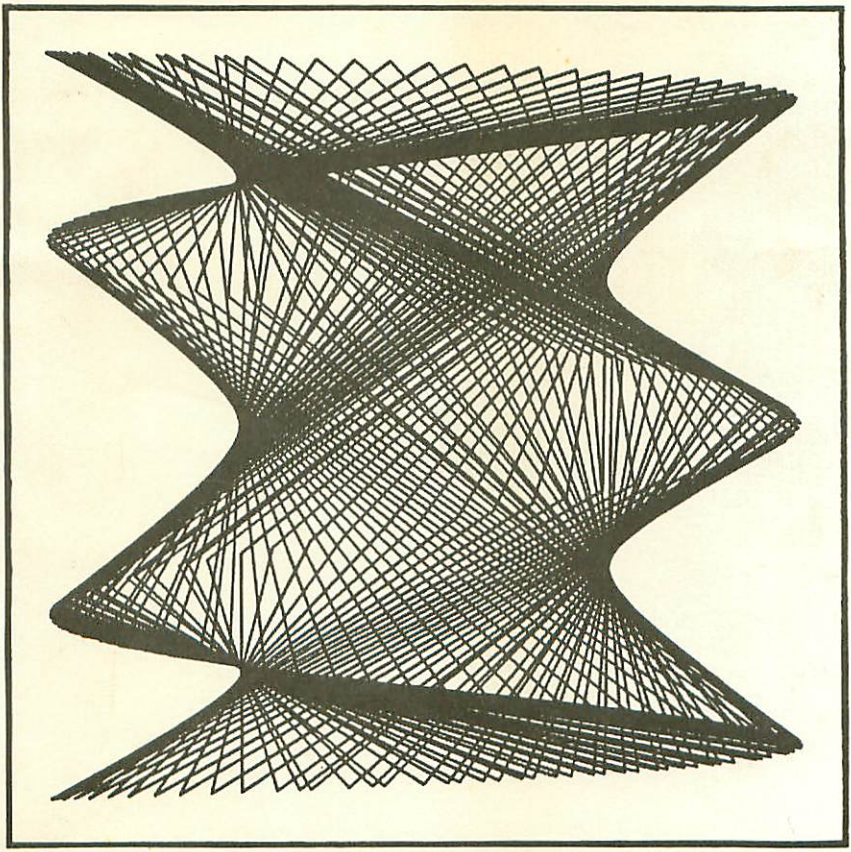
Der NDR-Klein-Computer

BASIC für Z 80



Franzis' SoftwareService

8 KByte - BASIC

für den NDR - KLEIN - COMPUTER

Das 8 KByte-BASIC entspricht in etwa dem Microsoft-Standard. Es wird auf zwei Eproms 2732 geliefert, die anstelle der beiden Grundsoftware-Eproms auf der SBC-2-Platine verwendet werden. Da der Interpreter auch den Graphikprozessor steuern sollte, mußten aufgrund des knappen Speicherplatzes Abstriche am Bedienungskomfort gemacht werden. Der Interpreter benötigt zusätzlich Speicherplatz bis zur Adresse 88C4H, ab 88C5H beginnt der Programmspeicher.

Sedezimalzahlen ("Hexzahlen") werden im Text durch ein nachgestelltes "H" gekennzeichnet. BASIC-Befehle werden in der Beschreibung groß geschrieben, dürfen jedoch beliebig als Groß- oder Kleinbuchstaben eingegeben werden. Drücken der RETURN-Taste wird durch die Kurzfassung <CR> dargestellt. Kontroll-Funktionen werden ähnlich abgekürzt, z.B. <CTRL-Q>; diese Schreibweise symbolisiert das Drücken der CONTROL-Taste und das gleichzeitige Drücken der Q-Taste.

Alphabetische Liste der BASIC-Befehle

Die nach einigen Befehlen angegebene Klammer soll darauf hinweisen, daß dem BASIC-Befehl ein Ausdruck in Klammern folgen muß, sonst erfolgt die Fehlermeldung "?SN Fehler" (SYNTAX ERROR), also falsche Schreibweise eines BASIC-Befehls.

1) ABS(
2) AND
3) ASC(
4) ATN(
5) CALL
6) CHR$(
7) CLEAR
8) CLRS
9) CONT
10) COS(
11) CSAVE
12) CLOAD
13) DATA
14) DEF FN
15) DIM
16) DRAWTO
17) END
18) EXP(
19) FRE(
20) FOR
21) GOSUB
22) GOTO
23) HEX(
24) IF
25) INP(
26) INPUT
27) INT(
28) LEFT$(
29) LEN(
30) LET
31) LIST
32) LLIST
33) LOG(
34) LPRINT
35) MID$(
36) MOVETO
37) NEW
38) NEXT
39) NOT
40) NULL
41) ON
42) OR
43) OUT
44) PAGE
45) PEEK(
46) POKE
47) POS(
48) PRINT
49) READ
50) REM
51) RESTORE
52) RETURN
53) RIGHT$(
54) RND(
55) RUN
56) SGN(
57) SIN(
58) SPC(
59) SQR(
60) STR$(
61) STOP
62) TAB(
63) TAN(
64) USR(
65) VAL(
66) WAIT
67) ? PRINT

START des BASIC-Interpreters, Warmstart

Nach dem Einschalten des Rechners meldet sich der Interpreter mit einem Fragezeichen. Beim ersten Mal muß mit C (Großbuchstabe!, C für "Coldstart") geantwortet werden, worauf folgende Meldung auf dem Bildschirm erscheint:

```
8K BASIC 1.3
RDK 83
o.k.
>
```

Nach dem Drücken der Reset-Taste (SBC-2-Platine) können Sie mit der Eingabe von "W" (für Warmstart) den Interpreter starten, ohne das Programm im Speicher zu zerstören. (Meldung: "o.k." und ">". Beim Warmstart umgeht der Interpreter die Initialisierungsroutinen.

Kontrollfunktionen:

Während der Programmausführung fragt der Interpreter ständig die Tastatur ab; bei Eingabe von <CTRL-S> stoppt die weitere Programmausführung. Mit <CTRL-Q> wird dann wieder gestartet. Dies gilt auch für den Befehl LIST.

Wird während der Programmausführung die Escape-Taste <ESC> betätigt, bewirkt dies eine Unterbrechung nach dem gerade ausgeführten Befehl sowie die Meldung "abgebrochen in Zeile XY". Wie auch bei dem Befehl STOP kann man jetzt die Variablen inspizieren und modifizieren und das Programm mit dem Befehl CONT fortsetzen, sofern es nicht verändert wurde.

Der INPUT-Befehl kann auf diese Weise nicht unterbrochen werden, sondern nur durch die Betätigung der RETURN-Taste; der Interpreter geht daraufhin in die BASIC-Anweisungsebene zurück (o.k.-Meldung), kann aber das Programm (per CONT) mit dem INPUT-Befehl fortsetzen, sofern es zwischenzeitlich nicht modifiziert worden ist.

VARIABLEN

Im Gegensatz zu einem Taschenrechner, der immer nur fest vorgegebene Zahlenwerte miteinander verknüpfen kann, läßt BASIC die Verarbeitung variabler Größen zu, d. h. die Rechenvorschrift wird in allgemeiner Form angegeben (z. B. mit a, b, x, y usf.) und für die Variablen werden je nach Bedarf unterschiedliche Werte eingesetzt.

BASIC kann dabei nicht nur Zahlen als variable Größen verarbeiten, sondern auch Buchstaben und Texte (sogenannte "Strings"). Variablen für Zahlen werden durch eine maximal zweistellige, alphanumerische Zeichenkombination abgekürzt, wobei an erster Stelle immer ein Buchstabe stehen muß. String-Variablen (maximal zulässige Länge: 255 Zeichen bei entsprechender Speicherplatzreservierung) werden dadurch gekennzeichnet, daß an ihre (höchstens zweistellige) Abkürzung ein "$" angehängt wird. Es ist zulässig, in einem Programm gleichzeitig die numerische Variable "A" und die Stringvariable "A$" zu benutzen.

Die Zahlendarstellung erfolgt mit sechs Stellen (einschließlich Dezimalpunkt) plus Vorzeichen plus zweistelligem, vorzeichenbehafteten Exponenten. Daher ergibt die Eingabe von PRINT 123.456 <CR> die Anzeige "123.45" und die Eingabe von PRINT 123.456 + 123.456 <CR> die Anzeige "246.91". Allerdings erhält man bei der Eingabe von PRINT 1 ^ 100000 <CR> dann auch wirklich den Wert 1.

Vor Zahlen wird ein positives Vorzeichen nicht dargestellt, sondern an dessen Stelle ein Leerzeichen ausgegeben. Es können Zahlen im Bereich von ca. -5E-39 bis ca. +5E+38 verarbeitet werden; diese scheinbar willkürlichen Grenzbereiche resultieren aus der internen binären Darstellungsform der Zahlen (drei Bytes für Mantisse und Vorzeichen, ein viertes Byte für den Exponenten plus Vorzeichen). Bei Gleitkommazahlen werden die Nachkommastellen nicht durch ein Komma, sondern durch einen Punkt abgetrennt. Das Komma ist in BASIC dafür reserviert, zwei voneinander unabhängige Größen (z. B. Variablen in einer Aufzählung) gegeneinander abzugrenzen.

Bei Winkelfunktionen muß der Winkel in Bogenmaß (rad) angegeben werden. Die Umrechnung erfolgt durch Division des Winkelwertes (Grad) mit 180 und Multiplikation mit der Zahl PI (3.1415).

OPERATOREN

Da in BASIC sämtliche Eingaben zeilenweise erfolgen, ist z. B. bei Formeln kein Bruchstrich oder Wurzelzeichen möglich. Daher weichen einige Operatoren von der gewohnten algebraischen Notation ab, und durch den Einsatz von Klammern werden bestimmte Ausdrucke (z. B. im Nenner oder unter einer Wurzel stehende) zusammengefaßt.

`+` Addition von Variablen; Strings können durch Addition aneinandergereiht werden

`-` Subtraktion von (numerischen) Variablen

`*` Multiplikation von (numerischen) Variablen

`/` Division von (numerischen) Variablen

`=` hat außer der arithmetischen Bedeutung (Gleichheitszeichen) in BASIC noch die Zuweisungsfunktion, d. h. einer links vom Gleichheitszeichen stehenden Variablen wird die rechts stehende Zahl, Funktion oder Zeichenkette zugewiesen

`<` Vergleichsoperator für "kleiner als"

`<=` Vergleichsoperator für "kleiner als oder gleich" (Reihenfolge der beiden Operatoren beliebig)

`>` Vergleichsoperator für "größer als"

`>=` Vergleichsoperator für "größer als oder gleich" (Reihenfolge der beiden Operatoren beliebig)

`<>` Operator für "ungleich"

E definiert die nachgestellte Zahl (keine Variable!) als Exponent zur Basis Zehn; kann nicht allein, sondern nur in Verbindung mit einer vorangestellten Zahl (nicht Variablen!) verwendet werden
BEI EINGABEN MUSS DAS EXPONENTEN-E GROSS GESCHRIEBEN WERDEN!

↑ definiert die nachgestellte Variable als Exponenten zur vorangestellten Basis

PI (Kreiszahl) ist nicht fest in BASIC gespeichert und muß bei Bedarf definiert werden:PI = 3.1415

BASIC-BEFEHLE UND -FUNKTIONEN

1) A B S (X)

Bildet den Absolutwert von X.

```
PRINT ABS(-523) <CR>                Anzeige: 523
PRINT ABS(-5.1234*10^4) <CR>          "    : 51234
```

2) X AND Y

Bildet auf Maschinen-Ebene bitweise die UND-Verknüpfung aus dem binären Äquivalent von X und Y, d. h. im Ergebniswort steht nur an der Stelle eine 1, an der die korrespondierenden Bits in beiden Operanden X u n d Y eine 1 hatten.

```
PRINT 101 AND 95 <CR>               Anzeige: 69
```

3) A S C ("X") oder A S C (A$)

Erzeugt den zum ASCII-Zeichen "X" gehörenden Binärcode und gibt diesen dezimal aus.

```
PRINT ASC("H") <CR>                 Anzeige: 72
PRINT ASC("Meier") <CR>               "    : 77
```

4) A T N (X)

Bildet den Arkustangens vom Argument X. Das Ergebnis wird im Bogenmaß angegeben.

```
PRINT ATN(2) <CR>                   Anzeige: 1.1071
```

5) C A L L X

Springt in das bei der Adresse X beginnende Unterprogramm in Maschinensprache, das mit dem Z-80-Befehl "Return" (C9H) abgeschlossen sein muß. Die Adresse X ist dezimal zu verstehen (0...65535, keine Zeilennummer!); soll die Startadresse sedezimal genannt werden, ist dies mit HEX("X") möglich. Nach dem Rücksprung aus dem Maschinen-Unterprogramm geht die Programmausführung in BASIC bei dem Befehl weiter, der hinter der CALL-Anweisung steht.

6) C H R $ (X)

Das der Zahl X entsprechende Zeichen nach ASCII wird ausgegeben. Bei Angaben für X, die außerhalb 0...255 (8-Bit-Wortlänge) liegen, wird mit der Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR) die Bereichsüberschreitung gemeldet.

PRINT CHR$(65) <CR> Anzeige: A

7) C L E A R

Setzt alle Variablen auf Null. Vorhandene Speicherplatzreservierungen werden nicht geloscht.

C L E A R X

Setzt alle Variablen auf Null und reserviert X Bytes für die Variablen-Speicherung; X kann eine (positive) Variable sein, darf aber die Anzahl der freien Speicherstellen nicht überschreiten, sonst erfolgt die Fehlermeldung "?OM Fehler" (OUT OF MEMORY). Zu kleine Reservierung ergibt "?OS Fehler" (OUT OF STRING).

CLEAR 200 <CR> PRINT FRE(A$) <CR> Anzeige: 200

8) C L R S

Loschen des Bildschirms

9) C O N T

Veranlaßt den Interpreter, ein zuvor per STOP-Befehl abgebrochenes Programm beim nachstfolgenden Befehl fortzusetzen. In der Zwischenzeit dürfen die Variablen oder Zwischenergebnisse inspiziert werden, allerdings sind vor dem CONT-Befehl keine Anderungen im Programm zulässig, sonst erfolgt Fehlermeldung "?CN Fehler" (CONTINUE ERROR). Beim INPUT-Befehl bewirkt die <CR>-Taste (RETURN-Taste) einen Abbruch (Meldung "o.k."). Mit CONT wird der INPUT-Befehl wiederholt.

10) C O S (X)

Bildet den Kosinus vom Argument X, das im Bogenmaß angegeben oder umgerechnet werden muß.

PRINT COS(60*3.1415/180) <CR> Anzeige: .5002
PRINT COS(60) <CR> " :-.95241

11) C S A V E

Ausgabe des gesamten gespeicherten Programms auf Magnetband-Kassette (ohne Angabe eines Programmnamens und ohne Steuerung des Laufwerks). Es empfiehlt sich, das Programm doppelt abzuspeichern.

12) C L O A D

Einlesen eines auf Magnetband-Kassette gespeicherten Programms (ohne Angabe eines Programm-Namens und ohne Steuerung des Laufwerks). Ein zuvor im Speicher vorhandenes Programm wird geloscht.

13) D A T A X,Y,Z...

Definiert eine Datenzeile mit verschiedenen, durch Komma getrennten Zahlenwerten.

D A T A A$, B$, C$ ...

Definiert eine Datenzeile mit verschiedenen, durch Komma getrennten Texten.

14) D E F F N A(X)=Y

Definiert den Variablennamen A als eine neue Funktion A, die die Rechenvorschrift Y (mathematischer Ausdruck) zusammenfaßt und diese beim Aufruf auf Variable anwendet. (X) ist ein Dummy-Argument, für das jeder beliebige alphanumerische Wert eingesetzt werden kann.

Beispiel: 10 DEF FN Q1(X) = X*B+B
20 INPUT X,B:PRINT FN Q1(X)

15) D I M X(z)

Reserviert für die numerische Variable "X" z Feldelemente, beginnend bei 0 und endend bei (z-1).

D I M X(i,j,k)

Reserviert für die numerische Variable "X" eine mehrdimensionale Matrix mit i*j*k Feldelementen, die jeweils mit dem Index 0 beginnen und mit (i-1) bzw. (j-1) bzw. (k-1) enden.
D I M A$(X)

Reserviert für die String-Variable "A$" X Feldelemente, beginnend bei 0 und endend bei (X-1).

D I M A$(i,j,k)

Reserviert für die String-Variable "A$" eine mehrdimensionale Matrix mit i*j*k Feldelementen, die jeweils mit dem Index 0 beginnen und enden bei (i-1) bzw. (j-1) bzw. (k-1).

16) D R A W T O X,Y

Zeichnet ein Gerade vom augenblicklichen Standpunkt des Cursors nach (x,y). Der Punkt (x,y) wird neuer Standpunkt. Der Bildschirm hat 512*256 Punkte. Die linke untere Bildecke besitzt die Koordinaten 0,0. Mit dem PAGE-Befehl wird die Schreibseite voreingestellt.

17) E N D

Zeigt dem Interpreter an, daß dieser die Programmausführung beenden und in die BASIC-Anweisungs-Ebene zurückspringen soll. Dieses Statement ist entbehrlich, wenn am Programmende keine weiteren Programmzeilen folgen (z. B. die von Unterprogrammen); DATA-Anweisungen können hinter einem Programm stehen, ohne daß davor ein END eingefügt werden muß.

18) E X P (X)

Bildet die X-te Potenz zur Basis e (= 2.7182); bei zu großem X erfolgt die Fehlermeldung "?OV Fehler" (OVERFLOW).

PRINT EXP(20) <CR> Anzeige: 4.8516E+08

19) F R E (X)

Ermittelt ab Adresse 8800H die (dezimale) Anzahl freier Speicherplätze. Bei der SBC-2-Baugruppe mit zwei Speicherbausteinen (RAM) ergibt sich jeweils nach dem Kaltstart folgende Zahl:

PRINT FRE(0) <CR> Anzeige: 1784

20) F O R X = A T O Z S T E P N

Definiert den Anfang einer Programmschleife, in der die Laufvariable X die Werte von A bis Z annehmen und bei jedem Durchlauf um die Schrittweite N erhöht werden soll; im Falle A = Z wird die Schleife einmal durchlaufen, und bei fehlender Angabe der Schrittweite N wird die Zahl 1 angenommen.

Während der Schleifendurchläufe kann die Laufvariable X verändert werden und in Zuweisungen innerhalb der Schleife darf sie nur rechts vom Gleichheitszeichen stehen. Es ist zulässig, mehrere derartige Schleifen ineinander zu verschachteln.

21) G O S U B X

Springt in das bei Zeile X beginnende Unterprogramm, aus dem bei Erreichen des RETURN-Befehls (s.u.) automatisch der Rücksprung ins aufrufende Programm erfolgt; die Programmausführung geht mit nächster BASIC-Zeile weiter, d.h.es darf nach GOSUB X kein weiterer Befehl folgen.

22) G O T O X

Setzt die Programmausführung bei Zeile X fort (unbedingter Sprungbefehl).

23) H E X ("X") oder H E X (A$)

Setzt die Sedezimalzahl ("Hexzahl") "X" in das dezimale Äquivalent um; mehr als vierstellige Angaben führen zu Fehlinterpretationen. Die Sedezimalzahl 0 - 7FFF ergibt 0 bis 32767, 8000 - FFFF ergibt -32768 bis -1. Die sedezimalen Zeichen A ... F müssen in Großbuchstaben eingegeben werden.

24) I F X=Y T H E N (G O T O) Z

Setzt die Programmausführung bei Zeile Z fort, wenn die Bedingung X = Y erfüllt ist; andernfalls geht es bei der nächstfolgenden BASIC-Zeile weiter (bedingter Sprung).

I F X<>Y T H E N (G O T O) Z

Setzt die Programmausführung bei Zeile Z fort, wenn die Bedingung X<>Y (X ungleich Y) erfüllt ist; andernfalls geht

es bei der nächstfolgenden BASIC-Zeile weiter (bedingter Sprung).

I F X<Y T H E N (G O T O) Z

Setzt die Programmausführung bei Zeile Z fort, wenn die Bedingung X<Y erfüllt ist, anderntalls geht es bei der nächstfolgenden BASIC-Zeile weiter (bedingter Sprung).

25) I N P (X)

Liest Daten von demjenigen Eingabe-Kanal ein, dem die dezimale Adresse X (Ø bis 255) zugeordnet ist. Die Angabe einer sedezimalen Adresse ist durch HEX ("X") möglich.

26) I N P U T X

Dieser Befehl ist nicht im Direkt-Modus (ohne Zeilennummer) anwendbar. Die CR-Taste (RETURN-Taste) bricht die Programmausführung ab (Fortsetzung durch Eingabe von CONT). Der Befehl erwartet die Eingabe einer numerischen Variablen, die anschließend unter der Bezeichnung "X" geführt wird. Es ist darauf zu achten, daß Nachkomma-Stellen nicht durch ein Komma, sondern einen Punkt abgetrennt werden, weil das Komma zur Trennung zweier aufeinanderfolgender Eingaben dient (Fehlermeldung "zu viel"). Nicht-numerische Eingaben (z. B. Buchstaben oder Sonderzeichen) werden zurückgewiesen (Fehlermeldung "neue Eingabe").

I N P U T X,Y

Erwartet die Eingabe zweier numerischer Variablen, die durch ein Komma voneinander getrennt werden müssen und anschliessend unter der Bezeichnung "X" und "Y" geführt werden. Die Reaktionen auf Falscheingaben erfolgen sinngemäß wie bei INPUT X.

I N P U T "ABC"; X

Wie INPUT X, aber mit vorheriger Ausgabe des Textes "ABC" auf dem Bildschirm, gefolgt vom Fragezeichen. Dies ist eine recht elegante Eingabeform, wei der Computer regelrecht nach der zu einem Text (z. B. "Lastwiderstand=?") gehörenden Zahl "fragt".

I N P U T A$

Erwartet die Eingabe einer Zeichenkette mit max. 79 Zeichen, die beliebige Zeichen enthalten darf, also auch Ziffern, jedoch kein Komma, weil das zur Trennung zweier aufeinanderfolgender Eingaben dient. Soll der String auch ein Komma enthalten, ist er in Anführungszeichen zu setzen.

27) I N T (X)

Liefert bei einer Gleitkommazahl X den nächstkleineren, ganzzahligen Zahlenwert (Integer). Bei negativen Dezimalzahlen ergibt sich die betragsmäßig kleinere Zahl.

```
PRINT INT(123.55)                    Anzeige: 123
PRINT INT(-0.5)                         "   : -1
```

28) L E F T $ (A$,n)

Spaltet vom String A$ die linksbündigen n Zeichen ab. Wenn der String nicht n Zeichen lang ist, werden entsprechend weniger genommen, ohne daß eine Fehlermeldung erfolgt. Zahlen für n <= 0 ergeben die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR).

PRINT LEFT$("MEIER",3) <CR> Anzeige: MEI

29) L E N (A$)

Ermittelt die Anzahl der im String A$ enthaltenen Zeichen.

PRINT LEN("MEIER") <CR> Anzeige: 5

30) L E T X=ABC

Weist der numerischen Variablen X den rechts vom Gleichheitszeichen stehenden Wert zu. Der Begriff "LET" ist hierbei zwar entbehrlich, kann unter Umständen aber die übersichtlichkeit eines Programms erhöhen.

L E T A$ = "ABC"

Weist der String-Variablen A$ die in Anführungszeichen stehende Zeichenkette zu. Auch hier ist der Begriff "LET" entbehrlich, dient aber u. U. der übersichtlichkeit eines Programmes.

31) L I S T

Bewirkt die Ausgabe des gesamten gespeicherten Programms zeilenweise auf dem Bildschirm und kann mit der Taste <ESC> abgebrochen werden.

L I S T X

Wie LIST, jedoch beginnend bei der Zeilennummer X. Um nur einen kleinen Programmausschnitt ausgeben zu lassen, gibt man "LIST X" und <CR> ein, hält die CTRL-Taste während der Return-Auslösung gedrückt und betätigt sofort (zusätzlich zur CTRL-Taste) zum Anhalten die Taste "S" und zum weiteren Anschauen die Taste "Q".

32) L L I S T

Wie LIST, nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

L L I S T X

Wie LIST X, nur erfolgt die Ausgabe auf den Drucker.

33) L O G (X)

Bildet den natürlichen Logarithmus von X (zur Basis e = 2.7182); bei negativem X erfolgt die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR); bei zu großem X die Meldung "?OV Fehler" (OVERFLOW).

PRINT LOG(1000) <CR> Anzeige: 6.9077

34) L P R I N T X

Wirkt genauso wie PRINT, nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

Für LPRINT gilt die abkürzende Form des Fragezeichens nicht.

Das diesem Befehl vorangestellte "L" steht als Abkürzung für engl. "Lineprinter" (Zeilendrucker).

L P R I N T "XYZ"

Wie PRINT "XYZ", nur erfolgt die Ausgabe nicht auf dem Bildschirm,sondern auf dem Drucker.

L P R I N T A$

Wie PRINT A$, nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

L P R I N T X;Y

Im Gegensatz zu PRINT X;Y erfolgt die Ausgabe nicht auf dem Bildschirm,sondern auf dem Drucker.

L P R I N T X,Y

Im Gegensatz zu PRINT X,Y erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

35) M I D $ (A$,n,m)

Löst aus dem String A$, beginnend beim n-ten Zeichen, m Zeichen heraus. Wenn nicht n oder m Zeichen vorhanden sind, werden entsprechend weniger genommen, ohne daß eine Fehlermeldung erfolgt. Zahlen für n bzw. m <= 0 ergeben die Meldung "?FC Fehler" (FUNCTION CALL ERROR).

PRINT MID$("ABCDEFGH,2,3) <CR> Anzeige: BCD

36) M O V E T O X,Y

Nur bei Graphikausgabe wird der Bildpunkt mit den Koordinaten X,Y neuer Cursor-Standpunkt, von dem aus mit dem Befehl DRAWTO U,V eine Linie zum Bildpunkt an der Stelle U,V gezeichnet wird.

37) N E W

Initialisiert den Interpreter neu (Buffer und Variablen löschen, internen Stack definieren u. a.) und wirkt wie das Löschen des Programmspeichers. Tatsächlich aber wird der Programmspeicher nicht gelöscht, sondern an den Beginn des Programmspeichers (die Startadresse steht in den RAM-Zellen 8857H/58H) werden zwei Bytes "00" eingeschrieben. Außerdem wird das Ende des Programmspeichers (die Endadresse steht in den RAM-Zellen 8887H/88H) hinter das zweite gelöschte Byte gesetzt (also um zwei Plätze höher als der Programmanfang).

Das bedeutet, daß das Programm (bis auf die ersten beiden Bytes) noch im Speicher steht, vom Interpreter aber *nicht* mehr ausgeführt werden kann, weil Anfangs- und Endmarkierung durch die NEW-Anweisung zusammengelegt worden sind. Beim Einbau in ein Programm wirkt dieser Befehl wie ein Software-Selbstmord: Das Programm löscht sich quasi selbst aus.

38) N E X T X

Definiert das Ende der mit FOR...TO...STEP begonnenen Programmschleife. Die Nennung der Laufvariablen X kann hierbei entfallen.

39) N O T X

Invertiert im binären Äquivalent von X jedes einzelne Bit; durch die dezimale Anzeige der Ergebnisse kommt es oft zu Mißverständnissen, weil das Ergebnis von "NOT 55" gleich -56 ist. Das liegt daran, daß der Interpreter ein HIGH im Höchstwertigen Bit eines Ergebniswortes als negatives Vorzeichen wertet.

40) N U L L X

Der Befehl NULL X <CR> verschiebt den linken Rand auf dem Bildschirm um X Zeichen nach rechts. Längere Zeichenketten werden auf der linken Bildschirmseite fortgesetzt.

41) O N X G O T O A,B,C...

Setzt die Programmausführung bei Zeile A fort, wenn X = 1 ist, bei Zeile B, wenn X = 2 ist, bei Zeile C, wenn X = 3 ist, usw. (Bedingte Verzweigung).

42) X O R Y

Bildet auf Maschinen-Ebene bitweise die ODER-Verknüpfung aus dem binären Äquivalent von X und Y, d. h. im Ergebniswort steht überall dort eine 1, wo an der korrespondierenden Stelle des Operanden X oder Y eine 1 war.

PRINT 37 OR 16 <CR> Anzeige: 53

43) O U T X,Y

Gibt den zu Y gehörenden Zahlenwert (in binärer Form) an demjenigen Ausgabe-Kanal aus, dem die Adresse X zugeordnet ist; X ist hierbei dezimal zu verstehen (0...255; die Angabe einer sedezimalen Adresse ist durch HEX("X") bzw. HEX("Y") möglich.

44) P A G E N,M

Der GDP64 wird voreingestellt. Dabei wird auf die Seiten N geschrieben und die Seite M angezeigt. Voreingestellt ist die Seite 1 (für Textmodus). Auf diese Seite wird geschaltet, wenn Basic sich mit "o.k." meldet. Nach diesem Befehl darf kein Input-Befehl folgen, da sonst die Seiteneinstellung nicht mehr stimmt. Mit dem Befehl PAGE 0,0 in einem BASIC-Graphik-Programm wird der Bildaufbau sichtbar, mit POKE HEX("87C5"),0 wird das Blinken des Cursors (Graphik)

abgeschaltet.

45) P E E K (X)

Liest den Inhalt der Speicherzelle mit der Adresse X; X ist hierbei dezimal zu verstehen (0 ... 65535; die Angabe einer sedezimalen Adresse ist durch HEX ("X") möglich).

```
PRINT PEEK(HEX("87C5"))          Anzeige: 10
```

46) P O K E X,Y

Schreibt den Wert Y in die Speicherzelle mit der Adresse X; X und Y sind dabei dezimal zu verstehen (0...65535 bzw. 0...255; die Angabe sedezimaler Zahlen ist durch HEX ("X") bzw. HEX ("Y") möglich.

```
POKE HEX("87C5"),2 <CR>   Cursorblinkfrequenz schneller
```

47) P O S (X)

Ermittelt die Cursor-Position in der laufenden Zeile, nennt also die Anzahl der bereits ausgegebenen Zeichen. (X) ist ein Dummy-Argument, für das jeder beliebige alphanumerische Wert eingesetzt werden kann.

48) P R I N T X

Dient im Direkt-Modus dazu, nach Return unmittelbar eine Ergebnisanzeige auf dem Bildschirm zu erzeugen (ohne vorherigen Programmstart), z. B. das Ergebnis einer Rechenoperation oder die Darstellung von Zwischenergebnissen bzw. Variablen nach einer Programmunterbrechung. Dabei kann "X" eine im Programm verwendete Variable sein oder eine Rechenvorschrift (z. B. 275*1.14) oder bei "X$"eine Zeichenkette oder eine beliebige Zeichenfolge, die dann in Anführungszeichen zu setzen ist. Beim Einsatz dieses Befehls in einem Programm ergibt sich eine Fülle von Varianten. Zur Abkürzung kann man anstelle von "PRINT" einfach ein Fragezeichen "?" eingeben.

P R I N T "XYZ"

Bewirkt die Ausgabe der in Anführungszeichen stehenden Zeichenkette auf dem Bildschirm.

P R I N T A$

Bewirkt die Ausgabe der zur String-Varablen A$ gehörenden Zeichenkette auf dem Bildschirm.

P R I N T X;Y

Bewirkt die Ausgabe der entsprechenden Zahlenwerte für X und Y hintereinander. Somit können auch numerische und String-Variablen zusammen ausgegeben werden, z. B. das Ergebnis einer Rechnung mit einem passenden Text. Zwischen beide Ausgaben wird ein Leerzeichen zur Trennung eingefügt; vor positiven Zahlenwerten steht ein weiteres Leerzeichen, weil das positive Vorzeichen unterdrückt wird.

P R I N T X,Y

Bewirkt die Ausgabe der entsprechenden Zahlenwerte in einer Zeile, wobei die zweite (und jede folgende) Ausgabe bei der nächsten Tabulator-Position anfängt (neue Tabulator-Position: alle 14 Spalten).

49) R E A D N

Ruft das jeweils nächste Daten-Element (in diesem Fall eine Zahl) ab; bei jeder Ausführung des READ-Befehls wird ein interner Daten-Pointer um Eins erhöht, um für den folgenden Zugriff das nächste Element zu adressieren. Findet der Interpreter kein Daten-Element mehr, weil mehr READ-Befehle ausgeführt wurden als Daten bereitgestellt sind, erfolgt die Fehlermeldung "?OD Fehler" (OUT OF DATA).

R E A D N$

Wie READ N, jedoch hier Abruf von Texten.

50) R E M

Kleinbuchstaben bleiben nach dem REM-Befehl erhalten. Dieser definiert die laufende Zeile als Kommentarzeile, d. h. nach "REM" kann jeder beliebige, erläuternde Text stehen. Der Interpreter übergeht eine Kommentarzeile bei der Programmausführung, jedoch kann eine solche Zeile als Sprungziel dienen.

10 REM Gitternetz oder 10 PRINT A: REM Ausgabe

51) R E S T O R E

Bewirkt das Rücksetzen des DATA-Zählers, der mit jedem READ-Befehl um Eins erhöht wird und damit stets auf die nächste, per DATA definierte Konstante weist. Wird bei READ keine durch DATA definierte Variable mehr gefunden, erfolgt die Fehlermeldung "?OD Fehler" (OUT OF DATA); das vorherige Rücksetzen per RESTORE vermeidet dies.

52) R E T U R N

Schließt ein BASIC-Unterprogramm ab und bewirkt den Rücksprung an die zuvor mit dem Befehl GOSUB verlassene Stelle im aufrufenden Programm (s. o.).

53) R I G H T $ (A$,n)

Spaltet vom String A$ die rechtsbündigen n Zeichen ab. Wenn der String nicht n Zeichen lang ist, werden entsprechend weniger genommen, ohne daß eine Fehlermeldung erfolgt. Zahlen für n <= 0 ergeben die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR).

54) R N D (X)

Erzeugt eine Gleitkomma-Pseudo-Zufallszahl zwischen 0 ... 1; es ist sichergestellt, daß bei verschiedenen Programmdurchläufen nicht jedesmal dieselbe Zahl bzw. dieselbe Zahlenfolge auftritt. (X) ist ein Dummy-Argument. Negative Zahlen

ergeben konstante Pseudozufallszahlen.

```
PRINT RND(-1) <CR>                    Anzeige: 7.6594E-06
```

55) R U N

Veranlaßt den Interpreter, ein gespeichertes Programm auszuführen, beginnend bei der niedrigsten Zeilennummer.

R U N X

Veranlaßt den Interpreter, ein gespeichertes Programm auszuführen, beginnend bei der Zeilennummer X.

56) S G N (X)

Liefert das Vorzeichen von X; + 1 bei positivem X, - 1 bei negativem X und Null bei X = 0 (Signum-Funktion).

57) S I N (X)

Bildet den Sinus vom Argument X, das im Bogenmaß angegeben oder umgerechnet werden muß.

```
PRINT SIN(45*3.1415/180) <CR>          Anzeige: .70709
PRINT SIN(45)            <CR>            "    : .8509
```

58) S P C (X)

Rückt den Cursor um X Stellen nach rechts und füllt den Zwischenraum mit Leerzeichen (Blanks) auf.

59) S Q R (X)

Bildet die Quadratwurzel aus dem (positiven) Argument X. Bei negativem X erfolgt die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR) für die Bereichsüberschreitung und bei zu großem X wird "?OV Fehler" (OVERFLOW) angezeigt. Die Bezeichnung "SQR" steht als Abkürzung für engl. "Square Root" = Quadratwurzel.

```
PRINT SQR(2) <CR>                     Anzeige: 1.4142
```

60) S T R $ (X)

Wandelt die numerische Variable X in eine String-Variable um und ermöglicht damit die Anwendung von String-Operationen auf Zahlen. Nach erfolgter Umwandlung kann mit der neu definierten String-Variablen keine mathematische Operation mehr durchgeführt werden, auch wenn es sich augenscheinlich um eine Zahl handelt.

61) S T O P

Bewirkt nach der Ausführung dieses Befehls die Unterbrechung des Programms mit der Meldung "abgebrochen in Zeile xyz"; anschließend ist die Inspektion und Modifikation von Variablen mit darauffolgender Fortsetzung des Programms möglich (per CONT), allerdings darf dabei das Programm nicht modifiziert werden (programmierte Programmunterbrechung).

62) T A B (X)

Rückt den Cursor vom linken Bildrand aus um X Stellen nach rechts. Im Beispiel kann daher der zweite TAB-Befehl nicht ausgeführt werden.

```
PRINT TAB(10);"*";TAB(5);"*" <CR>
Anzeige:          **
```

63) T A N (X)

Bildet den Tangens vom im Bogenmaß angegebenen Wert X.

```
PRINT TAN(45*3.1415/180) <CR>        Anzeige: .99995
PRINT TAN(45)            <CR>          "    : 1.6197
```

64) U S R (X)

Ruft die bei der Adrese X beginnende Anwender-Funktion auf und übergibt (im Gegensatz zu CALL) das Ergebnis im Gleitkomma-Akkumulator des Interpreters (Adressen 886E...8870H für Mantisse und Vorzeichen und Adresse 8871H für den vorzeichenbehafteten Exponenten).

65) V A L (A$)

Wandelt die String-Variable A$ in eine numerische Variable um und ermöglicht anschließend wieder die Anwendung mathematischer Operationen. Von A$ wird allerdings nur derjenige numerische Anteil berücksichtigt (Folge von Zahlen), der keine ASCII-Zeichen enthält, d. h. alle Zeichen, die im ursprünglichen String A$ rechts vom ersten nicht-numerischen Zeichen stehen, werden bei der Typ-Umwandlung nicht berücksichtigt.

```
PRINT VAL("1234ABCD") <CR>           Anzeige: 1234
PRINT VAL("ABC12")    <CR>             "    : 0
```

66) W A I T X,Y,Z

Wie INP X, aber mit anschließender Exklusiv-ODER- Verknüpfung mit Z, gefolgt von der UND-Verknüpfung mit Y. Der Interpreter führt den nächsten Befehl erst dann aus, wenn das so entstandene Ergebnis ungleich Null ist. Im Falle Z = 0 (oder bei fehlender Z-Angabe) werden die eingelesenen Daten nur mit Y UND-verknüpft. Die angegebenen Werte für X, Y und Z sind dezimal zu verstehen (0...255); die Angabe von sedezimalen Operanden ist durch HEX ("X") bzw. HEX ("Y") bzw. HEX ("Z") möglich.

Der Befehl WAIT 70,4,0 bewirkt eine Warteschleife, die erst verlassen wird, wenn Bit 2 des Statusregisters mit der Adresse 70H des Graphikprozessors EF9366 den Wert 1 besitzt.

F E H L E R B E H A N D L U N G

Bei der Umsetzung und Ausführung eines gespeicherten Programms prüft der Interpreter ständig, ob bei der Formulierung der einzelnen Befehle das vorgeschriebene Eingabeformat eingehalten worden ist (z. B. Setzen von Kl'ammern, Anhängen eines Dollar-Zeichen o. ä.). Derartige Syntax-Fehler sind für das Programm ohne Schwierigkeiten erkennbar, weil es die Eingaben nur mit einem vorgegebenen Muster zu vergleichen braucht.

Darüber hinaus gibt es Fehler des Bedieners, die eine Programmausführung unmöglich machen (z. B. wenn der Programm- oder Variablenspeicher voll ist und keine Daten mehr aufgenommen werden können). Ebenso gehört die Prüfung auf verbotene Rechengänge (Teilen durch Null, Wurzel aus negativer Zahl ziehen), fehlende Definitionen oder Bereichsüberschreitungen zu den überwachungsaufgaben des Interpreters.

Fehlermeldung: ?FF FEHLER (IN ZEILE XYZ)

Es wurde einer der oben genannten Fehler erkannt, für dessen Identifikation ein zweistelliger Fehlercode "FF" ausgegeben wird.

Im Direkt-Modus erfolgt nur die Ausgabe "?FF Fehler", während im Programmlauf auch noch die Zeilennummer angegeben wird, in der der Fehler auftritt: "?FF Fehler in Zeile XYZ"; bis auf die (international üblichen) zweistelligen Fehlercodes erfolgt diese Meldung in deutsch. Nach einer derartigen Fehlermeldung wird ein Programmlauf sofort abgebrochen, und der Interpreter geht zurück in den BASIC-Anweisungs-Modus (o.k.-Meldung).

ZUVIEL

Wenn bei der INPUT-Anweisung, die eine Bediener-Eingabe verlangt, mehr Eingaben gemacht werden als angefordert (mehrere Eingaben werden durch Komma voneinander getrennt), dann erfolgt die Meldung "zuviel". In diesem Fall ignoriert der Interpreter die überzähligen Eingaben und fährt mit der Programmausführung fort.

Diese Meldung ergeht auch dann, wenn eine Dezimalzahl anstelle des Dezimalpunktes ein Komma enthält, da der Interpreter dies als z w e i Eingaben versteht (Trennzeichen Komma).

NEUE EINGABE

Wenn bei Eingaben der falsche Variablen-Typ gewählt wird (bei erwarteten numerischen Daten die Eingabe von Buchstaben oder umgekehrt), dann erfolgt die Meldung "neue Eingabe". In diesem Fall ignoriert der Interpreter die falschen Eingaben und erwartet die Eingabe des richtigen Datentyps; danach fährt er mit der Programmausführung fort.

LISTE UND BEDEUTUNG DER FEHLERMELDUNGEN

1) BS BAD SUBSCRIPT Undefiniertes Feldelement

falsche Indizierung; ein aufgerufenes Matrix-Element liegt außerhalb der durch DIM festgelegten Grenzen

2) CN CONTINUE ERROR kein CONT möglich

Die Fortsetzung eines zuvor unterbrochen Programms per CONT ist nicht möglich, weil entweder ein Fehler vorliegt oder das Programm selbst zwischenzeitlich modifiziert worden ist.

3) DD DOUBLE DIMENSION Mehrfachdefinition eines Feldes

Dasselbe Feld wird im Programm nocheinmal dimensioniert.

4) FC FUNCTION CALL ERROR Rechenfehler

Bei einem Funktionsaufruf liegt ein Parameter außerhalb des zulässigen Bereichs, z. B. beim Wurzelziehen aus einer negativen Zahl. (Rechenfehler).

5) ID ILLEGAL DIRECT als Direktbefehl nicht erlaubt

Die gewählte Anweisung ist im Direkt-Modus nicht zulässig; eine Funktions-Definition beispielsweise kann nur innerhalb eines Programms, nicht aber im Direkt-Modus aufgerufen werden.

6) LS LONG STRING String zu lang

Ein String überschreitet die maximal zulässige Länge von 255 Zeichen.

7) NF NEXT WITHOUT FOR Next ohne For

Eine Programmschleife ist unvollständig programmiert worden.

8) OD OUT OF DATA zu wenig Daten

Es wurden mehr READ-Befehle ausgeführt als Daten bereitgestellt waren. Entweder müssen mehr Daten eingeführt werden oder der Daten-Pointer ist mittels RESTORE an den Beginn der Datenreihe zurückzusetzen.

9) OM OUT OF MEMORY Variablenspeicher voll
Ein Bereich des Arbeitsspeichers ist voll; das kann ein zu langes Programm sein, eine Überschreitung des Variablen-Speichers oder auch eine Überfüllung des vom Interpreter benutzten Stacks (z. B. durch zu viele ineinander verschachtelte Unterprogramme oder FOR...NEXT-Schleifen).

10) OS OUT OF STRING-SPACE Stringspeicher voll

Der für Strings reservierte Speicherbereich ist voll; das kann durch zu viele oder zu lange Strings passieren.

11) OV OVERFLOW Überlauf beim Rechnen

Das Ergebnis einer mathematischen Operation überschreitet den max. möglichen Zahlenbereich.

12) RG RETURN WITHOUT GOSUB Return ohne gosub

Es taucht ein RETURN-Befehl auf, ohne daß zuvor ein Unterprogramm-Aufruf erfolgt ist.

13) SN SYNTAX ERROR Syntax-Fehler

Es liegt eine Verletzung des vorgeschriebenen Eingabe-Formats vor, z. B. die Eingabe "CHR(X)" statt "CHR$(X)".

14) ST STRING TOO COMPLEX Fehler bei Stringverarbeitung

Ein String ist zu lang; er muß kürzer gefaßt oder in mehrere kürzere aufgeteilt werden.

15) TM TYPE MISMATCH unterschiedliche Variablentypen

In einer Zuweisung kollidieren unterschiedliche Variablen-Typen; anstelle einer erwarteten numerischen Variablen wurde ein String übergeben oder umgekehrt.

16) UF UNDEFINED FUNCTION undefinierte Funktion

Für eine im Programm aufgerufene Funktion fehlt zuvor die entsprechende Definition.

17) US UNDEFINED STATEMENT Sprungziel fehlt

Es wurde eine falsche, nicht im BASIC-Befehlssatz enthaltene Anweisung eingegeben (bzw. eine richtige gemeinte Anweisung wurde falsch geschrieben) oder es wurde im Programm ein Sprungziel aufgerufen, das garnicht existiert.

18) /0 DIVISION BY ZERO Teilung durch 0

Jemand hat verbotenerweise versucht, durch Null zu teilen.

GRAPHIKBEISPIELE

1) quadratisches Gitternetz (Bild 1)

```
 1 REM GITTERNETZ 25*25 FELDER
 5 CLRS
10 PAGE 0,0                          :REM SCHREIBSEITE 0
20 FOR I = 0 TO 500 STEP 20          :REM SENKRECHTE LINIEN
30 MOVETO I,0
40 DRAWTO I,250
50 NEXT
60 FOR I = 0 TO 500 STEP 10          :REM WAAGRECHTE LINIEN
70 MOVETO 0,I
80 DRAWTO 500,I
90 NEXT
100 GOTO 100
```

2. Zufallsquadrate (Bild 2)

```
 5 CLRS:PAGE 0,0
10 X = RND(1)*511
20 Y = RND(1)*250
30 MOVETO X,Y
40 GOSUB 100
50 GOTO 10
100 REM WUERFEL
110 X = X + 10:GOSUB 200
120 Y = Y +  5:GOSUB 200
130 X = X - 10:GOSUB 200
140 Y = Y -  5:GOSUB 200
200 DRAWTO X,Y:RETURN
```

3. Sinusschwingungen (Bild 3)

```
10 CLRS:PI = 3.1415
20 INPUT"Anzahl der Schwingungen";S: S = S*2
30 PAGE 0,0:FOR X = 0 TO 511
40 Y = INT(SIN(X/511*PI*S)*120+120)
50 MOVETO X,120:DRAWTO X,Y
60 NEXT
70 POKE HEX("87C5"),0:          REM Cursor aus
```

4. Sinusgraphik (Bild 4)

```
10 CLRS: PAGE 0,0
20 FOR X = 0 TO 511
25 PI = 3.1415
30 Y = INT(SIN(X/511*PI*4)*COS(X/511*17*PI)*100+100)
40 Y1 = INT(SIN(X/511*PI*4)*100+100)
50 MOVETO X,100
60 DRAWTO X,Y
70 MOVETO X,Y1:DRAWTO X,Y1
75 NEXT
80 POKE HEX("87C5"),0
```

5. Textausgabe (Bild 5)

```
10 CLEAR 100: CLRS
20 GDP=HEX("70")
30 INPUT "TEXT:";A$
```

```
40 INPUT "SCHRIFTGROESSE:";X
50 PAGE 0,0
60 OUT GDP+3,X+X*16:              REM Port 73 = Vergrößerung
70 MOVETO 0,0:                    REM Linke untere Ecke
80 FOR I = 1 TO LEN(A$)
85 OUT GDP,ASC(MID$(A$,I,1)):     REM Ausgabe der ASCII-Werte PORT 70
90 WAIT GDP,4,0:                  REM Warteschleife bis GDP fertig
100 NEXT I
110 OUT GDP+3,1+16                REM alte Schriftgröße
120 POKE HEX("87C5"),0:           REM Cursor aus
130 INPUT"Nochmal";X$
140 IF X$ = "J" THEN 10
```

6. Hex-Monitor (Bild 6)

```
10 REM Hex-Monitor
20 CLEAR 100:DIM H$(16):DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
30 FOR I = 0 TO 15:READ H$(I):NEXT I
50 TZ$="0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F"
60 TS$="dez.  hex.   ":CLRS
100 PRINT"Speicher anschauen    =  1":PRINT
110 PRINT"Speicher aendern      =  2":PRINT
130 INPUT"Zahl";A: IF A>2 OR A<1 THEN 60
1000 CLRS: INPUT"ab Hex-Adr";B$:B = HEX(B$):PRINT
1015 IF B<0 THEN B=65536+B
1016 IF A=2 THEN 2000
1018 B=INT(B/16)*B
1020 INPUT"bis:";C$:C = HEX(C$):CLRS
1025 IF C<0 THEN C = 65536 + C
1030 PRINT TS$,TZ$
1040 FOR I = B TO C
1080 IF I/16 = INT(I/16) THEN PRINT:ZR=I:GOSUB 6010
1085 D=PEEK(I):S=0:GOSUB 5017
1090 NEXT I:PRINT:INPUT"nochmal";X$
1100 IF LEFT$(X$,1) ="j" THEN 1000
1110 GOTO 60
1500 REM AENDERN
2000 ZR=B:GOSUB 6010
2003 D=PEEK(B):S=0:GOSUB 5017
2005 INPUT"hex";F$:F=HEX(F$):IF F>255 THEN 2005
2010 POKEB,F:B=B+1:GOTO 2000
3000 REM DEZ-HEX
5000 S = INT(D/256):gosub 5100
5015 PRINT H$;
5017 S = D - S*256:GOSUB 5100
5020 PRINT H$ + "  ";:RETURN
5100 L = S AND 15: H = (S AND 240)/16
5150 H$ = H$(H)+H$(L):RETURN
6000 REM Format
6010 ZR$=STR$(ZR$):PRINT MID$(ZR$,2,6);SPC(8-LEN(ZR$));
6020 D=ZR:GOSUB 5000
6030 RETURN
```

Speichererweiterung für BASIC-Programme

Bei der SBC-2-Baugruppe stehen als Datenspeicher zwei statische Speicherbausteine 6116 als RAM (Random Access Memory) mit einem Speicherplatz von 4 KByte zur Verfügung. Diese belegen die Adressen B000H - B7FFH (IC 8) und B800H - BFFFH (IC 9). Bei längeren BASIC-Programmen reicht bald dieser Platz nicht mehr aus und man wünscht sich eine Speichererweiterung.

Eine einfache und billige Möglichkeit ergibt sich durch die Verwendung einer weiteren SBC-2-Platine, auf der nur die Speicherbausteine mit Sockel sowie ein Pufferelko eingesetzt werden. Die Datenleitungen D0-D7, die Adressenleitungen A0-A7 und die Leseleitung -RD (Read Acces) sowie die Schreibleitung -WR (Write Acces) beider Platinen sind über den BUS (Trägerplatine) sowieso miteinander verbunden. Zusätzlich werden noch die drei Adressenleitungen A8-A10 und die Auswahlleitungen (-CS = Chip Select) für die bis zu vier zusätzlichen Speicherbausteine benötigt. Die Verbindung kann entweder mittels eines siebenadrigen Flachkabels oder über weitere BUS-Leitungen vorgenommen werden. Auf der Erweiterungsplatine muß dann nur noch die -WR-Leitung zu IC6 und IC7 geführt werden. Wenn auf der Erweiterungsplatine vier Speicherbausteine 6116 eingesetzt sind, ergibt sich ein Speicher bis zur Adresse AFFFH. Allerdings ist er nicht bis zum Ende nutzbar, da dort der STACK liegt. Der BASIC-Interpreter meldet mit dem Befehl PRINT FRE(0) <CR> 9976 Bytes anstelle von 1784 Bytes, da 8192 Bytes (4*2048) Speicherplatz dazugekommen sind. Es ergibt sich folgende Adressenzuordnung auf der Erweiterungsplatine, wobei die IC-Bezeichnungen IC 6 bis IC 9 beibehalten wurden.

IC 6 9000H - 97FFH
IC 7 9800H - 9FFFH
IC 8 A000H - A7FFH
IC 9 A800H - AFFFH

Im Einzelnen müssen nun folgende Verbindungen zwischen der SBC-2-Baugruppe und der Erweiterungsplatine geschaffen werden:

SBC-2			<----->	Erweiterung		
	IC	Pin	<----->		IC	Pin
IC5	74LS138	13		IC6	6116	18
IC5	"	12		IC7	"	18
IC5	"	11		IC8	"	18
IC5	"	10		IC9	"	18
IC9	6116	23		IC9	6116	23
IC9	"	22		IC9	"	22
IC9	"	19		IC9	"	19

Auf der Erweiterungsplatine:

IC7	6116	21	IC8	6116	21

Besser als ein Flachbandkabel zwischen beiden Platinen ist jedoch die Verdrahtung ausschließlich über die BUS-Leitungen. Auf der SBC-2-Platine kann nämlich noch ein Stift (Ø), falls er fehlen sollte, (Bestückungsseite, rechter Rand der Steckerleiste) eingebaut werden. Da die BUS-Leitungen für 12 Volt in der Grundausbaustufe nicht benötigt werden, stehen somit am linken Steckerteil drei und am rechten Steckerteil vier Stifte bzw. BUS-Leitungen zur Verfügung. Nun ergibt sich eine kurze Leitungsführung auf jeder Platine und man kann die Erweiterungsplatine ohne Lötarbeiten entnehmen.

Wie nun die Verbindungen zu legen sind, zeigen die Bilder 1 und 2. Die zusätzlichen Stifte wurden mit den Ziffern Ø - 3 und 4 - 6 gekennzeichnet. Auf jeder Platine müssen die mit den gleichen Ziffern gekennzeichneten Lötaugen miteinander mittels dünner Litze verbunden werden.

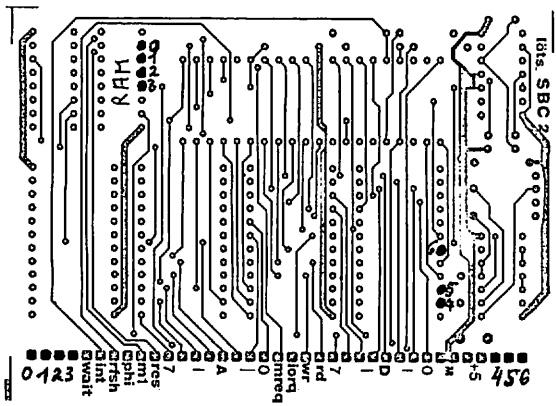


Bild 1. Lötseite der SBC-2-Baugruppe

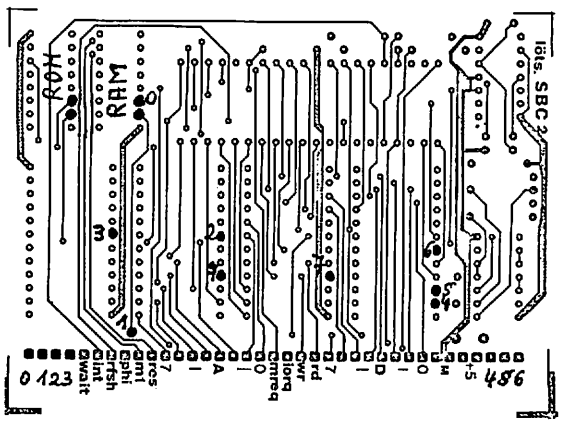


Bild 2. Lötseite der Erweiterungsplatine

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden*).

Alle Schaltungen und technischen Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, daß sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung eventueller Fehler sind Autor und Verlag jederzeit dankbar.

---

*) Bei gewerblicher Nutzung ist vorher die Genehmigung des möglichen Lizenzinhabers einzuholen.

BJ/SS/285/0006/5°