IV. Softwarebeschreibung des BASIC Betriebssystems mit internationalem Vorbild

0. Aufbau der Software

Das gesamte Betriebssystem umfaßt einen Speicherbereich von 16 KByte und kann in einem bzw. mehreren EPROMs oder ROMs untergebracht werden. Unter dem Betriebssytem eines Rechners versteht man nicht etwa die Sprache, mit der man sich mit ihm verständigt, sondern lediglich den Teil der Software, welcher die physischen Einheiten des Systems steuert und verwaltet. Hierunter sind die Resource Hauptspeicher, die Eingabeeinheiten, also Tastatur, externes Speichermedium und andere, sowie die Ausgabeeinheiten, also Bildschirm, Drucker und wiederum externe Speichermedien zu fassen. Bei den meisten Rechnern ist dies Betriebssystem im Speicher von anderen Teilen weitestgehend getrennt (z.B. bei der Softwarevariante für den Hardwareminimalen Kleinstcomputer mit Microsoft-BASIC). Bei der hier beschriebenen Softwarevariante ist keine exakte Trennung zwischen dem Sprachinterpreter und dem Betriebssystem möglich. Anlage 4 enthält das komplette Assemblerlisting des Programms.

1. Der BASIC-Sprachinterpreter

Der BASIC-Interpreter leistet die Aufgaben der Zeileninterpretation, der Befehlsausführung, der Berechnung mathematischer Ausdrücke, wozu er die arithmetischen Routinen und den Gleitkommarechner enthält, welcher unter anderem die Polynomberechnung der trigonometrischen Funktionen usw. ausführt. Den gesamten Befehlsvorrat des Interpreters enthält Anlage 1. Besonders hervorzuheben ist die sehr gute Unterstützung der Graphikmöglichkeiten des Rechners durch den Interpreter. Die Befehle PLOT, DRAW, CIRCLE gestatten es, sehr schnell und unkompliziert Geraden, Kreisbögen sowie Kreise zu zeichnen. Außerdem bietet die Möglichkeit der nutzerdefinierten Graphiksymbole eine Grundlage für die effektvolle Bildgestaltung von Programmen auf BASIC-Niveau. Der BASIC-Interpreter weist in Bezug auf die Eingabe eine Besonderheit auf. Die BASIC-Befehlswörter werden alle durch einen einzigen Tastendruck ausgelöst. Das heißt, für jeden BASIC-Befehl gibt es eine Funktionstaste. Dies führt zu einer enormen Mehrfachbelegung der Tasten. Gesteuert wird die jeweilige Funktion der Taste durch den Interpreter sowie zwei Shift-Tasten, was zusätzlich durch verschiedene Kursorsymbole angezeigt wird. Durch diese Mehrfachbelegung wird der Eingabeteil des Interpreters in seinem Umfang stark reduziert und Speicherbereich für andere Funktionen frei.

2. Das Betriebssystem

2.1. Der Tastaturanschluß

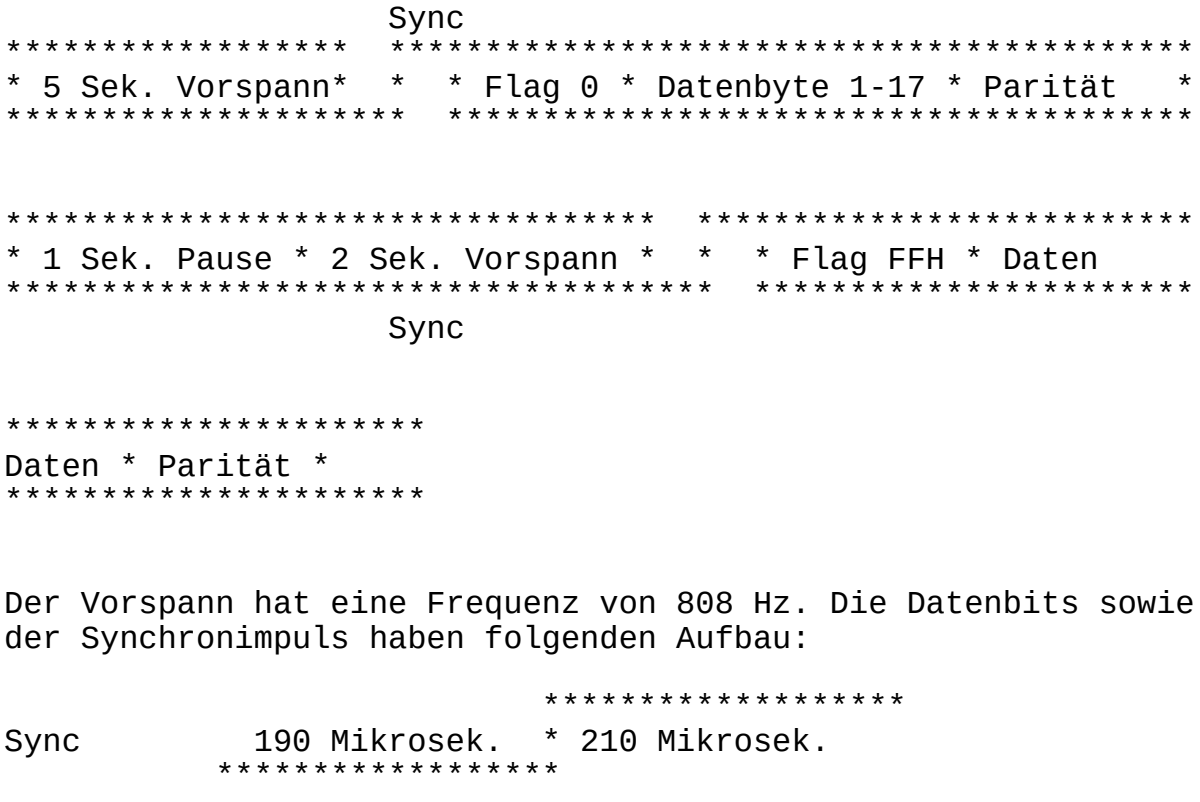
Die Software zur Anpassung des jeweiligen Tastaturtyps (eine Matrix von max. 12 Spalten und max. 8 Zeilen) an den Betriebssystemkern ist in ein hardwarebezogenes Programm und eine Tabelle von Daten gegliedert, die den Zusammenhang zwischen dem Aufbau der Tastaturmatrix und den benötigten Hexa-Werten wiederspiegelt. Das Programm (TAST) ermittelt jeweils die Stelle der gedrückten Taste in der Matrix, sowie eine eventuell gedrückte Shift-Taste und bildet daraus einen Zeiger in die Tabelle (CODTAB), in der der vom Betriebssystem zur Bildung des ASCII-Codes benötigte Wert steht. Durch variieren der Tabelle lassen sich verschiedene Tastaturtypen anpassen. Die Tastatur benötigt mindestens 41 Tasten. Bei 40 dieser Tasten erscheint abhängig von der Syntaxprüfung und den beiden verschiedenen Shift-Tasten eine von bis zu sechs möglichen Funktionen. Diese Sechsfachbelegung der Tasten erscheint zumindest in der Anfangsphase der Beschäftigung mit dem Rechner etwas schwierig, ermöglicht aber eine einfache Programmrealisierung.
Anlage 2 enthält den Aufbau der Tabelle CODTAB.

2.2. Der Kassettenrecorderanschluß

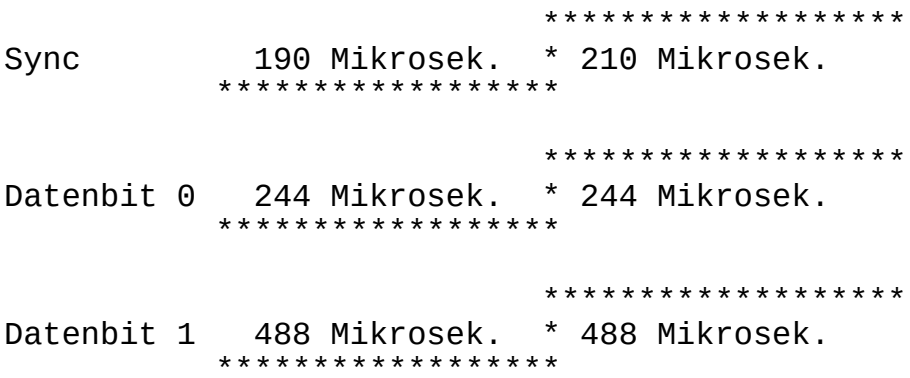
Der Kassettenrecorderanschluß wird durch umfangreiche Software unterstützt. Diese Software ermöglicht ein Laden, sowie Abspeichern von BASIC-Programmen, numerischen Feldern, String Feldern, sowie Speicherblöcken auf Kassettenmagnetband. Abgespeichert werden die Programme in zwei Teilen, dem Programmkopf bestehend aus 17 Bytes sowie dem Datenteil.
Der Aufbau des Kopfes sieht folgendermaßen aus:

```
Byte 1       kennzeichnet den Datentyp
     0       - BASIC-Programm
     1       - numerisches Feld
     2       - String Feld
     3       - Speicherblock
Byte  2-11   Name des Programms
Byte 12-13   Gesamtlänge
Byte 14-15   enthält bei Byte 1 = 0 - Autostartzeilennummer
                                = 1 - Adresse des Namens + 64
                                = 2 - Adresse des Namens + 128
                                = 3 - Startadresse
Byte 15-16   Länge des BASIC-Programms
```

Das Format der gesamten Aufzeichnung sieht folgendermaßen aus:



Der Vorspann hat eine Frequenz von 808 Hz. Die Datenbits sowie der Synchronimpuls haben folgenden Aufbau:



2.3 Bildschirmanschluß

Die Software des Betriebssytems legt den Bildwiederholspeicher in den Bereich von Adresse 4000h bis 57FFH. Dieser hat somit eine Länge von 6144 Bytes. Das entspricht 192 Zeilen zu 256 Punkten, da ein Byte aus 8 Punkten besteht. Die 192 Zeilen entsprechen 24 Textzeilen. Vom Interpreter wird in oberen und unteren Bildschirmteil unterschieden. Dem Benutzer sind innerhalb eines BASIC-Programms die 22 oberen Textzeilen zugänglich also 176 vertikale Punktpositionen. Der untere Teil mit 2 Textzeilen dient den Eingaben. Dieser untere Teil kann allerdings dynamisch vergrößert werden, wenn entsprechend mehr eingetippt wird.
Der Bildschirmspeicher enthält nur Punkte. Es lassen sich somit Zeichen graphisch addieren und das Zeichnen von Linien

über vorhandene Zeichen ist kein Problem. Graphische Figuren und Texte sind also identisch. Auf dem Speicherbereich der Punkte folgen die Attribute mit 768 Bytes. Es gibt pro Zeichenfeld (8x8 Punkte) ein Attribut. Jedes Byte dieses Speicherbereiches enthält folgende Information:

Bit 7 Flashing (Blinken) 1 - an 0 - aus
Bit 6 Bright (Kontrast) 1 - scharf 0 -normal
Bit 3-5 Paperfarbe (Untergrund)
Bit 0-2 Inkfarbe (Vordergrund)

2.4. Zeichensatz

In den letzten 768 Bytes des ROMs bzw. EPROMs von Adresse 3D00H bis 3FFFH stehen die darstellbaren Text- und Graphikzeichen. Da die Zeichen in einer geläufigen 8x8 Matrix wiedergegeben werden, sind pro Zeichen acht Bytes notwendig, insgesamt sind es also 96 Zeichen.

2.5. Softwaremäßige Steuerung der Hardewarekomponenten

Dieser Softwareteil umfaßt mehrere Aufgaben:

4.1 Initialisierung der Hardware

Mittels des Programms INIT wird eine Einstellung der Betriebsarten des PIO-Schaltkreises sowie der Start der CTC-Kanäle vorgenommen. Weiterhin wird die RAM-Größe ermittelt und der RAM definiert geladen. Die Videosignalerzeugung wird vorbereitet und in den Interpreter übergegangen.

2.5.2. Videosignalerzeugung entsprechend des Bildwiederholspeicherinhaltes

Für die Ansteuerung eines Bildschirms werden ein Synchron- und ein Videosignal benötigt. Die Bildaustastsignale für Zeilen- und Bildrücklauf werden nicht gesondert erzeugt, aber während dieser Zeit besitzt das Videosignal den Pegel, der dem höchsten Schwarzwert entspricht. Die Synchronimpulse werden von der CTC und einem Doppelmonoflop erzeugt. Der CTC-Kanal 2 wird so initialisiert, daß ca. 70 Zeilen nach dem Bildimpuls ein Nulldurchgang erfolgt. Dadurch wird in Verbindung mit der Hardware alle 20 ms ein nichtmaskierter Interrupt (NMI) ausgelöst. Die NMI-Routine realisiert mit der Hardware die Videosignalerzeugung. Anlage 3 zeigt den PAP der Routine. Dieses NMI-Programm läuft völlig unabhänig von den anderen Pro-

grammen, unterbricht diese allerdings alle 20 ms für ca. 10 ms, was eine Verschlechterung der Dynamik verursacht. Von der NMI-Routine werden keine Register oder Speicherplätze zerstört. Es tritt lediglich eine zusätzliche Belastung des Stackbereiches ein. Der Inhalt des Interruptflipflops wird vor der Rückkehr ins unterbrochene Programm wieder hergestellt. Innerhalb der NMI-Routine erfolgt bei EI (enable Interrupt) eine Tastaturabfrage und eine Erhöhung einer Zeitzählerzelle.

2.5.3. Festlegung der Speicherstruktur

Aufgrund der in I. beschriebenen Flexibilität macht sich eine softwaremäßige Einstellung der Zuordnung der Adreßbereiche zu den Speicherelementen erforderlich. Bei dieser Softwarevariante wird die Hardware so gesteuert, daß ständig in den ersten 16 KByte des Adreßbereiches der EPROM bzw. ROM angesprochen wird. In den weiteren 48 KByte Adreßbereich werden (je nach Bestückung) 16 KByte bzw. 64 KByte RAM angesprochen.

BASIC-Befehlsatz

Befehl	Beschreibung
BEEP a,b	erzeugt einen Ton am Musikausgang mit der Tonhöhe a und der Länge b wenn der Bildaufbau ausgeschaltet ist
BRIGHT n	Kontrast n=0 - normal n=1 - scharf n=8 - transparent
CIRCLE a,b,c	Zeichnen eines Kreises mit dem Mittelpunkt a,b und dem Radius c
CLEAR n	löscht alle Variablen, CLS und löscht die letzte Plotposition, setzt den RAMTOP auf n
CLS	löscht den Bildschirm
CONTINUE	fortsetzen eines BASIC-Programms nach BREAK
COPY	kopiert den Bildschirminhalt auf den Drucker
DATA	Variablenliste für den READ-Befehl
DEF FN	Funktionsdefinition
DIM a(...)	dimensioniert ein numerisches Feld, dessen Elemente zu Null gesetzt werden
DIM a$(...)	dimensioniert ein String-Feld
DRAW a,b,c	zeichnet eine Gerade von der aktuelle Plotposition x,y zur Position x+a, y+b; ist c ungleich 0, so wird ein Kreisausschnitt über den Winkel c (Bogenmaß) gezeichnet
FLASH n	n=1 - Blinken des Zeichens n=0 - Blinken aus
FOR x=a TO b STEP c	Schleifenanweisung mit der Anfangsvariablen a, der Endvariablen b und der Schrittweite c
GOSUB n	Unterprogrammaufruf, n ist die Zeilennummer
GOTO n	Sprung zur Zeile n
IF a THEN b	a Bedingung; b ist die Anweisung

INK n	Vordergrundfarbe n=0 schwarz n=1 blau n=2 rot n=3 magenta n=4 grün n=5 cyan n=6 gelb n=7 weiß
INPUT	Eingabe über die Tastatur
INVERSE n	n=1 Vorder- und Hintergrund gewechselt n=0 normal
LET a=b	Variablenzuweisung
LIST	Auflisten des BASIC-Programms
LIST n	Auflisten ab Zeile n
LLIST	Auflisten des Programms auf dem Drucker
LOAD p	Laden des BASIC-Programms mit dem Namen p von Kassette
LOAD p DATA	Laden eines numerischen Feldes
LOAD p DATA$	Laden eine String-Feldes
LOAD p CODE r,s	Laden eines Speicherblockes p mit der Länge von s Bytes ab Adresse r
LOAD p CODE r	Laden eines Speicherblocks ab Adresse r
LOAD p CODE	Laden eines Speicherblocks auf die im aufgezeichneten Programm angegebene Position
LOAD SCREEN$	Laden eines Bildes
LPRINT	Ausgabe auf den Drucker
MERGE p	wie LOAD p, jedoch ohne das sich im Speicher befindende Programm zu löschen
NEW	Neuinitialisierung
NEXT x	Endekennzeichnung einer Schleife
OUT x,y	gibt am Port x das Byte y aus
OVER n	n=1 die Zeichen werden beim Überschreiben nicht gelöscht n=0 normal

PAPER n	steuert die Hintergrundfarbe (siehe INK)
PAUSE n	Pause für n Zeiteinheiten (20 ms)
PLOT s,z	setzt eine Punkt auf die Position s,z
POKE a,w	schreibt das Byte w auf den Speicherplatz a
PRINT	Ausgabe auf den Bildschirm
PRINT AT x,y	Ausgabe mit Positionsangabe
PRINT TAB n	Ausgabe mit Tabulatorangabe
RAND n	initialisiert den Zufallsgenerator
READ	Variablenzuweisung in Verbindung mit DATA
REM	Kommentarzeile
RESTORE a	legt den Zeiger bei der READ-Anweisung auf den ersten Wert der Zeile a; ohne a --> in der ersten Zeile
RETURN	Rückkehr aus Unterprogrammen
RUN n	löscht die Variablen und startet ab Zeile n
SAVE p	speichert ein BASIC-Programm
SAVE p LINE a	speichert ein BASIC-Programm mit der Autostart Zeilennummer a
SAVE p DATA	speichert ein numerisches Feld
SAVE p DATA$	speichert ein String-Feld
SAVE p CODE r,s	speichert den Speicherblock der Länge s beginnend ab der Speicheradresse r
SAVE p SCREEN$	speichert den Bildschirminhalt
STOP	stoppt das Programm
SYSTEM n	schaltet bei n=255 den Bildschirm aus bzw. an; setzt bei n=0 bis 8 die Farbe der Eingabezeilen
VERIFY p	vergleicht das Programm auf dem Magnetband mit dem im Speicher

Anweisungen	Beschreibung
ABS x	Absolutbetrag von x
ACS x	Arcuscosinus von x
AND x	log. UND-Verknüpfung
ASN x	Arcussinus von x
ATN x	Arcustangens von x
ATTR x,y	Ermittlung des Attributes der Position x,y
BIN	Binärzahl
CHR$ x	ergibt das ASCII-Zeichen des Codes x
CODE a$	ergibt den ASCII-Code des Zeichens x
COS x	Cosinus von x
EXP x	Exponent von x
FN a(x)	Anwenderdefinierte Funktion a(x); siehe DEF FN
IN x	Eingabe von Port x
INKEY$	fragt nach gedrückter Taste ohne Quittung
INT x	Integerteil von x
LEN a$	Länge des Strings a$
LN x	natürlicher Logarithmus von x
OR x	log. ODER-Verknüpfung
PEEK x	Inhalt der Speicherzelle x
PI	die Zahl PI
POINT (s,z)	ergibt eins, wenn Punkt s,z INK COLOR besitzt
RND	Zufallszahl zwischen 0 und 1

SCREEN$ (x,y)	ergibt das an Stelle x,y abgebildete Zeichen
SGN x	Signum von x
SIN x	Sinus von x
SQR x	Quadratwurzel von x
STR $x	Verwandelt x in einen String
TAN x	Tangens von x
USR a	Sprung zur Adresse a
USR a$	gibt die Adresse für das Bitmuster eines selbstdefinierten Zeichens an
VAL a$	numerischer Wert des Strings a$ als Zahl
VAL $ a$	numerischer Wert des String a$ als String

Weitere Tastenfunktionen:

BREAK	unterbricht ein laufendes BASIC-Programm
BREAK + CAPS-SHIFT	unterbricht jedes Programm

Aufbau der Tabelle CODTAB

Jedem Speicherplatz in der Tabelle ist ein Kreuzungspunkt auf der Matrix zugeordnet. Bezeichnet werden die Kreuzungspunkte hier mit den Kontakten des Steckers für den Tastaturanschluß

```
CODTAB:   A2  - B3
          A2  - B4
          A2  - B5
          A2  - B6
          A2  - B7
          A2  - B8
          A2  - B9
          A2  - B10

          A3  - B3
              :
              :
          A3  - B10

          A4  - B3
              :
              :
          A4  - B10

          A5  - B3
              :
              :
          A5  - B10

          A6  - B3
              :
              :
          A6  - B10

          A7  - B3
              :
              :
          A7  - B10

          A8  - B3
              :
              :
          A8  - B10

          A9  - B3
              :
              :
          A9  - B10

          A10 - B3
              :
              :
          A10 - B10
```

```
A11 - B3
     :
     :
A11 - B10

A12 - B3
     :
     :
A12 - B10

A1  - B3
     :
     :
A1  - B10
```

An diesen Punkten sind für folgende Tasten folgende Werte einzutragen.

A	26H	Q	25H	6	03H
B	00H	R	0DH	7	0BH
C	0FH	S	1EH	8	13H
D	16H	T	05H	9	1BH
E	15H	U	0AH	Symbol	
F	0EH	V	07H	Shift	18H
G	06H	W	1DH	Caps	
H	01H	X	17H	Shift	27H
I	12H	Y	02H	Space	20H
J	09H	Z	1FH	Enter	21H
K	11H	0	23H		
L	19H	1	24H		
M	10H	2	1CH		
N	08H	3	14H		
O	1AH	4	0CH		
P	22H	5	04H		

PAP der NMI-Routine