Hans Kohorst

Der geknackte CBM

Während z. B. Tandy klammheimlich ab und zu die ROM-Software seiner Computer ändert und auch selbst keinerlei Dokumentation darüber verbreitet, existieren bei Commodores CBM heute drei Betriebssysteme (2000/3000/4000), die wenigstens zum Teil in den Handbüchern dokumentiert sind. Will man etwas tiefer einsteigen oder Maschinenprogramme adaptieren, reichen die Handbuch-Angaben aber meist nicht aus. Hier folgt deshalb eine Gegenüberstellung von ROM-Adressen in den Versionen 3000 und 4000 des CBM; letztere ist mit dem CBM 8000 identisch.

Basic 3.0	Basic 4.0	Beschreibung
C000-C045	B000-B065	Tabellen der Basic-Befehle
C046-C073	B066-B093	Tabellen der Basic-Funktionen
C074-C091	B094-B0B1	Hierarchie und „Action-Adressen“
C092-C192	B0B2-B20C	Tabelle der Basic-Worte
C193-C2A9	B20D-B321	Error-Meldungen
C2AA-C2D7	B322-B34F	Sucht im Stack nach FOR- oder GOSUB-Aktivitäten
C2D8-C31A	B350-B392	Schafft Raum für neue Zeile
C31B-C327	B393-B39F	Test: Stack zu tief
C328-C354	B3A0-B3CC	Verfügbaren Speicher prüfen
C355	B3CD	Holt Fehlermeldung (ab C193) und vollzieht
C389-C3AA	B3FF-B41E	Warmstart: „READY.“
C3AB-C441	B41F-B4B5	Bearbeitet neue Basic-Zeile
C442-C46E	B4B6-B4E1	Verketten von Basic-Zeilen nach Löschen bzw. Einfügen
C46F-C494	B4E2-B4FA	Holt Zeile von Tastatur
C495-C52B	B4FB-B5A2	Keywords vergleichen und durch 1-Byte-Befehl ersetzen
C52C-C55A	B5A3-B5D1	Sucht nach vorhandenen Zeilennummern
C55B	B5D2	Führt NEW aus und
C577-C5A6	B5EC-B621	generiert CLR
C5A7-C5BA	B622-B62F	Startet Basic-Programm
C5B5-C657	B630-B6DD	Generiert LIST
C658-C6FF	B6DE-B784	Generiert FOR
C700-C72F	B785-B7B6	Ausführung eines Basic-Statements
C730-C73E	B7B7-B7C5	Generiert RESTORE
C73F-C76A	B7C6-B7ED	Generiert STOP oder END
C76B-C784	B7EE-B807	Generiert CONT
C785-C78F	B808-B812	Generiert RUN
C790-C7AC	B813-B839	Generiert GOSUB
C7AD-C7D9	B83A-B85C	Generiert GOTO
C7DA	B85D	Generiert RETURN: dann
C7F3-C80D	B883-B890	Generiert DATA: Überspringen der Zeile
C80E-C810	B891-B893	Sucht nach nächstem Statement
C811-C82F	B894-B8B2	Sucht nach nächster Zeile
C830	B8B3	Generiert IF: wenn, dann
C843-C852	B8C6-B8D5	Generiert REM: überspringen
C853-C872	B8D6-B8F5	Generiert ON
C873-C8AC	B8F6-B92F	Holt Integer-Zahl
C8AD-C927	B930-BA87	Generiert LET
C98B-C990	BA88-BA8D	Generiert PRINT#
C991-C9A4	BA8E-BAA1	Generiert CMD
C9A5-CA1B	BAA2-BB1C	Generiert PRINT
CA1C-CA38	BB1D-BB39	Ausgabe eines STRINGS (Adresse im Akku und Y-Reg.)
CA39-CA4E	BB3A-BB4B	Ausgabe eines Zeichens (Inhalt des Akkus)
CA4F-CA7C	BB4C-BB79	Bearbeitung falscher Eingaben
CA7D-CAA6	BB7A-BBA3	Generiert GET
CAA7-CAC0	BBA4-BBBD	Generiert INPUT#
CAC1-CAF9	BBBE-BBF4	Generiert INPUT
CAFA-CB06	BBF5-BC01	Eingabe annehmen und beantworten
CB07-CBFB	BC02-BCF6	Generiert READ
CBFC-CC1F	BCF7-BD18	Meldung EXTRA IGNORED/REDO FROM START
CC20-CC78	BD19-BD71	Generiert NEXT
CC79-CC9E	BD72-BD97	Ausgabe: TYPE MISMATCH (wenn erforderlich)
CC9F-CDEB	BD98-BEE8	Holt Strings und numerische Ausdrück
CDEC-CDF1	BEE9-BEEE	Behandelt Ausdruck in Klammern
CDF2-CE02	BEEF-BEFF	Sucht nach Komma
CE03-CE07	BF00-BF0B	Ausgabe SYNTAX ERROR: verläßt Programm
CE08-CE0E	C047-C085	Erkennt Funktion und stellt Bezug dazu her
CE0F-CE88	BF8C-C046	Sucht nach Variablen-Namen
CEC8-CEF7	C086-C0B5	Generiert OR und AND
CEF8-CF5F	C0B6-C11D	Führt Vergleiche aus
CF60-CF6C	C11E-C12A	Generiert DIM
CF6D-CFF6	C12B-C1BF	Sucht nach Variablen
D001-D077	C1C0-C2C7	Setzt neue Variable
D078-D088	C2C8-C2D8	Subroutine für Array-Pointer
D089-D08C	C2D9-C2DC	Zahl 32768 in binärer Form
D08D-D0AB	C2DD-C2FB	Umrechnung Fließkomma/Integer
D0AC-D227	C2FC-C447	Suche/Aufbau eines Arrays
D228-D258	C477-C4A7	Bearbeitet ARRAY
D259	C4A8	Generiert FRE
D26D-D279	C4BC-C4C8	Wandelt Integer in Fließkomma um
D27A-D27F	C4C9-C4CE	Generiert POS
D280-D28C	C4CF-C4DB	Prüft, ob direkt. oder indirekt. Befehl. Ausg.: ILLEGAL DIRECT
D28D-D2BA	C4DC-C509	Generiert DEF
D2BB-D2CD	C50A-C51C	Überprüft FN-Syntax
D2CE-D33E	C51D-C58D	Bearbeitet FN
D33F-D34E	C58E-C59D	Generiert STR$
D34F-D360	C59E-C5AF	Setzt STRING-Vektor
D361-D3CD	C5B0-C61C	Sucht nach STRING-Elementen
D3EC-D3FF	C61D-C669	Schafft Freiraum für String
D400-D496	C66A-C74E	Entfernt überflüssige Strings
D517-D553	C74F-C78B	Concatenation-Routine (String-Addition)
D554-D57C	C78C-C7B4	Speichert String
D57D-D5B4	C7B5-C810	Überspringt nicht gebrauchte Strings
D5B5-D5C5	C811-C821	Säubert Descriptor-Stack
D5C6-D5D9	C822-C835	Generiert CHR$
D5DA-D605	C836-C861	Generiert LEFT$

Basic 3.0	Basic 4.0	Beschreibung
	C862-C86C	Generiert RIGHT$
	C86D-C896	Generiert MID$
	C897-C8B1	Holt String-Data
	C8B2-C8B7	Generiert LEN
	C8B8-C8C0	Holt Länge des Strings
	C8C1-C8D0	Generiert ASC
	C8D1-C8E2	Holt 1-Byte-String aus Basic
	C8E3-C920	Generiert VAL
	C921-C92C	Holt 2 Parameter für POKE und WAIT
	C92D-C942	Prüft, ob Bereich 0-65535 für POKE oder WAIT eingehalten
	C943-C959	Generiert PEEK
	C95A-C962	Generiert POKE
	C963-C97E	Generiert WAIT
	C97F-C985	Addiert 0.5 zu Akku 1
	C986-C997	Generiert Subtraktion
	C998-CA7C	Generiert Addition
	CA7D-CAB3	Komplementiert Akku 1
	CAB4-CAB8	Overflow
	CAB9-CAF1	Multipliziert ein Byte
	CAF2-CB1F	Enthält Fließkomma-Konstante
	CB20-CB5D	Generiert LOG
	CB5E-CBC1	Generiert Multiplikation
	CBC2-CBEC	Lädt Akku 2 mit Memory (Adresse in Akku und Y-Register)
	CBED-CC09	Testet und justiert Akku 1 und 2
	CC0A-CC17	Behandelt Overflow und Underflow
	CC18-CC2E	Multiplikation mit 10
	CC2F-CC33	10 in binärer Form
	CC34-CC3C	Division durch 10
	CC3D-CC44	Generiert Division durch x
	CC45-CCD7	Generiert Division in x und y
	CCD8-CCFC	Lädt Akku 1 aus Memory
	CCFD-CD31	Lädt Memory mit Akku 1
	CD32-CD41	Bringt Akku 2 nach Akku 1
	CD42-CD50	Bringt Akku 1 nach Akku 2
	CD51-CD60	Rundet Akku 1
	CD61-CD6E	Holt Vorzeichen von Akku 1
	CD6F-CD8D	Generiert SGN
	CD8E-CD90	Generiert ABS
	CD91-CDD0	Vergleicht Akku 1 mit Memory
	CDD1-CE01	Wandelt Fließ- in Fest-Komma
	CE02-CE28	Generiert INT
	CE29-CEB3	Wandelt ASCII-String in Fließkomma-Zahl
	CEB4-CEE8	Holt neue ASCII-Stelle
	CEE9-CEF8	Enthält Konstante
	CF78	Ausdruck IN, dann
	CF7F-CF92	Ausdruck Basic-Zeilennummer
	CF93-D0C6	Wandelt Fließkommazahl in ASCII
	D0C7-D107	Enthält Konstanten
	D108-D111	Generiert SQR
	D112-D14A	Generiert Exponential-Funktionen
	D14B-D155	Generiert Negation
	D156-D183	Enthält Konstanten
	D184-D1D6	Generiert EXP
	D1D7-D220	Testet Mehrfach-Funktionen
	D221-D228	Enthält RND-Konstanten
	D229-D281	Generiert RND
	D282-D288	Generiert COS
	D289-D2D1	Generiert SIN
	D2D2-D2FD	Generiert TAN
	D2FE-D32B	Enthält Konstante
	D32C-D35B	Generiert ATN
	D35C-D398	Enthält Konstante
	D399-D3B5	CHRGET für Zero-Page
	D3B6-D471	Basic-Kaltstart

Basic 3.0	Basic 4.0	Beschreibung
E1B7-E1DD	D448-D471	Enthält BYTES FREE und ###COMMODORE BASIC###
Diese	D7AC-D802	Generiert RECORD
Funktionen	D803-D837	Testet Disk-Parameter
sind im	D838-D872	Dummy Disk Control Message
Basic 3.0	D873-D919	Generiert CATALOG oder DIRECTORY
nicht	D91A-D92E	Ausgabe
enthalten.	D92F-D941	Sucht freie Sekundäradresse
	D942-D976	Generiert DOPEN
	D977-D990	Generiert APPEND
	D991-D9D1	Holt Disk Status
	D9D2-DA06	Generiert HEADER
	DA07-DA30	Generiert DCLOSE
	DA31-DA64	Führt Disk-Aufzeichnung durch
	DA65-DA7D	Generiert COLLECT
	DA7E-DAA6	Generiert BACKUP
	DAA7-DAC6	Generiert COPY
	DAC7-DAD3	Generiert CONTACT
	DAD4-DB0C	Fügt Kommando ein
	DB0D-DB39	Generiert DSAVE
	DB3A-DB65	Generiert DLOAD
	DB66-DB98	Generiert SCRATCH
	DB99-DB9D	Tested DIRECT COMMAND
	DB9E-DBD6	Ausgabe: ARE YOU SURE?
	DBD7-DBE0	Ausgabe: BAD DISK
	DBE1-DBF9	Löscht DS$ und ST
	DBFA-DC67	Setzt Disk-Kommando um
	DC68-DE29	Führt Basic-DOS aus
	DE2C-DE48	Holt DEVICE-Nummer
	DE49-DE86	Holt File-Namen
	DE87-DE9C	Holt Variablen-Parameter
* Nur Einstiegs-Punkte für den Bereich E1DE (E000) bis E6E4 (E600) *		
E1DE	E000	Initialisiert Register (Clear screen, Reset-Routine)
E285	E0A7	Nimmt Eingabe von Tastatur an
E2F2	E116	Nimmt Eingabe vom Bildschirm an
E3D8	E202	Gibt ein Zeichen auf Bildschirm
E61B	E442	INTERRUPT (Einstieg)
E62E	E455	Hardware-Interrupt-Routinen (Uhr, Cursor, Tastenfeld)
E6E4	E600	INTERRUPT (Ausgang)
E76A-E7FF	D717-D7AB	MLM subroutine
F000-F0B5	F000-F0D1	File-Messages
F0B6-F0B9	F0D2-F0D4	Sendet „TALK"
F0BA-F0BB	F0D5-F0D6	Sendet „LISTEN"
F0BC-F0ED	F0D7-F108	IEEE-Steuerzeichen
F0EE-F127	F109-F142	Sendet 1 Byte zum IEEE-Bus
F128-F135	F143-F150	Sendet Byte, löscht ATN
	F151-F16B	Option: TIMEOUT oder WAIT
F136-F13F	F16C-F16F	Ausgabe: DEVICE NOT PRESENT
F140-F155	F170-F184	Lesetakt, löscht Steuerleitungen
F156-F163	F185-F192	Sendet File-Messages
F164-F16E	F193-F19D	Sendet Byte, löscht Steuerleitung
F16F-F17E	F19E-F1AD	Sendet IEEE-Zeichen
F17F-F18B	F1AE-F1BF	Schaltet IEEE-Device ab
F18C-F1D0	F1C0-F204	Holt Byte vom IEEE-Bus (INPUT)
F1D1-F1E0	F205-F214	Holt ein Byte (GET)
F1E1-F231	F215-F265	Holt ein Byte (INPUT)
F232-F26D	F266-F2A1	Gibt ein Byte aus
F26E-F283	F2A2-F2B5	Verläßt alle Files (kein CLOSE)
F284-F28C	F2B6-F2C0	Restore benutzte I/O Devices
F28D-F2A8	F2C1-F2DC	Sucht nach File-Daten
F2A9-F300	F2DD-F334	Generiert CLOSE
F301-F30E	F335-F342	Abfrage: RUN/STOP-Taste
F30F-F314	F343-F348	Bearbeitet RUN/STOP-Taste
F315-F31C	F349-F350	Prüft Direktmodus und gibt Meldung aus

Basic 3.0	Basic 4.0	Beschreibung
F31D-F321	F351-F355	Test, ob Direktmodus
F322-F3C1	F356-F400	Programmlade-Unterprogramm
F3C2-F409	F401-F448	Generiert LOAD
F40A-F42D	F449-F46C	Ausgabe: SEARCHING
F42E-F43D	F46D-F47C	Ausgabe: LOADING oder VERIFYING
F43E-F45F	F47D-F4A4	Holt LOAD/SAVE-Parameter
F466-F493	F4A5-F4D2	Sendet NAME zum IEEE-Bus
F494-F4B6	F4D3-F4F5	Findet speziellen Tape-Header
F4B7-F4CD	F4F6-F50C	Generiert VERIFY
F4CE-F50D	F50D-F55F	Holt OPEN/CLOSE-Parameter
F521-F5A5	F560-F5E4	Generiert OPEN
F5A6-F5D9	F5E5-F618	Findet nächsten Tape-Header
F5DA-F63B	F619-F67A	Schreibt Tape-Header
F63C-F655	F67B-F694	Holt Start/End-Adr. des Headers
F656-F66B	F695-F6AA	Setzt Buffer-Adresse
F66C-F683	F6AB-F6C2	Setzt Buffer-Start- und End-Adr.
F684-F68C	F6C3-F6CB	Generiert SYS
F68D-F69D	F6CC-F6DC	Setzt Tape-Write (Start und Endadr.)
F69E-F728	F6DD-F767	Generiert SAVE
F729-F76F	F768-F7AE	Stellt Uhr nach
F770-F7BB	F7AF-F7FD	Ordnet INPUT-Device zu
F7BC-F805	F7FE-F84A	Ordnet OUTPUT-Device zu
F806-F811	F84B-F856	Tape-Buffer-Pointer betätigen
F812-F834	F857-F879	Wartet, bis PLAY gedrückt
F835-F846	F87A-F88B	Testet, ob Recorder angeschaltet
F847-F854	F88C-F899	Wartet, bis RECORD und PLAY gedrückt
F855-F885	F89A-F8CA	Initial. Tape-Read (liest 192 B)
F886-F89A	F8CB-F8DF	Initial. Tape-Write (schr. 192 B)
F89B-F8E5	F8E0-F92A	Tape-I/O-Routine
F8E6-F8EF	F92B-F934	Wartet auf norm. Interrupt (wenn I/O beendet)
F8F0-F8FF	F935-F944	Prüft, ob STOP-Taste gedrückt
F900-F930	F945-F975	Read-Timing-Subroutine
F931-FA56	F976-FA9B	Liest Bits zum Tape
FA57-FB75	FA9C-FBBA	Liest Zeichen vom Tape
FB76-FB7E	FBBB-FBC3	Löscht Tape-Read-Addresse
FB7F-FB83	FBC4-FBC8	Error-Flag nach ST
FB84-FB92	FBC9-FBD7	Löscht Zähler für neues Byte
FB93-FBAE	FBD8-FBF3	Schreibt ein Bit auf Band
FBAF-FC40	FBF4-FC85	Tape-Write-Routine
FC41-FC7A	FC86-FCBF	Schreibt Tape-Leader
FC7B-FC95	FCC0-FCDA	Ende der Tape-Aufz., Interrupt-Vektor rückspeichern
FC96-FCA5	FCDB-FCEA	Setzt Interrupt-Vektor
FCA6-FCB3	FCEB-FCF8	Schaltet Recorder-Motor aus
FCB4-FCC5	FCF9-FD0A	Überprüft die Prüfsumme
FCC6-FCD0	FD0B-FD15	Erhöht LOAD/SAVE-Pointer
FCD1-FCFD	FD16-FD4B	Power-On-Reset
FD01-FD10	FD4C-FD5C	Tabelle der Interrupt-Vektoren
FD11-FFB0	D472-D716	Maschinensprache-Monitor (TIM)

Sprungtabelle:

	FF93-FF9E	CONTACT, DOPEN, DCLOSE, RECORD
	FF9F-FFAA	HEADER, COLLECT, BACKUP, COPY
	FFAB-FFB6	APPEND, DSAVE, DLOAD, CATALOG
	FFB7-FFBC	RENAME, SCRATCH
	FFBD	Holt Disk-Status
FFC0	FFC0	OPEN
FFC3	FFC3	CLOSE
FFC6	FFC6	Setzt INPUT-Device
FFC9	FFC9	Setzt OUTPUT-Device
FFCC	FFCC	Restored benutzte I/O-Devices
FFCF	FFCF	INPUT ein Byte
FFD2	FFD2	OUTPUT ein Byte
FFD5	FFD5	LOAD
FFD8	FFD8	SAVE
FFDB	FFDB	VERIFY

Basic 3.0	Basic 4.0	Beschreibung
FFDE	FFDE	SYS
FFE1	FFE1	Testet Stop-Taste
FFE4	FFE4	Holt ein Byte
FFE7	FFE7	Verläßt alle Files (kein CLOSE)
FFEA	FFEA	Stellt Uhr nach
FFFA-FFFF	FFFA-FFFF	ROM-Vektoren: NMI, RESET, INT

Adressen, die unter der Rubrik Basic 3.0 nicht erscheinen, sind nur für Disketten-Operationen bestimmt.
Bei Verwendung von Betriebs-Subroutinen ist darauf zu achten, daß die erste angegebene Adresse nicht immer mit der Einstiegsadresse übereinstimmen muß.

Beispiel:

CA39 (BB3A) Ausgabe eines Zeichens
Um diese Routine zu benutzen, ist folgendes notwendig:
1. Laden des Akkus (direkt oder indirekt) mit dem auszugehenden Wert.
2. Ansprung der Routine nach Adresse CA45 (BB46)

Aber:

CA1C (BB1D) Ausgabe eines Strings
Hier ist folgendes erforderlich:
1. Akku mit Lower-Order-Byte der Adresse des Strings laden.
2. Y-Register mit Higher-Order-Byte der Adresse des Strings laden.
3. Aufruf der Routine durch CA1C (BB1D).

(Der String muß mit 00 enden!)

Beispiel-Programme:

1. Ausgabe von 256 Byte

033A	A2 00	LDX # $ 00	Schleifenzähler
033C	BD 00 10	LDA $ 1000,X	Byte in Akku laden
033F	20 45 CA	JSR $ CA45	Ausgabe eines Bytes
0342	E8	INX	Schleifenzähler + 1
0343	D0 F7	BNE $ 033C	Schleifenzähler = 0?
0345	60	RTS	Zurück nach Basic

Die Bytes der Adressen 1000 bis 10FF werden auf den Bildschirm ausgegeben.

2. Ausgabe eines Strings

033A	A9 00	LDA #$ 00	Akku mit L-Byte laden
033C	A0 10	LDY #$ 10	Y-Register mit H-Byte
033E	20 1C CA	JSR $ CA1C	Ausgabe d. adress. Str.
0341	60	RTS	Zurück nach Basic

1000	0D 43 42 4D 20 42 41 53
1009	49 43 00 AA AA AA AA AA

Die Bytes der Adressen 1000 bis 100A werden als String ausgegeben (0D = Line-Feed, AA = Inhalt nach dem Einschalten; nicht wichtig

☐ Weitere Cross-Assembler für den 6809 und Z-8 sind verfügbar.
☐ Ein schneller und leistungsfähiger Basic-Compiler wird angeboten. Er erlaubt lange Variablen-Namen, optionale Zeilennummern, if-then-else-Strukturen, INCLUDE-Files (d. h. das Einfügen von getrennten Source-Texten nach Anweisung im Programm), virtuelle Arrays, getrennte Übersetzung von Basic-Unterprogrammen und ist P-Code-kompatibel mit dem Pascal und dem Fortran 77.
☐ Pascal, Fortran, Basic sowie Assembler-Code können zusammengelinkt (verbunden) werden.
☐ Implementierung für die Prozessoren 8086/8087 und 6800 sind angekündigt für Ende 1981.
Es wird auch angekündigt, daß künftig die Umwandlung von P-Code in den eigenen Maschinencode der jeweiligen Prozessoren möglich sein soll, um die endgültig ausgetesteten Programme noch schneller zu machen. Eine Benutzergruppe, der man zum Jahresbeitrag von 20 $ beitreten kann, gibt es unter der folgenden Adresse:
USUS, Chip Chain Secretary,
P.O. Box 1148, La Jolla,
Ca. 92038.

Literatur

[1] *Gilbreath, J.*:A High-Level Language Benchmark. Byte 1981, Heft 9, S. 180.
[2] Softech Microsystems: UCSD-Pascal. Users Manual. Softech Microsystems, 9494 Black Mountain Road, San Diego, CA 92126, USA.
[3] *Bowles, K.L.*: Microcomputer Problem Solving Using Pascal. Springer-Verlag, Berlin, Heidelberg, New York.
[4[*Bowles, K.L.*: Beginners Guide for the USCD-Pascal System. McGraw-Hill, New York.
[5] *Jensen, K., Wirth, N.*: Pascal User Manual and Report. Second edition. Springer-Verlag, Berlin, Heidelberg, New York.

Schnelle Stichwortsuche beim PET

Im Heft 22/1980 der FUNKSCHAU wurde ein Maschinenprogramm für den CBM vorgestellt, das einen für Suchvorgänge interessanten Befehl implementierte, die „INSTRING"-Funktion [3]. Diese neue Funktion gestattet in Verbindung mit dem in 1/1980 vorgestellten Editor eine schnelle Stichwortsuche [4]. Nachfolgend findet sich eine Version, die auf dem alten Betriebssystem des PET lauffähig ist (*Bild 1*).
Die Funktion „INSTRING" bietet auf elegante Weise die Möglichkeit, String-Arrays nach mehreren Schlüsselworten zu durchsuchen. Nur wenn alle Schlüsselworte im String enthalten sind, wird dieser ausgegeben.
Damit kann man z. B. ein Literaturverzeichnis all jener Veröffentlichungen erstellen, die sich mit einem bestimmten Thema befassen. Werden nur wenige Schlüsselworte angegeben, ist der Literaturnachweis möglicherweise sehr groß. Mit zunehmender und damit einschränkender Anzahl von Schlüsselworten schrumpft die Liste auf die wirklich das Problem treffenden Beiträge zusammen. Die Implementierung der Schlüsselwortsuche in das Texteditor-Programm zeigt der Programmausschnitt in *Bild 2*. Ohne Änderung sind damit bis zu 11 Schlüsselworte verarbeitbar. Da die Suche innerhalb eines Strings sofort abgebrochen wird, wenn eines der Schlüsselworte nicht vorkommt, erhöht sich die Suchzeit nur unwesentlich. Das Anpassen des Maschinen-Programms an das Betriebssystem des PET war dank der im Sonderheft „Mikrocomputer-Anwendungen" abgedruckten Übersicht über die Betriebssystem-Routinen von PET und CBM ohne allzu große Probleme möglich [5]. *Peter Weber*

```
270 J=-1:I=J:REM SUCHROUTINE
280 PRINT"SUCHBEGRIFFE: (MIT 'END' ABSCHLIESSEN)
290 I=I+1:INPUTB$(I):IFB$(I)<>"END"ANDI<10THEN290
300 I=I-1
310 J=J+1:IFA$(J)="END"THENPRINT:RETURN
320 FORII=0TOI:S=@INST(A$(J),B$(II)):IFS=0THENII=I
330 NEXT:IFS=0THEN310
340 PRINTA$(J)
350 GETE$:IFE$<>" "THEN310
360 PRINT"■ F=FORTSETZUNG,S=STOP■"
362 GETE$:IFE$<>"S"ANDE$<>"F"THEN362
364 IFE$="F"THEN310
366 RETURN
READY.
```

Bild 2. So erfolgt die Einbindung der neuen Suchfunktion in den Texteditor, den die FUNKSCHAU in Heft 22/1980 veröffentlichte

```
100 PRINT"BITTE WARTEN
110 X=7821:Y=PEEK(135)-32:X=256*Y+X
120 FORI=0TO370
130 READA:B=B+A:IFA<0THENA=Y-A
140 POKE(X+I),A
150 NEXTI
160 IFB<>45619THENPRINT"□□□□ACHTUNG■: EINGABE FEHLER":END
170 SYS(7821+256*Y)
180 PRINT"□□□□□INSTRING-FUNKTION JETZT AKTIV
190 REM
1000 DATA 169,141,133,134,169,-30,133,135,169,076,133,203,169,162,133,204,169
1010 DATA -30,133,205,096,201,064,240,036,201,058,176,247,076,207,000,230,201
1020 DATA 208,002,230,202,096,165,201,208,002,198,202,198,201,096,032,173,-30
1030 DATA 208,003,032,180,-30,160,000,177,201,096,164,202,192,002,240,214,169
1040 DATA 000,133,001,133,193,032,194,-30,201,178,240,003,076,028,206,032,167
1050 DATA 204,165,008,072,165,150,133,160,165,151,133,170,032,173,-30,160,005
1060 DATA 032,194,000,217,242,-31,208,226,136,208,245,032,189,-30,032,040,206
1070 DATA 032,169,204,032,223,-31,160,000,177,150,133,002,200,177,150,141,122
1080 DATA -31,141,157,-31,200,177,150,141,123,-31,141,158,-31,032,017,206,032
1090 DATA 040,206,032,169,204,032,223,-31,160,000,177,150,133,165,200,177,150
1100 DATA 141,119,-31,141,152,-31,200,177,150,141,120,-31,141,153,-31,032,197
1110 DATA -30,201,041,240,039,201,044,240,003,076,028,206,032,173,-30,032,184
1120 DATA 204,032,167,204,032,171,208,165,180,133,001,197,002,144,002,176,060
1130 DATA 032,197,-30,201,041,240,003,076,028,206,166,001,173,048,004,221,034
1140 DATA 004,240,009,232,228,002,208,243,162,000,240,029,134,161,169,000,133
1150 DATA 171,230,161,230,171,164,171,196,165,240,012,185,048,004,164,161,217
1160 DATA 034,004,208,221,240,234,232,134,193,165,160,133,150,165,170,133,151
1170 DATA 104,016,014,160,000,169,000,145,150,200,165,193,145,150,076,212,-31
1180 DATA 169,000,133,177,164,193,132,178,162,144,056,032,027,219,166,150,164
1190 DATA 151,032,166,218,104,104,104,104,032,173,-30,076,131,198,096,224,000
1200 DATA 208,251,162,008,189,247,-31,032,210,255,202,208,247,160,040,076,123
1210 DATA 245,040,084,083,078,073,071,078,073,082,084,083,063,013
```

Bild 1. Dieses Basic-Programm initialisiert ein Maschinenprogramm, das beim alten PET-2001-Betriebssystem das Suchen nach praktisch beliebig vielen Begriffen in Strings zuläßt

Literatur

[1] Basic-Texteditor, FUNKSCHAU 1980, Heft 1.
[2] Basic-Texteditor. FUNKSCHAU 1980, Heft 22.
[3] *Martin, Reinhold:* Schnelle Stichwortsuche beim CBM, FUNKSCHAU 1980, Heft 22.
[4] Schnelles Suchen beim Texteditor. FUNKSCHAU 1980, Heft 23.
[5] *Martin, Reinhold; Smode, Dieter:* ROM und RAM bei PET und CBM. Sonderheft „Mikrocomputer-Anwendungen", Franzis-Verlag.

Werner Lang

AIM-65 decodiert DCF 77

Das folgende Programm ermöglicht es, unter Verwendung eines geeigneten Empfängers für den Zeitzeichensender DCF 77 die ausgestrahlte amtliche Zeit auf dem Display des Mikrocomputers KC-100 bzw. AIM-65 zur Anzeige zu bringen. Ferner ist die Programmierung acht verschiedener Alarmzeiten möglich.

```
(↑)=0000 20 13 00 A9 00 85 F6 8D 03 A0 8D 0C A0 8D 0D A0
( ) 0010 4C 29 00 A9 00 85 F9 85 FB 85 FC 85 FD 85 FE A9
( ) 0020 99 85 FA A9 00 85 F8 60 A9 01 8D 0D A0 2C 0D A0
( ) 0030 F0 FB A9 00 85 FD A9 80 8D 09 A0 A9 20 2C 0D A0
( ) 0040 F0 FB E6 FD A5 FD C9 20 D0 EC A9 80 2C 01 A0 F0
( ) 0050 D7 A9 01 8D 0D A0 2C 0D A0 F0 FB E6 F9 18 F8 A9
( ) 0060 01 65 FA 85 FA D8 A5 F9 C9 01 D0 03 20 00 04 A5
( ) 0070 F9 C9 15 30 03 4C 7E 00 20 3F 03 4C 51 00 A5 F9
( ) 0080 C9 3B D0 06 20 3F 03 4C 29 02 20 3F 03 A9 00 85
( ) 0090 FD A9 20 8D 09 A0 2C 0D A0 F0 FB E6 FD A5 FD C9
( ) 00A0 06 D0 EE A9 80 2C 01 A0 F0 03 4C 5E 03 A9 00 85
( ) 0080 FD A9 20 8D 09 A0 2C 0D A0 F0 FB E6 FD A5 FD C9
( ) 00C0 0C D0 EE A4 FC C0 04 30 04 A9 05 85 F8 A5 FB C5
( ) 00D0 F8 F0 03 4C 00 02 4C 0F 02 00 FF 00 FF 00 FF FF
(↑)=0200 E6 FB AD 01 A0 29 80 05 FE 4A 85 FE 4C 51 00 AD
( ) 0210 01 A0 29 80 05 FE 49 FF A6 FC 95 F0 E6 FC A9 00
( ) 0220 85 FE A9 01 85 FB 4C 51 00 A9 00 85 FD A9 50 8D
( ) 0230 08 A0 A9 C3 8D 09 A0 A9 20 2C 0D A0 F0 FB E6 FD
( ) 0240 A9 14 C5 FD D0 E7 18 F8 A9 01 65 FA 85 FA D8 20
( ) 0250 3F 03 A9 A0 8D 09 A0 A9 20 2C 0D A0 F0 FB A9 80
( ) 0260 2C 01 A0 F0 0A A9 01 85 F6 20 13 00 4C 72 02 4C
( ) 0270 5E 03 A5 F0 29 0F 8D CB 03 A5 F0 29 70 20 1E 03
( ) 0280 8D CC 03 A5 F1 29 0F 8D CD 03 A5 F1 29 30 20 1E
( ) 0290 03 8D CE 03 A5 F1 2A 2A 29 01 85 F5 A5 F2 0A 29
( ) 02A0 0F 85 F5 8D CF 03 A5 F2 29 18 20 1F 03 8D 00 03
( ) 02B0 A5 F2 29 E0 20 1D 03 8D 05 03 A5 F3 29 0F 8D 01
( ) 02C0 03 A5 F3 29 10 20 1E 03 8D 02 03 A5 F4 29 08 85
( ) 02D0 F5 A5 F3 20 1D 03 05 F5 8D 03 03 A5 F4 20 1E 03
( ) 02E0 8D 04 03 A2 00 BD CB 03 09 30 9D CB 03 E8 E0 0A
( ) 02F0 D0 F3 A0 00 C8 CC D5 03 D0 FA 98 0A A8 B9 A6 03
( ) 0300 8D D5 03 C8 B9 A6 03 8D D6 03 A2 00 BD CB 03 BC
( ) 0310 9C 03 99 D7 03 E8 E0 0C D0 F2 4C 51 00 4A 4A 4A
( ) 0320 4A 4A 60 A9 01 25 F6 D0 01 60 A2 00 86 F7 BD D7
( ) 0330 03 09 80 20 7B EF E6 F7 A6 F7 E0 12 D0 F0 60 A9
( ) 0340 01 25 F6 D0 01 60 A5 FA 20 1E 03 09 B0 A2 12 20
( ) 0350 7B EF A5 FA 29 0F 09 B0 A2 13 20 7B EF 60 20 13
( ) 0360 EA A2 00 BD B6 03 20 7A E9 E8 E0 05 D0 F5 4C 00
( ) 0370 00 20 13 EA A2 00 BD BB 03 20 7A E9 E8 E0 10 D0
( ) 0380 F5 A9 2E 8D 09 03 8D DC 03 A9 3A 8D E5 03 8D E8
```

```
(↑)=0390 03 A9 20 8D DF 03 8D E2 03 4C 00 00 10 0F 0D 0C
( ) 03A0 01 00 04 03 07 06 09 0A 4D 4F 44 49 4D 49 44 4F
( ) 03B0 46 52 53 41 53 4F 45 52 52 4F 52 50 4C 45 41 53
( ) 03C0 45 20 57 41 49 54 2E 2E 2E 2E 2E 00 FF 10 7F 00
(↑)=0400 20 23 03 A0 00 C8 C0 09 F0 25 B9 34 04 8D 13 04
( ) 0410 A2 00 BD CE 04 DD D7 03 D0 EB E8 E0 11 D0 F3 A9
( ) 0420 C0 8D 0B A0 A9 80 8D 02 A0 A9 04 8D 05 A0 60 A9
( ) 0430 00 8D 08 A0 60 50 62 74 86 98 AA BC CE BD FB F7
```

Bild 1. Hex-Listing des Programms. Die Prüfsummen der einzelnen Teile sind: FROM 0 TO D9 [illegible]; FROM 200 TO 3CB = 38C3; FROM 400 TO 43D = 1C51 (vgl. mc 2/1981, S. 36)

Da sich die Funkempfänger für den Zeitzeichensender DCF 77 (77,5 kHz) immer größerer Beliebtheit erfreuen und man selbst von diesem Fieber nicht ganz verschont bleibt, liegt es nahe, deren Funktionen, soweit es sich um die digitale Signalverarbeitung handelt, von seinem Mikroprozessorsystem übernehmen zu lassen.

Im vorliegenden Fall ist das Programm (*Bild 1*) hierfür zwar für den AIM-65 geschrieben, es läßt sich aber ohne größere Schwierigkeiten auch auf andere 6502-Systeme umschreiben, insbesondere, wenn sie einen User-VIA 6522 besitzen, da bis auf die Interruptflag-Abfrage, die Anzeige- und Timerroutinen keine speziellen betriebssystemspezifischen Programmteile benutzt werden. Die wenigen genannten kann man sich z. B. mit Hilfe der in [1] tabellierten Monitor-Unterprogramme an sein System anpassen.

Der Empfänger für 77,5 kHz

Der AIM-65 mag zwar Erstaunliches leisten, aber ganz ohne einen geeigneten Empfänger, der das amplitudenmodulierte Langwellensignal demoduliert und mit TTL-Pegel bereitstellt, geht es nicht. Doch sollte man sich von dieser kleinen Hürde nicht davon abhalten lassen, das Programm in Betrieb zu nehmen. Wenn man nicht allzu hohe Anforderungen an die Zuverlässigkeit stellt, läßt sich solch ein Empfänger mit nur geringem Aufwand aufbauen. So z. B. verwendet der Verfasser mit Erfolg eine Schaltung nach eigenem Entwurf, die sich für ca. 15 DM Bauelementekosten realisieren läßt. Für eine hohe Betriebssicherheit sollte man aber schon einen Empfänger mit Quarzfilter vorsehen, da sich sonst vor allem Probleme mit der 5. Harmonischen der Fernsehzeilenfrequenz ergeben können [2]. Wer den Selbstbau scheut, kann einen entsprechenden Schaltungsaufbau auch erwerben. Solch ein Empfänger wird z. B. von der Firma Völkner (Braunschweig) angeboten, bei dem man allerdings noch einen Demodulator nachschalten muß. Wichtig ist, daß das vom Empfänger abgegebene Digitalsignal der Hüllkurve

des Sendesignals entspricht. Der Sekundenbeginn ist folglich durch eine abfallende Flanke und die Trägerabsenkung durch den TTL-Pegel Low gekennzeichnet. Ist dies bei dem verwendeten Empfänger umgekehrt, kann man zwar auch durch ein paar kleine Veränderungen im Programm zu Erfolg kommen, einfacher und schneller dürfte aber eine Invertierung des Signals z. B. durch ein 7400-Gatter sein.

Das Programm in 6502-Maschinensprache

Die vom Zeitzeichensender seriell im BCD-Code ausgestrahlte Information wird einschließlich der Prüfbits innerhalb der 21. und 58. Sekunde eines Minutendurchgangs übermittelt. Dabei handelt es sich jeweils um die Daten der nächstfolgenden Minute. Somit hat man, ganz gleich ob z. B. ein Schaltjahr oder der Wechsel zur Sommerzeit vorliegt, immer die exakte Zeit.
Nachdem durch die fehlende Trägerabsenkung zur 59. Sekunde der bevorstehende Minutenwechsel erkannt wurde, wird ab der 21. Sekunde jeweils 150 ms nach der negativen Flanke der Port PA7 abgefragt und dessen Zustand dem Zwischenspeicher FE übergeben. Erst nachdem dieser nach acht Abfragen aufgefüllt ist, wird sein Inhalt in einem weiteren Speicher ab F0 abgelegt, um die nachfolgenden Datenbits aufnehmen zu können. Die Decodierung der formlos aneinandergereiht abgespeicherten Einlesewerte vom BCD-Code in die zur Anzeige notwendige Form wird vor jedem Minutenwechsel mittels einer Maskenabfrage und einer einfachen Umcodierungsvorschrift in den ASCII-Code vorgenommen. Sodann werden die Zeichen durch eine Umordnungstabelle in das gewünschte Anzeigeformat gebracht und dem Auslesespeicher übergeben, von wo sie zu Beginn der neuen Minute das Display übernimmt.
Die Anzeige der Sekunden geschieht durch dezimales Hochzählen nach jeder negativen Flanke und entsprechender Umcodierung in das ASCII-Format. Lediglich nach der 58. Sekunde muß ein Timer gestartet werden, der die 59., nicht ausgesendete Sekunde festlegt.
Zu jeder neuen Minute wird der Speicherbereich für die Alarmzeiten abgefragt und abhängig vom Vergleichsergebnis für die Dauer einer Minute ein Alarmton erzeugt.
Zur Fehlererkennung von gestörten Signalen ist während der Dateneinlesephase jeweils etwa 50 ms nach der abfallenden Flanke eine Abfrage des Ports vorgesehen. Ist der Signalzustand High, muß ein Fehler vorliegen. Ein durch eine Störung ausgebliebener Sekundenimpuls wird spätestens nach einer Minute erkannt, wenn nämlich die dann nicht vorhandene Synchronität zwischen dem tatsächlichen und dem vermeintlichen Minutenwechsel wiederum durch eine Portabfrage hervorgeht. In beiden Fällen gibt AIM-65 eine Fehlermeldung aus und wartet auf einen neuen Minutenanfang, um seinen Einlesezyklus fortzusetzen.
Der Verfasser hat durch eine Reihe von simulierten Störungen ermittelt, daß durch diese Fehlererkennungsmethode ein unkorrektes Empfangssignal mit großer Wahrscheinlichkeit erkannt wird.
Auf eine Auswertung der Prüfbits wurde bewußt verzichtet, da gestörte Signale hierdurch nur mit einer relativ geringen Sicherheit zu analysieren wären.
Anwender, die nur die 1-KByte-Version des AIM besitzen, müssen auf die Möglichkeit der Alarmzeiteingabe verzichten. Für sie endet das Programmlisting bei 03CA. Es ist lediglich bei 006C die Unterprogrammadresse von 0400 auf 0323 zu ändern.
Das komplette Programm benötigt außer den aus dem Hexlisting hervorgehenden Speicherplätzen die RAM-Bereiche:
00F0...00FE
03CB...03E8
0450...04E0

Anschluß und Bedienung der Alarmeinrichtung

Das vom Empfänger kommende TTL-Signal wird auf Port PA7 (J1, Pin 8) geführt. Zudem ist PA7 mit CA2 (J1, Pin 21) zu verbinden.
Zur Erzeugung des Alarmtons ist ein Kleinlautsprecher über PB7 (J1, Pin 15) anzusteuern (*Bild 2*). Zu empfehlen ist ein Schalter, mit dem er sich vorzeitig abschalten läßt.
Nach dem Start mit F1 bei 0371 bittet der AIM 65, zu warten, denn er braucht wie alle Funkuhren zunächst etwa zwei Minuten Zeit, um die gesendeten Daten einzulesen, und meldet sich dann prompt mit der empfangenen Zeitinformation.
Falls von der Möglichkeit des Alarmtons Gebrauch gemacht werden soll, sind die entsprechenden Zeiten mit dem Text-Editor vor dem Start ab 0450 nacheinander zeilenweise einzugeben. Dabei ist streng darauf zu achten, daß das Eingabeformat mit dem Anzeigeformat genau übereinstimmt. Auch die beiden Leerplätze vor und nach dem Wochentag sind einzutippen. Lediglich nach dem Minuten-Einer folgt bereits das Zeilenendzeichen Return, da die Alarmzeiten nur minutengenau zu programmieren sind. Ein Beispiel soll dies verdeutlichen: 22.05.81 FR 19:35...
Die zeitliche Ordnung ist dabei uninteressant; ebenso die Anzahl der Alarmzeiten. Sind mehr als acht Zeiten eingegeben, werden die restlichen einfach ignoriert.

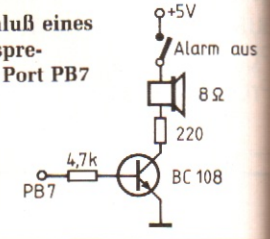


Bild 2. Anschluß eines kleinen Lautsprechers an den Port PB7

Literatur

[1] ROM und RAM in KIM und AIM, in: „Mikrocomputer-Anwendungen", Franzis-Sonderheft Nr. 33.
[2] Mikrocomputergesteuerte Funkuhr im Mini-Format. FUNKSCHAU 1979, Heft 14, S. 839.

Video-Computer-System

Texas Instruments hat seinen Computer TI99/4 aufgemöbelt und eine Version A daraus gemacht. Hervorstechendstes Merkmal dieses neuen Computersystems ist die Schreibmaschinentastatur. Daran ist jetzt auch ein größeres Programmierprojekt ermüdungsfrei zu erledigen. Innen sitzt der bewährte Prozessor 9900. In der Grundversion wird der Computer mit 16 KByte RAM, 26 KByte ROM mit Betriebssystem, Interpreter für die Grafiksprache GPL, 4 KByte Basic-Interpreter (TI-Basic-ANSI-Min), 4 KByte Monitor für 9900-Maschinensprache geliefert. Der Video-Ausgang liefert jetzt ein PAL-Signal, also das Signal, das jeder Farbfernseher an seiner AV-Buchse in unserem Land erwartet. R

Der neue Texas-Computer mit Schreibmaschinentastatur

Herwig Feichtinger

Kreuzkorrelation per Programm

Im letzten Heft haben wir bereits eine sehr effiziente Methode zur Erkennung gestörter, verrauschter Signale im Tonfrequenzbereich kennengelernt: die Autokorrelation. Hier folgt nun ein noch leistungsfähigeres Verfahren zur Tonerkennung per Software, das auf der Kreuzkorrelation beruht.

Während die Autokorrelation [1] das digitalisierte Eingangssignal zeitverschoben mit sich selbst vergleicht, wird bei der sogenannten Kreuzkorrelation ein empfangsseitig erzeugtes Signal bzw. Bitmuster als Referenz verwendet. Dies hat den Vorteil, daß das Referenzsignal nicht schon ebenso gestört ist wie das Eingangssignal. Deshalb lassen sich auch stark verrauschte Signale mit der Kreuzkorrelation noch erkennen, bei denen die Autokorrelation längst versagt. Außerdem fällt die Kreuzkorrelation nicht auf Harmonische, also ganzzahlige Vielfache der Eingangsfrequenz herein.

Die Phasenlage interessiert nicht

Um die Übereinstimmung des Referenzsignals mit dem Eingangssignal prüfen zu können, müßte die Phasenlage des Eingangssignals bekannt sein, was normalerweise nicht der Fall ist. Während bei einer reinen Hardware-Kreuzkorrelation zur Vermeidung dieses Problems meist ein sin/cos-Referenzsignal nebst zwei Multiplizierern eingesetzt wird [2], verschiebt die hier vorgestellt Softwarelösung das 8-Bit-Referenzmuster so lange, bis eine optimale Übereinstimmung mit dem digitalisierten Eingangssignal festgestellt wird. Im Beispiel von *Bild 1* ist dazu ein zweimaliges Rotieren des Referenzmusters, das stets genau eine Periode des erwarteten Signals darstellt, erforderlich. Der Referenz-Grundwert hex F0 entspricht dem Bitmuster 11110000, was eine digitalisierte Sinus-Periode bzw. ein symmetrisches Rechtecksignal darstellt.

Parameter leicht änderbar

Das in *Bild 2* als Assembler-Listing abgedruckte 6502-Programm ist für die Adressenbelegung der Computer AIM-65 bzw. PC-100 ausgelegt und verwendet wie [1] deren Kassetten-Port als Eingang, so daß keinerlei zusätzliche Hardware erforderlich ist.
Das Programm zählt einfach innerhalb eines gewissen Zeitintervalls, dessen Dauer in Zelle 0000 einstellbar ist, die Anzahl der mit dem Referenzmuster übereinstimmenden Bits. Diese erreicht natürlich bei der Sollfrequenz, die durch den Inhalt von Zelle 0001 festgelegt ist, ein Maximum. Das Referenz-Bitmuster, normalerweise hex F0, ist in Zelle 0002 zu schreiben, und der Inhalt der Zelle 0003 bestimmt schließlich, wieviele Bytes nacheinander mit dem digitalisierten Eingangssignal aufgefüllt werden, bevor der Test auf Übereinstimmung mit dem Referenzmuster erfolgt. Dies bestimmt im wesentlichen die Bandbreite, hat aber leider wegen der Beeinflussung der Programmlaufzeit auch Einfluß auf die Sollfrequenz. Die Frequenz ergibt sich zu:

$$f = \frac{10^6}{8 \cdot (19 + 11\ \mathrm{LEN} + 5\ \mathrm{FREQ})}\ \mathrm{Hz}$$

Die am Schluß des Listings in Bild 2 angegebene Bytefolge stellt die Parameter auf die Erkennung der bei Funkübertragungen üblichen Ruffrequenz von 1750 Hz ein. Je nach LEN (0003) benötigt das Programm eine entsprechende Zahl von Bytes ab Adresse 000A als Software-Schieberegister für das Eingangssignal.

Programmkern voll relokatibel

Der „Kern" des 6502-Programms, der von 0C00 bis 0C62 reicht, ist in seiner Adressenlage frei verschiebbar. Systemspezifisch ist lediglich die Adresse des Eingangsports (hier A800). Als Ergebnis steht die Zahl aller innerhalb der Meßdauer als gleich erkannter Bits hexadezimal vierstellig in ERGH und ERGL zur Verfügung.
Diese Zahl wird vom letzten Programmabschnitt, der wegen der Adressen von NUMA und CRLF AIM-65/PC-100-spezifisch ist, auf das Display ausgegeben, was hier nur zu Demonstrationszwecken dient. Als Kriterium für das Vorhandensein eines Tones mit der Sollfrequenz kann beim angegebenen Beispiel das Überschreiten eines Wertes von etwa hex 0300 gewertet werden.
Die Empfindlichkeit läßt sich wie bei der Autokorrelation durch Verlängern der Meßdauer in 0000 steigern. Es ist kein Problem, noch Signale zu erkennen, die die gleiche Amplitude wie das störende Rausch-Signal besitzen – ein Beweis, daß die digitale Signalverarbeitung der analogen keineswegs unterlegen sein muß.

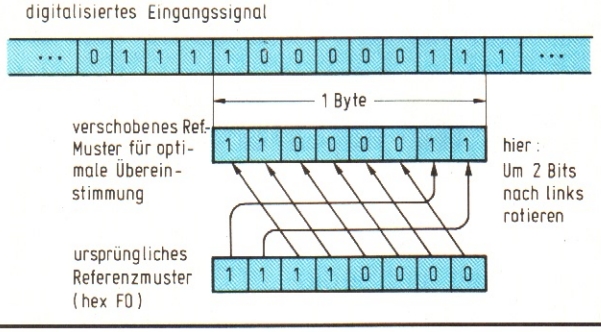


Bild 1. Elimination der Phaseninformation durch Verschieben des Referenz-Bitmusters

Literatur

[1] Tonerkennung per Software. mc 1981, Heft 4.
[2] Kreuzkorrelator zur Messung von Nf-Spektren. ELEKTRONIK 1981, Heft 12.

```
0000            ;KORRELATION MIT
0000            ;REFERENZ-BITMUSTER
0000                    *=0
0000            ;BENUTZERVARIABLE
0000 PER                *=*+1           ;MESSDAUER
0001 FREQ               *=*+1           ;FREQUENZ
0002 REFP               *=*+1           ;REFERENZ
0003 LEN                *=*+1           ;BYTEZAHL
0004            ;PROGRAMMVARIABLE
0004 SUM                *=*+1
0005 CNT                *=*+1           ;ZAEHLER
0006 CNT1               *=*+1
0007 MAX                *=*+1           ;MAX.SUM
0008 ERGL               *=*+1           ;ERGEBNIS
0009 ERGH               *=*+1
000A SMPL               *=$10C          ;SAMPLES
010C
010C            ;AIM-65-ADRESSEN
010C            4C000C  JMP CCOR        ;TASTE F1
010F CRLF               =$E9F0
010F NUMA               =$EA46          ;DISPLAY
010F PB                 =$A800          ;PB7=INPUT
010F                    *=$C00
0C00
0C00            ;PROGRAMMSTART
0C00 CCOR       A900    LDA #0          ;ALLES
0C02            8508    STA ERGL        ;LOESCHEN
0C04            8509    STA ERGH
0C06            A500    LDA PER         ;MESS-
0C08            8506    STA CNT1        ;DAUER
0C0A            ;BITMUSTER LESEN
0C0A REC        A503    LDA LEN
0C0C            0A      ASL A           ;LEN X 8
0C0D            0A      ASL A           ;ERGIBT
0C0E            0A      ASL A           ;BIT-
0C0F            8505    STA CNT         ;ANZAHL
0C11 REC1       A603    LDX LEN
0C13            AD00A8  LDA PB
0C16            2A      ROL A
0C17 REC2       3609    ROL SMPL-1,X
0C19            CA      DEX
0C1A            D0FB    BNE REC2
0C1C            A401    LDY FREQ        ;VER-
0C1E REC3       88      DEY             ;ZOEGERUNG
0C1F            D0FD    BNE REC3
0C21            C605    DEC CNT
0C23            D0EC    BNE REC1
0C25            ;MUSTERVERGLEICH
0C25            A908    LDA #8          ;REF. 8MAL
0C27            8505    STA CNT         ;SCHIEBEN
0C29            4A      LSR A           ;MAX=4
0C2A            8507    STA MAX
0C2C COM0       A502    LDA REFP        ;REF.
0C2E            2A      ROL A           ;ROTIEREN
0C2F            2602    ROL REFP
0C31            A900    LDA #0
0C33            8504    STA SUM         ;SAMPLES
0C35            A603    LDX LEN         ;MIT
0C37 COM1       A008    LDY #8          ;REFER.
0C39            B509    LDA SMPL-1,X    ;VERGL.
0C3B            4502    EOR REFP
0C3D COM2       2A      ROL A           ;GLEICHE
0C3E            B002    BCS COM3        ;BITS
0C40            E604    INC SUM         ;ZAEHLEN
0C42 COM3       88      DEY
0C43            D0F8    BNE COM2
0C45            CA      DEX
0C46            D0EF    BNE COM1
0C48            A504    LDA SUM         ;MAXIMAL-
0C4A            C507    CMP MAX         ;WERT
0C4C            9002    BCC COM4        ;SPEICHERN
0C4E            8507    STA MAX
0C50 COM4       C605    DEC CNT
0C52            D0D8    BNE COM0
0C54            A507    LDA MAX         ;MAX.-WERT
0C56            18      CLC             ;ZUM ER-
0C57            6508    ADC ERGL        ;GEBNIS
0C59            8508    STA ERGL        ;ADDIEREN
0C5B            9002    BCC NOCY
0C5D            E609    INC ERGH
0C5F NOCY       C606    DEC CNT1
0C61            D0A7    BNE REC
0C63            ;AUSGABE AUF AIM-65
0C63            20F0E9  JSR CRLF        ;DISPLAY
0C66            A509    LDA ERGH        ;LOESCHEN
0C68            2046EA  JSR NUMA
0C6B            A508    LDA ERGL        ;ERGEBNIS
0C6D            2046EA  JSR NUMA        ;ANZEIGEN
0C70            4C000C  JMP CCOR
0C73            ;DATEN FUER 1750 HZ
0C73            ;0000: 20 02 F0 04
0C73                    .END
```

Bild 2. Assemblerlisting des AIM-65/PC-100-Demonstrationsprogramms

Jürgen Plate

Suchen und Sortieren in Pascal und Basic

5. Te

Unser Literaturverzeichnis-Programm stellt eine Zusammenfassung des bisher Gelernten dar. Seine Funktionsweise und Bedienung wurde bereits im letzten Heft besprochen; als Abschluß folgen nun noch die kompletten Programmlistings.

Gleichzeitig ist unsere kleine Serie übe
Such- und Sortierverfahren in den zwe
wichtigsten höheren Tischcomputer-
Programmiersprachen beendet; wir hof
fen, es hat Ihnen ein wenig Spaß gem

...gehörige Maschinenprogramm im ...findet, z. B. für SIN. Dazu bedient ...sich einer Tabelle, die ebenfalls im ...steht: Der komprimierte 1-Byte-Ba...befehl dient als Index in diese Tabel... Beim CBM steht die Tabelle ab der ...imalen Adresse 49152, und die ...esse des SIN-Maschinenprogramms ...let die CPU in den beiden Bytes bei ...244 und 49245. Die 16-Bit-Zahl dort ...gibt die Adresse 57311, und dort ...springt der Interpreter hin, um den Si...-Wert des nachfolgenden Ausdrucks ...berechnen.
Das Microsoft-CBM-Basic komprimiert ...die Basic-Befehle; andere Basic-In...preter, z. B. das DAI-8080-Basic, wan...auch Dezimalzahlen in arithmetischen Ausdrücken schon während der Editierphase in Binärzahlen um.
Der CBM muß diese Dezimal-Binärumwandlung während des Programmlaufs tun, was ihn natürlich mehr Zeit kostet. Andere Interpreter, etwa der des HP-85, arbeiten grundsätzlich im BCD-Code, also dezimal, und können daher auf diese Umwandlung ganz verzichten – allerdings unter Inkaufnahme eines größeren Speicherplatzbedarfs für mehrstellige Zahlen.

Der Interpreter erkennt Fehler

Manche Computer, z. B. ZX-80 und ABC-80, erkennen syntaktische Fehler bei der Programmeingabe schon mit dem Editor, d. h. bevor das Programm mit RUN gestartet wird. Der CBM hingegen nimmt die Prüfung auf richtige Syntax erst nach RUN vor. Stünde etwa hinter SIN keine Klammer, so würde sich der Interpreter darüber mit „Syntax Error" beschweren.
Bei all diesem Komfort heutiger Basic-Interpreter sollte man aber nicht vergessen: Erkannt werden stets nur syntaktische Fehler, d. h. Fehler in der Schreibweise von Befehlen, und grobe Fehler in der Programmlogik, etwa der Versuch, ein Variablen-Array zweimal mit DIM zu dimensionieren. Echte logische Fehler merkt der Computer leider nicht – und manchmal auch nicht der Benutzer...

Unterprogramme ohne RETURN

...weilen steht man vor dem Problem, ...einem Unterprogramm an eine be...mmte Zeile des Basic-Hauptpro...gramms direkt zurückspringen zu müs...n, z. B. wegen eines Eingabefehlers. ...man dies statt RETURN mit einem ...TO-Befehl, so stimmt der Stapelzei...(Stackpointer) nicht mehr. Bei ei...folgenden Return-Befehl wird ...lich zurückgesprungen. Der Rück...rung in höhere Schachtelebenen ist ...nicht mehr möglich, falls eine mit ...SUB angesprungene Unterroutine ...einem GOTO verlassen wird.
In Maschinensprache läßt sich das Pro...em auf einfache Weise umgehen, in...man z. B. mit zweimaligem PLA ...Stackpointer wieder korrigiert. Die ...ten Basic-Interpreter sehen diese ...glichkeit aber leider nicht vor. Eine ...Maschinensprache-Erweiterung ...schafft hier Abhilfe.

Die Wirkung des Maschinenprogrammes

...der dem Befehl SYS(826) findet ...exakt die gleichen Verhältnisse ...FOR-NEXT-, GOSUB-Hierarchie) ...hätte das Programm den das zuge...hörige GOSUB beendenden RETURN-...Befehl absolviert.

```
                *
                * PLA PLA FUER BASIC
                * P. HEUMOS  DG2MAZ
                *

                INDEX EPZ 71        $47
                * 2001: 153         $99
                * 8001: 71          $47

                SUCHE EQU 49834     $C2AA
                * 2001: 49836       $C2AC
                * 8001: 45858       $B322

                TITEL EQU 23202     $5AA2
                * 2001: 24482       $5FA2
                * 8001: 23202       $5AA2

                ERROR EQU 50007     $C357
                * 2001: 50009       $C359
                * 8001: 46031       $B3FC

                INTER EQU 50884     $C6C4
                * 2001: 50869       $C6B5
                * 8001: 46922       $B74A

033A: A9FF      LDA #255            ; INITIALISIERT FUER ROUTINE
033C: 8547      STA INDEX
033E: 20AAC2    JSR SUCHE           ; SUCHT ERKENNUNGSBYTE AUF STAPEL
0341: 9A        TXS
0342: C98D      CMP #141            ; ERKENNUNGSBYTE FUER RETURN ?
0344: F008      BEQ RE1
0346: A216      LDX #22             ; NEIN: LADE ZEIGER FUER MELDUNG
0348: 2CA25A    BIT TITEL           ;  'RETURN WITHOUT GOSUB ERROR'
034B: 4C57C3    JMP ERROR           ; DRUCKE MELDUNG, READY

034E: A207    RE1 LDX #7            ; LETZTES RETURN VOM STAPEL NEHMEN
0350: 68      RE2 PLA
0351: CA        DEX
0352: D0FC      BNE RE2
0354: 4CC4C6    JMP INTER           ; SPRUNG IN DIE INTERPRETERSCHLEIFE


INDEX       $47
SUCHE       $C2AA
TITEL       $5AA2
ERROR       $C357
INTER       $C6C4
RE1         $034E
RE2         $0350
```

Bild 1. Maschinenprogramm zur Stack-Korrektur für das Betriebssystem CBM 3001

```
W:033A-0341  A9 FF 85 99 20 AC C2 9A
W:0342-0349  C9 8D F0 08 A2 16 2C A2
W:034A-0351  5F 4C 59 C3 A2 07 68 CA
W:0352-0359  D0 FC 4C B5 C6 00 00 00
```

Bild 2. So sieht der Hex-Dump des PET-2001-Programms aus; es unterscheidet sich geringfügig von der 3001-Version

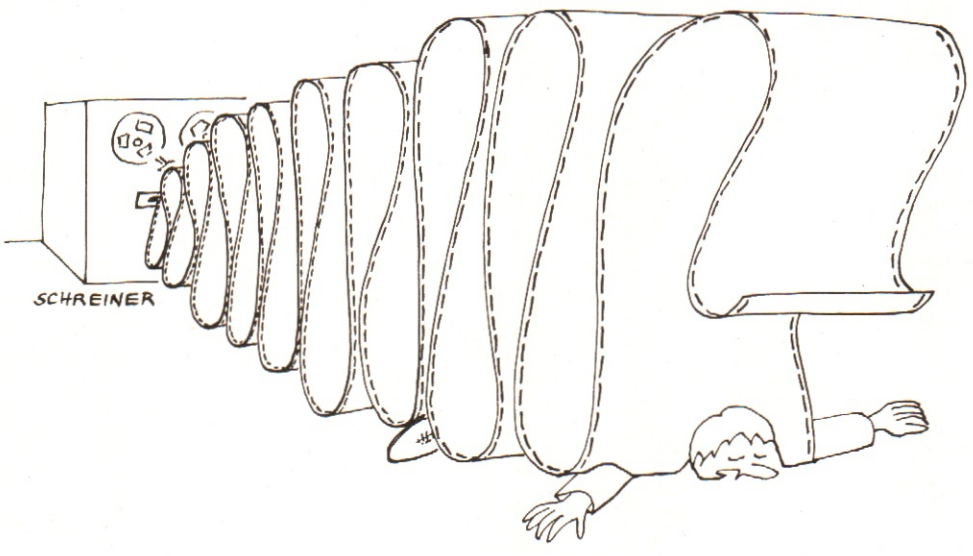
```
W:033A-0341  A9 FF 85 47 20 22 B3 9A
W:0342-0349  C9 8D F0 08 A2 16 2C A2
W:034A-0351  5A 4C FC B3 A2 07 68 CA
W:0352-0359  D0 FC 4C 4A B7 00 00 00
```

Bild 3. Version für das Betriebssystem CBM 4001/8001

```
100 STOP
110 TESTPROGRAMME STARTEN MIT :
120 RUN170
130 RUN200
140 RUN230
150 RUN280
160 :
170 GOSUB180
180 A=A+1:PRINT"A="A:GOTO170
190 :
200 GOSUB210
210 SYS826:A=A+1:PRINT"A="A:GOTO200
220 :
230 GOSUB240: PRINTA:NEXTA:END
240 FORA=0TO10
250 IFA<5THEN PRINTA:NEXTA
260 RETURN
270 :
280 GOSUB290: PRINTA:NEXTA:END
290 FORA=0TO10
300 IFA<5THEN PRINTA:NEXTA
310 SYS826:   PRINTA:NEXTA:END
READY.
```

Bild 4. Die vier Basic-Teilprogramme demonstrieren die hilfreiche Wirkung der Stack-Korrektur mit SYS 826

In *Bild 1* ist die Erweiterung für Computer mit dem Betriebssystem CBM 3001 disassembliert aufgelistet. *Bild 2* enthält als Hex-Dump die entsprechende Version für den PET 2001, *Bild 3* für CBM 4001 und CBM 8001. Die Unterschiede ergeben sich aus unterschiedlichen Adressenbelegungen. Zu beachten ist, daß bestimmte Basic-Befehle des CBM 4001/8001 den Kassettenbuffer verwenden, in dem das Maschinensprache-Programm steht; sollten sich Schwierigkeiten ergeben, muß man das Programm in einen anderen Speicherbereich, z. B. geschützt ans RAM-Ende legen, wobei keine Veränderungen erforderlich sind. *Bild 4* zeigt schließlich ein Basic-Programm, das einige Testbeispiele enthält. RUN 170 demonstriert den Stack-Überlauf beim Rücksprung mit GOTO. RUN 230 bei falsch abgeschlossenen FOR-NEXT-Schleifen. RUN 200 und RUN 280 zeigen, wie der Einsatz der Maschinensprache-Erweiterung das Problem löst. *Paul Heumos*



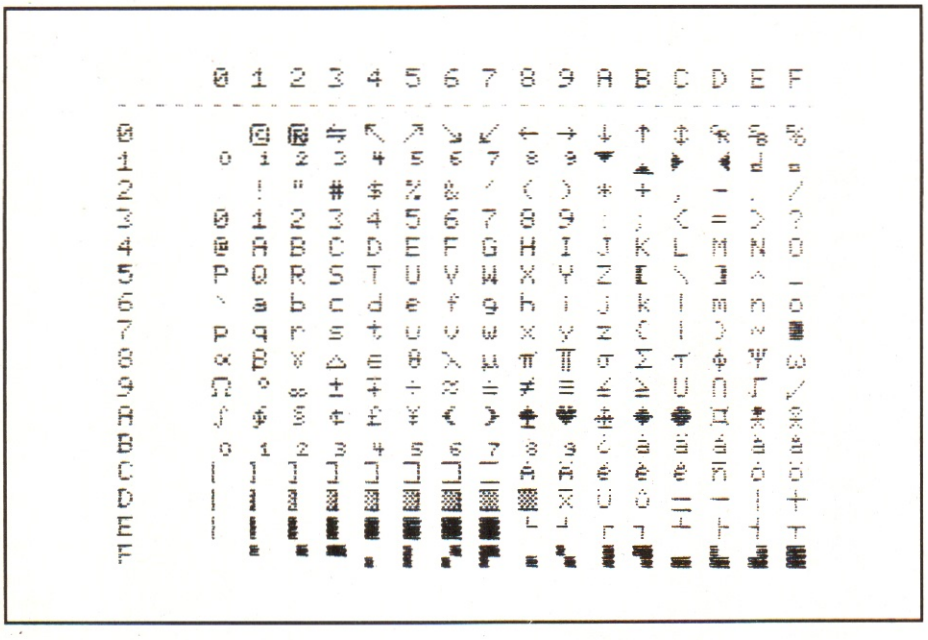
Herwig Feichtinger

Ein würdiger Nachfolger

Rockwell's AIM-65/40

Der Mikrocomputer AIM-65 hat sich vor allem als preiswertes 6502-Entwicklungssystem einen Namen gemacht. Viele seiner Beschränkungen wurden nun in einer größeren (und leider dreimal teureren) Version behoben: Der AIM-65/40 besitzt ein vierzigstelliges Display, einen breiteren Drucker und drei Mikroprozessoren.

Der AIM-65/40 soll die Lücke zwischen dem preiswerten Entwicklungssystem AIM-65, das inzwischen auch von mehreren OEM-Firmen als Komplettgerät im Gehäuse angeboten wird, und dem teureren „System 65" füllen. Gegenüber dem bekannten AIM-65 (Siemens vertreibt ihn als PC-100-Kit) besitzt die neue Version ein vierzigstelliges, grünleuchtendes 16-Segment-Fluoreszenz-Display mit Dezimalpunkten, die hier zur Kennzeichnung der sonst nicht darstellbaren Kleinbuchstaben dienen. Der Drucker bringt ebenfalls 40 Zeichen in eine Zeile, kann Kleinbuchstaben und zahlreiche Sonderzeichen (z. B. Indexziffern, Wurzel u. v. a.) darstellen und vor allem auch plotten – 280 Punkte horizontale Auflösung schafft er. Die Tastatur wurde auf den vollen ASCII-Zeichensatz erweitert, und außerdem stehen jetzt acht Funktionstasten (F1...F8) zum direkten Aufruf von Programmen zur Verfügung. Auch eine Reset- und eine NMI-Taste befinden sich auf der absetzbaren Tastatur. Eine rastende Taste „All Caps" dient zum Umschalten auf Nur-Großschreibung während des Programmierens, sie hat aber intelligenterweise keinen Einfluß auf die Wirksamkeit der Shift-Taste bei Ziffern- und Zeichentasten.



Der Zeichensatz des AIM-65/40, wie er vom eingebauten Selbsttest-Programm ausgegeben wird

Wem das Display nebst Drucker nicht ausreicht, dem stehen zwei RS-232-Schnittstellen zum Anschluß eines Terminals und eines größeren Druckers zur Verfügung; hierfür besitzt Systemsoftware auch besondere bildschirmorientierte Befehle. Das Kassetten-Interface schließlich arbeitet mit 1200 Baud sehr zuverlässig und ist mit jenem des AIM-65 voll kompatibel.
Der AIM-65/40 wird übrigens als offene Platine ohne Netzteil (5 V/3 A und 24 V/1 A) geliefert; um das Gehäuse muß man sich also selbst kümmern.

Jetzt dynamisches RAM

Zusätzlich zu den Monitorprogramm-ROMs kann der Anwender ROMs für die höheren Programmiersprachen Basic und Forth, für den 6502-Assembler oder auch für eigene EPROM-Programme auf die Platine stecken; die Fassungen eignen sich für 2-, 4- oder 8-KByte-EPROMs. Das standardmäßig mitgelieferte Monitorprogramm enthält einen komfortablen Texteditor, einen Line-by-line-Assembler für mnemonische Befehlseingabe und einen Disassembler, liegt im Adressenbereich A000...BFFF. Außerdem enthält die Platine ein ROM mit I/O-Software im Bereich F000...FFFF. Ein ROM mit Mathematik-Routinen (E000...EFFF) ist auf Wunsch erhältlich, ebenso der Assembler (9000...9FFF). *Tabelle 1* zeigt die Speicheraufteilung.
Das beim AIM-65/40 nun dynamische RAM liegt im Bereich 0000...8FFF (36 KByte max.); normalerweise sind bei Lieferung jedoch „nur" 16 KByte RAM bestückt.
Die Adressen der I/O-Ports, der Timer und der Monitor-Unterprogramme sind ebenso wie die optionalen Zusatz-ROMs in keiner Weise mit dem AIM-65 kompatibel. Vorhandene ROMs z. B. für Basic lasen sich also für den AIM-65/40 leider nicht weiterverwenden. Ebenso müssen alle Maschinenprogramme erst an das neue Monitorprogramm angepaßt werden (*Tabelle 2*).

Drei CPUs in einem Rechner

Der AIM-65/40 arbeitet mit der bekannten CPU 6502 als Hauptprozessor; sie dient hauptsächlich zum Ausführen von Benutzerprogrammen, fragt aber über einen VIA-Baustein 6522 auch die ASCII-Tastatur ab, die (im Gegensatz zum AIM-65) wirklich alle ASCII-Zeichen enthält, wenn auch nicht die deutschen Sonderzeichen wie ä, ö, ü und ß.

...cker nicht
...i RS-232-
...uß eines Te...
...Druckers zu...
...Systemsoft-
...chirmorien-
...en-Interface
...00 Baud sehr
...em des AIM-

...ens als offene
...3 A und 24 V
...äuse muß ma...

...M

...rprogramm-
...r ROMs für d...
...achen Basic
...Assembler o...
...Programme a...
...assungen eig...
...KByte-
...äßig mitgelie...
...nthält einen
...einen Line-b...
...ionische Be...
...lisassembler
...A000...BFFF
...atine ein RO...
...ich
...it Mathemati...
...st auf Wunsch
...sembler
...zeigt die Spei...

...n dynamische...
...00...8FFF (3...
...eise sind bei
...16 KByte RA...

...rts, der Timer
...rogramme sin...
...en Zusatz-RO...
...n AIM-65 kom...
...s z. B. für Bas...
...AIM-65/40 leic...
...Ebenso müss...
...me erst an da...
...n angepaßt we...

Rechner

...mit der bekan...
...tprozessor; sie...
...m Ausführen v...
...fragt aber übe...
...auch die ASC...
...gensatz zum A...
...-Zeichen enth...
...eutschen Sond...
...d ß.

...weiterer Prozessor, nämlich die vom ...-EMUF bekannte CPU 6504, dient ...schließlich zur Steuerung des Druk...(Bild), und ein zweiter 6504 bedient ...stellige alphanumerische Fluores...-Display.
...drei Prozessoren sind nicht über ...essen- und Datenbus, sondern über ...ASCII-Schnittstelle miteinander ...unden, so daß sich Drucker, Display ...Zentraleinheit prinzipiell auch ge...verwenden ließen. Mit Steuerbe...läßt sich der Drucker aber auch ...Grafik-Modus mit 280 Punkten hori...ler Auflösung betreiben. Alle Steu...chen entsprechen weitgehend der ...Norm.

...mentation vorbildlich

...schon beim AIM-65, so ist auch ...AIM-65/40 die mitgelieferte Doku...tion in Form eines rund 7 cm dik...Bücherstapels wahrhaft vorbildlich, ...dies ist sicher ein Argument für ...strielle Anwendungen: Ein „User's ...ual", das alle Gerätefunktionen, den ...aufbau und die Bedienung be...ibt; ein Programmierhandbuch, das ...die Software der Prozessoren 6502 ...6504 eingeht und – das ist ein ...deres Lob wert – zwei Büchlein ...kompletten, kommentierten ...-Assemblerlistings. Mitgeliefert ...auch ein 23seitiges „Summary

...belle 1: Speicheraufteilung beim ...-65/40

I/O-ROM
Math-Pack-ROM (optional)
Basic-, PL-65- oder Forth-ROMs (optional)
Monitorprogramm- und Texteditor-ROMs
Assembler (optional)
Freies RAM
Reserviert für externe Erweiterungen
System-RAM
Stack (Page 1)
Page Zero (weitgehend frei)

Tabelle 2: Einige Monitor-Unterprogramme

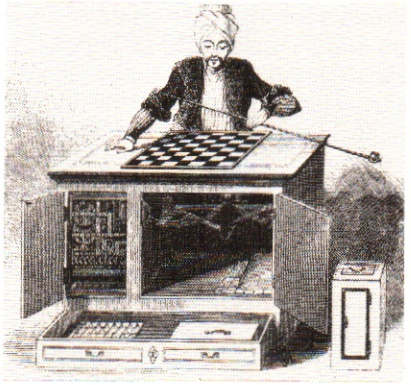
AIM-65	AIM-65/40	Name	Wirkung
E993	F233	INALL	Ein Zeichen vom aktiven Input Device einlesen
EA46	F3A4	NUMA	Ein Byte als zwei ASCII-Zeichen an das aktive Output Device senden
E9BC	F32B	OUTALL	Ein ASCII-Zeichen an das aktive Output Device ausgeben
E9F0	F38F	CRLOW	CR/LF an Display/Printer ausgeben

Booklet", das in Kurzform über Monitor-, Editor- und Assemblerbefehle informiert sowie die wichtigsten Systemadressen enthält, ein Schaltbild und eine 6500-Programmierkarte mit allen CPU-Befehlen. Dazu kommen noch ausführliche Datenblätter der Bausteine 6522 (VIA), 6502/6504 (CPU), und 6551 (ACIA).
Derzeit steht die Dokumentation für den AIM-65/40 bis auf das Programmierhandbuch nur in englischer Sprache zur Verfügung; eine Übersetzung ins Deutsche ist aber bereits geplant.
Ein komfortabler Basic-Interpreter ist von GWK lieferbar; diese Firma stellte uns auch freundlicherweise das Testgerät zur Verfügung.

Für wen ist der Super-AIM gebaut?

Wie schon erwähnt, unterscheidet sich das Konzept des AIM-65/40 grundsätzlich von dem anderer Tischcomputer. Wer nur in Basic arbeiten möchte, sollte sich besser einen „herkömmlichen" Computer zulegen. Wer aber vorwiegend industrielle Steuerungen, Meßgeräte-Abfragen oder auch Einplatinen-Computer wie den mc-EMUF programmieren möchte, für den ist der AIM-65/40 mit Sicherheit besser geeignet als z. B. ein CBM mit seiner randvoll gefüllten „Zero Page" und nur acht I/O-Leitungen. Eine Stärke des AIM-65/40 ist sein superkomfortables Monitorprogramm und der sehr brauchbare Assembler – diese Dinge machen ihn prädestiniert für das Arbeiten in 6502-Maschinensprache. Nur der Laie hält momentan das 40stellige Display gegenüber einem Bildschirm für eine Beschränkung; die praktische Erfahrung zeigt schnell, daß es zusammen mit dem Thermodrucker völlig ausreicht. Fazit: Viel Leistung, leider nicht ganz billig.

Spruch des Monats



Eine Schachspielmaschine, die lernt, wird eine Vielfalt in ihrem Spiel zeigen, die von der Qualität der Spieler abhängt, gegen die man sie hat kämpfen lassen. Die beste Art, sie zu einer Meistermaschine zu machen, wäre wahrscheinlich die, sie gegen die allerverschiedensten guten Schachspieler einzusetzen. Auf der anderen Seite könnte eine wohlersonnene Maschine durch unüberlegte Wahl ihrer Gegner mehr oder weniger ruiniert werden. Auch ein Pferd wird durch die unüberlegte Wahl seiner Reiter zugrundegerichtet.

Norbert Wiener, 1949
(Norbert Wiener gilt als der Vater der Kybernetik)