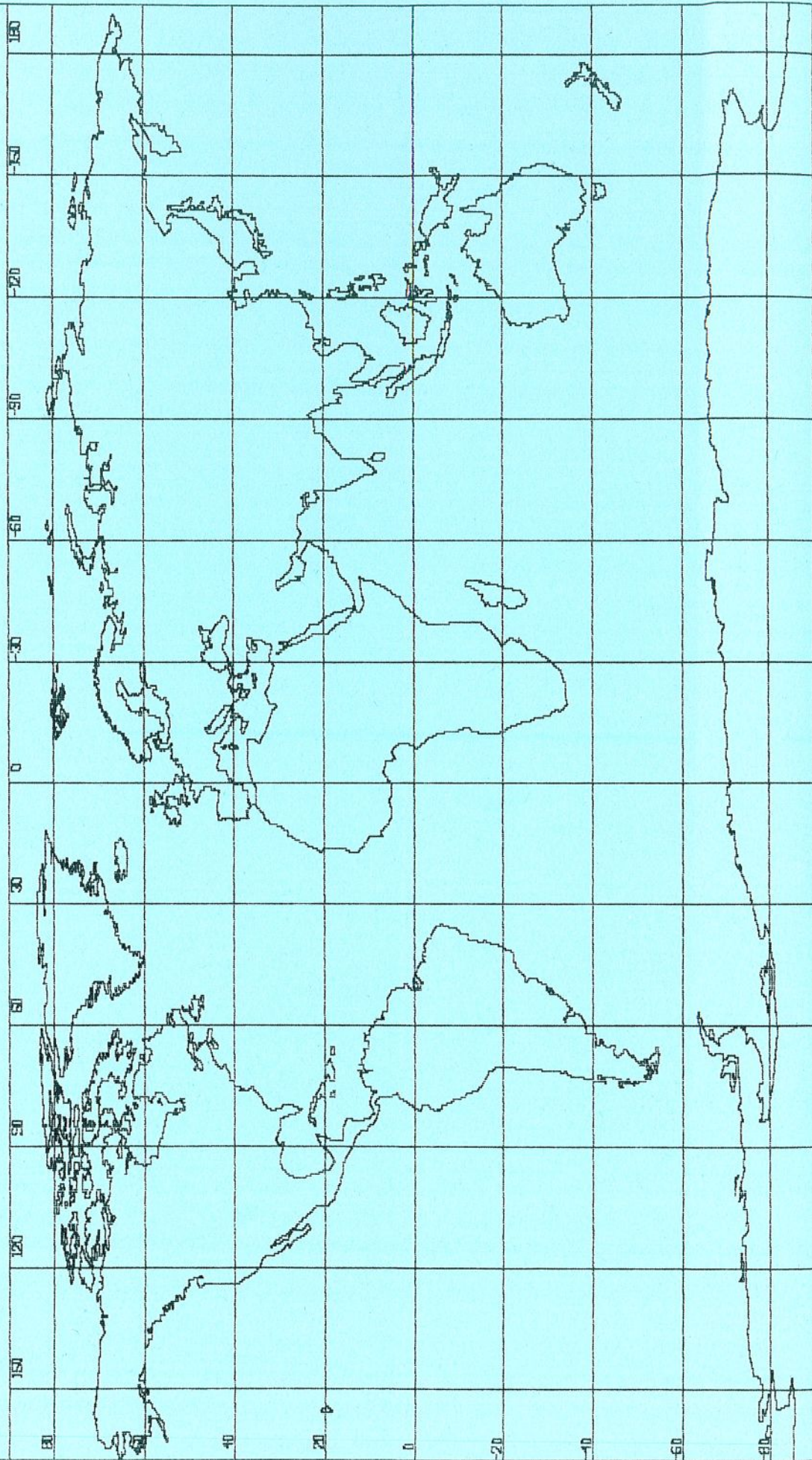


CLUBZEITUNG

21. AUSGABE



AUS DEM INHALT:

Hochauflösende Softgraphic von Jürgen Degenhardt	Teil 2	3
Bericht KOMTEK 1 von Wolfgang Reichelsdorfer		8
Zeichnen von Linien mit der Blockgraphik von Jürgen Degenhardt		10
4 Drives und trotzdem Doppelkopflaufwerke von Kurt Trappschuh		13
Hochauflösende Softgraphic von Jürgen Degenhardt	Teil 3	15
Simulation des HELP-Befehls von Bernd Niedermeier		17
Hochauflösende Softgraphic von Jürgen Degenhardt	Teil 4	19
Im String stehende BASIC-Befehle abarbeiten von Bernie Niedermeier		23
Bessere BASIC-Listings vom Video-Genie aus MICRO-EXTRA		25
Mitglieder Adressliste		27

Termine für Clubtreffen

Mittwoch 23.11.83 19.00 Uhr
Mittwoch 28.12.83 19.00 Uhr
Mittwoch 25.01.84 19.00 Uhr

Alle Treffen finden statt in der

**Gaststätte Kriegersiedlung
Albert-Roßhaupterstr. 61
8000 München 2**

Achtung! Neue Adresse: Postfach 1140 8011 Kirchseeon
Bitte verwenden Sie künftig nur noch diese neue Anschrift.

DISKETTEN

Z.Z. kann ich anbieten

MULTILIFE - Disketten
mit Verstärkungsringen

DM 5.50 Stk.
+ Versandkosten

Verstärkungsringe einzeln
stabilere Ausführung als oben

DM -.50 "

Ein Werkzeug zum nachträglichen Anbringen der Ringe ist im Club
ausleihbar.

DISKLOCHER

Im Club ist ein Werkzeug zum nachträglichen Lochen von Disket-
ten (zwecks beidseitiger Benutzung), ausleihbar.

ZU VERKAUFEN

TRS-80 Mod.1 Level 2 mit 10er-Tastatur und Kleinschreibung
Expansion-Interface mit 48K und 1 Disklaufwerk.
Die Anlage ist neuwertig. VB DM 3500.--

Alfred Brühbach, Haydnstr. 5 3501 Fuldabrück Tel. 0561/41929

SUCHE

Bei wem kann ich die Bücher

"BASIC faster & better & other Mysteries" und
"Machine Language Disk I/O & other Mysteries"

ausleihen?

Alfred Brühbach, Haydnstr. 5 3501 Fuldabrück Tel. 0561/41929

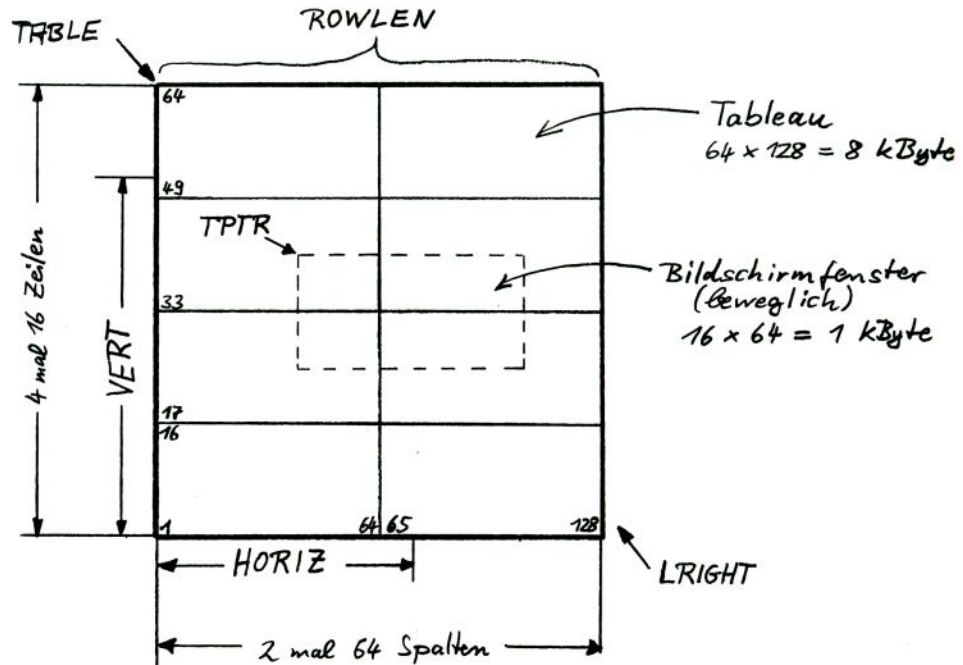
ZU VERKAUFEN:

Tandy Expansion interface mit 32K RAM.

Manfred Gebert Gautingerstr. 9 8031 Geisenbrunn

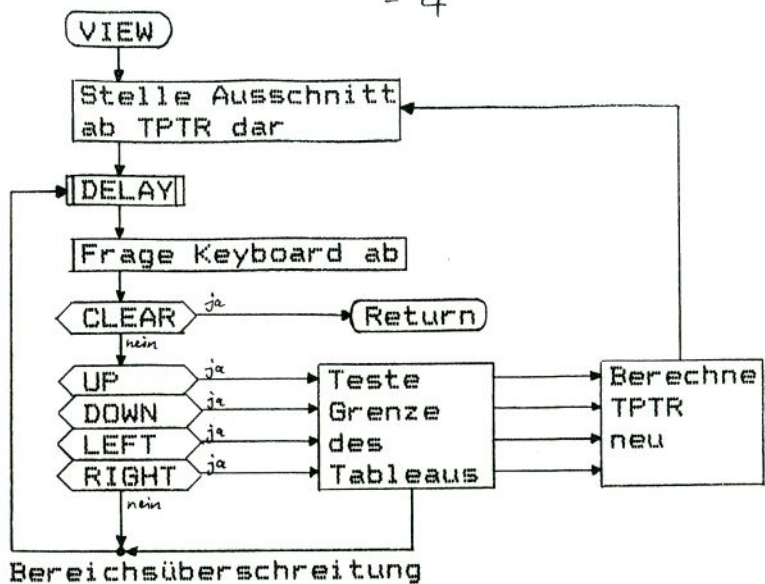
Hochauflösende Softgraphic für den TRS80 Model I (Teil 2)

Im Teil 1 stellte ich die Routinen CLEAR, die das Graphikableau löscht, und HDCOPY vor, die das Tableau auf dem Drucker ausgibt. Bevor wir überhaupt etwas ins Tableau hineinschreiben, möchte ich zuerst noch VIEW vorstellen, die Routine zum Darstellen des Tableaus auf dem Bildschirm. Wie schon gesagt, ist es 8 mal so groß, wie der Bildschirm. Es wird deshalb jeweils nur ein Ausschnitt betrachtet. Mit Hilfe der Pfeiltasten kann man dieses Fenster über das ganze Tableau fahren.



Damit beim Hin- und Herschieben der Bildschirm nicht über die Grenzen des Tableaus fährt, wird mit den beiden Zählern VERT und HORIZ der erlaubte RAM-Bereich abgesteckt und getestet (Zeilen 250 bis 460). Zusätzlich zu diesen beiden Zählern dient TPTR als Zeiger auf die linke obere Ecke des momentan dargestellten Bildschirmbereichs. Das folgende grobe Flußdiagramm und die Kommentare zum Listing dürften "die letzten Klarheiten beseitigen".

Hier ein Wort zu den Kommentaren in Englisch. Ich habe sie nicht in Deutsch geschrieben, weil es in Englisch viel kürzer und prägnanter geht. Um eine gewisse Portion (amerikanisches) Englisch kommt man ja ohnehin nicht herum beim "Computern", oder?



Nun zu den Routinen, mit denen man ein Pixel setzt (SET), löscht (RESET) oder testet (POINT). Alle drei Aufgaben sind sich sehr ähnlich und es liegt daher nahe, die drei Routinen zu einer zusammenzufassen. Nur in dem Moment, wo auf das berechnete Bit eingewirkt wird (Zeilen 1720 und 1730), muß ein entsprechend modifizierter Opcode ausgeführt werden. Die Ähnlichkeit und Kompatibilität der Z80-Befehle RES, SET und BIT kommt einem hierbei sehr entgegen.

Die Routine besitzt drei verschiedene Einsprungpunkte (Zeile 1550, 1570 und 1590) entsprechend diesen drei verschiedenen Opcodes, welche zunächst gespeichert und später mit der Information über das anzuprechende Pixelbit versehen werden (Zeilen 1670 bis 1700).

Der zwischen die Einsprungpunkte eingeschobene "Dummy-Opode" (DEFB 1) bewirkt, daß die CPU je nach Einsprungpunkt den folgenden Code "überliest" (01 XX XX bedeutet LD BC,XXXX !). Ein Trick übrigens, mit dem wohl kaum ein Disassembler fertig werden dürfte.

Die Routine ETKBM dient dazu, aus den von BASIC in den Variablen X% und Y% zur Verfügung gestellten Koordinatenwerten das entsprechende Byte und eine Maske hierfür zu berechnen.

Zunächst testet sie auf Bereichsüberschreitung (Rückmeldung -2 über Routine OVRFLW) und berechnet dann das anzusprechende Byte. Wenn die Routine EVBYTE in den Reigen der von BASIC aufrufbaren Routinen aufgenommen wird, kann sie uns später wichtige Informationen liefern. Die Routine ETKBM wird später einmal noch für STRMOV gebraucht.

ETKBM liefert auch noch in den Registern C und D eine Maske, mit Hilfe derer der Opcode in der RESET/SET/POINT-Routine verändert werden kann.

Wie das im Einzelnen gemacht wird, ist gar nicht einfach zu erklären. Ich möchte dies dem interessierten Leser überlassen.

Im Unterschied zur BASIC-Bildschirmgraphik habe ich den Punkt (x=0,y=0) in die linke untere Ecke des Tableaus gelegt; ich finde das sinnfälliger.

POINT meldet übrigens genau wie in BASIC eine -1, falls das Pixel gesetzt war.

Im Teil 3 wird es um die Routinen FUNCIN und EVAL gehen - zwei Routinen, die das Plotten von beliebigen Funktionen sehr erleichtern.

Tschüss bis dann

Euer Jürgen Degenhardt

0080		00100	ROWLEN	EQU	128	; SEE ALSO "TEIL 1"
3C00		00110	SCREEN	EQU	3C00H	; FIRST BYTE OF VIDEO RAM
0000	DD00	00120	TPTR	DEFW	TABLE	; POINTER TO UPPER LEFT COR-
		00130				;NER OF CURRENT VISIBLE
		00140				;PART OF THE TABLEAU
0002	31	00150	VERT	DEFB	49	; 0 < VERT < 50
		00160				; VERT POINTS TO THAT ROW OF
		00170				; THE TABLEAU WHICH IS CUR-
		00180				; RENTLY TO BE SEEN ON THE
		00190				; BOTTOM OF THE SCREEN
0003	01	00200	HORIZ	DEFB	1	; 0 < HORIZ < 66
		00210				; HORIZ POINTS TO THAT CO-
		00220				; LUMN OF THE TABLEAU WHICH
		00230				; IS CURRENTLY TO BE SEEN
		00240				; ON THE LEFT OF THE SCREEN
0004	1180FF	00250	UP	LD	DE, -ROWLEN	; PREPARE FOR SHIFTING ONE
0007	3A0200	00260		LD	A, (VERT)	; ROW UP
000A	3C	00270		INC	A	; BUT FIRST INCREMENT AND
000B	FE32	00280		CP	50	; TEST VERT
000D	1807	00290		JR	CVERT	
000F	118000	00300	DOWN	LD	DE, ROWLEN	; PREPARE FOR SHIFTING ONE
0012	3A0200	00310		LD	A, (VERT)	; ROW DOWN, BUT FIRST
0015	3D	00320		DEC	A	; DECREMENT AND TEST VERT
0016	2B33	00330	CVERT	JR	Z, DELAY	; NO SHIFT IF OUT OF BOARDER
0018	320200	00340		LD	(VERT), A	; SAVE NEW ROW VALUE
001B	1817	00350		JR	SHIFTG	; GO, SHIFT AND DISPLAY
		00360				
001D	110100	00370	RIGHT	LD	DE, 1	; PREPARE FOR SHIFTING ONE
0020	3A0300	00380		LD	A, (HORIZ)	; COLUMN RIGHT
0023	3C	00390		INC	A	; BUT FIRST INCREMENT AND
0024	FE42	00400		CP	66	; TEST HORIZ
0026	1807	00410		JR	CHORIZ	
0028	11FFFF	00420	LEFT	LD	DE, -1	; PREPARE FOR SHIFTING ONE
002B	3A0300	00430		LD	A, (HORIZ)	; COLUMN LEFT, BUT FIRST
002E	3D	00440		DEC	A	; DECREMENT AND TEST HORIZ
002F	281A	00450	CHORIZ	JR	Z, DELAY	; NO SHIFT IF OUT OF BOARDER
0031	320300	00460		LD	(HORIZ), A	; SAVE NEW COLUMN VALUE
		00470				
0034	19	00480	SHIFTG	ADD	HL, DE	; HL=NEW UPPER LEFT CORNER
0035	220000	00490		LD	(TPTR), HL	; OF VISIBLE PART OF TABLEAU
		00500				
0038	2A0000	00510	VIEW	LD	HL, (TPTR)	; ENTRY POINT OF VIEW !!!!
003B	11003C	00520		LD	DE, SCREEN	
003E	014000	00530		LD	BC, 64	
0041	EDB0	00540	MOVIT	LDIR		; MOVE 16 ROWS - 64 BYTES
0043	014000	00550		LD	BC, 64	; EACH - TO THE SCREEN
0046	09	00560		ADD	HL, BC	
0047	7A	00570		LD	A, D	; D=40H IS USED AS AN END-
0048	B9	00580		CP	C	; MARKER
0049	20F6	00590		JR	NZ, MOVIT	
		00600				
004B	CD6000	00610	DELAY	CALL	60H	; ROM DELAY ROUTINE
004E	3A403B	00620	SCAN	LD	A, (3840H)	; SCAN KEYBOARD ROW
0051	0F	00630		RRCA		; THE CLEAR KEY (A=2) IS
0052	0F	00640		RRCA		; USED AS A RETURN KEY
0053	DB	00650		RET	C	
0054	2A0000	00660		LD	HL, (TPTR)	; PREPARE FOR SHIFTING
0057	010040	00670		LD	BC, 4000H	; BC=DELAY COUNTER
005A	0F	00680		RRCA		
005B	0F	00690		RRCA		
005C	38A6	00700		JR	C, UP	; JUMP IF UP ARROW
005E	0F	00710		RRCA		
005F	38AE	00720		JR	C, DOWN	; DOWN ARROW
0061	CB3B	00730		SRL	B	
0063	0F	00740		RRCA		
0064	38C2	00750		JR	C, LEFT	; LEFT ARROW
0066	0F	00760		RRCA		
0067	38B4	00770		JR	C, RIGHT	; RIGHT ARROW
0069	18E0	00780		JR	DELAY	; DEBOUNCE AND SCAN AGAIN
		00790				
		00800				

```

00810
00820
00830
00840
00850
00860
00870
00880
260D      00890  VARPTR  EQU    260DH
0000 58    00900  VARX    DEFM   'X%'
0002 00    00910          NOP
0003 59    00920  VARY    DEFM   'Y%'
0005 00    00930          NOP
00940
0006 CD1400 00950  EVBYTE  CALL   ETKBM      ;evaluate byte and
0009 C39A0A 00960    JP      OA9AH    ;return value to BASIC
00970
000C F1     00980  OVRFLW  POP    AF          ;clear stack
000D F1     00990  OVERFL  POP    AF          ;don't return to caller
000E 21FEFF 01000    LD      HL,-2     ;OVERFLOW ERROR CODE
0011 C39A0A 01010    JP      OA9AH    ;return to BASIC
01020
0014 210300 01030  ETKBM   LD      HL,VARY    ;points to BASIC's
0017 CD0D26 01040    CALL   VARPTR     ;Y% - variable
001A 13     01050    INC    DE          ;test MSB
001B 1A     01060    LD      A,(DE)    ;of Y%
001C B7     01070    OR     A           ;must be zero!
001D 20EE   01080    JR     NZ,OVRFLW
001F 47     01090    LD      B,A       ;B=0, used later
0020 1B     01100    DEC    DE          ;points to LSB
0021 1A     01110    LD      A,(DE)    ;get value of Y% LSB
0022 FEC0   01120    CP     192        ;must be less than 192
0024 30E7   01130    JR     NC,OVRFLW
0026 04     01140  DIVIDE  INC     B          ;B = row number =
0027 D603   01150    SUB    3           ;INT(Y%/3)+1
0029 30FB   01160    JR     NC,DIVIDE
002B C604   01170    ADD   A,4         ;A = remainder + 1 =
002D F5     01180    PUSH  AF          ;row within byte
01190
002E 118000 01200    LD      DE,ROWLEN ;adjust HL to the
0031 218220 01210    LD      HL,LRIGHT ;row given by the
0034 ED52   01220  YSHIFT SBC    HL,DE       ;value in B
0036 10FC   01230    DJNZ  YSHIFT
0038 E5     01240    PUSH  HL
01250
0039 210000 01260    LD      HL,VARX   ;points to BASIC's
003C CD0D26 01270    CALL   VARPTR     ;X% - variable
003F E1     01280    POP    HL
0040 1A     01290    LD      A,(DE)    ;get LSB of X%
0041 4F     01300    LD      C,A       ;into C
0042 13     01310    INC    DE          ;test MSB
0043 1A     01320    LD      A,(DE)    ;of X%
0044 B7     01330    OR     A           ;must be zero!
0045 20C5   01340    JR     NZ,OVRFLW
0047 57     01350    LD      D,A       ;D=0, used later
0048 47     01360    LD      B,A       ;BC = X% value
0049 CB39   01370    SRL   C           ;halve the X% value and
004B 3002   01380    JR     NC,ODD     ;set D=B if pixel is in
004D 1608   01390    LD      D,B       ;first column within byte
004F 09     01400  ODD    ADD   HL,BC      ;HL points to desired byte
01410
0050 C1     01420    POP    BC         ;B = row within byte
0051 0E40   01430    LD      C,64      ;this loop builds a mask
0053 CB39   01440  MASKIT SRL   C           ;to modify the instruction
0055 10FC   01450    DJNZ  MASKIT     ;for manipulating
0057 CB99   01460    RES   3,C         ;the right pixel
0059 C9     01470    RET
01480
01490
01500
01510

```

- 4 -

```

01520
01530
01540
005A 3E86 01550 RESET LD A,86H ;'RES B,(HL)
005C 01 01560 DEFB 1 ;dummy opcode
005D 3EC6 01570 SET LD A,0C6H ;'SET B,(HL)
005F 01 01580 DEFB 1 ;dummy opcode
0060 3E46 01590 POINT LD A,46H ;'BIT B,(HL)
0062 327800 01600 LD (INSTR),A ;save instruction
0065 CD1400 01610 CALL ETKBM ;evaluate & test coor-
;dinates, evaluate byte
01620
01630
0068 CB7E 01640 BIT 7,(HL) ;and mask for pixel
006A 2002 01650 JR NZ,MODIFY ;test if byte is a
;graphics character
006C 3680 01660 LD (HL),80H ;shift to graphic
006E 3A7800 01670 MODIFY LD A,(INSTR) ;load instruction
0071 B1 01680 OR C ;and combine it with
0072 B2 01690 OR D ;the masks
0073 327800 01700 LD (INSTR),A ;modify the routine
0076 AF 01710 XOR A ;set zero flag
0077 CB 01720 DEFB 0CBH ;opcode for SET, RESET
0078 00 01730 INSTR NOP ;or POINT
0079 60 01740 LD H,B ;zeros HL
007A 68 01750 LD L,B
007B 2801 01760 JR Z,EXIT ;if the pixel is on,
007D 2B 01770 DEC HL ;POINT returns a -1
007E C39A0A 01780 EXIT JP 0A9AH ;return to BASIC
01790
0081 01800 TABLE EQU $ ;TABLEAU BEGINS HERE
2082 01810 LRIGHT EQU TABLE+2001H ;LOWER RIGHT CORNER
0080 01820 ROWLEN EQU 128
402D 01830 END 402DH
00000 TOTAL ERRORS
22600 TEXT AREA BYTES LEFT

```

BØ-MICRO UMLAUF:

Immer wieder kommen Klagen, daß der Zeitschriftenumlauf nicht funktioniert. Ich bitte deshalb nochmals alle Teilnehmer, die Hefte nicht allzu lange zu behalten, sondern möglichst bald an den Nächsten weiterzuschicken.

Wenn die Hefte zum Schluß wieder bei mir sind, könnt Ihr sie gerne auch über einen längeren Zeitraum ausleihen.

CLUB-VERBUND:

Es ist möglich, daß in der nächsten Zeit Kontakte zur TRS-BØ User-Group Bremerhaven entstehen. Dabei geht es hauptsächlich um den Austausch von Beiträgen für die Clubzeitung.

Wer Einwände gegen die Veröffentlichung seiner Beiträge in der Bremerhavener Club-Info hat, muß mir dies mitteilen.

Eine derartige Verbindung besteht ja bereits zur AMMS eV. Wie wichtig und fruchtbar diese Verbindung bisher war, zeigen die letzten Ausgaben der Clubzeitung.

Wolfgang Reichelsdorfer
Marienbaderstr.21
8858 Neuburg/Donau
8.9.1983

Bericht KOMTEK1

Als ich vor kurzem bei einem Bekannten den KOMTEK1 genannten Computer sah, beschloß ich, diesen unter die Lupe zu nehmen. Das Gerät soll ja TRS80-kompatibel sein. Ich ließ mir einen Rechner einige Tage aus und spielte ein wenig damit herum.

Erster Eindruck: Beiges Gehäuse, graue ASCII-Tastatur, hellblaue Funktionstasten. Das ganze ist recht gewichtig. Warum? Aha. Die Bodenplatte ist Stahlblech, der Netztrafo gleich eingebaut! Eine Netzleitung und ein Kassettenkabel mit den üblichen Klinkernsteckern hängen am Gerät, ein Monitor und ein TV-Kabel (CINCH) liegen der Verpackung bei, ebenfalls ein Bedienungshandbuch und ein BASIC-Kurs, beides in Deutsch. Ein paar Fingerübungen auf den Tasten zeigen auch hier robuste Ausführung, fast haben die Entwickler ein wenig zu viel des Guten getan: Die Tasten gehen etwas schwer, fast wie bei einer mechanischen Schreibmaschine. Dafür gibt es zusätzlich eine Control-, eine unbeschriftete Funktions- und eine Shift-Lock-Taste. Die RESET-Funktion wird ebenfalls hier ausgeführt, liegt aber neben NEWLINE, BREAK und CLEAR. Recht gefährlich also.

Von der Rückseite betrachtet wirkt das Gerät noch interessanter: Jede Menge Anschlußmöglichkeiten! Vier Sensoreingänge (Klinke), TV-HF und Video (Cinch), Parallel-Printer, 50-Pin CPU-Expander, Kasette, Floppy (und sechs Schaltausgänge (Klinke).

Die Innereien: Mutterplatine mit Z80A, 2 MHz, 16 bis 64 KByte, HF-Modulator, ein 8255 PIO als Sensor-, Control- und Printerinterface, vier PROMS, Clock und Speicher. Weiterhin vorhanden, doch nicht belegt, Slots für Floppy-Controller, HIRES-Graphik und Farbzusatz!. Natürlich ist auch eine RS232/V24 Schnittstelle möglich.

Nun zur Weichware. Nach dem Einschalten und dem vorgeschriebenen Drücken der RESET-Taste meldet sich ein SCS-BASIC. Dieses umfaßt 12K und soll kompatibel zu Level II des TRS80 Modell und Video Genie sein. Darüber hinaus gibt es ein PROM mit 1.5K Hilfsroutinen. Diese sind mit SYSTEM 12464 aufrufbar und beinhalten eine Tastenentprellung (etwas langsam, Tasten prellen auch ohne nicht), Tastenwiederholung (sehr schnell) und Umlauttreiber. Die Graphikzeichen sind über Tastatur zugänglich und die Echtzeituhr läßt sich per POKE auch ohne Diskbasic programmieren. Im Speicher darüber befinden sich in altbekannter Weise noch DCBs und Video-RAM.

Der Bildschirm gliedert sich in 64 * 16 Zeichenplätze auf. Der Aufruf der Breitschrift ergibt lediglich größere Zeichenabstände (ähnlich Genie). Die Schrift ist gut lesbar, jedoch scheint mir ein normaler Fernseher mit 1024 Zeichen überfordert zu sein. Es empfiehlt sich ein Monitor. Leider konnte ich meinen TRS80 Monitor nicht anschließen, da ich keinen Adapter von DIN auf CINCH zur Hand hatte.

Sehr gespannt war ich auf die 'Kompatibilität' des KOMTEK1 mit meinen Programmen. So lud ich einige BASIC- und SYSTEM-Files von meinem CTR 80 in das Gerät und siehe da, alles lief einwandfrei. Die Druckerschnittstelle funktionierte über BASIC ebenfalls anstandslos. Der ITOH druckte alles so, wie ich es wollte.

Die Preise:Das Grundgerät mit 32K, Printer und Control-Interface kostet etwa 1160.- DM, für die Floppykarte werden 500.- DM fällig, HIRES und Farbplatine sind ab etwa November lieferbar.

Weitere Details können der beigefügten Produktinformation entnommen werden. Anfragen bitte an die dort angegebene Adresse richten.

Anmerkung: Da mir der KOMTEK1 aufgrund seiner Auslegung sehr gut gefällt (einzigster Schwachpunkt bisher ist der fehlende 10er-Block), werde ich demnächst einen kleinen Bericht über die Diskettenversion verfassen. Ein Gerät ist mir zu diesem Zwecke schon zugesagt.

Produktinformation Komtek I

- 3 -



Standartausrüstung :

- + Z - 80 CPU, 2MHz
- + 16 K RAM nder Grundausrüstung
- + 12 K Basic Level II im ROM
- + echte Schreibmasch.Tastatur ASCII
- + 64 x 16 Zeichen
- + Groß-und Kleinschreibung
- + echte Unterlängen
- + Blockgraphik
- + progr. Tongenerator
- + Echtzeituhr
- + getrennte Anschlüsse für Monitor und Fernsehgerät
- + Anschluß für alle handelsüblichen Recorder

- + voll Software - kompatibel mit TRS-80 (TANDY) und Video-Genie (EACA)
- + bei entsprechender Ausrüstung bis zu 4 Floppy-Laufwerke anschließbar
- + Pascal und Fortran in Kürze erhältlich

Zusatzausrüstung (zum Geräteeinbau, keine ext.Boxen, kein Kabelgewirr)

RS 232 c Interface
 High-Resolution-Graphik 256x192
 Printer Interface
 Double Density Adapter

Colour Interface
 Sensor-und Schaltfunktion
 Expander Board/Floppy-Controller
 Druckeraktivierung + Contr.Interface

Best.Nr	Bezeichnung	Endverk.Pr incl MWST.
501001/1	Grundgerät mit 16 K RAM	857.--
501002/1	Grundgerät mit 64 K RAM	1090.--
501001/2	Grundgerät mit 16 K RAM und Sensor/ Schaltfunktion	925.--
501002/2	wie 501001/2 jedoch mit 64 K RAM	1160.--
<hr/>		
501800	Floppy Controller	499.--
501806	Color Interface	Ab November 83 lieferbar
501807	RS-232-c Interface	Ab November 83 lieferbar
501809	Double Density Adapter	299.--
501864	64 K RAM /Einbau im Werk (BRD)	250.--
501823	High Resolution Graphik	ist in Vorbereitung
501827	Drucker Kabel	100.--
502800	Sensor-und Schaltfunkt.Interface	100.--
502801	Schaltbox für Schaltfunktion	125.--
502802	Druckeraktivierung+Contr.Interf. 502800	172.--

Es gelten unsere Verkaufs-und Lieferbedingungen
 Lieferung erfolgt ab Neuburg/Donau
 Berechnung der Verpackung und Versandgeb.erfolgt zum Selbstkostenpr.

Zeichnen von Linien mit der Blockgraphik

(nach einer Idee aus ELCOMP 1/83, S. 124)

Längst nicht alle Homecomputer, die einen Zeichensatz für eine Bildschirmgraphik haben, unterstützen das Arbeiten damit durch einen ausreichenden BASIC-Befehlssatz. Für ein effektives Arbeiten mit einer hochauflösenden Graphik sind Befehle wie DRAW, MOVE, etc. unumgänglich und sind in diesen Systemen meistens auch implementiert. Bei den billigeren Systemen mit sogenannter Blockgraphik darf man dagegen schon froh sein, wenn es Befehle zum Setzen (SET) und Löschen (RESET) eines Pixels (kleinstes anzusprechendes Graphikfeld) gibt. Andere Homecomputer wie z.B. der CBM haben zwar die Möglichkeit zu einer Blockgraphik (neben einem umfangreichen Sortiment anderer Graphikzeichen), unterstützen diese aber nicht im geringsten. Man ist also gezwungen, sich einen Satz von Unterprogrammen zuzulegen, die in der Lage sind, diese "höheren Graphikbefehle" durchzuführen.

Der wohl am häufigsten benötigte Befehl ist DRAW, d.h. das Verbinden zweier Punkte durch eine Linie. Natürlich wird diese Linie mehr oder weniger gezackt aussehen, je nachdem wie grob das Raster der Pixel ist.

Die beiden zu verbindenden Punkte dürfen jede beliebige Lage zueinander haben. Diese Forderung führt zu 9 verschiedenen Fällen (Bild 1), je nach Vorzeichen und Größe der Differenz beider Koordinaten. Das zu entwickelnde Programm muß alle diese Fälle verarbeiten können.

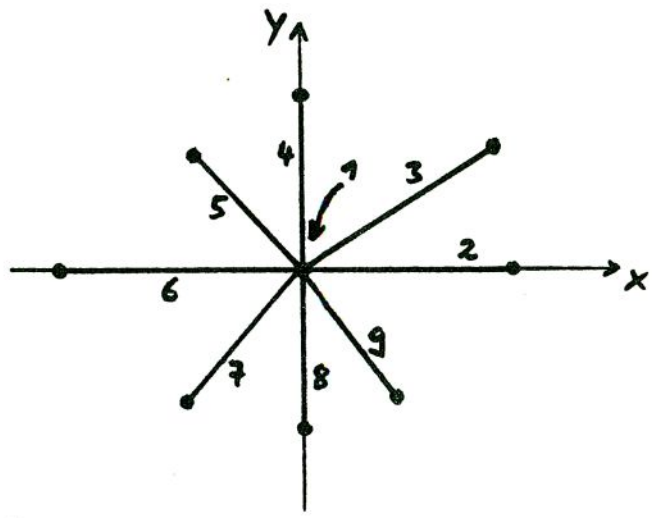


BILD 1

Fall	Koordinaten		Schrittweite	
	X	Y	X	Y
1	$X_1=X_2$	$Y_1=Y_2$	0	0
2	$X_1<X_2$	$Y_1=Y_2$	+1	0
3	$X_1<X_2$	$Y_1<Y_2$	+DX	+DY
4	$X_1=X_2$	$Y_1<Y_2$	0	+1
5	$X_1>X_2$	$Y_1<Y_2$	-DX	+DY
6	$X_1>X_2$	$Y_1=Y_2$	-1	0
7	$X_1>X_2$	$Y_1>Y_2$	-DX	-DY
8	$X_1=X_2$	$Y_1>Y_2$	0	-1
9	$X_1<X_2$	$Y_1>Y_2$	+DX	-DY

Was passiert eigentlich, wenn einer oder beide Punkte gar nicht im Bereich der Koordinaten des Bildschirms liegen? Beim TRS80 führt ja der Versuch den Befehl SET(-1,130) auszuführen zu einer Fehlermeldung und damit zum Abbruch des Programms. Das ist oft sehr unerwünscht und man muß per Programm dafür sorgen, daß unerlaubte Koordinaten keinen Schaden anrichten können, z.B. durch vier Abfragen, etwa

```
IF X<0 OR X>127 OR Y<0 OR Y>47 THEN ... .
```

Eleganter ist es in jedem Fall, den ON ERROR GOTO ... - Befehl auszunutzen, denn schließlich enthält die Routine des SET-Befehls selbst schon diese vier Abfragen in irgendeiner Form. Diese in BASIC zu wiederholen wäre also überflüssig. Übrigens enthält SET auch die INTeGer-Funktion.

Wir müssen uns nun noch überlegen, wie wir die Koordinaten derjenigen Pixel ermitteln, die für die Verbindungslinie zwischen den beiden Endpunkten benötigt werden. Die Fälle 2, 4, 6 und 8 sowie 1 sind trivial, da hierbei ausgehend vom Startpunkt (X1,Y1) jeweils nur eine Koordinate in- bzw. decremementiert werden muß bis der Endpunkt (X2,Y2) erreicht ist. Wir suchen jedoch einen Algorithmus, der die komplizierteren Fälle 3, 5, 7 und 9 erfüllt und dann die trivialeren mit erfüllt. Vorsicht, der Artikel in der oben erwähnten ELCOMP tut dieses nicht! Unsere Verbindungslinie soll durch eine ununterbrochene Reihe von Pixeln dargestellt werden, die sich mindestens "übereck" berühren (Bild 2a). In einer zweiten Version kann man auch fordern, daß sich die Pixel immer mit den Kanten berühren (Bild 2b).

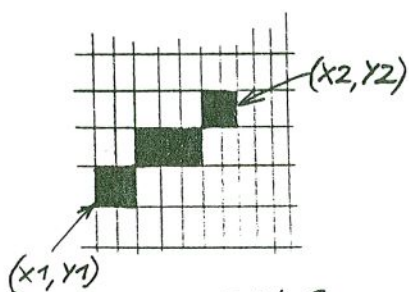


Bild 2a

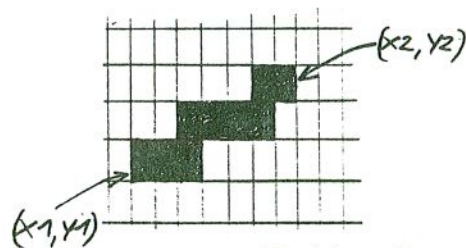


Bild 2b

Im ersten Fall sieht man wohl leicht ein, daß die Anzahl der benötigten Pixel einfach aus der Differenz der X-Koordinaten gewonnen werden kann. Aber Vorsicht! Es kann ja sein, daß diese aus dem Bild abgeleitete Regel nicht für die übrigen 8 Fälle zutrifft. Allgemein gilt nämlich: Die Anzahl der benötigten Pixel ist gleich dem größten ganzzahligen Absolutwert der Koordinatendifferenzen.

Die Schrittweite in der jeweils anderen Koordinatenrichtung (im Bild 2a also die Y-Richtung) erhält man dann (vorzeichenrichtig) ganz einfach dadurch, daß man die Differenz in dieser Richtung durch die Anzahl der Pixel teilt. Listing 1 enthält den Algorithmus für den Bild 2a entsprechenden Fall. Es sei dem Leser überlassen, sich zu überlegen, wie man die Anzahl der benötigten Pixel für den Fall nach Bild 2b erhält. Listing 2 enthält den dafür entwickelten Algorithmus.

Listing 1

=====

```
10 IF ABS(X2-X1) > ABS(Y2-Y1)
   THEN D=(Y2-Y1)/ABS(X2-X1):
       FOR X1 = X1 TO X2 STEP SGN(X2-X1):
           SET(X1,Y1):Y1=Y1+D:
       NEXT
   ELSE D=(X2-X1)/ABS(Y2-Y1):
       FOR Y1 = Y1 TO Y2 STEP SGN(Y2-Y1):
           SET(X1,Y1):X1=X1+D:
       NEXT

20 X1=X2: Y1=Y2: SET(X1,Y2): RETURN
```

Listing 2:

=====

```
10 ND=ABS(X2-X1) + ABS(Y2-Y1):
   IF ND THEN DX=(X2-X1)/ND: DY=(Y2-Y1)/ND

20 FOR I = 1 TO ND:
   SET(X1,Y1): X1=X1+DX: Y1=Y1+DY:
   NEXT:
   X1=X2: Y1=Y2: SET(X1,Y1): RETURN
```

Es muß noch bemerkt werden, daß alle Variable Fließkommavariablen sein müssen, und daß X1 und Y1 nach dem Programmlauf die Werte von X2 und Y2 haben, was das Programmieren von Polygonzügen erleichtert.

Was diesem Programm noch fehlt, ist neben der Schnelligkeit noch eine Art Schalterfunktion, die aus DRAW ein UNDRAW, d.h. "lösche Linie", macht. In meinem Artikel über die "Hochauflösende Softgraphik" wird der SET- bzw. RESET-Vorgang über einen USR-Aufruf getätigt. Die Übergabevariable dieses Aufrufs kann also diese Schalterfunktion gut übernehmen.

Jürgen Degenhardt



Kurt Trappschuh
Reineckestraße 6
8036 Herrsching

27.08.83 ☎ 08152/2512

Liebe Clubfreunde,

da ich zufällig in den Besitz einer 4. Drive kam, in meinem System aber bereits ein Doppelkopflaufwerk Dienst tut, war ich gezwungen, mir etwas einfallen zu lassen.

Normalerweise kann man an einen TRS-80 entweder 4 einseitige oder bis zu 3 doppelseitige Laufwerke anschließen. Sobald also auch nur 1 Doppelkopflaufwerk im System ist, können nur noch 3 Drives angeschlossen werden. Das kommt daher, daß zur Kopfumschaltung von Doppelkopflaufwerken die Leitung DS 3 (Drive Select 3) benutzt wird, die ansonsten die 4. Drive anspricht. Wird trotzdem ein 4. Laufwerk angeschlossen, so fühlt sich dieses bei jedem Zugriffsversuch auf die Rückseite eines Doppelkopflaufwerkes ebenfalls angesprochen, was natürlich zum großen "Datenschrott" führt.

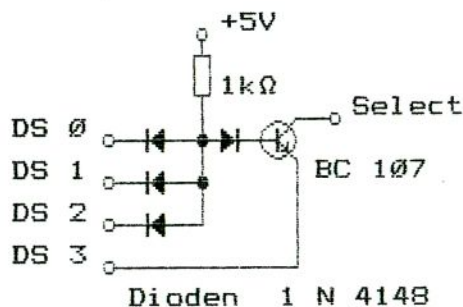
Nach einigen Überlegungen und Versuchen habe ich nun folgendes herausgefunden:

- ein Zugriff auf die Rückseite eines Doppelkopflaufwerkes steuert auch die Leitung DS 3 durch (Logisch 0 !), dadurch fühlt sich Drive 3 ebenfalls angesprochen, was zu Fehlermeldungen führt.
- wird Drive 3 über DS 3 angesprochen, so werden zwar die Köpfe der Doppelkopflaufwerke auf die Rückseite eingestellt, da diese Drives aber zu diesem Zeitpunkt nicht angesprochen sind (DS 0-2), geht diese Aktion in Ordnung.
- Drive 3 kann nur ein einseitiges Laufwerk sein, da es mit der Kopfumschaltleitung (DS 3) ja bereits angesprochen wird, Drive 0-2 können doppelseitig sein.
- Drive 3 darf nur dann angesteuert werden, wenn die Leitung DS 3 allein Logisch 0 ist, ist zusätzlich eine der Leitungen DS 0-2 ebenfalls Logisch 0, so ist ja nicht Drive 3 sondern die Rückseite einer der Drives 0-2 gemeint.

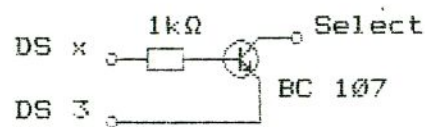
Um den letzten Punkt erfüllen zu können, ist ein kleiner Eingriff in Drive 3 erforderlich:

- die meisten Laufwerke sind nicht von Haus aus für den Betrieb als Drive 3 vorbereitet. Die Leitung DS 3 ist normalerweise mit Head Select (Kopfschaltung) (auch bei einseitigen Laufwerken !) verbunden. Um Probleme zu vermeiden empfehle ich, diese Leitung kurz hinter dem Stecker durchzutrennen.
- befinden sich mehrere Doppelkopflaufwerke im System, so ist die Schaltung nach Bild 1 "freifliegend" in die Drive einzulöten. Außer +5V und DS 3 sind alle Leitungen an dem Punkt anzutreffen, wo durch Jumper, Schalter oder Drahtbrücken die Drive normalerweise eingestellt wird. +5V kann an irgendeinem in der Nähe befindlichen 14poligen IC am Pin 14 abgenommen werden. DS 3 muß unter Umständen kurz hinter dem Stecker angezapft werden. Die Diode an der Basis des Transistors ist notwendig, da sonst die Basis nicht tief genug "heruntergezogen" werden kann, um den Transistor zu sperren.
- befindet sich nur 1 Doppelkopflaufwerk im System, so kann auch die vereinfachte Schaltung 2 eingebaut werden. DS x ist dann diejenige der Leitungen DS 0-2, an der das Doppelkopflaufwerk hängt.

Schaltung 1



Schaltung 2



Die hier vorgestellte Schaltung arbeitet bei mir schon seit einigen Monaten zu meiner besten Zufriedenheit. Änderungen am Betriebssystem sind nicht erforderlich.

fröhliches Löten !

Kurt Krausschuh

```
10 DEFUSR=&H xxxx
20 INPUT "X-Koordinate";X%
30 INPUT "Y-Koordinate";Y%
40 Z=USR(code)
```

Dabei ist 'xxxx' natürlich abhängig von der Startadresse des Maschinenprogramm-Verteilers und 'code' muß die Zahl sein, die im Verteiler zur Routine SET führt.

Nach Z=USR(view) konnte man sich dann anschauen, ob auch wirklich das richtige Pixel gesetzt wurde (Pfeil-Tasten nicht vergessen), und mit der CLEAR-Taste sprang man zurück nach BASIC. 'view' mußte dabei auf die VIEW-Routine zeigen.

In diesem Teil möchte ich Routinen vorstellen, die das Plotten von Funktionen sehr erleichtern.

Fangen wir mit EVAL an. Diese Routine ruft dreimal die Zuweisungsroutine im BASIC-ROM auf, d.h. sie führt dreimal einen Code aus, der genauso gut auch in einem BASIC-Programm stehen könnte. Diese Methode ist letztlich jedoch schneller, kürzer und entlastet den Programmierer. Dieser muß sich nun nicht mehr um X% und Y%, sondern um X und Y kümmern. Die richtige Rundung nimmt ihm das Programm ab.

Der erste Aufruf der LET-Routine in EVAL ist jedoch etwas ganz besonderes. Hier wird ein gewandelter BASIC-Code ausgeführt, der in einem besonderen Speicher (FBUFF) steht bzw. mit Hilfe der Routine FUNCIN dorthin gebracht wurde. FUNCIN wiederum holte sich den Code dafür aus dem String X\$, der im gerade laufenden BASIC-Programm z.B. per LINEINPUT eingelesen wurde. Der Sinn dieser Zeremonie? Nun - lesen wir die Sache einmal in umgekehrter Reihenfolge:

Zuerst fragt uns das BASIC-Programm "Welche Funktion soll geplottet werden?" und wir geben "Y=30*SIN(X)+96" ein. Diese Funktionsgleichung wird in X\$ gespeichert, FUNCIN wird einmal aufgerufen und dann läuft eine FOR...NEXT-Schleife ab mit X=0 bis 255 (vielleicht mit der Schrittweite 0.1). In der Schleife wird jedesmal EVAL und anschließend SET oder abkürzend FSLOOP aufgerufen. Anschließend präsentiert uns das Programm ein Menue, in dem wir die Wahl haben zwischen VIEW, HDCOPY, CLEAR und der Eingabe einer neuen Funktion. Das Programm braucht dabei nicht angehalten zu werden, etwa um die neue Funktion als DEFFNY=... zu programmieren!

Natürlich muß, bis alles läuft, der Assemblercode aus den drei Teilen meines Artikels vorher zusammengestellt, mit dem richtigen ORG-Statement assembliert und geladen werden. Dabei ist zu bedenken, daß für den Speicher FBUFF 256 und für das Tableau 8 kByte zu reservieren sind. Das in diesem Teil verwendete 'ORG OF000H' ist also zu groß! Ein paar Zeilen sind auch zu streichen, z.B. die Zeilen 510 und 840 bis 880, da sie doppelt vorkommen. Aber darauf macht uns ja der Assembler ohnehin aufmerksam.

Nun fehlt nur noch STRMOV, doch für heute

Tschüss bis zum Teil 4

Euer Jürgen Degenhardt

F000		00300	ORG	0F000H	
F000	2122F0	00310	FUNCIN	LD	HL,XSTRNG ;BASIC's X# contains
F003	CD0D26	00320		CALL	VARPTR ;function y=f(x)
F006	E7	00330		RST	20H ;make sure it's a string
F007	C29719	00340		JP	NZ,SYNERR ;else SYNTAX ERROR
F00A	EB	00350		EX	DE,HL ;HL points to X#
F00B	1125F0	00360		LD	DE,FBUFF ;DE points to buffer
F00E	D5	00370		PUSH	DE
F00F	D5	00380		PUSH	DE
F010	CDC829	00390		CALL	29C8H ;move X# to FBUFF
F013	7D	00400		LD	A,L ;A=L=0
F014	12	00410		LD	(DE),A ;terminate function string
F015	E1	00420		POP	HL ;HL points to FBUFF
F016	CDC01B	00430		CALL	1BC0H ;tokenisation of X#
F019	D1	00440		POP	DE ;DE points to FBUFF
F01A	0D	00450		DEC	C ;decrement code length
F01B	0D	00460		DEC	C ;twice
F01C	23	00470		INC	HL ;first byte in workarea
F01D	EDB0	00480		LDIR	;move tokenised code to
F01F	C39A0A	00490		JP	0A9AH ;FBUFF and return
		00500			
260D		00510	VARPTR	EQU	260DH ;ROM VARPTR routine
1997		00520	SYNERR	EQU	1997H ;SYNTAX ERROR routine
F022	58	00530	XSTRNG	DEFM	'X#'
F024	00	00540		NOP	
0100		00550	FBUFF	DEFS	256 ;256 byte buffer for code
		00560			
F125	2125F0	00570	EVAL	LD	HL,FBUFF ;points to tokenised code
F128	CD211F	00580		CALL	LETSPG ;evaluate function
F12B	2137F1	00590	ROUND	LD	HL,ROUNDX ;points to x%=x+.5
F12E	CD211F	00600		CALL	LETSPG ;make an integer argument
F131	213FF1	00610		LD	HL,ROUNDY ;points to y%=y+.5
F134	C3211F	00620		JP	LETSPG ;integer value and return
		00630			
F137	58	00640	ROUNDX	DEFM	'X%'
F139	D5	00650		DEFB	0D5H ; = token
F13A	58	00660		DEFM	'X'
F13B	CD	00670		DEFB	0CDH ; + token
F13C	2E	00680		DEFM	'.5'
F13E	00	00690		NOP	
F13F	59	00700	ROUNDY	DEFM	'Y%'
F141	D5	00710		DEFB	0D5H ; = token
F142	59	00720		DEFM	'Y'
F143	CD	00730		DEFB	0CDH ; + token
F144	2E	00740		DEFM	'.5'
F146	00	00750		NOP	
		00760			
F147	CD25F1	00770	FRLOOP	CALL	EVAL ;these are special
F14A	C325F1	00780		JP	RESET ;functions used in
F14D	CD25F1	00790	FSLOOP	CALL	EVAL ;FOR...NEXT loops
F150	C325F1	00800		JP	SET
F153	CD25F1	00810	FPLOOP	CALL	EVAL
F156	C325F1	00820		JP	POINT
		00830			
F125		00840	RESET	EQU	EVAL ;RESET, SET & POINT are
F125		00850	SET	EQU	EVAL ;subroutines used in
F125		00860	POINT	EQU	EVAL ;'Teil 2'! So delete
		00870			;these 3 lines when
		00880			;appending 'Teil 2'!
1F21		00890	LETSPG	EQU	1F21H ;ROM LET routine
402D		00900		END	402DH
00000	TOTAL	ERRORS			
27850	TEXT	AREA	BYTES	LEFT	

-17-

SIMULATION DES BEFEHLS HELP VERSION 1.0 VOM 13.12.1982

Bernd Niedermeier
Hirschbergweg 9
8011 Heimstetten
Tel.: (089) 903 57 31

Hallo Clubfreunde,
hier habe ich ein manchmal recht brauchbares Hilfsmittel, um irgendwelche Fehler in einem Programm aufzufinden. Wie oft hat man schon ein Programm geschrieben, das aus vielen Multiple Statement Lines besteht und dann auch noch Befehle wie CMD'O' und anderen Stringoperationen, mehrere komplizierte Formeln in einer Zeile usw. enthaelt. Wenn nun einmal in solch einer komplizierten Zeile ein Fehler auftritt, so steht man meist eine Weile ratlos dem Problem gegenueber und zerlegt eine Formel nach der anderen, um den Fehler zu finden. Hier kann nun eine wenig geholfen werden.

Laedt man das Programm vorher und setzt den Anfang des UP's auf FFC0H (DEFUSR=FFC0 fuer DOS bzw POKE 16526,192:POKE 16527,255), so gibt man nach Auftreten eines Fehlers Z=USR(0) ein. Es wird dann die Fehlerzeile gelistet und bei dem Befehl, wo der Fehler auftrat, ein Level2-Cursor ausgegeben. Das sieht nun so aus, als ob die Zeile jetzt diesen Cursor enthalten wuerde. Dem ist aber nicht so: bei LIST Zeilennr ist der urspruengliche Inhalt nicht veraendert worden.

Man kann sich nun ganz speziell an den Ausdruck heranmachen, der unmittelbar bei dem Cursor beginnt. Nachdem das UP die Zeile gelistet hat, steht statt dem BASIC-Befehl der Cursor, da BASIC Befehle bekanntlich als ein Byte abgespeichert werden.

Ich werde versuchen, die Ausgabe der Zeile zunaechst so zu gestalten, dass der Cursor vor dem BASIC-Befehl steht. Spaeter moechte ich mich immer mehr auf die eigentliche Fehlerquelle hinarbeiten (z.B. fehlende Klammern).

Und damit mal wieder HAPPY DATAS

von Bernd Niedermeier



```

00100 ;*****
00110 ;**** SIMULATION DES BEFEHLS 'HELP' ****
00120 ;**** (AUFFINDEN EINES FEHLERS IN EINER PROGRAMM- ****
00130 ;**** ZEILE UND ANZEIGE DESSELBEN AUF DEM BILD- ****
00140 ;**** SCHIRM ****
00150 ;**** BERND NIEDERMEIER HIRSCHBERGWEG 9 ****
00160 ;**** 8011 HEIMSTETTEN TEL.: (089) 903 57 31 ****
00170 ;**** VERSION 1.0 VOM 13.12.82 ****
00180 ;*****
00190 ;
00200 ;

```

```

FFC0 00210 ORG 0FFC0H
FFC0 2AF540 00220 LD HL,(40F5H) ;ERRORLINE=>HL
FFC3 E5 00230 PUSH HL ;HL SICHERN
FFC4 CDAF0F 00240 CALL 0FAFH ;AUSGABE DER Z.NR. DEZ.
FFC7 3E20 00250 LD A,' ' ;SPACE
FFC9 CD2A03 00260 CALL 32AH ;AUSGEBEN
FFCC E1 00270 POP HL ;HL ZURUECKHOLEN
FFCD 5D 00280 LD E,L ;HL IN DE UEBERTRAGEN
FFCE 54 00290 LD D,H
FFCF CD2C1B 00300 CALL 1B2CH ;SUCHE NACH ADRESSE DER
00310 ;ZEILE MIT NR HL
00320 ;(STEHT IN BC)
FFD2 03 00330 INC BC ;BC AUF ERSTES ZEICHEN
FFD3 03 00340 INC BC ;DES PROGR.TEXTS BRINGEN
FFD4 03 00350 INC BC
FFD5 03 00360 INC BC
FFD6 2AEE40 00370 LD HL,(40EEH) ;ADR DES BEFEHLS WO ERROR
00380 ;AUFTRAT
FFD9 3E3A 00390 LD A,':'
FFDB BE 00400 CP (HL) ;ZEICHEN=': '?
FFDC 2808 00410 JR Z,INC1 ;JA, EINMAL 'INC HL'
FFDE AF 00420 XOR A
FFDF BE 00430 CP (HL) ;ZEICHEN=NULL?
FFE0 2005 00440 JR NZ,WEITER ;NEIN:KEIN 'INC HL'
FFE2 23 00450 INC HL ;HL AUF PROGRAMMTEXT
FFE3 23 00460 INC HL
FFE4 23 00470 INC HL
FFE5 23 00480 INC HL
FFE6 23 00490 INC1 INC HL
FFE7 E5 00500 WEITER PUSH HL ;HL SICHERN
FFE8 5E 00510 LD E,(HL) ;BYTE SICHERN
FF09 D5 00520 PUSH DE ;E SICHERN
FFEA 365F 00530 LD (HL),5FH ;LEVEL2 CURSOR LADEN
FFEC 69 00540 LD L,C ;HL=>BC
FFED 60 00550 LD H,B
FFEE CD7E2B 00560 CALL 2B7EH ;UMWDLG DER ZEILE IN
00570 ;BASIC KEYWORDS UND BE-
00580 ;REITSTELLUNG IM I/O BUFF
FFF1 2AA740 00590 LD HL,(40A7H) ;ADR DES I/O BUFFERS=>HL
FFF4 CD752B 00600 CALL 2B75H ;GIBT ZEILE AUS
FFF7 CDFE20 00610 CALL 20FEH ;CARRIAGE RETURN
FFFA D1 00620 POP DE ;E ZURUECKHOLEN
FFFB E1 00630 POP HL ;ADR D. BEFEHLS ZURUECK
FFFC 73 00640 LD (HL),E ;ORIGINALBYTE ZURUECK
FFFD C9 00650 RET ;=>BASIC
0000 00660 END

```

```

000000 TOTAL ERRORS
33360 TEXT AREA BYTES LEFT

```

```

INC1 FFE6 00490 00410
WEITER FFE7 00500 00440

```

Hochauflösende Softgraphic für den TRS80 Model I (Teil 4)

Zum Abrunden des Programmpakets, das ich in den letzten 3 Folgen vorgestellt habe, möchte ich heute noch eine Routine vorstellen, die es ermöglichen soll, Textstrings auf das Tableau zu schreiben, z.B. zur Beschriftung von Diagrammen und Kurven.

Vor dem Aufruf von STRMOV wird in der BASIC-Variablen X\$ der Text bereitgestellt. Die Routine arbeitet nicht wie PRINT @, d.h. man muß nicht die Anfangsposition relativ zur oberen linken Ecke des Tableaus berechnen. Vielmehr werden die Koordinaten dieser Position in den Variablen X% und Y% benötigt. STRMOV bestimmt daraus selbst die Anfangsadresse durch Aufruf von ETKBM (s. Teil 2). Es dürfte klar sein, daß jeweils 6 verschiedene Koordinatenpaare zur gleichen String-Anfangsadresse führen.

Nachdem anschließend X\$ lokalisiert worden ist, wird untersucht, ob der gesamte String auf dem Tableau Platz findet oder ob er gekürzt werden muß. Letzteres führt zu einem etwas anderen Einsprungpunkt der uns schon von früher bekannten TRNSFR-Routine (s. Teil 3, FUNCIN).

Ich habe die Routine STRPTR als Subroutine ausgeführt, da sie auch von FUNCIN benutzt werden kann. Das Rücksetzen des Integer-Flags ist übrigens nötig, da es von VARPTR verändert wurde. Würde es nicht gesetzt, ergäbe sich ein TYP MISMATCH ERROR bei der Rückkehr nach BASIC. Aus eben diesem Grunde wurde in FUNCIN der Rücksprung über 0A9AH gewählt. Wenn FUNCIN nun STRPTR aufruft, kann man mit RET zurückspringen - und schon wurden wieder ein paar Byte gespart:

Ersetze	FUNCIN LD HL, XSTRNG	durch	FUNCIN CALL STRPTR
	...		
	JR NZ, SYNERR		
und	JP 0A9AH	durch	RET

Da wir gerade beim Bytesparen sind. Ersetzt doch bitte auch VARY bzw. VARX in den Zeilen 1030 bzw. 1260 des 2. Teils durch ROUNDY bzw. ROUNDX von Teil 3. Ihr könnt dann die Zeilen 900 bis 930 von Teil 2 ebenfalls streichen und VARPTR funktioniert immer noch! ... 4 Byte gespart!

Noch ein Wort zur Behandlung der Fehlermeldungen, die in ETKBM beim Überschreiten der Grenzen des Tableaus auftreten. Die im Teil 2 vorgeschlagene Lösung ist nicht sehr schön, und zwar aus folgenden Gründen:

Die (Unter-)Routine OVERFL gibt lediglich einen Fehlercode "nach oben". Dieser muß dann vom Hauptprogramm (BASIC) abgefragt werden - und zwar bei jedem Rücksprung! Der Fehlercode "-2" darf dabei keine "normale", zulässige Information sein. Beim Aufruf von EVBYTE wird das deutlich: Die "-2" muß streng getrennt werden von der Menge der "erlaubten" Tableaupositionen - konkret, das Tableau darf nicht den Speicherplatz "-2" bzw OFFFEH enthalten!

Aber diese Methode der Fehlerbehandlung ist auch noch aus einem anderen Grunde nicht zu empfehlen. Ich hatte ursprünglich beim Test der X-Koordinate in Zeile 1340 von ETKBM "JR NZ, OVERFL" geschrieben, mit der Folge, daß z.B. bei negativen X-Werten die Routine mit einem POP zu wenig verlassen wurde und das System zusammenbrach!

Der Programmierer muß also peinlich darauf achten, auf welcher

Stack-Ebene eine Fehlermeldung generiert werden soll. Als möglichen Ausweg könnte man sich folgende Programmstruktur denken: Auf der/einer hohen Programmebene wird die Stellung des Stackpointers "notiert" bevor in die Tiefen der Unterprogramme hinabgestiegen wird. Tritt dann ein Fehler auf, so wird eine Fehlerbehandlungsroutine gerufen, die neben der Erzeugung einer angemessenen Meldung die wichtige Aufgabe hat, den "alten", notierten Stack wiederherzustellen bevor "nach oben" weitergegeben wird.

Klingt gut, nicht wahr? Ich kann nur sagen, daß ich erst ab dem Moment, als ich mir dieses überlegt hatte, die ON ERROR GOTO Befehle von BASIC so richtig verstanden habe. Und das Tollste - mit den Aufrufen von SYNERR hatte ich diese Lösung schon dauernd praktiziert!

SYNERR kann tatsächlich von jeder Unterprogrammebene aus angesprungen werden, ohne daß es "Stacksalat" gibt. Aufgeräumt wird dann in BASIC - und zwar dann und nur dann, wenn ein Fehler auftritt. Der (fehlerbehaftete) Rücksprung erfolgt gleichsam "aus einer anderen Richtung".

Also frisch ans Werk: Streicht im Teil 2 die Zeilen 970 bis 1010, ersetzt in den Zeilen 1080, 1130 und 1340 das "JR N...,OV..." durch "JP N...,OVRFLW" und fügt die neue Zeile "895 OVRFLW EQU 7B2H" ein. Dem OV-ERROR wird damit zwar eine neue, zusätzliche Bedeutung gegeben, aber das läßt sich bestimmt verkraften.

Vermißt Ihr die Routine ROUND? Nun, sie ist ganz einfach ein weiterer Einsprungpunkt in der Routine EVAL aus Teil 3. Vielleicht kann sie uns beim Erstellen eines BASIC-Programms noch nützlich sein.

In der Routine CLEAR von Teil 1 sollte es übrigens besser "LD BC, TABLEN - 1" heißen.

So, das waren nun alle Routinen, die ich vorstellen wollte. Wer schreibt weitere? Wer schreibt ein BASIC-Programm, das die Möglichkeiten dieses Programmpakets voll ausschöpft und dem Benutzer "serviert"?

Wer Schwierigkeiten hat beim Zusammenstellen der vier Teile, kann von mir ein komplettes Listing "in einem Guß" bekommen bei Einsendung eines adressierten und fankierten Rückumschlags (1,30 DM Porto).

Viel Spaß nun!

Euer Jürgen Degenhardt

00190	TRNSFR	EQU	29C8H	;transfer string
00220	VARPTR	EQU	260DH	;variable pointer
00250	SYNERR	EQU	1997H	;Syntax Error Routine
00360	FLAG	EQU	40AFH	;variable typ flag
00370				
02460	STRMOV	CALL	ETKBM	;evaluate destination
02470		EXX		;save HL
02480		CALL	STRPTR	;look at BASIC's X\$
02490		PUSH	DE	;save VARPTR(X\$)
02500		EXX		;restore destination (HL)
02510		PUSH	HL	;save it again
02520		LD	DE,LRIGHT	;DE = end of tableau + 1
02530		EX	DE,HL	;evaluate differenz betw.
02540		SBC	HL,DE	;dest. & end of tableau
02550		LD	B,L	;B = LSB of differenz
02560		LD	A,H	;A = MSB --"
02570		POP	DE	;DE = destination
02580		POP	HL	;HL = VARPTR(X\$)
02590		OR	A	;more than 255 bytes?
02600		JR	NZ,ENOUGH	;yes, no more testing
02610		LD	A,(HL)	;is LEN(X\$) > bytes
02620		SUB	B	;left in tableau?
02630		LD	A,B	;prepare if shorter
02640		JP	P,TRNSFR+1	;fill in, but truncate X\$
02650	ENOUGH	JP	TRNSFR	;fill in whole X\$
02660				
02670	STRPTR	LD	HL,XSTRNG	;points to BASIC's
02680		CALL	VARPTR	;X\$ - variable
02690		RST	20H	;test string
02700		JP	NZ,SYNERR	;else SYNTAX ERROR
02710		LD	A,2	;reset integer typ flag
02720		LD	(FLAG),A	
02730		RET		;return to caller
02740				
02750	XSTRNG	DEFM	'X\$'	
02760		NOP		

=====

Habt Ihr schon einmal ein Programm geschrieben, bei dem der Benutzer "menuegeführt" wird? Dann wißt Ihr ja auch, daß dabei immer wieder das Problem entsteht, daß der Benutzer auch wieder zum Hauptmenue zurückfinden muß - und das (fast) immer wenn es ihm gerade in den Sinn kommt.

Die gängigste und einzige in BASIC anwendbare Methode ist wohl die, daß man dem Benutzer in den diversen Untermenues mitteilt, welche Taste er drücken kann, um zum Hauptmenue zurückzukehren. Das muß dann eventuell Schrittweise von einer Programmebene zur nächsten geschehen. Oft benötigt man auch eine Hilfsvariable, mit deren Hilfe übergeordnete Programme erkennen können, daß "nach oben weiter durchgereicht werden soll". In FOR...NEXT-Schleifen tritt das Problem ähnlich auf.

Kein Problem, werdet Ihr jetzt sagen, wozu hat man NEWDOS! Dort gibt es Befehle wie CMD"F=POPS". Richtig, jetzt sind wir auf dem richtigen Weg!

Wenn wir uns nun noch einigen könnten, welche Taste wir für den "Rücksturz" in allen Untermenues benutzen wollen, dann könnte man eventuell darauf verzichten, immer den Hinweis zu geben: "Bitte Taste <m> drücken zur Rückkehr ins Hauptmenue ...".

Wie wäre es denn mit der BREAK-Taste? Die Aufschrift allein dürfte ja schon sinnfällig genug sein! Wie sieht es mit der Dekodierung aus? BREAK wird intern als ASCII 1 codiert. Die Abfrage müßte also etwa lauten:

```
A$=INKEY$: IF A$=CHR$(1) THEN ...
```

Doch nun erleben wir Schiffbruch. Es gibt nämlich schon eine Routine, die die BREAK-Taste abfragt, und die pfuscht uns nun ins Handwerk.

Aber könnten wir diese Routine nicht vor unseren Karren spannen? Wenn wir folgende zwei Bytes im NEWDOS 80 V2.0 ändern, so erreichen wir damit, daß nun nicht mehr das Programm "normal" gestoppt wird, sondern es wird ein L3-ERROR erzeugt (, der dann das Programm stoppt). Der L3-ERROR ist ja höchst überflüssig in DISK-Systemen, oder?

```
Ändere in SYS0/SYS,8,D0      C3 12 43
in                            C3 2D 01.
```

Mit einer ERROR-Trap-Routine fangen wir dann das Programm ab (ERROR-Code 44), führen ein CMD"F=POPS" durch und springen direkt zurück ins Hauptmenue. Hier jedoch sollten wir die Möglichkeit einbauen, mit einer anderen (nicht BREAK!) Taste ein END-Statement anzuspringen, da wir sonst das Programm überhaupt nicht mehr anhalten können.

Noch ein Hinweis: Wenn dieser Modus einmal eingebaut ist, sollte man sich hüten mit AUTO zu arbeiten, da dieser Befehl ja ebenfalls nur mit BREAK gestoppt werden kann.

Viel Spaß!

Euer Jürgen Degenhardt

```

99 CMD"F=POPS"
100 REM ***** Hauptmenue *****
110 ON ERROR GOTO 1000
120 REM Es folgt das Menue und ein Verteiler auf Unterprogramme
.
.
.
199 END
200 REM ***** Beginn eines Unterprogramms *****
.
.
.
1000 REM ***** ERROR TRAP ROUTINE *****
1010 IF ERR=44 RESUME 99 ELSE ...
```

BER/NIE Software Bernd Niedermeier
Hirschbergweg 9
8011 Heimstetten
Tel.: (089) 903 57 31

Hallo Clubfreunde,
hier habe ich einen Programmiertrick, der zwar nicht auf
meinem eigenen Mist gewachsen ist, aber unbedingt der
Veroeffentlichung bedarf.

Es handelt sich um eine simple Maschinenunterroutine, mit
der man in Strings stehende BASIC-Befehle abarbeiten kann.
Verwirrend?

Folgendes Anwendungsbeispiel:

Man will ein Funktionsplotprogramm schreiben, sich das
muehsame Reinpoken in eine vorher mit ':' aufgefullte
Programmzeile ersparen. Nach Laden des unten aufgelisteten
Programms kann man z.B. folgendes Programm schreiben:

```
10 DEFUSR=&HFFC0 'Unterroutinenanfang
20 CLEAR 300 'Reservierung von Stringspace, da String
               bis zu 255 Zeichen lang sein kann
30 LININPUT'Funktion f(x)=';F$
40 F$='F(X)='+F$+'GOTO 70' 'das Anhaengen von GOTO xxxxx
                           ist notwendig, da nach Aufruf
                           der Unterroutine sonst eine
                           READY-Meldung erfolgen wuerde.
                           xxxxx ist die Zeilenr, mit der
                           nach Aufruf der Unterroutine
                           fortgefahren werden soll. Bei
                           diesem Beispiel Zeilenr 70.

50 FOR X=0 TO 10
60 Z=USR(F$)
70 NEXT X
```

Den Anwendungsmoeglichkeiten duerften nur die Phantasie als
Grenze gegenuebrstehen.

*Die Funktionswerte von 0-10 stehen im Feld F().
Eine Dimensionierung muß vorher natürlich erfolgt sein.*

Im folgenden nun das Assemblerlisting:

```
100          ORG          0FFC0H
110          LD           HL,(4121H)
120          INC          HL
130          LD           A,(HL)
140          INC          HL
150          LD           H,(HL)
160          LD           L,A
170          DEC          HL
180          POP          AF
190          POP          AF
200          POP          AF
210          POP          AF
220          POP          AF
230          POP          AF
240          POP          AF
250          JP           1A81H
260          END
```

Die Routine wird nach setzen von HIMEM vom DOS mit CMD*LOAD programmname* geladen.

Damit mal wieder HAPPY DATAS...
Euer Bernie Keybounce



BEITRAGSERHÖHUNG:

Wie bereits in Ausgabe 18 dieser Zeitung mitgeteilt, erhöht sich ab Januar 84 der monatliche Clubbeitrag auf 4.- DM. Sind eingehende Beitragszahlungen nicht ganzzahlig durch die Beitragssätze teilbar, wird der Rest auf das Zusatzkonto gebucht.

UMLAUT-CHIP

Es gibt ihn noch - den Zeichengenerator für den deutschen Zeichensatz - zum vernünftigen Preis. Es handelt sich hierbei um eine Clubeigene Entwicklung und nicht um einen Nachbau. Hier das Wichtigste in Stichpunkten:

- Umlaute und echte Unterlängen auf dem Bildschirm.
- zwei Zeichensätze deutsch/international umschaltbar.
- betriebsfertig - komplett mit Schalter.
- ideal für Umlaut-SCRIPSIT Anwender.
- einfach einzubauen - alten Zeichengeneratorchip aus Steckfassung (Tastatur) entfernen, neuen Umlautchip hineinstecken. Nur bei sehr alten Geräten ist für den Character-Generator (Z29) keine Fassung vorhanden.
- in begrenzten Umfang kann ich auf Sonderwünsche bei der Zeichendarstellung eingehen.
- voll ausnutzbar nur mit Kleinschrift-Umbausatz.
- Preis DM 50.-

Bessere BASIC-Listings vom Videogenie

von Ulrich Heidenreich

Nachdruck mit freundlicher Genehmigung der MICRO EXTRA-Redaktion

Sei es aus Speicherplatzgründen, oder daß der Autor schlechthin unübersichtlich programmiert: ein solches Listing (Abbildung 1) ist nur mühsam zu lesen und zu verstehen. Dieser Beitrag stellt nun eine Z-80-Routine vor, die aus beliebigen BASIC-Listings eine übersichtliche Form wie in Abbildung 2 erstellt.

```

40 CLEAR1873;DEFINT A-Z;DEFSTRH,R-T;DIMH(255),T(255);H="0123456789ABCDEF"
80 FORI=1TO16:FORJ=1TO16:H(K)=MID$(H,I,1)+MID$(H,J,1)+" ";IFK<320RK>95,T(K)="."E
LSET(K)=CHR$(K)
90 K=K+1:NEXT:NEXT
100 INPUT"STARTADRESSE;ENDADRESSE (NICHT MIT KOMMA TRENNEN)
";S;R=LEFT$(S,4);GOSUB1050;A=D;R=RIGHT$(S,4);GOSUB1050;O=D
300 FORI=ATOOSTEP14:P=I/256;PRINTH(P)"H(I-P*256) ";;FORJ=IT0J+13:PRINTH(PEEK(J
));NEXT:PRINT" ";;FORJ=IT0J+13:PRINTT(PEEK(J));NEXT:NEXT:PRINT:GOTO100
1050 D=0;FORL=0TO3:J=ASC(MID$(R,L+1,1));IFJ<58J=J-48ELSEJ=J-55
1060 D=D+J*16A(3-L);NEXT:RETURN

```

Abb. 1: Beispiel eines unübersichtlichen BASIC-Listings

```

40 CLEAR1873
DEFINT A-Z
DEFSTRH,R-T
DIMH(255),T(255)
H="0123456789ABCDEF"
80 FORI=1TO16
FORJ=1TO16
H(K)=MID$(H,I,1)+MID$(H,J,1)+" "
IFK<320RK>95,T(K)="."ELSET(K)=CHR$(K)
90 K=K+1
NEXT
NEXT
100 INPUT"STARTADRESSE;ENDADRESSE (NICHT MIT KOMMA TRENNEN)
";S
R=LEFT$(S,4)
GOSUB1050
A=D
R=RIGHT$(S,4)
GOSUB1050
O=D
300 FORI=ATOOSTEP14
P=I/256
PRINTH(P)"H(I-P*256) ";
FORJ=IT0J+13
PRINTH(PEEK(J));
NEXT
PRINT" ";
FORJ=IT0J+13
PRINTT(PEEK(J));
NEXT
NEXT
PRINT
GOTO100
1050 D=0
FORL=0TO3
J=ASC(MID$(R,L+1,1))
IFJ<58J=J-48ELSEJ=J-55
1060 D=D+J*16A(3-L)
NEXT
RETURN

```

(Anm.: 'A' entspricht '1')

Abb. 2: Mit Formatierungs-Programm ausgegebenes BASIC-Listing

Insbesondere, wenn das BASIC-Programm unterschiedlich lange Zeilennummern besitzt und die Programmzeile die maximale Zeilenlänge überschreitet, macht es sich störend bemerkbar, daß das Standard-BASIC keine starre Spalteneinteilung wie z. B. FORTRAN besitzt; eine optische Trennung zwischen Label (hier: Zeilennummer) und Inhalt der Programmzeile fehlt! Zur Übersicht trägt auch die Aneinanderreihung vieler Einzelbefehle innerhalb einer Programmzeile keineswegs bei.

Ein Ausgabeprogramm, das folgende Forderungen erfüllt, kann zur Lesbarkeit eines Programmlistings erheblich beitragen:

1. Die Zeilennummern sollten rechtsbündig ausgegeben werden.
2. Die Programmzeilen sollten linksbündig ab einer festen TAB-Position (hier TAB[6]) folgen.
3. Jede Programmzeile sollte jeweils nur ein Statement enthalten.

Die hier gezeigte Maschinenroutine (Abbildung 3) bietet diese Vorteile.

-----		00000		-----	
00010		:		Formatierte Ausgabe von BASIC-Listings	
00020		-----		-----	
00030		-----		-----	
00040		ORG		7F71H (MEM-SIZE : 32025 fuer 16 KByte)	
00050		-----		-----	
00060		LINVAL EQU		0040H (KONST. :Zeilenlaenge (hier:BLD))	
40EC		00070 PUNKT EQU		40ECH (ADRESSE:Zeile fuer Punkt-Option)	
41DF		00080 DOSVEK EQU		41DFH (VEKTOR :DOS-Vektor zum Einbinden)	
1D9B		00090 TASTE EQU		1D9BH (UP :Kontrolle Tastendruck)	
0FAF		00100 OUTHL EQU		0FAFH (UP :HL dezimal ausgeben)	
032A		00110 OUTA EQU		032AH (UP :Zeichen aus A ausgeben)	
2B7E		00120 TRANSL EQU		2B7EH (UP :aus Token uebersetzen)	
1B2C		00130 LISEEK EQU		1B2CH (UP :Adr.von Zeile DE -> BC)	
06CC		00140 BASIC EQU		06CCH (ADRESSE:BASIC-Warmstart)	
40A7		00150 IOPUFF EQU		40A7H (VEKTOR :Zeiger auf I/O-Puffer)	
20FE		00160 CR EQU		20FEH (UP :CR mit LF ausgeben)	
09C2		00170 HLBCDE EQU		09C2H (UP : (HL...) -> E,D,C,B)	
00180		-----		-----	
7F71 3EC3		00190 START LD		A,0C3H (RET durch JP ersetzen)	
7F73 32DF41		00200 LD		(DOSVEK),A (Staradresse "ANFANG")	
7F75 217F7F		00210 LD		HL,ANFANG (schreiben)	
7F79 22E041		00220 LD		(DOSVEK+1),HL	
7F7C C3CC06		00230 JP		BASIC	
7F7F B7		00240 ANFANG OR		A (Wenn A=0, dann RET, da	
7F80 C8		00250 RET		Z (Aufruf von INPUT)	
7F81 2B		00260 DEC		HL	
7F82 2B		00270 DEC		HL	
7F83 E5		00280 PUSH		HL ((SP):Zeiger erste Zeile	
7F84 CD2C1B		00290 CALL		LISEEK	
7F87 E1		00300 POP		HL (HL:Zeiger erste Zeile	
7F88 C5		00310 PUSH		BC ((SP):Zeiger letzte Zeil.	
7F89 CDC209		00320 LINE CALL		HLBCDE	
7F8C 7A		00330 LD		A,D (Zeiger = NULL bedeutet	
7F8D B3		00340 OR		E (Programm-Ende)	
7F8E CACC06		00350 JP		Z,BASIC	
7F91 D5		00360 PUSH		DE ((SP):Zeiger folgende Z.	
7F92 E5		00370 PUSH		HL ((SP):Zeiger auf Text	
7F93 ED43EC40		00380 LD		(PUNKT),BC (BC:aktuelle Zeilen-Nr.	
7F97 CB78		00390 BIT		7,B	
7F99 201A		00400 JR		NZ,NOBLAN	
7F9B 50		00410 LD		D,B	
7F9C 59		00420 LD		E,C (DE:Zeilen-Nummer	
7F9D 210900		00430 LD		HL,9 (vor Zeilennummer 0...4	
7FA0 CDF97F		00440 CALL		OUTBLA (Blanks ausgeben)	
7FA3 216300		00450 LD		HL,99	
7FA6 CDF97F		00460 CALL		OUTBLA	
7FA9 21E703		00470 LD		HL,999	
7FAC CDF97F		00480 CALL		OUTBLA	
7FAF 210F27		00490 LD		HL,9999	
7FB2 CDF97F		00500 CALL		OUTBLA	
7FB5 2AEC40		00510 NOBLAN LD		HL,(PUNKT) (HL:Zeilen-Nummer	
7FB8 CDAF0F		00520 CALL		OUTHL	
7FBB 3E20		00530 LD		A, / /	
7FBD CD2A03		00540 CALL		OUTA	
7FC0 E1		00550 POP		HL (HL:Zeiger auf Text	
7FC1 CD7E2B		00560 CALL		TRANSL	
7FC4 2AA740		00570 LD		HL,(IOPUFF) (HL:Zeiger auf I/O-Puffer	
7FC7 063A		00580 INITB LD		B,LINLEN-6 (B:Zeichenzaehler	
7FC9 7E		00590 OUTCHA LD		A,(HL) (A:Zeichen	
7FCA 23		00600 INC		HL (HL:Zeiger auf Zeichen	
7FCB B7		00610 OR		A	
7FCC 281E		00620 JR		Z,NEXLIN (NULL=Zeilen-Ende	
7FCE FE0A		00630 CP		10	
7FD0 280C		00640 JR		Z,LFEEED	
7FD2 FE3A		00650 CP		/ / (Doppelpunkt durch	
7FD4 2808		00660 JR		Z,LFEEED (Linefeed ersetzen	
7FD6 CD2A03		00670 CALL		OUTA	
7FD9 CD9B1D		00680 CALL		TASTE	
7FDC 10EB		00690 DJNZ		OUTCHA	
7FDE CDFE20		00700 LFEEED CALL		CR	
7FE1 0606		00710 LD		B,6	
7FE3 3E20		00720 LD		A, / /	
7FE5 CD2A03		00730 BLANK CALL		OUTA	
7FE8 10FB		00740 DJNZ		BLANK	
7FEA 18DB		00750 JR		INITB	
7FEC CDFE20		00760 NEXLIN CALL		CR	
7FEF D1		00770 POP		DE (DE:Zeiger aktuelle Zeile	
7FF0 E1		00780 POP		HL (HL:Zeiger letzte Zeile	
7FF1 E5		00790 PUSH		HL ((SP):Zeiger letzte Zeile	
7FF2 DF		00800 RST		18H (Vergleich HL-DE	
7FF3 EB		00810 EX		DE,HL (HL:Zeiger aktuelle Zeile	
7FF4 FACC06		00820 JP		M,BASIC (letzte Zeile erreicht ?	
7FF7 1890		00830 JR		LINE	
7FF9 DF		00840 OUTBLA RST		18H (Vergleich HL-DE	
7FFA 3E20		00850 LD		A, / /	
7FFC D42A03		00860 CALL		NC,OUTA	
7FFF C9		00870 RET			
7F71		00880 END		START	

Abb. 3: Maschinenroutine

Das Einbinden der Routine ins Betriebssystem erfolgt über den DOS-Vektor an 41DFH. In der Grundversion mit Cassettenbetrieb stört das Ändern dieses Vektors nicht weiter; zusammen mit dem DOS muß das „RETZ“ in Assemblerzeile 250 durch ein „JP Z,nn“ ersetzt werden; nn ist hierbei die Sprungadresse an 41E0/E1H. „LIST“ und „LLIST“ ergeben dann das entsprechend modifizierte Listing.

Individuell angepaßt werden muß evtl. nur die maximale Zeilenlänge des gewählten Ausgabegerätes durch Ändern der Konstante LINLEN. Hier wurden die 64 Zeichen des Bildschirms voreingestellt. Je nach Drucker (z.B. EG 3085/Itoh 8510 : Pica 80; Elite 96; Compressed 136) muß LINLEN auf die entsprechende Länge begrenzt bzw. erweitert werden.

Da die LLIST/LIST-Routinen größtenteils durch das Formatierungsprogramm ersetzt werden, ergibt sich bei „LIST-nn“ für den Spezialfall, daß die Zeilennummer nn nicht existiert, folgende Abweichung vom „normalen“ LIST: Gelistet wird nicht nur bis zur vorhergehenden, sondern bis zur folgenden Zeile. Die „-“-Option und der Stop mit SHIFT bleiben erhalten.

Diese Eigenschaften der Ausgaberroutine werden im Einzelnen durch folgende Teilroutinen bewerkstelligt:

- START: Einbinden über den DOS-Vektor 41DFH
- ANFANG: Übergabe der Listparameter
- LINE ff.: Kontrolle auf Programmende
Ausgabe von 0...4 Blanks und nachfolgender Zeilennummer
Rückübersetzung aus Zwischencode
- OUTCHA: Ausgabe von LINLEN-6-Zeichen
- NEXLIN: Kontrolle, ob Endzeile überschritten

(Das in Abbildung 2 gezeigte Programm ist übrigens ein Schneller HEX-DUMP mit Übersetzung druckbarer ASCII-Zeichen.)

===== MITGLIEDER-ADRESSLISTE (ALPHABETISCH) =====

NAME =====	VORNAME =====	ADRESSE =====	WOHNORT =====	TELEFON =====
BALLARIN	GREGOR	OWINGERSTR. 6	777 UEBERLINGEN	07551/63919
BAMIEDEMANN	KARL	PERETSHOFENERSTR. 7	8000 MUENCHEN 71	089/7913535
BERGBAUER	RUDOLF	GULDEINSTR. 52	8000 MUENCHEN 2	089/508147
BERGER	FRANZ	SCHUBERTSTR. 5	8037 OLCHING	08142/16876
BOEHLER	SEPP	MEMELWEG 21	7400 TUEBINGEN	07071/31825
BONENBERGER	PETER	WALDBLICKSTR. 15	7912 WEISSENHORN	07309/5570
BOVERMANN	KLAUS	OBERFOEHRINGERSTR. 107	8000 MUENCHEN 81	089/952239
BRANDES	HANS-DIETER	KOETNERHOLZWEG 47	3000 HANOVER 91	0511/2100547
BRUEBACH	ALFRED	HAYDNSTR. 5	3501 FULDABRUECK	0561/41929
BUERGMAYR	MARKUS	MUENCHNERSTR. 22/2	8019 STEINHOERING	08094/1204
DEGENHARDT	JUERGEN	HILDEBRANDSTR. 34	3300 BRAUNSCHWEIG	0531/325700
DENZ	KLAUS	NELL.-SCHIERBERG 74	2846 NEUENKIRCHEN	05493/665
DUNKE	ANDREAS	PFANNMUELLERWEG 19	6100 DARMSTADT	06151/717700
EICKENBERG	GUSTAVO	JOHANN CLANZESTR. 43/W73	8000 MUENCHEN 70	089/7692251
EISENBERGER	KARL-HEINZ	GARTENSTR. 3	8011 GRASBRUNN 1	089/465621
ENDRES	MICHAEL	BRUCHSTR. 54	6920 SINSHEIM	07261/63666
FRANZ	WOLFGANG	J.BAPTIST ZIMMERMANNSTR 4	8018 GRAFING	08092/5303
GEBERT	MANFRED	GAUTINGERSTR. 8	8031 GEISENBRUNN	
GIESELMANN	WILLHELM	AHRWEG 20	5142 HUECKELHOVEN	02433/85579
GRAESSLE	WILHELM	RACHELSTR. 34	8313 VILSBIBURG	08741/7450
GRENSING	WOLFGANG	HOMBERGER HOF	7776 OWINGEN	07551/62410
GREUBEL	KARL-HEINZ	OBENER WEG 9	8730 BAD KISSINGEN	0971/9380
GRIES	ULRICH	SILBERSTEINSTR. 92	1000 BERLIN 44	030/6253625
GROSSEGESSE	HANS JORDAN	WOLFRATSHAUSENER-STR. 68A	8000 MUENCHEN 70	089/7231905
HAIBLE	BERNHARD	SCHOENHUTWEG 5	7170 SCHWAEBISCH HALL	0791-43703
HANNE	BRUND	BIRKENSTR. 2	3014 LAATZEN 1	0511/867681
HERZOG	BENEDICT	STRASSBURGER STR. 77	2800 BREMEN 1	04221/344954
HESS	BERNHARD	KAZMAIRSTR. 30	8000 MUENCHEN 2	089/503125
HOMBERGER	RUDOLF	ROSEGGERSTR. 9	8900 AUGSBURG 21	0821/84173
HORNUNG	GUENTHER	KREUZBERGWEG 2	5568 DAUN	06592/1623
HUBER	HANS	HURTOEST 14	8225 TRAUNREUT	08669/5805
KART	RENATE	DEROYSTR. 6	8000 MUENCHEN 2	089/185903
KIRCHNER	PETER	BLUMENSTR. 11	8938 BUCHLOE	08241/2332
KOSTHORST	ALFONS	DORFBAUERNGEOHEFT 58	4236 HAMINKELN 2	02852/4519
KRAML	KLAUS	SCHOENSTR. 20	8000 MUENCHEN 90	089/6518617
KRANZ	GISELA	POSTFACH 1170	8218 UNTERWOESSEN	08641/8221
KRETSCHMAR	GUENTER	LEITENWEG 16	8190 WOLFRATSHAUSEN	08171/18457
KRONSCHNABL	KURT	VEILCHENWEG 5	8037 NEU-ESTING	08142/20656
LUECKEL	MANFRED	OSTERFELDERSTR. 13	4250 BOTTROP	02041/22324
MADER	MARTIN	SEBASTIAN-FRANCK-STR. 5	8850 DONAUOERTH	0906/66731
MAIER	GERHARD	NEUBIBERGER STR. 58/2	8011 PUTZBRUNN	089/6015887
MAYRING	DR. LOTHAR	KARLSTR. 43/III	8000 MUENCHEN 2	089/595170
MILICZEK	KARL-HEINZ	HEITERWANGER STR. 46	8000 MUENCHEN 70	089/7602966
MODEL	KLAUS	YORCKSTR. 73	1000 BERLIN 61	030/7851837
MOEBIUS	WALTER	ZUR BREITE 14	7753 ALLENBACH	07533/5591
NETZ	BERND	LAUINGERSTR. 10	8000 MUENCHEN 50	089/1491221
NIEDERMEIER	BERND	HIRSCHBERGWEG 9	8011 KIRCHHEIM	089/9035731
ORTHUBER	WOLFGANG	CHR.-PROBST STR. 16/1016	8000 MUENCHEN 40	089/3233263
PENTENRIEDER	FRANZ JOSEF	WILDMOSSTR. 9	8130 STARNBERG-WANGEN	08151/89071
PFEIFFER	WOLFGANG	BERNHARDIRING 7	8852 KAISHEIM	09009/1064
RAUCH	NORBERT	ERNST-HAECKEL-STR. 69 B	8000 MUENCHEN 50	089/8123081
REICHELSDORF	WOLFGANG	MARIENBADERSTR. 21	8858 NEUBURG/DONAU	08431/7846
RESSEL	JOSEF	EFFNERSTR. 75/C	8000 MUENCHEN 81	089/981408
RIEGER	LEONHARD	INN TALSTR. 4	8018 GRAFING	08092/5412

===== MITGLIEDER-ADRESSLISTE (ALPHABETISCH) =====

NAME	VORNAME	ADRESSE	WOHNORT	TELEFON
====	=====	=====	=====	=====
SAGNER	RAINER	ANSELWEG 10	8050 PULLING	00161/1546
SALDER	WOLF-MARKO	KREMHILDENSTR. 2 /5	8034 GERMERING	009/8412448
SCHAARSCHMIT	BERNHARD	RAIFFEISENSTR. 62	8044 UNTERSCHLEISSHEIM	009/3101484
SHELLHORN	KURT	DONNERSBERGERSTR. 32	8000 MUENCHEN 2	009/165394
SCHICK	KLAUS	RHEINGAUSTR. 6	6238 HOFHEIM	06192/7500
SCHIER	REINHOLD	PAPPENHEIMSTR. 12	8000 MUENCHEN 2	009/194926
SCHITTENHELM	GERHARD	REUSSENACHSTR. 23	7778 MARKDORF	07544/3170
SCHLADEBACH	GERT	GERBERGASSE 1	7845 BUGGINGEN	07631/5379
SCHNEIDER	WOLFGANG	KRUENERSTR. 31	8000 MUENCHEN 70	009/7604120
SCHRAMM	VOLKER	PFRUENDESIEDLUNG 17	8311 GERZEN	00744/226
SCHUMMEL	MICHAEL	BRENERSTR. 143	2940 WILHELMSHAVEN	04421/25978
SCHWARM	HANS-MARTIN	ROLLNERSTR. 50	8500 NUERNBERG 10	0911/335820
SEIBOLD	RUDI	SEMPTWEG 2	8011 KIRCHHEIM	009/9037351
SEITZ	PETER	BONAMESSER STR.69	6000 FRANKFURT 50	
SPIES	KARL	LUDWIG-STEUB-STR. 7	8025 UNTERHACHING	009/6115575
SYLVIO	WALDAMERO	POSTFACH 402004	8000 MUENCHEN 40	009/5804184
THALMEIER	GREGOR	POSTFACH 1140	8011 KIRCHSEEON	00091/9085
TRAPPSCHUH	KURT	REINECKESTR. 6	8036 HERRSCHING	00152/2512
VOGELSBANG	MANFRED H.	POSTFACH 280	8316 FRONTENHAUSEN	00732/514
VOIGTS	FRIEDEMANN	ESCHENSTRASSE 4	8034 GERMERING	009/8414991
WIMMER	FRANZ	RINGSTR.20	8031 MAISACH	00142/13876
WINKLER	HERMANN	ASTALLERSTR. 6	8000 MUENCHEN 2	009/5024853
WIRTZ	WOLFGANG	SCHANDERLWEG 7	8000 MUENCHEN 82	009/4304324

CLUBKONTO

Postscheckamt München
 BLZ: 700 100 80
 Kontonr.: 3452 35-800
 Inhaber: G. Thalmeier