbezogen werden.

REDAKTIONS- UND ANZEIGENSCHLUSS FÜR NR. 7 ⇨ 1. JUNI 1979.

Für die nächsten Hefte sind folgende Themen geplant: Leitfaden für die Programmierung - 6522 VIA in verschiedenen Anwendungen - Systemprogramme für AIM 65 - Assembler des AIM - vielseitiges Sortierprogramm - Makros u.a.m..

65xx MICRO MAG brachte bereits: Advanced Subroutine Package ab Heft 2 - Leitfaden für die Programmierung ab Heft 4 - Heft 1: Verschieblich laden - Universal-Timer - Etikettensuche/Statistik mit Magnetband - Heft 2: QUICKDUMP/VERSALOAD schnelles leistungsfähiges Programmpaket für die Speicherung - SUMMARY und Stringhunter (Magnetband) - Zahlenwandlung - Heft 3: Makros für 65xx - Timer und Interrupt - RAM Test with random patterns - Disassembler - Zahlenwandlung - Heft 4: Printerprogramm für Mini Dot - Speicherauszug mit Printer - ALPHASORT- Heft 5: Anwenderberichte AIM 65 und SYM-1 - zwei Systemprogramme für AIM - Hinweise auf den AIM USER's GUIDE u.v.a.m..