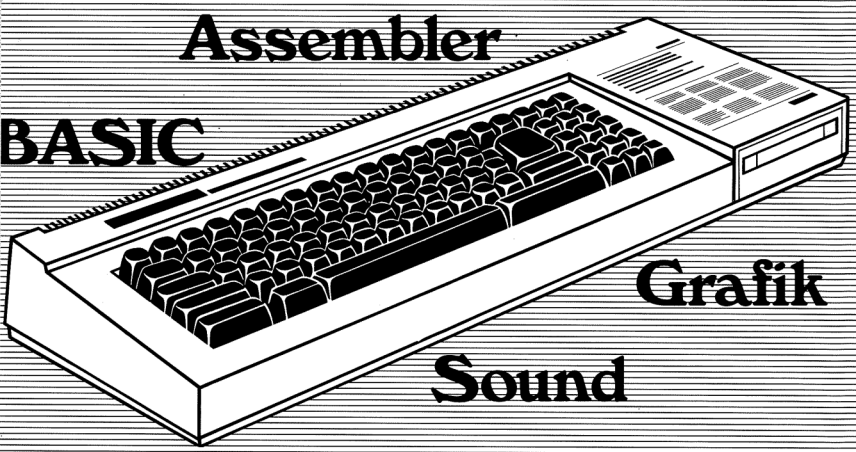
Andreas Dripke

SCHNEIDER CPC 464, 664, 6128

Programmier-begleiter



Vieweg

Andreas Dripke

Schneider CPC 464, 664, 6128
Programmierbegleiter

Aus dem Programm Mikrocomputer

Entwerfen von Programmen
von Gerhard Oetzmann

Wie arbeite ich mit dem Schneider CPC 464
von Wolfgang Schneider

BASIC-Wegweiser für den Schneider CPC 464, 664 und 6128
von Ekkehard Kaier

Wie arbeite ich mit dem Schneider Joyce
von Wolfgang Schneider

BASIC-Wegweiser für den Commodore 128
von Ekkehard Kaier

Wie arbeite ich mit dem Commodore 128
von Wolfgang Schneider

BASIC-Wegweiser für den Commodore 64
von Ekkehard Kaier

Commodore 64-Programmierbegleiter
von Andreas Dripke

Spaß mit Algorithmen
von Johann Weilharter

Einführung in die Anwendung des Betriebssystems CP/M
von Wolfgang Schneider

Vieweg

Andreas Dripke

Schneider CPC 464, 664, 6128 Programmierbegleiter

BASIC
Assembler
Grafik
Sound

Friedr. Vieweg & Sohn Braunschweig / Wiesbaden

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Dripke, Andreas:
Schneider-CPC-464, -664, -6128-Programmierbegleiter: BASIC, Assembler, Grafik, Sound / Andreas Dripke. — Braunschweig; Wiesbaden: Vieweg, 1986.
ISBN 3-528-04490-X

1986

Umschlaggestaltung: Ludwig Markgraf, Wiesbaden
Druck und buchbinderische Verarbeitung: W. Langelüddecke, Braunschweig
Printed in Germany

ISBN 3-528-04490-X

Inhaltsverzeichnis

Vorwort

Jeder, der einen Computer programmiert, benötigt dazu spezifische Informationen über sein jeweiliges Gerät. Das vorliegende Werk enthält in einer wohlüberlegten, praxisgerechten Auswahl alle Daten, die für die Programmierung der CPC-Computer von Schneider relevant sind. Dabei werden sämtliche Aspekte von der BASIC-Programmierung über Assembler bis hin zu Grafik und Sound berücksichtigt: BASIC-Befehlsauflistung mit Erklärung, Assembler Begriffserläuterungen, Z80 Mnemonics und Adressierungsarten, CPU Blockdiagramm, logische Operationen, Speicherbelegung, Grafik-Chip (Beschreibung, Registerauflistung), Gate Array (Beschreibung, Registerauflistung), Farbcodes, Soundchip (Beschreibung, Registerauflistung), Hüllkurvenaufbau, Notentabelle, ASCII-Auflistung u. a. Unterschiede bei den verschiedenen CPC-Modellen sind eingearbeitet.

Da wir uns selbst seit Erscheinen der CPC-Geräte mit der Programmierung dieser Computer beschäftigen, glauben wir, Ihnen eine praxisorientierte Aufbereitung der Informationen garantieren zu können. Da es jedoch bekanntlich nichts gibt, was nicht noch verbessert werden könnte, haben wir für Anregungen aller Art stets ein offenes Ohr. Schreiben Sie bitte kurz an den Verlag. Wir werden Ihren Brief sorgfältig studieren und in zukünftigen Auflagen einarbeiten. Bitte haben Sie jedoch Verständnis dafür, daß wir beim besten Willen nicht auf einzelne Schreiben antworten können.

So, genug der Vorrede. Jetzt wünschen wir Ihnen viel Freude und Erfolg mit dem vorliegenden „Wissenskonzentrat". Möge es Ihnen stets den Weg zum richtigen Bit zeigen!

Wiesbaden, im Juni 1986

Andreas Dripke
Michael Krause

BASIC

Notation

Argumente

arg Argumente ohne Klammern müssen angegeben werden.

[*arg*] Argumente in eckigen Klammern sind optional, d. h. die Argumente können, brauchen aber nicht angegeben zu werden.
Die Klammern werden nicht mit eingegeben.

{*arg/arg*} In geschweiften Klammern stehen durch Schrägstrich getrennte Alternativen, von denen eine auszuwählen ist. Die Klammern und der Schrägstrich werden nicht mit eingegeben.

(*arg*) Die Argumente von Funktionen werden in runde Klammern eingeschlossen. Die Klammern müssen mit eingegeben werden.

... Die Punkte zeigen, daß der in eckigen Klammern eingeschlossene Term mehrfach wiederholt werden darf.

CPC 664 Der angegebene Befehl ist nur mit dem Computer CPC 664 mit eingebauter Floppy verfügbar. Alle übrigen Befehle funktionieren sowohl auf dem CPC 464 wie dem CPC 664.

Terme

TERME in Großbuchstaben werden wie angegeben verwendet.
terme in Kleinbuchstaben werden wie folgt ersetzt:

adresse Position im Speicher (0<=*adresse*<=65535).

anweisung BASIC-Anweisung.

ausdruck Berechenbare Formel oder numerischer Wert. Der Wertebereich für Ausdrücke liegt zwischen $2.9*10^{\wedge}-39$ (kleinster positiver Wert) und $1.7*10^{\wedge}+38$ (größter positiver Wert). Eventuelle Wertebereichsbeschränkungen sind im Einzelfall angegeben. Bsp.: 3+4*A.

*ausdruck$* Auswertbarer Stringausdruck. Der Stringinhalt (Zeichenfolge) muß stets in Anführungszeichen eingeschlossen werden. Bsp.: A$+"xyz".

bereich {*zeilennummer*[–[*zeilennummer*]]/–*zeilennummer*}

block Folge von Einzelanweisungen, die jeweils durch Doppelpunkt voneinander getrennt werden.

booleausdruck Berechenbarer, boolescher Wert.
Ergebnis: 0 ist logisch falsch (*false*), –1 ist logisch wahr (*true*).

buchstbereich Folge von Buchstaben für DEF-Kommandos.
Gültige Buchstabenreihen sind z. B.: A, C–F, AA–AX.

byteausdruck Ausdruck mit einem Wert zwischen 0 und 255.

dateityp Art einer Datei, anzugeben bei SAVE:

Typ	Art der Datei
A	ASCII-Datei
B	Binärdatei
P	geschütztes Programm

dummy Pseudoparameter (Wert wird nicht verwendet).

eaeinheit Nummer einer Ein-/Ausgabeeinheit.

Wert	Ein-/Ausgabeeinheit
0	Bildschirm
1–7	Bildschirmfenster
8	Drucker
9	Diskettenstation

farbmodus Bei bestimmten Befehlen kann beim CPC 664 der Parameter *farbmodus* mit angegeben werden, welcher zu zeichnende Farben als deckend oder transparent definiert.

farbnummer Wert zur Festlegung der Bildschirmfarbe.

Nr.	Farbe	Nr.	Farbe
0	schwarz	13	weiß
1	blau	14	pastellblau
2	hellblau	15	orange
3	rot	16	rosa
4	magenta	17	pastellmagenta
5	hellviolett	18	hellgrün
6	hellrot	19	seegrün
7	purpur	20	helles blaugrün
8	helles magenta	21	limonengrün
9	grün	22	pastellgrün
10	blaugrün	23	pastellblaugrün
11	himmelblau	24	hellgelb
12	gelb	25	pastellgelb
		26	leuchtendweiß

farbstift In Abhängigkeit vom Bildschirmmodus können einer festgelegten Anzahl von Farbstiften Farben der Farbtabelle zugeordnet werden, um diese dann als Parameter in Zeichenkommandos zu verwenden.

Modus	Farbstifte
0	16
1	4
2	2

folge Daten zur Definition einer Hüllkurve zur Tonerzeugung. Die Daten können aus bis zu fünf Teilbeschreibungen mit je drei durch Komma getrennten Byteausdrücken bestehen. Weitere Informationen siehe ENT und ENV.

hexzahl Zahl, welche nur aus Ziffern und den Buchstaben 'a' bis 'f' besteht.

hüllkurve Auswahl einer bestimmten Hüllkurve zur Tonerzeugung. Der Wert kann zwischen 0 und 15 liegen. Die Hüllkurve 0 ist fest definiert und kann deshalb mit ENT und ENV nicht verändert werden.

intausdruck Ganzzahlausdruck im Bereich von −32768 bis +32767.

kanal Zahl, welche einen Musik-Kanal auswählt.
1 = Kanal A, 2 = Kanal B und 3 = Kanal C.

konstante Konstante.

koordinaten Festlegung der Koordinaten für ein Fenster. Durch vier, jeweils durch Komma getrennte Werte werden die linke, rechte, obere und untere Grenze des Fensters in dieser Reihenfolge festgelegt.

schritt Schrittweite im Bereich von 1 bis 255.

text Aneinanderreihung von Buchstaben, Ziffern und Sonderzeichen außer dem Anführungszeichen.

uhrnummer Nummer einer Uhr im Bereich von 0–3, die für EVERY und AFTER Kommandos verwendet wird. Diese Kommandos können sich auch gegenseitig unterbrechen. Dabei hat die Uhr 3 die höchste, Uhr 0 die niedrigste Priorität.

var Numerische Variable.

var$ Stringvariable.

variablenliste Folge von ein oder mehreren durch Komma oder Semikolon getrennten Numerischen und/oder Stringvariablen.

x,x1 Horizontale Auflösung.

Modus	Auflösung = Wertebereich
0	1–160
1	1–320
2	1–640

Die Auflösung bezieht sich jeweils auf den gesamten Bildschirm. 1 liegt links.

y,y1 Vertikale Auflösung (Wertebereich 1–200).
Die Auflösung bezieht sich dabei auf den gesamten Bildschirm. 1 liegt oben.

xr,yr Relative Koordinaten für relative Grafikbefehle.

zeile1...8 Folge von acht, jeweils durch Komma getrennte Byteausdrücke, welche bitweise das Muster für ein selbstdefiniertes Zeichen ergeben.

zeilennummer Nummer einer BASIC-Zeile.

zuweisung Wertzuweisung an eine Variable.

Beispiele:

'POKE *adresse,byteausdruck'* beschreibt z. B. einen Ausdruck der Form 'POKE 2047,23'. Dabei ist 'POKE' das BASIC-Befehlswort, welches das Abspeichern eines Wertes in einer Speicherstelle des Hauptspeichers auslöst, '2047' die Adresse, in die der Wert übertragen werden soll und '23' der Wert selbst.

'INPUT [*#eaeinheit,*][;][*"text"* {;/,}] {*var/var$*}[, {*var/var$*}...]' beschreibt einen Ausdruck der Form **'INPUT** "Ihre Eingabe";A$,B$'. In diesem Falle ist 'INPUT' das Befehlswort zum Einlesen einer Benutzereingabe. ' "Ihre Eingabe" ' ein auszugebender Text und 'A$,B$' zwei Variablen, in die zwei durch Komma getrennte Texte eingelesen werden sollen. Da keine Ein-/Ausgabeeinheit spezifiziert wurde, bezieht sich der Befehl auf den Bildschirm.

Variablennnamen

Variablennamen beginnen mit einem Buchstaben und können danach durch Buchstaben und Ziffern fortgesetzt werden. Sie können eine Länge bis zu 40 Zeichen haben. BASIC-Befehlswörter dürfen nicht als Variablennamen verwendet werden.

Befehle

Editierbefehle

AUTO [*zeilennummer*[,*byteausdruck*]]
Schaltet die Automatische Zeilennummerierung beginnend bei *zeilennummer* in Schritten des *byteausdruck* ein. Jeweils nach Eingabe einer Programmzeile wird nun die nächste Zeilennummer vorgegeben. Ersatzwert für *zeilennummer* und Schrittweite ist jeweils 10. Der Abbruch dieses Modus erfolgt durch das Drücken der ESC-Taste. Ist die vorgegebene Zeile schon vorhanden, so wird nach der Zeilennummer ein '*' ausgegeben.

CAT
Gibt für alle auf der Cassette befindlichen Dateien Name, Blocknummer und Dateikennzeichen aus.

Kennz.	Art der Datei
$	BASIC-Programm
*	ASCII-Datei
&	Binärdatei
%	geschütztes Programm

Ist ein Diskettenlautwerk angeschlossen, so werden alle Dateien mit Namen und Größe ausgegeben.

CHAIN *ausdruck$[,zeilennummer]*
Lädt das Programm mit Namen *ausdruck$* in den Speicher und startet es. Dabei wird das bisherige Programm gelöscht. *zeilennummer* gibt an, in welcher Zeile das Programm nach dem Laden gestartet werden soll (Ersatzwert ist die erste Zeile im Programm). Die Variablen werden nicht gelöscht, aber ON ERROR GOTO wird ausgeschaltet und RESTORE ausgeführt.

CHAIN MERGE *ausdruck$[,zeilennummer]*[**,DELETE** *bereich*]
Verbindet zwei Programme miteinander; das erste steht im Speicher, das zweite mit Namen *ausdruck$* wird mit CHAIN MERGE hinzugeladen. *zeilennummer* gibt an, in welcher Zeile das Programm nach dem Laden gestartet werden soll (Ersatzwert ist die erste Zeile im Programm). Mit DELETE *bereich* kann ein Programmteil angegeben werden, welcher vor dem Nachladen gelöscht wird. Die Variablen werden nicht gelöscht, aber ON ERROR GOTO wird ausgeschaltet und RESTORE ausgeführt.

CONT
Setzt den durch zweimaliges Drücken der ESC-Taste oder durch den Befehl STOP unterbrochenen Programmablauf fort.

DELETE *bereich*
Löscht eine oder mehrere Programmzeilen.

EDIT *zeilennummer*
Bearbeiten einer Programmzeile. Die Zeile wird auf dem Bildschirm ausgegeben und der Cursor auf das erste Zeichen der Zeile gestellt. Nun kann mit Hilfe der Cursortasten jede beliebige Position innerhalb der Zeile erreicht und der Text verändert werden.

KEY *byteausdruck,ausdruck$*
Belegt die Taste des Ziffernblocks mit der Nummer *byteausdruck* mit dem Text *ausdruck$*. Ein einzelner Text darf bis zu 32 Zeichen lang sein, der Text aller Tasten zusammen 100 Zeichen.

Nr.	Taste	Nr.	Taste
128	0	135	7
129	1	136	8
130	2	137	9
131	3	138	.
132	4	139	Enter
133	5	140	Ctrl+
134	6		Enter

KEY DEF *byteausdruck1,* {0/1}*,byteausdruck2*
Definiert die Taste mit der Nummer *byteausdruck1* so um, daß sie ab sofort das Zeichen mit dem Code *byteausdruck2* liefert. Eins als zweiter Parameter schaltet die Repeat-Funktion der Taste ein, Null schaltet diese aus.

LIST [*bereich*][*,#eaeinheit*]
Zeigt das aktuelle Programm. Mit der ESC-Taste kann das Listing angehalten werden. Es wird dann durch Drücken der Leertaste fortgesetzt oder durch nochmaliges Drücken von ESC ganz abgebrochen.

LOAD *ausdruck$*[*,adresse*]
Lädt das Programm mit Namen *ausdruck$* von Cassette oder Diskette. Wird *adresse* angegeben, so wird eine binäre Datei ab dieser Adresse in den Speicher geladen.

MERGE [*ausdruck$*]
Verbindet zwei Programme miteinander; das erste steht im Speicher, das zweite mit Namen *ausdruck$* wird mit MERGE von Cassette oder Diskette hinzugeladen. Die Variablen werden dabei gelöscht, ON ERROR GOTO wird ausgeschaltet und RESTORE ausgeführt.

NEW
Löscht das aktuelle BASIC-Programm.

RENUM [*zeilennummer1*][,[*zeilennummer2*][,*schritt*]]
Numeriert einen Teil des Programms ab *zeilennummer2* neu durch. Der neue Programmteil beginnt ab *zeilennummer1* mit dem Abstand *schritt* (Standardwerte sind 10,0,10).

RUN {*ausdruck$*/[*zeilennummer*]}
Wenn *ausdruck$* angegeben ist, wird das Programm mit Namen *ausdruck$* von Cassette oder Diskette geladen und gestartet. RUN alleine oder mit *zeilennummer* startet das Programm im Speicher bei der ersten oder der angegebenen Zeilennummer.

SAVE *ausdruck$*[*,dateityp*][*,adresse,intausdruck*]
Speichert ein Programm unter dem Namen *ausdruck$* auf Cassette oder Diskette. Bei einer binären Datei sind noch Startadresse und Länge (*intausdruck*) anzugeben.

SPEED KEY *byteausdruck1,byteausdruck2*
Setzt die Parameter für die Tastenwiederholfunktion. *byteausdruck1* * 0.02 sec ist die Ansprechverzögerung, *byteausdruck2* * 0.02 sec die Wiederholungsperiode. Ersatzwert ist jeweils 10.

SPEED WRITE {0/1}
Legt die Schreibgeschwindigkeit auf Cassette auf 1000 Baud (0) oder 2000 Baud (1) fest.

TRON/TROFF
Aktiviert (TRON) bzw. desaktiviert (TROFF) den TRACE-Modus, in welchem vor Ausführung einer Programmzeile deren Nummer in eckigen Klammern an der aktuellen Cursorposition auf dem Bildschirm ausgegeben wird.

Siehe auch unter folgenden Funktionen: FRE, HIMEM.

Ein-/Ausgabe-Befehle

CLEAR INPUT CPC 664
Löscht den Tastaturpuffer.

CLOSEIN
Schließt die offene Eingabedatei auf Cassette oder Diskette.

CLOSEOUT
Schließt die offene Ausgabedatei auf Cassette oder Diskette.

COPYCHR$ CPC 664
Kopiert ein Zeichen von einer Bildschirmposition zu einer anderen.

CURSOR CPC 664
Schaltet den Cursor ein/aus.

DATA *konstante*[,*konstante...*]
Signalisiert den Anfang der Datenangaben für READ.

INPUT [*#eaeinheit,*][,][*"text"* {;/,}] {*var/var$*} [, {*var/var$*}...]
Liest ein oder mehrere durch Komma oder Leerzeichen getrennte Eingaben von der Tastatur oder der angegebenen *eaeinheit* in ebensoviele Variablen ein. Ein Semikolon nach 'INPUT' bzw. '*#eaeinheit,*' unterdrückt den Zeilenvorschub nach der Eingabe. Als Eingabeaufforderung kann *text* auf dem Bildschirm ausgegeben werden. Wird nach *text* ein ';' gesetzt, so wird automatisch ein '?' mit ausgegeben.

LINE INPUT [*#eaeinheit,*][;][*"text"* {;/,}]*var$*
Liest eine Zeile ohne Rücksicht auf Leerzeichen und Kommata in eine Textvariable *var$* ein. Die Parameter entsprechen denen des INPUT-Befehls.

OPENIN *ausdruck$*
Eröffnet eine Eingabedatei auf Cassette oder Diskette mit dem Namen *ausdruck$*.

OPENOUT *ausdruck$*
Eröffnet eine Ausgabedatei auf Cassette oder Diskette mit dem Namen *ausdruck$*.

OUT *adresse,byteausdruck*
Gibt *byteausdruck* zur Ausgabe an das mit *adresse* angesprochene Interface.

PRINT [*#eaeinheit,*]*variablenliste*
Gibt Text bzw. Variablenwerte auf dem Bildschirm oder dem Gerät *eaeinheit* aus.

PRINT [*#eaeinheit,*] **USING** *ausdruck$;variablenliste*
Gibt Text bzw. Variablenwerte gemäß dem in *ausdruck$* angegebenen Format auf dem Bildschirm oder dem Gerät *eaeinheit* aus.

Zch.	Bedeutung des Formatzeichens
#	Ziffer
.	Position des Dezimalpunktes
+	Vorzeichen immer ausgeben
–	Vorzeichen ausgeben, wenn Zahl negativ
**	Maximal zwei führende Druckstellen werden mit '*' statt mit Leerzeichen ausgegeben.
$$	Vor dem ersten Zeichen wird '$' ausgegeben.
,	Tausendertrennung
^^^^	Ausgabe im Exponentialformat
!	Erstes Zeichen des Textes ausgeben.
\ \	Gibt soviele Zeichen aus, wie das Format belegt.
&	Der gesamte Textausdruck wird ausgegeben.

Bsp.: 'PRINT –8.272 USING '**$#,##+"' ' führt zu '*$8,27–'.

READ {*var/var$*} [, {*var/var$*}...]
Liest Daten aus DATA-Anweisungen.

RESTORE [*zeilennummer*]
Setzt den Datenlesezeiger auf die erste DATA-Anweisung im Programm. Ist eine Zeilennummer angegeben, so wird der Datenlesezeiger auf diese Zeile gestellt.

Siehe auch unter folgenden Funktionen: INKEY, INKEY$, INP, JOY, POS, SPC, TAB, VPOS.

Diskettenbefehle

Die Diskettenbefehle können nur ausgeführt werden, wenn eine Diskettenstation angeschlossen und zum Zeitpunkt des Einschaltens des Computers aktiv ist. In diesem Falle beziehen sich auch andere Befehle wie z. B. LOAD und SAVE auf die Diskettenstation. Diese Zuordnung kann mit den Befehlen |TAPE und |DISC verändert werden. Alle Diskettenbefehle werden durch einen vorangestellten senkrechten Balken ('|') gekennzeichnet.

|A
Umschaltung auf Laufwerk A.

|B
Umschaltung auf Laufwerk B.

|CPM
übergibt die Kontrolle an das CP/M Betriebssystem.

|DIR [*ausdruck$*]
Zeigt das Inhaltsverzeichnis der Diskette und den freien Speicherplatz auf dem Bildschirm.

|DISC [. {**IN/OUT**}]
Lenkt alle Ein-/Ausgabebefehle auf die Diskette. Wird .IN oder .OUT spezifiziert, so bezieht sich die Umlenkung nur auf Ein- bzw. Ausgabebefehle.

|DRIVE,*ausdruck$*
Setzen des Standard-Laufwerks.

|ERA,*ausdruck$*
Löscht die Datei mit Namen *ausdruck$*.

|REN,*ausdruck1$,ausdruck2$*
Gibt der Datei mit Namen *ausdruck1$* den neuen Namen *ausdruck2$*.

|TAPE[. {**IN/OUT**}]
Lenkt alle Ein-/Ausgabebefehle auf die Cassette. Wird .IN oder .OUT spezifiziert, so bezieht sich die Umlenkung nur auf Ein- bzw. Ausgabebefehle.

|USER,*ausdruck*
Reserviert für spezielle Anwendungen.

Strukturanweisungen

AFTER *intausdruck* [,*uhrnummer*] **GOSUB** *zeilennummer*
Springt nach *intausdruck* * 0.02 sec das Unterprogramm bei *zeilennummer* an. *uhrnummer* gibt die Uhr an, welche verwendet wird.

CALL *adresse* [,*parameter*]
Springt in ein Maschinenspracheunterprogramm, das bei *adresse* beginnt.

END
Beendet die Abarbeitung des Programms.

ERROR *byteausdruck*
Simuliert das Auftreten eines Fehlers mit der Nummer *byteausdruck* (siehe ON ERROR GOTO).

EVERY *intausdruck* [,*uhrnummer*] **GOSUB** *zeilennummer*
Springt alle *intausdruck* * 0.02 sec das Unterprogramm bei *zeilennummer* an. *uhrnummer* (Bereich 0–3) gibt die Uhr an, welche verwendet wird.

FOR *zuweisung* **TO** *ausdruck1* [**STEP** *ausdruck2*]
Zählschleifenanweisung, bei der die zwischen FOR und TO initialisierte Variable bei jedem Schleifendurchlauf um *ausdruck2* (Ersatzwert ist 1) erhöht wird, bis der nach TO stehende Endwert *ausdruck1* erreicht ist. Das Ende des Schleifenblocks wird durch NEXT [*var*] gekennzeichnet. Bsp.: 'FOR X=0 TO 10 STEP 2' zählt 0, 2, 4, 6, 8, 10.

GOSUB *zeilennummer*
Springt in das bei *zeilennummer* beginnende Unterprogramm.

GOTO *zeilennummer*
Springt in die Zeile *zeilennummer.*

IF *booleausdruck* **THEN** *block* [**ELSE** *block*]
Ist die zwischen IF und THEN stehende Bedingung erfüllt, so wird der nach THEN folgende Teil der Programmzeile, andernfalls der hinter ELSE folgende Anweisungsblock abgearbeitet.

NEXT [*var*[,*var*...]]
Schließt ohne Variablenangabe die innerste, sonst die angegebene(n) FOR-Schleife(n).

ON *byteausdruck* {**GOTO/GOSUB**} *zeilennummer* [,*zeilennummer*...]
Der angegebene *byteausdruck* bestimmt, auf die wievielte der aufgezählten Zeilennummern sich der GOTO/GOSUB-Befehl beziehen soll. Sind nicht genügend Zeilennummern vorhanden, so wird der Befehl ignoriert.

ON BREAK {**GOSUB** *zeilennummer*/ **STOP**}
Springt in das bei *zeilennummer* beginnende Unterprogramm, wenn zweimal die ESC-Taste gedrückt wird. ON BREAK STOP schaltet wieder in den Modus, in dem bei zweifachem Drücken der ESC-Taste das Programm abgebrochen wird.

ON ERROR GOTO {zeilennummer/0}
Initialisiert eine Fehlerbehandlungsroutine, die bei Auftreten eines Fehlers während der Ausführung eines Programms ab jetzt angesprungen wird, durch Angabe der ersten Zeilennummer der Routine. Zeilennummer 0 schaltet dies aus. Die Funktionen ERL liefert die Nummer der fehlerhaften Programmzeile und ERR die Nummer des Fehlers gemäß folgender Tabelle:

Code	Bedeutung	Code	Bedeutung
01	Unexpected next	16	Stringexpr. too complex
02	Syntax	17	Cannot CONTinue
03	Unexpected RETURN	18	Unknown user function
04	DATA exhausted	19	RESUME missing
05	Improper argument	20	Unexpected RESUME
06	Overflow	21	Direct command found
07	Memory full	22	Operand missing
08	Line does not exist	23	Line too long
09	Subscript out of range	24	EOF met
10	Array already dim'd	25	File type error
11	Division by zero	26	NEXT missing
12	Invalid direct command	27	File already open
13	Type mismatch	28	Unknown command
14	String space full	29	WEND missing
15	String too long	30	Unexpected WEND

ON SQ (*kanal*) **GOSUB**
Springt zum Unterprogramm an der angegebenen Zeile, falls bei der Tonausgabe in der Warteschlange des angegebenen Kanals Platz frei ist. Damit kann während des Programmlaufs und während Musik gespielt wird in den Programmteil gesprungen werden, der die Musik erzeugt.

REM [*text*]
Kennzeichnet den Beginn einer Kommentarzeile. Die folgenden Zeichen bis zum Ende der Zeile bleiben unbeachtet.

RESUME [{*zeilennummer*/**NEXT**}]
Beendet die durch ON ERROR GOTO festgelegte Fehlerbehandlungsroutine. Ohne Parameter wird die zum Fehler führende Anweisung noch einmal ausgeführt, RESUME NEXT setzt das Programm hinter der fehlerhaften Anweisung fort, Bei Angabe einer *zeilennummer* wird zu dieser Programmzeile verzweigt.

RETURN
Kennzeichnet das Ende eines Unterprogramms. Es erfolgt die Rückkehr zum Hauptprogramm.

STOP
Unterbricht den Programmablauf mit Angabe der Zeilennummer für Testzwecke.

WHILE *booleausdruck*:*block* **WEND**
WHILE setzt den Anfang einer Schleife, welche solange wiederholt wird, wie *booleausdruck* wahr ist. Das Ende der Schleife wird durch WEND bezeichnet.

Variablenbearbeitung

CLEAR
Löscht alle Variablen und Felder und vergißt alle Dateien.

DEF FN {*var1/var1$*} ({*var2/var2$*})= {*ausdruck/ausdruck$*}
Definiert eine numerische (oder String-) Benutzerfunktion *ausdruck* (*ausdruck$*) mit Namen FN *var1* (FN *var1$*), die wie z. B. ABS oder INT in arithmetischen Ausdrücken (LEFT$ oder STR$ in Stringausdrücken) gebraucht werden kann. *var2* (*var2$*) gilt ausschließlich in der Variablendefinition und steht dort stellvertretend für den Wert, der beim Funktionsaufruf der Funktion übergeben wird.
Bsp.: 'DEF FN A (X) = 2 * X' definiert eine Funktion, welche das Argument mit zwei multipliziert. '? FNA(7–3)' liefert also den Wert 8.

DEFINT [*buchstbereich*]

DEFREAL [*buchstbereich*]

DEFSTR [*buchstbereich*]
Legt fest, daß alle Variablen, die mit den angegebenen Buchstaben beginnen, vom Typ Integer (DEFINT), Real (DEFREAL) bzw. String (DEFSTR) sind.

DEG
Schaltet die Berechnung der trigonometrischen Funktionen (ATN, COS, SIN) auf Winkelgrad um.

DIM {*var/var$*} (*intausdruck*[,*intausdruck*...])
Vereinbart die Größe des Datenfelds *var* bzw. *var$*.

ERASE {*var/var$*} [, {*var/var$*} ...]
Löscht die angegebenen Variablenfelder mit Namen *var* oder *var$* und gibt den ihnen zugeordneten Speicherplatz wieder frei.

[**LET**] {*var=ausdruck/var$=ausdruck$*}
Führt eine Wertzuweisung an eine Variable aus.

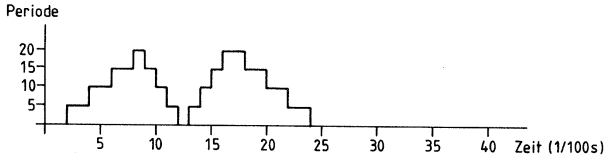
RAD
Schaltet die Berechnung der trigonometrischen Funktionen (ATN, COS, SIN) zurück auf Bogenmaß (Standard).

RANDOMIZE [*ausdruck*]
Setzt einen neuen Startwert für Zufallsfolgen.

Siehe auch unter folgenden Funktionen: FN.

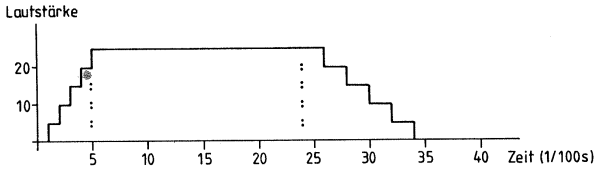
Soundbefehle

ENT *hüllkurve, folge*
Variiert einen Ton in seiner Höhe. Es können bis zu fünf Folgen angegeben werden. Sie bestehen jeweils aus der Schrittanzahl (Bereich 0–239), die die Anzahl der durchzuführenden Einzelveränderungen darstellt, der Schrittgröße (Bereich –128 bis +127) und der jeweiligen Zeitdauer (Bereich 0–255), welche in 1/100 Sekunden angegeben wird.
Beispiel für den Befehl ENT 1,4,5,2,4,–5,1,4,5,1,4,–5,2



Abschnitt	Zeitspanne	Schrittanzahl	Schrittgröße	Dauer
1	0– 8	4	5	2
2	8– 12	4	–5	1
3	12– 16	4	5	1
4	16– 24	4	–5	2
⇒ Befehl: ENT 1,4,5,2,4,–5,1,4,5,1,4,–5,2				

ENV *hüllkurve, folge*
Spezifiziert den Lautstärkenverlauf einer Hüllkurve. Es können bis zu fünf Folgen angegeben werden. Sie bestehen jeweils aus der Schrittanzahl (Bereich 0–127), die die Anzahl der durchzuführenden Einzelveränderungen darstellt, der Schrittgröße (Bereich –128 bis +127) und der jeweiligen Zeitdauer (Bereich 0–255), welche in 1/100 Sekunden angegeben wird.
Beispiel für den Befehl ENV 1,5,5,1,1,0,19,5,–5,2



Abschnitt	Zeitspanne	Schrittanzahl	Schrittgröße	Dauer
1	0– 5	5	5	1
2	5– 24	1	0	19
3	24– 34	5	–5	2
⇒ Befehl: ENV 1,5,5,1,1,0,19,5,–5,2				

RELEASE *byteausdruck*
Hebt den eventuellen Wartezustand eines oder mehrerer Kanäle auf. Dabei wird *byteausdruck* bitweise wie folgt ausgewertet: Bit 0 = Kanal A, Bit 1 = Kanal B und Bit 2 = Kanal C. Bsp.: RELEASE 3 hebt den Wartezustand von Kanal A und B auf.

SOUND *a,b,c,d,e,f,g*
Spielt einen Ton oder reiht ihn in die Warteschlange des betreffenden Kanals ein. Dabei werden folgende Parameter verarbeitet:

a: Kanal Status (Bereich 0–255).

Dezimal	Bit	Bedeutung
1	0	sende SOUND nach Kanal A
2	1	sende SOUND nach Kanal B
4	2	sende SOUND nach Kanal C
8	3	Rendezvous mit Kanal A
16	4	Rendezvous mit Kanal B
32	5	Rendezvous mit Kanal C
64	6	Haltezustand
128	7	Kanal aktiv

b: Ton Periode (Bereich 0–4095).
Legt eine Periode und damit die Frequenz des zu spielenden Tons fest.
Dabei gilt: Frequenz = 125000/Periode.

c: Tondauer (Bereich –32768 bis +32767, Standard ist 20).
Bei Werten größer Null ist der Wert die Tondauer in 1/100 Sekunden. Null bedeutet, daß die Tondauer von der Hüllkurve gesteuert wird und ein negativer Wert wiederholt die Lautstärkenhüllkurve so oft wie der Wert angibt.

d: Lautstärke (Bereich 0–15, Standard ist 12).
Legt die Lautstärke des Tones fest. Null bedeutet keine Lautstärke.

e: Lautstärkenhüllkurve (Bereich 0–15, Standard ist 0).
Auswahl einer mit ENV definierten Lautstärkenhüllkurve (1–15) oder der Standardhüllkurve 0.

f: Tonhüllkurve (Bereich 0–15, Standard ist 0).
Auswahl einer mit ENT definierten Tonhüllkurve (1–15) oder der Standardhüllkurve 0.

g: Geräuschperiode (Bereich 0–15, Standard ist 0).
Legt das Geräusch fest, das dem Sound hinzugefügt wird. 0 bedeutet kein Geräusch.

Beispiel: 'SOUND 2,284,100,7' spielt eine Sekunde lang den Kammerton A (440 Hz) auf Kanal B mit größtmöglicher Lautstärke.

Siehe auch unter folgenden Funktionen: REMAIN, SQ.

Grafikbefehle

BORDER *farbnummer1*[,*farbnummer2*]
farbnummer1 legt die Bildschirmrandfarbe fest. Wird zusätzlich auch *farbnummer2* angegeben, so wechselt der Bildschirmrand zwischen den beiden Farben. Die Intervalle werden mit dem Kommando SPEED INK festgelegt.

CLG [*farbstift*]
Füllt den Bildschirm mit der *farbstift* zugeordneten Farbe. Fehlt die Angabe des Farbstiftes, so wird der des letzten CLG-Kommandos verwendet.

CLS [*#eaeinheit*]
Füllt den mit *eaeinheit* angegebenen Bildbereich bzw. wenn keine angegeben ist, den ganzen Bildschirm mit der Hintergrundfarbe.

DRAW *x,y*[,*farbstift*]

DRAW *x,y*[,*farbstift*[,*farbmodus*]] CPC 664
Zeichnet eine Linie von der augenblicklichen Cursorposition zur absoluten Position *x,y*. Beim CPC 664 kann zusätzlich der Parameter *farbmodus* angegeben werden.

DRAWR *xr,yr*[,*farbstift*]

DRAWR *xr,yr*[,*farbstift*[,*farbmodus*]] CPC 664
Zeichnet eine Linie von der augenblicklichen Cursorposition zur relativ um *xr,yr* versetzten Position. Beim CPC 664 kann zusätzlich der Parameter *farbmodus* angegeben werden.

FILL CPC 664
Füllt ein definiertes Feld mit einer Farbe aus.

FRAME CPC 664
Synchronisiert die Bildschirmausgabe mit dem Strahlrücklauf auf dem Monitor. Dadurch können flüssige Bewegungsabläufe flimmerfrei dargestellt werden.

INK *farbstift,farbnummer*[,*farbnummer*]
Ordnet *farbstift* die neue Farbe *farbnummer* zu.

LOCATE [*#eaeinheit,*]*x,y*
Positioniert den Cursor auf *x,y* relativ zum Ursprung des Bildschirmfensters *eaeinheit*. Die linke obere Ecke des Fensters hat dabei die Koordinaten 1,1.

MASK *byteausdruck* CPC 664
Definiert ein Muster für das Zeichnen von Linien. *byteausdruck* gibt dabei in 8 Bit an, welche von acht nebeneinanderliegenden Punkten dargestellt werden sollen.

MODE {0/1/2}
Setzt den neuen Bildschirmmodus und löscht den Bildschirm.

Modus	Auflösung		Anzahl
	Text (S,Z)	Grafik (Z,S)	Farben
0	20 x 25	160 x 200	16
1	40 x 25	320 x 200	4
2	80 x 25	640 x 200	2

S=Spalten, Z=Zeilen.

MOVE *x,y*
Setzt den Grafikcursor auf die absolute Position *x,y*.

MOVER *xr,yr*
Setzt den Grafikcursor auf die Position *xr,yr* relativ zur aktuellen Position.

ORIGIN *x,y*[,*koordinaten*]
Bestimmt mit *x,y* den Startpunkt des Grafikcursors relativ zur linken unteren Ecke. Der optionale Teil legt die Grenzen eines Koordinatensystems, in dem die Koordinaten nach rechts und oben ansteigen, fest. Zeichenbefehle, die sich auf Werte außerhalb dieser Grenzen beziehen, werden nicht ausgeführt.

PAPER [*#eaeinheit,*]*farbstift*
Legt die Hintergrundfarbe für einen Bildschirmbereich fest. Wird in den Bereich ein Zeichen geschrieben, so wird der zugehörige Hintergrund mit der Farbe von *farbstift* beschrieben.

PEN [*#eaeinheit,*]*farbstift*

PEN [*#eaeinheit,*]*farbstift*[,*farbmodus*] CPC 664
Ordnet dem Schreib- oder Zeichenstift den angegebenen *farbstift* zu. Beim CPC 664 kann zusätzlich der Parameter *farbmodus* angegeben werden.

PLOT *x,y*[,*farbstift*]

PLOT *x,y*[,*farbstift*[,*farbmodus*]] CPC 664
Plottet jeden Einzelpunkt von der aktuellen Grafikcursorposition bis *x,y*. Beim CPC 664 kann zusätzlich der Parameter *farbmodus* angegeben werden.

PLOTR *xr,yr*[,*farbstift*]

PLOTR *xr,yr[,farbstift[,farbmodus]]* CPC 664
Plottet relativ von der aktuellen Grafikcursorposition bis *xr,yr*. Beim CPC 664 kann zusätzlich der Parameter *farbmodus* angegeben werden.

SPEED INK *ausdruck1,ausdruck2*
Legt die Zeitintervalle der Farbwechsel für INK und BORDER Kommandos fest. *ausdruck1* ist die Zeitdauer (in 0.02 sec) für die erste und *ausdruck2* für die zweite Farbe.

SYMBOL *byteausdruck, zeile1...zeile8*
Definiert das Zeichen *byteausdruck* neu. Auf *byteausdruck* folgen durch Komma getrennt acht weitere Byteausdrücke. (*zeile1–8*), die jeweils eine Zeile des neuen Zeichens bitweise beschreiben. Das Setzen des höchstwertigsten Bits von *zeile1* bedeutet dabei, daß der linke obere Eckpunkt des Zeichens gesetzt ist. SYMBOL muß zuerst durch SYMBOL AFTER ermöglicht werden.

SYMBOL AFTER *byteausdruck*
Ermöglicht, daß alle Zeichen des Zeichensatzes, deren Code größer oder gleich *byteausdruck* ist, mit dem Befehl SYMBOL neu definiert werden können. Zunächst werden alle Zeichen auf den Standardzeichensatz eingestellt.

TAG [*#eaeinheit*]
Schaltet die Ausgabe von Text durch PRINT-Anweisungen von der aktuellen Textauf die aktuelle Grafikcursorposition um. Die linke obere Ecke des ersten Zeichens wird auf die aktuelle Grafikcursorposition gelegt. Damit ist Text und hochauflösende Grafik beliebig mischbar.

TAGOFF [*#eaeinheit*]
Schaltet die durch TAG auf den Grafikcursor umgelenkte Textausgabe wieder auf die aktuelle Textcursorposition um.

WIDTH *byteausdruck*
Stellt die Zeilenlänge für die Druckerausgabe ein. Wird diese erreicht, so wird eine neue Zeile begonnen.

WINDOW [*#eaeinheit,*]*koordinaten*
Legt ein Fenster im Bildschirm fest und weist ihm eine Fensternummer zu. Als *eaeinheit* sind die Werte zwischen 1 und 7 zulässig, so daß 7 Fenster möglich sind.

WINDOW SWAP *eaeinheit,eaeinheit*
Tauscht die Zuordnungen (z. B. für Ausgabe von Fehlermeldungen) zu zwei Fenstern aus.

Siehe auch unter folgenden Funktionen: TEST, TESTR, XPOS, YPOS.

Systembefehle

CALL *adresse*
Ruft ein Maschinenunterprogramm auf.

DI
Sperrt alle Unterbrechungsanforderungen durch EVERY und AFTER.

EI
Hebt die Sperrung der Unterbrechungsanforderungen wieder auf.

MEMORY *adresse*
Stellt die obere Speichergrenze für BASIC ein.

POKE *adresse, byteausdruck*
Speichert ein Byte an der angegebenen Speicherstelle (*adresse*).

WAIT *adresse, byteausdruck1*[*,byteausdruck2*]
Wartet bis ein bestimmter Wert in *adresse* steht. Dazu wird der Wert aus dem Speicher gelesen, mit *byteausdruck1* 'logisch und', wenn vorhanden mit *byteausdruck2* 'logisch exklusiv-oder' verknüpft und geprüft, ob das Ergebnis gleich 0 ist. Ist das nicht der Fall, so beginnt die Befehlsbearbeitung von neuem.

Siehe auch unter folgenden Funktionen: PEEK, HIMEM.

Befehle zur 128-Kbyte-Verwaltung (CPC 6128)

Die Hauptspeicherkapazität des CPC 6128 von 128 Kbyte ist in zwei Blöcke (Banks) von je 64 Kbyte unterteilt. Der erste Block entspricht dem Speicherbereich der Modelle CPC 464 und 664, der zweite Block kann mit RSX-Befehlen wahlweise als Speicher für Bildschirmseiten oder als RAM-Disk benutzt werden. Um die RSX-Befehle zu aktivieren, so daß sie aufgerufen werden können, muß man die 1A-Diskette einlegen und

RUN "BANKMAN,BAS"

eingeben. Von diesem Zeitpunkt an können die nachfolgend beschriebenen RSX-Befehle mit dem Zeichen "|" am Anfang jedes Befehlswortes wie normale BASIC-Befehle in Programmen genutzt werden.

|**BANKFIND**, *@var%,ausdruck$,anfang,ende*
Sucht in dem mit *anfang* und *ende* festgelegten Datensatzbereich nach einem Datensatz, dessen Inhalt mit *ausdruck$* übereinstimmt. In *var%* steht die Nummer des gefundenen Datensatzes oder ein Fehlercode, falls kein Satz mit *ausdruck$* identisch ist. Als Jokerzeichen, das heißt, als Ersatz für ein beliebiges anderes Zeichen, darf CHR$(0) in *ausdruck$* verwendet werden.

|**BANKOPEN**, *byteausdruck*
Eröffnet eine Datei mit relativem Zugriff mit der Datensatzlänge *byteausdruck*. Die Anzahl der Datensätze ist variabel, wird aber durch die Kapazität des zweiten Speicherblocks von 64 Kbyte begrenzt.

|**BANKREAD**,*@var%*[*,ausdruck*]
Liest den Inhalt des Datensatzes mit der Nummer *ausdruck* in die Variable *var$*. Dieser muß zuvor ein String zugewiesen werden, der genau so viele Leerzeichen enthält, wie der Datensatz lang ist. Wird *ausdruck* weggelassen, bezieht sich die Operation auf den aktuellen Datensatz, auf den der Datensatzzeiger *var%* deutet.

|**BANKWRITE**,*@var%,var$* [*,ausdruck*]
Schreibt den Inhalt von *var$* in den Datensatz mit der Nummer *ausdruck*. Die Variable *var$* muß einen String enthalten, dessen Länge exakt der Datensatzlänge entspricht. Nach dem Schreibvorgang wird der Datensatzzeiger in *var%* automatisch um eins erhöht. Wird *ausdruck* weggelassen, bezieht sich die Operation auf den aktuellen Datensatz, auf den der Datensatzzeiger *var%* deutet.

|**SCREENCOPY**,bildschirmseite a, bildschirmseite b
Kopiert den Inhalt von *bildschirmseite b* in *bildschirmseite a*. Es stehen fünf logische Bildschirmseiten mit Nummern von 1 bis 5 zur Verfügung; Seite 1 ist stets die Anzeigeseite, die auf dem Bildschirm sichtbar ist, die Seiten 2 bis 5 werden im zweiten 64-Kbyte-Block abgelegt.

|**SCREENSWAP**,*bildschirmseite a*,*bildschirmseite b*
Vertauscht die Inhalte der beiden angegebenen Bildschirmseiten miteinander. Es stehen fünf logische Bildschirmseiten mit Nummern von 1 bis 5 zu Verfügung; Seite 1 ist stets die Anzeigeseite, die auf dem Bildschirm sichtbar ist, die Seiten 2 bis 5 werden im zweiten 64-Kbyte-Block abgelegt.

Funktionen

ABS (*ausdruck*)
Liefert den Absolutwert (Betrag) des Ausdrucks.

ASC (*ausdruck*$)
Bestimmt den Wert des ersten Zeichens des Strings.

ATN (*ausdruck*)
Arcustangens in Bogenmaß (RAD aktiv) oder Grad (DEG aktiv).

BIN$ (*intausdruck*,*byteausdruck*)
Erzeugt einen String mit dem binären Wert des vorzeichenlosen *intausdruck*. Mit *byteausdruck* kann eine Länge (Bereich 1–16) angegeben werden, bis zu der der String mit führenden Nullen aufgefüllt wird. Sind weniger Stellen angegeben, als benötigt werden, die Zahl korrekt darzustellen, so wird *byteausdruck* ignoriert.

CHR$ (*byteausdruck*)
Erzeugt einen String aus einem Zeichen des ASCII-Codes.

CINT (*ausdruck*)
Liefert als Integer den gerundeten, ganzzahligen Wert von *ausdruck* im Bereich −32768 bis +32767.

COS (ausdruck)
Cosinus in Bogenmaß (RAD aktiv) oder Grad (DEG aktiv).

CREAL (*ausdruck*)
Gibt einen numerischen Ausdruck in reeller Form zurück.

DEC$ (*ausdruck*,*ausdruck*$) CPC 664
Wandelt *ausdruck* in einen String gemäß dem Format *ausdruck*$ um. Erklärung der Formatzeichen siehe PRINT USING.

DERR CPC 664
Liefert die Nummer des aufgetretenen Floppy-Fehlers wenn ON ERROR GOTO aktiv ist.

EOF
Liefert −1 (logisch wahr) zurück, wenn das Dateiende erreicht wurde.

ERR
Liefert die Nummer des aufgetretenen Fehlers, wenn ON ERROR GOTO aktiv ist. Liste siehe ON ERROR GOTO.

ERL
Liefert die Nummer der fehlerhaften Zeile, wenn ON ERROR GOTO aktiv ist.

EXP (*ausdruck*)
Exponentialfunktion.

FIX (*ausdruck*)
Schneidet die Nachkommastellen einer reellen Zahl ab.

FN var1(*var2*)
Benutzerdefinierte Funktion mit Namen *var1* und Ersatzvariable *var2* (siehe DEF FN).

FRE (*dummy*)
Liefert die Anzahl noch freier Bytes im BASIC-Arbeitsspeicher. Wird als *dummy* eine Stringvariable verwendet, so wird eine 'garbage collection' durchgeführt, d. h., der für Textvariablen verwendete Speicherbereich wird aufgeräumt und neu geordnet.

HEX$ (*adresse*[,*ausdruck*])
Erzeugt einen Hex-String des Werts *adresse*. Mit *ausdruck* (Bereich 0–16) kann die Länge des Strings festgelegt werden.

HIMEM
Liefert die höchste von BASIC erreichbare Speicheradresse.

INKEY (*ausdruck*)
Liefert als Rückgabewert den momentanen Zustand der Taste mit der Nummer *ausdruck*.

Wert	Shift	CTRL	Taste
−1	?	?	nicht ged.
0	nicht ged.	nicht ged.	gedrückt
32	gedrückt	nicht ged.	gedrückt
128	nicht ged.	gedrückt	gedrückt
160	gedrückt	gedrückt	gedrückt

INKEY$
Liefert das Zeichen der momentan gedrückten Taste in einem ein Zeichen langen String zurück. Ist keine Taste gedrückt, so wird ein Nullstring zurückgeliefert.

INP (*adresse*)
Liest einen Wert aus der Eingabeschnittstelle bei *adresse*.

INSTR ([*byteausdruck*,]*ausdruck1*$,*ausdruck2*$)
Sucht *ausdruck2*$ in *ausdruck1*$ ab Position *byteausdruck* (Ersatzwert 1). Ergebnis ist die Position der Übereinstimmung; 0 entspricht nicht gefunden.

INT (*ausdruck*)
Liefert die zu *ausdruck* nächstkleinere ganze Zahl.

JOY ({0/1})
Liefert den momentanen Zustand des Joysticks Nummer 0 oder 1:

Dezimal	Bit	Bedeutung
1	0	Vorwärts
2	1	Rückwärts
4	2	Links
8	3	Rechts
16	4	Feuer 2
32	5	Feuer 1

**LEFT$** (*ausdruck$*,byteausdruck)
Erzeugt einen String mit *byteausdruck* Zeichen von links her aus *ausdruck$*.

LEN (*ausdruck$*)
Errechnet die Länge des Strings.

LOG (*ausdruck*)
Liefert den natürlichen Logarithmus.

LOG10 (*ausdruck*)
Liefert den dekadischen Logarithmus.

**LOWER$** (*ausdruck$*)
Liefert einen String, der *ausdruck$* entspricht; die Großbuchstaben werden jedoch in Kleinbuchstaben umgewandelt.

MAX (*ausdruck,ausdruck*[,*ausdruck*...])
Ermittelt den größten Wert der angegebenen Ausdrücke.

**MID$** (*ausdruck$,byteausdruck1*[,*byteausdruck2*])
Erzeugt einen String aus *ausdruck$* ab Zeichen *byteausdruck1* mit der Länge *byteausdruck2* Zeichen (Ersatzwert ist die noch verbleibende Stringlänge).

MIN (*ausdruck,ausdruck*[,*ausdruck*...])
Ermittelt den kleinsten Wert der angegebenen Ausdrücke.

PEEK (*adresse*)
Liefert den Wert des Inhalts der angegebenen Adresse.

PI
Liefert π = 3.141592653468251.

POS (*dummy*)
Liefert die momentane Position des Cursors in der Zeile.

REMAIN (*uhrnummer*)
Gibt die Restzeit der Uhr *uhrnummer* zurück und schaltet die Funktion (EVERY, AFTER) ab. War die Uhr nicht eingeschaltet. so wird Null zurückgegeben.

**RIGHT$** (*ausdruck$,byteausdruck*)
Erzeugt einen String mit *byteausdruck* Zeichen von rechts her aus *ausdruck$*.

RND (*ausdruck*)
Erzeugt eine Zufallszahl zwischen 0 und 1.

ROUND (*ausdruck*[,*byteausdruck*])
Rundet *ausdruck* auf *byteausdruck* Nachkommastellen (Ersatzwert 0).

SGN (ausdruck)
Vorzeichenfunktion, liefert –1 für negative, 1 für positive Zahlen; 0 für Null.

SIN (*ausdruck*)
Sinur in Bogenmaß (RAD aktiv) oder Grad (DEG aktiv).

SPACE$ (*byteausdruck*)
Liefert einen *byteausdruck* Zeichen langen String aus Leerzeichen zurück.

SPC (*byteausdruck*)
Nur innerhalb der PRINT Anweisung: Druckt *byteausdruck* Leerzeichen.

SQ (*kanal*)
Gibt die Anzahl freier Plätze in der Tonwarteschlange des angegebenen *kanal* und den Status zurück. In der Warteschlange ist Platz für fünf SOUND-Kommandos. Im zurückgegebenen *byteausdruck* haben die Bits folgende Bedeutung:

Bit	Bedeutung
0– 2	Anzahl freie Einträge in Warteschlange
3– 5	ggf. Rendezvousstatus des 1. Eintrags
6	wenn gesetzt: Haltezustand
7	wenn gesetzt: Kanal aktiv

SQR (*ausdruck*)
Quadratwurzel.

STR$ (*ausdruck*)
Wandelt einen numerischen Ausdruck in einen String um.

**STRING$** (*byteausdruck1,ausdruck$*)
Erzeugt einen String aus *byteausdruck1* mal dem ersten Zeichen von *ausdruck$*.

TAB (*byteausdruck*)
Nur innerhalb der PRINT Anweisung: Tabuliert auf Position *byteausdruck*.

TAN (*ausdruck*)
Tangens in Bogenmaß.

TEST (*x,y*)
Liefert die Farbstiftnummer, die an der Position *x,y* verwendet wurde.

TESTR (*xr,yr*)
Liefert die Farbstiftnummer, die an der Position relativ zur aktuellen Position *xr, yr* verwendet wurde.

TIME
Liefert die seit dem Einschalten vergangene Zeit in 1/300 Sekunden.

UNT (*adresse*)
Gibt als Integer den gerundeten, ganzzahligen Wert von *adresse* (Bereich –32768 bis +65535) zurück. Werte größer als 32767 werden dabei in entsprechende negative umgewandelt (Zweierkomplement).

**UPPER$** (*ausdruck$*)
Liefert einen String, der *ausdruck$* entspricht; die Kleinbuchstaben werden jedoch in Großbuchstaben umgewandelt.

VAL (*ausdruck$*)
Liefert den Wert einer im String vorkommenden Konstante, sofern diese an der ersten Position im String beginnt.

VPOS (*#eaeinheit*)
Gibt die vertikale Position des Textcursors von *eaeinheit* zurück.

XPOS
Liefert die X-Koordinate des Grafikcursors.

YPOS
Liefert die Y-Koordinate des Grafikcursors.

& *hexzahl*
Wandelt eine Hexadezimalzahl (z. B. eine Speicher- oder Portadresse) in eine Dezimalzahl um.

Operatoren

Mit einem Operanden

−*ausdruck*	Ergibt negativen Wert von *ausdruck*.
NOT *ausdruck*	Ergibt −1 wenn *ausdruck* gleich 0, in allen anderen Fällen 0 (logische Negation).

Mit zwei Operanden

AND	Logisches Und.
OR	Logisches Oder.
XOR	Logisches Exklusiv-Oder.
=	Wertzuweisung bzw. logischer Test auf Gleichheit.
<> ><	Test auf ungleich.
<	Test auf kleiner.
<= =<	Test auf kleiner-gleich.
>	Test auf größer.
>= =>	Test auf größer-gleich.
+	Addition.
−	Subtraktion.
*	Multiplikation.
/	Division.
↑	Potenzierung.

Datentypen

Standardtypen

Gleitkommavariable, Darstellbare Werte von $2.9*10^{\wedge}-39$ (kleinster positiver Wert) bis $1.7*10^{\wedge}+38$ (größter positiver Wert) (analog für negative Werte).

% Ganzzahlvariable, Rechenbereich −32768 bis +32767.

$ Stringvariable mit 0 bis 255 Zeichen Länge.

Verbindungen von Einzeldaten

{*var/var$*} (*ausdruck*[,*ausdruck...*])
Mit der Anweisung DIM angelegtes, ein- oder mehrdimensionales Feld zur Speicherung von Datenlisten.

Assembler

Begriffe

Adresse
Eine Folge von zwei Bytes (LSB/MSB), die zusammen eine Speicherposition im Bereich von 0 ($0000) bis 65535 ($FFFF) angeben (256*LSB+MSB).

Adressierungsart
Art des Zugriffs auf den Speicher durch einen Assemblerbefehl.

Argument
Bestandteil eines Befehls, welcher den Wert festlegt, mit dem die angegebene Operation ausgeführt wird.

Assembler
Der Assembler ist ein Programm, welches die Mnemocodes und angegebenen Adressierungsarten in die interne Darstellung im Rechner umsetzt. Komfortable Assembler erlauben dabei auch die Definition von Lables und Macros.

Akku
Hauptrechenregister der CPU für arithmetische Operationen und Datenaustausch.

Befehl
Ein Assemblerbefehl besteht aus einem Befehlscode, welcher die Art der Operation festlegt und meist noch aus einem oder mehreren Operanden.

Bit
Kleinste in einem Computer speicherbare Einheit. Entspricht einem Zustand 0 oder 1 bzw. AUS oder EIN.

Byte
Folge von acht Bits. Alle Register und alle Speicherstellen des Computers sind jeweils ein Byte lang und können damit Werte zwischen 0 und 255 oder ein beliebiges Zeichen speichern.

CPU
Abk. für 'Central Processing Unit' (Zentraleinheit). Die CPU erkennt und verarbeitet die Assemblerbefehle.

Flag
Ein Flag ist ein Bit, das eine logische Entscheidung speichert, z. B. das Zeroflag speichert, ob der letzte geprüfte Wert gleich Null war.

Interrupt
Peripheriegeräte wie z. B. eine Diskettenstation oder interne Bausteine wie z. B. ein Uhrenbaustein (Timer) können sogenannte Interrupts auslösen, welche die normale Programmausführung unterbrechen, um eine spezielle Bearbeitungsroutine durchzuführen. Danach wird die Bearbeitung des normalen Programms fortgesetzt.

Interruptflagregister
Register der CPU, in welchem gespeichert wird, woher die Unterbrechungsanforderung (interrupt request) kam.

LSB (*l*east *s*ignificant *b*yte)
Im Speicher in zwei aufeinanderfolgenden Bytes abgelegte Werte (z. B. für indirekte Adressierung) werden immer mit dem niederwertigen (LSB) vor dem höherwertigen (MSB) Byte gespeichert (256*LSB+MSB).

LSN (*l*east *s*ignificant *n*ibble)
Die vier niederwertigen Bits (0–3) in einem Byte.

Mnemocode
Abkürzung für den Befehlsnamen. Bsp. LD für Load.

MSB (*m*ost *s*ignificant *b*yte)
Das höherwertige zweier Bytes. Siehe LSB.

MSN (*m*ost *s*ignificant *n*ibble)
die vier höherwertigen Bits (4–7) in einem Byte.

Programmzeiger (program counter)
Internes 16-bit-Register der CPU, welches die Adresse des momentan zu bearbeitenden Befehls enthält.

Register
Speicher innerhalb der CPU, in dem Werte für schnellen Zugriff und für spezielle Operationen gehalten werden. Es gibt folgende Register:

Name	Länge	Verwendung
A	8	arithmetische Operationen
B	8	Zählregister, Wertespeicherung
BC	16	Zeiger, Wertespeicherung
DE	16	Zeiger, Wertespeicherung
F	8	Flagregister, Statusregister
HL	16	Zeiger
I	8	Interruptflagregister
IX	16	Indexregister
IY	16	Indexregister
PC	16	Befehlszähler
SP	16	Stapelzeiger
A'–L'	div.	Alternativer Registersatz

Signflag
Bit des Statusregisters, welches anzeigt, ob ein Wert negativ, d. h. sein höchstwertigstes Bit gesetzt ist.

Stapel (stack)
Ein vom Benutzer festgelegter Speicherbereich ist der Prozessorstapel. Er wird für das Zwischenspeichern von Daten sowie Rücksprungadressen bei Unterprogrammaufrufen und Interrupts benötigt.

Stapelzeiger (stackpointer)
Internes 16-bit-Register, welches die Adresse der momentan aktuellen Daten im Stapel enthält.

Statusregister

Internes Register, welches über den momentanen Systemzustand Aufschluß gibt.

Bit	Name	Gesetzt, wenn ...
0	C	bei einer arithm. Operation ein Übertrag auftrat.
1	N	die letzte Operation eine Subtraktion war.
2	P/V	Parität ungerade, oder Überlauf bei arithm. Operationen.
4	H	bei einer arithm. Operation ein Übertrag von Bit 3 nach 4 auftrat (für BCD).
6	Z	das Ergebnis der Operation Null ist.
7	S	das höchstwertigste Bit des Ergebnisses Eins ist.

Übertragsflag

Bit des Statusregisters für arithmetische Operationen. Es speichert den Übertrag bei Addition und Subtraktion.

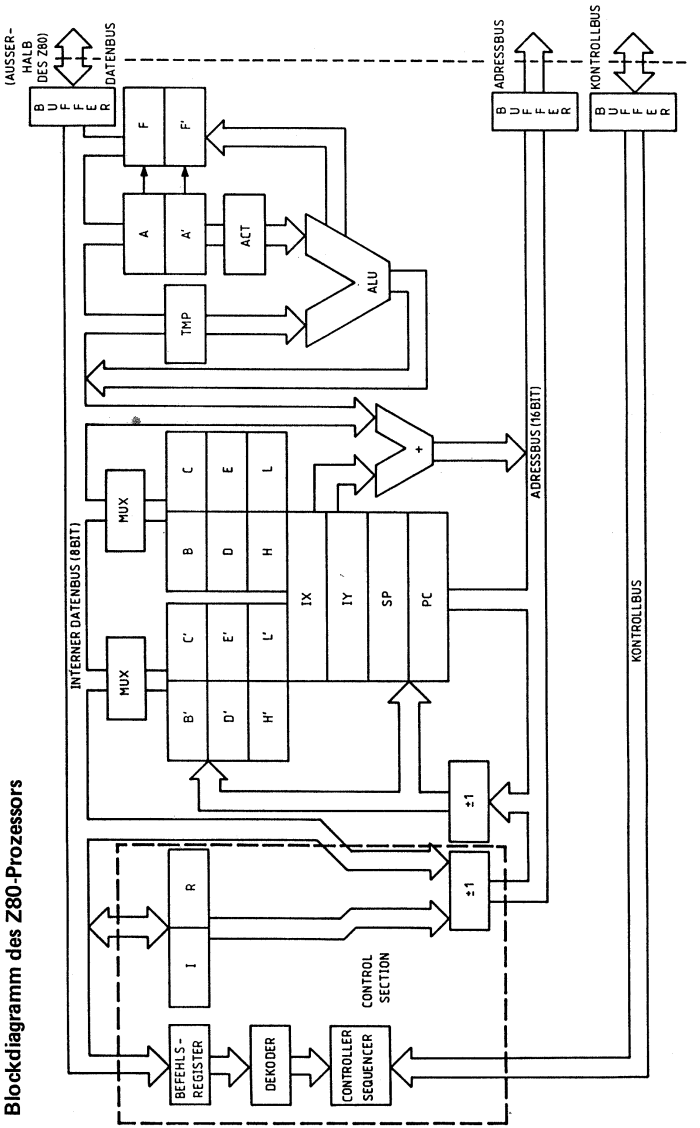
Zeroflag

Bit des Statusregisters, welches gesetzt wird, wenn der Wert eines Registers (d. h. alle 8 Bits) gleich Null ist.

Notation

Register

A	Akkumulator
B–E	8-bit-Datenregister
F	Flagregister, Statusregister
H/L	16-bit-Doppelregister für indirekten Speicherzugriff
IX	Indexregister
IY	Indexregister
M	Memory, Speicher
P	Prozessorstatusregister
SP	Stackpointer, Stapelzeiger
PC	Program Counter, Programmzähler
PCL	PC low (niederwertig)
PCH	PC high (hochwertig)
ADL	Adresse low (LSB)
ADH	Adresse high (MSB)
DATA	Datenregisterinhalt
ir	Indexregister (IX oder IY)



Blockdiagramm des Z80-Prozessors

Quelle: Zilog

Statusflags

N	Additions/Subtraktionsflag
Z	Nullflag (Zero)
C	Übertragsflag (Carry)
I	Interruptflag
P/V	Paritäts- oder Überlaufflag
H	Halbübertragsflag (half Carry)
S	Vorzeichenflag (Sign)
–	Not, Invertierung
0	Null, Bit gelöscht
1	Eins, Bit gesetzt

Operationen

→	Transfer nach
←	Transfer nach
↑	Transfer vom Stack
↓	Transfer auf den Stack
+	Addition
–	Subtraktion
∧	Logisch Und (AND)
∨	Logisch Oder (OR)
∀	Logisch Exklusiv-Oder (EOR)
=	Nimmt den Wert an

Adressierungsarten

abs	absolut
imp	implizit
( )	indirekt
(ir+d)	indirekt-indiziert
rel	relativ
imm	unmittelbar

Befehle (Mnemonics)

Speicherung

LD *rr, (nn)*
Lädt das Registerpaar rr mit dem Inhalt der durch nn adressierten Speicherstelle.

LD *rr,nn*
Lädt das Registerpaar rr direkt mit dem Zweibytewert nn.

LD *r,n*
Lädt das Register r mit dem Wert n.

LD *(BC),A*
Schreibt den Wert aus dem Akku in die Speicherstelle, auf die das BC-Register zeigt.

LD *(DE),A*
Schreibt den Wert aus dem Akku in die Speicherstelle, auf die das DE-Register zeigt.

LD *(HL),n*
Schreibt den Wert n in die Speicherstelle, auf die das HL-Register zeigt.

LD *(HL),r*
Schreibt den Inhalt des Registers r in die Speicherstelle, auf die das HL-Register zeigt.

LD *r,(ir+d)*
Lädt das Register r indirekt aus der indiziert adressierten Speicherstelle (ir+d).

LD *(ir+d),n*
Speichert den Wert n in der indiziert adressierten Speicherstelle (ir+d).

LD *(ir+d),r*
Speichert den Inhalt des Registers r in der indiziert adressierten Speicherstelle (ir+d).

LD *A,(nn)*
Lädt den Inhalt der Speicherstelle nn in den Akku.

LD *(nn),A*
Schreibt den Inhalt des Akkus in die Speicherstelle nn.

LD *(nn),rr*
Speichert den Inhalt des Registerpaars rr in die Speicherstellen nn und nn+1.

LD *(nn),HL*
Speichert den Inhalt des Registerpaars HL in die Speicherstellen nn und nn+1.

LD *(nn),ir*
Speichert den Inhalt des 16-bit-Indexregisters ir in die Speicherstellen nn und nn+1.

LD *A,(BC)*
Lädt den Akku mit dem Inhalt der Speicherstelle auf die das BC-Register zeigt.

LD *A,(DE)*
Lädt den Akku mit dem Inhalt der Speicherstelle auf die das DE-Register zeigt.

LD *HL,(nn)*
Lädt den Inhalt der Speicherstellen nn und nn+1 ins HL-Register.

LD *Ir,nn*
Lädt den 16-bit-Wert nn in das Indexregister ir.

LD *ir,(nn)*
Lädt den Inhalt der Speicherstellen nn und nn+1 in das Indexregister ir.

LD *r,(HL)*
Lädt das Register r mit dem Inhalt der Speicheradresse in HL.

LD *SP,HL*
Lädt den Stapelzeiger mit dem Inhalt des HL-Registers.

LD *SP,ir*
Lädt den Stapelzeiger mit dem Inhalt des Indexregisters ir.

LDD
Speichert den Inhalt der Speicherstelle, auf die das HL-Register zeigt, in die Speicherstelle, auf die das DE-Register zeigt. Danach werden das DE- und das HL-Register dekrementiert.

LDDR
Speichert den Inhalt der Speicherstelle, auf die das HL-Register zeigt, in die Speicherstelle, auf die das DE-Register zeigt. Danach werden das BC-, DE- und das HL-Register dekrementiert. Dieser Befehl wird solange wiederholt, bis der Inhalt des BC-Registers gleich Null ist.

LDI
Speichert den Inhalt der Speicherstelle, auf die das HL-Register zeigt, in die Speicherstelle, auf die das DE-Register zeigt. Danach werden das DE-Register inkrementiert und das HL-Register dekrementiert.

LDIR
Speichert den Inhalt der Speicherstelle, auf die das HL-Register zeigt. in die Speicherstelle, auf die das DE-Register zeigt. Danach werden das DE-Register inkrementiert und das BC- und das HL-Register dekrementiert. Dieser Befehl wird solange wiederholt, bis der Inhalt des BC-Registers gleich Null ist.

Registertransfer

EX *AF,AF'*
Vertauscht die Inhalte von Akkumulator und Flagregister mit dem der entsprechenden Zweitregister.

EX *DE,HL*
Vertauscht die Inhalte von DE- und HL-Register miteinander.

EXX
Vertauscht die Inhalte der BC-, DE- und HL-Register mit dem der jeweiligen Zweitregister.

LD *r,r'*
Lädt das Register r mit dem Inhalt von Register r'.

LD *A,I*
Lädt den Akku mit dem Inhalt des Interruptvektorregisters I.

LD *I,A*
Speichert den Inhalt des Akku im Interruptvektorregister I.

LD *A,R*
Lädt den Akku mit dem Inhalt des Memory-Refresh Register.

LD *R,A*
Speichert den Inhalt des Akku in das Memory-Refresh Register.

Peripherie

IN *r,(C)*
Lädt Register r aus Port (C).

IN *A,(C)*
Lädt Akkumulator aus Port (C).

IND
Speichert an der Adresse, auf die das HL-Register zeigt, den Wert, der von dem Port geladen wird, dessen Adresse im C-Register steht. Anschließend werden das Register B und das Registerpaar HL dekrementiert.

INDR
Speichert an der Adresse, auf die das HL-Register zeigt, den Wert, der von dem Port geladen wird, dessen Adresse im C-Register steht. Anschließend werden das Register B und das Registerpaar HL dekrementiert. Dieser Befehl wird solange wiederholt, bis der Inhalt des B-Registers gleich Null ist.

INI
Speichert an der Adresse, auf die das HL-Register zeigt, den Wert, der von dem Port geladen wird, dessen Adresse im C-Register steht. Anschließend wird das Registerpaar HL inkrementiert und das Register B dekrementiert.

INIR
Speichert an der Adresse, auf die das HL-Register zeigt, den Wert, der von dem Port geladen wird, dessen Adresse im C-Register steht. Anschließend wird das Registerpaar HL inkrementiert und das Register B dekrementiert. Dieser Befehl wird solange wiederholt, bis der Inhalt des B-Registers gleich Null ist.

OTDR
Gibt den Wert, der in der Adresse auf die das HL-Register zeigt steht, auf dem Port aus, dessen Adresse im C-Register steht. Anschließend werden das Register B und das Registerpaar HL dekrementiert. Dieser Befehl wird solange wiederholt, bis der Inhalt des B-Registers gleich Null ist.

OTIR
Gibt den Wert, der in der Adresse auf die das HL-Register zeigt steht, auf dem Port aus, dessen Adresse im C-Register steht. Anschließend wird das Registerpaar HL inkrementiert und das Register B dekrementiert. Dieser Befehl wird solange wiederholt, bis der Inhalt des B-Registers gleich Null ist.

OUT *(C),r*
Gibt den Inhalt des Registers r auf dem Port aus, dessen Adresse im Register C steht.

OUT *(C),A*
Gibt den Inhalt des Akkumulators auf dem Port aus, dessen Adresse im Register C steht.

OUTD
Gibt den Wert, der in der Adresse auf die das HL-Register zeigt steht, auf dem Port aus, dessen Adresse im C-Register steht. Anschließend werden das Register B und das Registerpaar HL dekrementiert.

OUTI
Gibt den Wert, der in der Adresse auf die das HL-Register zeigt steht, auf dem Port aus, dessen Adresse im C-Register steht. Anschließend wird das Registerpaar HL inkrementiert und das Register B dekrementiert.

Stapelbeeinflussung

EX *(SP),HL*
Vertauscht den Inhalt des HL-Registers mit dem Inhalt des obersten Stapelelements.

EX *(SP),ir*
Vertauscht den Inhalt des Indexregisters ir mit dem Inhalt des obersten Stapelelements.

LD *SP,HL*
Lädt den Stapelzeiger mit dem Inhalt des HL-Registers.

PUSH *qq*
Bringt den Inhalt des Registerpaares qq auf den Stapel.

PUSH *ir*
Bringt den Inhalt des Indexregisters ir auf den Stapel.

POP *qq*
Holt den Inhalt des Registerpaares qq vom Stapel.

POP *ir*
Holt den Inhalt des Indexregisters ir vom Stapel.

Arithmetik

ADC *A,s*
Addiert s zum Inhalt des Akkus. War das Übertragsflag gesetzt, so wird 1 hinzugezählt. Ist das Ergebnis größer als 255, so wird das Übertragsflag gesetzt. s kann sein: r, n, (HL), (ir+d).

ADC *HL,rr*
Addiert den Inhalt des Registerpaares rr zum Inhalt des HL-Registers. War das Übertragsflag gesetzt, so wird 1 hinzugezählt. Ist das Ergebnis größer als 65535, so wird das Übertragsflag gesetzt.

ADD *A,s*
Addiert s zum Inhalt des Akkus. s kann sein: r, n, (HL), (ir+d).

ADD *HL,rr*
Addiert den Inhalt des Registerpaares rr zum Inhalt des HL-Registers.

ADD *ir,rr*
Addiert den Inhalt des Registerpaares rr zum Inhalt des Indexregisters ir.

CCF
Komplementiert das Übertragsflag.

CPL
Komplementiert den Akkumulator.

DAA
Führt eine BCD-Korrektur des Akkumulators durch.

NEG
Negiert den Akkumulator.

DEC *m*
Verringert den Inhalt von m um 1. War der Inhalt 0, so ist der neue Wert 255. m kann sein: r, (HL), (ir+d).

DEC *rr*
Verringert den Inhalt des Registerpaares rr um 1. War der Inhalt 0, so ist der neue Wert 65535.

DEC *ir*
Verringert den Inhalt des Indexregisters ir um 1. War der Inhalt 0, so ist der neue Wert 65535.

INC *m*
Erhöht den Inhalt von m um 1. Findet ein Überlauf statt, so ist der neue Wert 0. m kann sein: r, (HL), (ir+d).

INC *rr*
Verringert den Inhalt des Registerpaares rr um 1. Findet ein Überlauf statt, so ist der neue Wert 0.

INC *ir*
Verringert den Inhalt des Indexregisters ir um 1. Findet ein Überlauf statt, so ist der neue Wert 0.

SBC *A,s*
Subtrahiert s vom Akku. War das Übertragsflag gelöscht, so wird zusätzlich 1 abgezogen. Ist das Ergebnis kleiner als 0, so wird das Übertragsflag gelöscht. s kann sein: r, n, (HL), (ir+d).

SBC *HL,ss*
Subtrahiert das Registerpaar ss vom Akku ohne Übertrag. ss kann sein: BC, DE, HL, SP.

SCF
Setzt das Übertragsflag.

SLA *s*
Schiebt alle Bits um eine Position nach links. Das höchstwertige Bit wird ins Übertragsflag übertragen; von rechts wird eine Null nachgeschoben. Dies entspricht einer Multiplikation mit 2. s kann sein: r, (HL), (ir+d).

SRA *s*
Schiebt alle Bits um eine Position nach rechts. Das niederwertigste Bit wird ins Übertragsflag übertragen; Bit 7 bleibt unverändert. s kann sein: r, (HL), (ir+d).

SRL *s*
Schiebt alle Bits um eine Position nach rechts. Das niederwertigste Bit wird ins Übertragsflag übertragen; von links wird eine Null nachgeschoben. Die Operation entspricht einer Division durch 2, wobei der Rest ins Übertragsflag übertragen wird. s kann sein: r, (HL), (ir+d).

SUB *s*
Subtrahiert s vom Akku, ohne einen Übertrag zu berücksichtigen oder zu erzeugen. s kann sein: r, n, (HL), (ir+d).

Vergleiche

CP *s*
Vergleicht den Inhalt des Akkus mit s. s kann sein: r, n, (HL), (ir+d).

CPD
Vergleicht Akku und Inhalt der Speicherstelle, deren Adresse im HL-Register steht und dekrementiert dann HL- und BC-Register.

CPDR
Vergleicht Akku und den Inhalt der Speicherstelle, deren Adresse im HL-Register steht und dekrementiert dann HL- und BC-Register. Dies wird solange wiederholt, bis BC=0 oder der Vergleich Übereinstimmung ergibt.

CPI
Vergleicht Akku und Inhalt der Speicherstelle, deren Adresse im HL-Register steht, inkrementiert das HL-Register und dekrementiert das BC-Register.

CPDR
Vergleicht Akku und den Inhalt der Speicherstelle, deren Adresse im HL-Register steht, inkrementiert das HL-Register und dekrementiert das BC-Register. Dies wird solange wiederholt, bis BC=0 oder der Vergleich Übereinstimmung ergibt.

Verzweigungen

CALL *bed,adr*
Aufruf eines Unterprogramms ab Speicherstelle adr, wenn die Bedingung bed erfüllt ist.

bed	Bedeutung
NZ	nicht gleich Null
Z	gleich Null
NC	Übertrag nicht gesetzt
C	Übertrag gesetzt
PO	ungerade Parität
PE	gerade Parität
P	positiv
M	negativ

CALL *adr*
Unbedingter Aufruf des Unterprogramms ab Speicherstelle adr.

DJNZ
Dekrementiert den Inhalt des Registers B. Ist das Ergebnis ungleich Null, so wird der Offset mit Zweier-Komplement-Arithmetik zum Befehlszähler addiert (so daß auch Rückwärtssprünge möglich sind) und damit ein relativer Sprung ausgeführt.

JP *bed,adr*
Springt, wenn die Bedingung bed (siehe CALL) erfüllt ist, nach adr.

JP *adr*
Unbedingter Sprung an die Adresse adr.

JP *(HL)*
Unbedingter Sprung an die Adresse, die im HL-Register steht.

JP *(ir)*
Unbedingter Sprung an die Adresse, die im Indexregister ir steht.

JR *bed,diff*
Springt an die Adresse PC+2+diff (wenn Bit 7 in diff gesetzt ist, so wird 256 abgezogen), wenn die Bedingung bed erfüllt ist.

JR *diff*
Springt unbedingt an die Adresse PC+2+diff (wenn Bit 7 in diff gesetzt ist, so wird 256 abgezogen).

RET
Rückkehr von einem Unterprogramm, d. h. der Programmzeiger wird auf die Position drei Bytes nach der Adresse des aufrufenden CALL gestellt.

RET *bed*
Bedingte Rückkehr von einem Unterprogramm, d. h. der Programmzeiger wird auf die Position drei Bytes nach der Adresse des aufrufenden CALL gestellt, wenn die Bedingung bed erfüllt ist.

RETI
Rückkehr von einer Interruptroutine. Der Programmzeiger wird wiederhergestellt.

RETN
Rückkehr von einem nicht maskierbaren Interrupt. Der Programmzeiger und das Interruptflagregister (I) wird wiederhergestellt.

RST *p*
Schneller Unterprogrammaufruf. p kann sein: 00H bis 38H in acht Byte Schritten.

Bitmanipulationen

AND *s*
Verknüpft s bitweise 'logisch UND' mit dem Akku. s kann sein: r, n, (HL), (ir+d).

BIT *b,(HL)*
Testet das Bit b der Speicherstelle, deren Adresse im HL-Register steht und setzt das Zeroflag.

BIT *b,(ir)*
Testet das Bit b der Speicherstelle, deren Adresse im Indexregister ir steht und setzt das Zeroflag.

BIT *b,r*
Testet das Bit b im Register r und setzt das Zeroflag.

OR *s*
Verknüpft s bitweise 'logisch ODER' mit dem Akku. s kann sein: r, n, (HL), (ir+d).

RES *b,s*
Setzt Bit b in s zurück. s kann sein: r, n, (HL), (ir+d).

RL *s*
Verschiebt die Bits in s zyklisch um 1 nach links durch das Übertragsflag. Das Übertragsflag wird ins niederwertigste Bit übertragen und nimmt sodann das höchstwertige Bit auf. s kann sein: r, n, (HL), (ir+d).

RLA
Verschiebt die Bits im Akku zyklisch um 1 nach links durch das Übertragsflag. Das Übertragsflag wird ins niederwertigste Bit übertragen und nimmt sodann das höchstwertige Bit auf.

RLC *r*
Verschiebt die Bits im Register r zyklisch um 1 nach links durch das Übertragsflag. Das Übertragsflag nimmt das höchstwertige Bit auf.

RLC *(HL)*
Verschiebt die Bits der Speicherstelle, auf die das HL-Register zeigt, zyklisch um 1 nach links. Das Übertragsflag nimmt das höchstwertige Bit auf.

RLC *(ir)*
Verschiebt die Bits der Speicherstelle, auf die das Indexregister ir zeigt, zyklisch um 1 nach links. Das Übertragsflag nimmt das höchstwertige Bit auf.

RLCA
Verschiebt die Bits im Akku zyklisch um 1 nach links. Das Übertragsflag nimmt das höchstwertige Bit auf.

RLD
Verschiebt die Bits in der Speicherstelle, auf die das HL-Register zeigt, und das LSN im Akku dezimal nach links, d. h. das LSN des Akku kommt in das LSN der Speicherstelle. Das dortige LSN kommt nach MSN und das MSN nach LSN des Akku.

RR *s*
Verschiebt die Bits in s zyklisch um 1 nach rechts durch das Übertragsflag. Das Übertragsflag wird ins höchstwertigste Bit übertragen und nimmt sodann das niederwertige Bit auf. s kann sein: r, n, (HL), (ir+d).

RRA
Verschiebt die Bits im Akku zyklisch um 1 nach rechts durch das Übertragsflag. Das Übertragsflag wird ins höchstwertige Bit übertragen und nimmt sodann das niederwertigste Bit auf. Das Argument bezeichnet die Adresse, auf die sich der Befehl bezieht. Bsp.: LD (2000H) – Es wird der Inhalt von Speicherstelle 2000 (hexadezimal) geladen.

RRC *r*
Verschiebt die Bits im Register r zyklisch um 1 nach rechts durch das Übertragsflag. Das Übertragsflag nimmt das niederwertigste Bit auf.

RRC *(HL)*
Verschiebt die Bits der Speicherstelle, auf die das HL-Register zeigt, zyklisch um 1 nach rechts. Das Übertragsflag nimmt das niederwertige Bit auf.

RRC *(ir)*
Verschiebt die Bits der Speicherstelle, auf die das Indexregister ir zeigt, zyklisch um 1 nach rechts. Das Übertragsflag nimmt das niederwertige Bit auf.

RRCA
Verschiebt die Bits im Akku zyklisch um 1 nach rechts. Das Übertragsflag nimmt das niederwertige Bit auf.

RRD
Verschiebt die Bits in der Speicherstelle, auf die das HL-Register zeigt, und das LSN im Akku dezimal nach rechts, d. h. das LSN des Akku kommt in das MSN der Speicherstelle. Das dortige MSN kommt nach LSN und das LSN nach LSN des Akku.

XOR *s*
Verknüpft s bitweise 'logisch EXKLUSIV ODER' mit dem Akku. s kann sein: r, n, (HL), (ir+d).

Interruptbehandlung

EI
Läßt die Ausführung von Interrupts wieder zu (enable interrupt).

DI
Verhindert die Ausführung von Interrupts (disable interrupt).

IM *0*
Setzt den Interrupt-Modus 0. Der unterbrechende Baustein kann einen Befehl zur Ausführung auf den Datenbus legen (8080-Modus).

IM *1*
Setzt Interrupt-Modus 1. Wenn ein Interrupt auftritt, wird der Befehl RST 0038H ausgeführt.

IM *2*
Setzt Interrupt-Modus 2. Tritt ein Interrupt auf, so wird ein Byte, das der unterbrechende Baustein auf dem Datenbus liefert, als LSB und der Inhalt von Register I als MSB des Sprungzieles eines sofort auszuführenden Sprungs genommen.

Verschiedenes

HALT
Hält die CPU an. Es werden solange NOP-Befehle durchgeführt, bis ein Interrupt oder Reset eintritt.

NOP
Es wird keine Operation durchgeführt (no Operation).

Adressierungsarten

ABSOLUT – absolute; *abs*
Das Argument bezeichnet direkt die Zieladresse. Bsp.: LDA, 2000

IMPLIZIT – implied; *imp*
Der Befehl hat kein Argument. Die Art der Operation geht aus dem Befehl selbst hervor. Bsp.: CCF – Das Übertragsflag wird komplementiert.

INDIREKT – indirect; *( )*
Das Argument bezeichnet ein Register, in dem das Argument für den Befehl zu finden ist. Bsp.: INC (HL) – Inkrementieren der Speicherstelle, deren Adresse im HL-Register steht.

INDIREKT INDIZIERT – indirect indexed; *(ir+d)*
Um die tatsächliche Adresse zu erhalten, wird zum Argument (Distanz d) der Inhalt des Indexregisters ir (IX oder IY) dazugezählt. Bsp.: LD A, (IX+2).

RELATIV – relative; *rel*
Die Zieladresse der (relativen) JR-Befehle ergibt sich aus der Summe der momentanen Adresse des Programmzählers+2 und dem (ein Byte langen) Argument, von der bei gesetztem Bit 7 256 subtrahiert werden. Damit können Sprünge 126 Byte rückwärts und 129 Byte vorwärts programmiert werden. Bsp.: JR 5 – Sprung um 5 Bytes vorwärts.

UNMITTELBAR – immediate; *imm*
Das Argument ist keine Adresse, sondern direkt der Wert für die angegebene Operation.
Bsp.: OR 80H – Es wird ein OR mit dem Wert 80 (hexadezimal) ausgeführt.

Befehlssatz des Z80 Prozessors

Code (hex)	Assemblerbefehl	
8E	ADC	A,(HL)
DD8E05	ADC	A,(IX+d)
FD8E05	ADC	A,(IY+d)
8F	ADC	A,A
88	ADC	A,B
89	ADC	A,C
8A	ADC	A,D
8B	ADC	A,E
8C	ADC	A,H
8D	ADC	A,L
CE20	ADC	A,n
ED4A	ADC	HL,BC
ED5A	ADC	HL,DE
ED6A	ADC	HL,HL
ED7A	ADC	HL,SP
86	ADD	A,(HL)
DD8605	ADD	A,(IX+d)
FD8605	ADD	A,(IY+d)
87	ADD	A,A
80	ADD	A,B
81	ADD	A,C
82	ADD	A,D
83	ADD	A,E
84	ADD	A,H
85	ADD	A,L
C620	ADD	A,n
09	ADD	HL,BC
19	ADD	HL,DE
29	ADD	HL,HL
39	ADD	HL,SP
DD09	ADD	IX,BC
DD19	ADD	IX,DE
DD29	ADD	IX,IX
DD39	ADD	IX,SP
FD09	ADD	IY,BC
FD19	ADD	IY,DE
FD29	ADD	IY,IY
FD39	ADD	IY,SP
A6	AND	(HL)
DDA605	AND	(IX+d)
FDA605	AND	(IY+d)
A7	AND	A
A0	AND	B
A1	AND	C
A2	AND	D
A3	AND	E
A4	AND	H
A5	AND	L

Code (hex)	Assemblerbefehl	
E620	AND	n
CB46	BIT	0,(HL)
DDCB0546	BIT	0,(IX+d)
FDCB0546	BIT	0,(IY+d)
CB47	BIT	0,A
CB40	BIT	0,B
CB41	BIT	0,C
CB42	BIT	0,D
CB43	BIT	0,E
CB44	BIT	0,H
CB45	BIT	0,L
CB4E	BIT	1(HL)
DDCB054E	BIT	1,(IX+d)
FDCB054E	BIT	1,(IY+d)
CB4F	BIT	1,A
CB48	BIT	1,B
CB49	BIT	1,C
CB4A	BIT	1,D
CB4B	BIT	1,E
CB4C	BIT	1,H
CB4D	BIT	1,L
CB56	BIT	2,(HL)
DDCB0556	BIT	2,(IX+d)
FDCB0556	BIT	2,(IY+d)
CB57	BIT	2,A
CB50	BIT	2,B
CB51	BIT	2,C
CB52	BIT	2,D
CB53	BIT	2,E
CB54	BIT	2,H
CB55	BIT	2,L
CB5E	BIT	3,(HL)
DDCB055E	BIT	3,(IX+d)
FDCB055E	BIT	3,(IY+d)
CB5F	BIT	3,A
CB58	BIT	3,B
CB59	BIT	3,C
CB5A	BIT	3,D
CB5B	BIT	3,E
CB5C	BIT	3,H
CB5D	BIT	3,L
CB66	BIT	4,(HL)
DDCB0566	BIT	4,(IX+d)
FDCB0566	BIT	4,(IX+d)
CB67	BIT	4,A
CB60	BIT	4,B
CB61	BIT	4,C
CB62	BIT	4,D

Code (hex)	Assemblerbefehl	
CB63	BIT	4,E
CB64	BIT	4,H
CB65	BIT	4,L
CB6E	BIT	5,(HL)
DDCB056E	BIT	5,(IX+d)
FDCB056E	BIT	5,(IY+d)
CB6F	BIT	5,A
CB68	BIT	5,B
CB69	BIT	5,C
CB6A	BIT	5,D
CB6B	BIT	5,E
CB6C	BIT	5,H
CB6D	BIT	5,L
CB76	BIT	6,(HL)
DDCB0576	BIT	6,(IX+d)
FDCB0576	BIT	6,(IY+d)
CB77	BIT	6,A
CB70	BIT	6,B
CB71	BIT	6,C
CB72	BIT	6,D
CB73	BIT	6,E
CB74	BIT	6,H
CB75	BIT	6,L
CB7E	BIT	7,(HL)
DDCB057E	BIT	7,(IX+d)
FDCB057E	BIT	7,(IY+d)
CB7F	BIT	7,A
CB78	BIT	7,B
CB79	BIT	7,C
CB7A	BIT	7,D
CB7B	BIT	7,E
CB7C	BIT	7,H
CB7D	BIT	7,L
DC8405	CALL	C,nn
FC8405	CALL	M,nn
D48405	CALL	NC,nn
C48405	CALL	NZ,nn
F48405	CALL	P,nn
EC8405	CALL	PE,nn
E48405	CALL	PO,nn
CC8405	CALL	Z,nn
CD8405	CALL	nn
3F	CCF	
BE	CP	(HL)
DDBE05	CP	(IX+d)
FDBE05	CP	(IY+d)
BF	CP	A
B8	CP	B
B9	CP	C
BA	CP	D
BB	CP	E
BC	CP	H
BD	CP	L
FE20	CP	n
EDA9	CPD	
EDB9	CPDR	

Code (hex)	Assemblerbefehl	
EDB1	CPIR	
EDA1	CPI	
2F	CPL	
27	DAA	
35	DEC	(HL)
DD3505	DEC	(IX+d)
FD3505	DEC	(IY+d)
3D	DEC	A
05	DEC	B
06	DEC	BC
0D	DEC	C
15	DEC	D
18	DEC	DE
1D	DEC	E
25	DEC	H
2B	DEC	HL
DD2B	DEC	IX
FD2B	DEC	IY
2D	DEC	L
38	DEC	SP
F3	DI	
T02E	DJNZ	I
FB	EI	
E3	EX	(SP),HL
DDE3	EX	(SP),IX
FDE3	EX	(SP),IY
08	EX	AF,AF
EB	EX	DE,HL
D9	EXX	
76	HALT	
ED46	IM	0
ED56	IM	1
ED5E	IM	2
ED78	IN	A,(C)
ED40	IN	B,(C)
ED48	IN	C,(C)
ED50	IN	D,(C)
ED58	IN	E,(C)
ED60	IN	H,(C)
ED68	IN	L,(C)
34	INC	(HL)
DD3405	INC	(IX+d)
FD3405	INC	(IY+d)
3C	INC	A
04	INC	B
03	INC	BC
0C	INC	C
14	INC	D
13	INC	DE
1C	INC	E
24	INC	H
23	INC	HL
DD23	INC	IX
FD23	INC	L
2C	INC	L
33	INC	SP
DB20	IN	a,(n)

Code (hex)	Assemblerbefehl	
EDAA	IND	DD7E05
EDBA	INDR	
FDA2	INI	
EDB2	INIR	
C38405	JP	nn
E9	JP	(HL)
DDE9	JP	(IX)
FDE9	JP	(IY)
DA8405	JP	C,nn
FA8405	JP	M,nn
D28405	JP	NC,nn
C28405	JP	NZ,nn
F28405	JP	P,nn
EA8405	JP	PE,nn
E28405	JP	PO,nn
CA8405	JP	Z,nn
382E	JH	C,p
302E	JR	NC,p
202E	JR	NZ,e
282E	JR	Z,e
182E	JR	e
02	LD	(BC),A
12	LD	(DE),A
77	LD	(HL),A
70	LD	(HL),B
71	LD	(HL),C
72	LD	(HL),D
73	LD	(HL),E
74	LD	(HL),H
75	LD	(HL),L
3620	LD	(HL),n
DD7705	LD	(IX+d),A
DD7005	LD	(IX+d),B
DD7105	LD	(IX+d),C
DD7205	LD	(IX+d),D
DD7305	LD	(IX+d),E
DD7405	LD	(IX+d),H
DD7505	LD	(IX+d),L
DD360520	LD	(IX+d),n
FD7705	LD	(IY+d),A
FD7005	LD	(IY+d),B
FD7105	LD	(IY+d),C
FD7205	LD	(IY+d),D
FD7305	LD	(IY+d),E
FD7405	LD	(IY+d),H
FD7505	LD	(IY+d),L
FD360520	LD	(IY+d),n
328405	LD	(nn),A
ED438405	LD	(nn),BC
ED538405	LD	(nn),DE
228405	LD	(nn),HL
DD228405	LD	(nn),IX
FD228405	LD	(nn),IY
ED738405	LD	(nn),SP
0A	LD	A,(BC)
1A	LD	A,(DE)
7E	LD	A,(HL)

Code (hex)	Assemblerbefehl	
DD7E05	LD	A,(IX+d)
FD7E05	LD	A,(IY+d)
3A8405	LD	A,(nn)
7F	LD	A,A
78	LD	A,B
79	LD	A,C
7A	LD	A,D
7B	LD	A,E
7C	LD	A,H
ED57	LD	A,I
7D	LD	A,L
3E20	LD	A,n
ED5F	LD	A,R
46	LD	B,(HL)
DD4605	LD	B,(IX+d)
FD4605	LD	B,(IY+d)
47	LD	B,A
40	LD	B,B
41	LD	B,C
42	LD	B,D
43	LD	B,E
44	LD	B,H
45	LD	B,L
0620	LD	B,n
ED4B8405	LD	BC,(nn)
018405	LD	BC,nn
4E	LD	C,(HL)
DD4E05	LD	C,(IX+d)
FD4E05	LD	C,(IY+d)
4F	LD	C,A
48	LD	C,B
49	LD	C,C
4A	LD	C,D
4B	LD	C,E
4C	LD	C,H
4D	LD	C,L
0E20	LD	C,n
56	LD	D,(HL)
DD5605	LD	D,(IX+d)
FD5605	LD	D,(IY+d)
57	LD	D,A
50	LD	D,B
51	LD	D,C
52	LD	D,D
53	LD	D,E
54	LD	D,H
55	LD	D,L
1620	LD	D,n
ED588405	LD	DE,(nn)
118405	LD	DE,nn
5E	LD	E,(HL)
DD5E05	LD	E,(IX+d)
FD5E05	LD	E,(IY+d)
5F	LD	E,A
58	LD	E,B
59	LD	E,C
5A	LD	E,D

Code (hex)	Assemblerbefehl	
5B	LD	E,E
5C	LD	E,H
5D	LD	E,L
1E20	LD	E,n
66	LD	H,(HL)
DD6605	LD	H,(IX+d)
FD6605	LD	H,(IY+d)
67	LD	H,A
60	LD	H,B
61	LD	H,C
62	LD	H,D
63	LD	H,E
64	LD	H,H
65	LD	H.L
2620	LD	H,n
2A8405	LD	HL,(nn)
218405	LD	HL,nn
ED47	LD	I,A
DD2A8405	LD	IX,(nn)
DD218405	LD	IX,nn
FD2A8405	LD	IY,(nn)
FD218405	LD	IY,nn
6E	LD	L,(HL)
DD6E05	LD	L,(IX+d)
FD6E05	LD	L,(IY+d)
6F	LD	L,A
68	LD	L,B
69	LD	L,C
6A	LD	L,D
6B	LD	L,E
6C	LD	L,H
6D	LD	L,L
2E20	LD	L,n
ED4F	LD	R,A
ED7B8405	LD	SP,(nn)
F9	LD	SP,HL
DDF9	LD	SP,IX
FDF9	LD	SP,IY
318405	LD	SP,nn
EDA8	LDD	
EDB8	LDDR	
EDA0	LDI	
EDB0	LDIR	
ED44	NEG	
00	NOP	
B6	OR	(HL)
DDB605	OR	(IX+d)
FDB605	OR	(IY+d)
B7	OR	A
B0	OR	B
B1	OR	C
B2	OR	D
B3	OR	E
B4	OR	H
B5	OR	L
F620	OR	n
ED8B	OTDR	

Code (hex)	Assemblerbefehl	
EDB3	OTIR	
ED79	OUT	(C),A
ED41	OUT	(C),B
ED49	OUT	(C),C
ED51	OUT	(C),D
ED59	OUT	(C),E
ED61	OUT	(C),H
ED69	OUT	(C),L
D320	OUT	(n),A
EDAB	OUTD	
EDA3	OUTI	
F1	POP	AF
C1	POP	BC
D1	POP	DE
E1	POP	HL
DDE1	POP	IX
FDE1	POP	IY
F5	PUSH	AF
C5	PUSH	BC
D5	PUSH	DE
E5	PUSH	HL
DDE5	PUSH	IX
FDE5	PUSH	IY
CB86	RES	0,(HL)
DDCB0586	RES	0,(IX+d)
FDCB0586	RES	0,(IY+d)
CB87	RES	0,A
CB80	RES	0,B
CB81	RES	0,C
CB82	RES	0,D
CB83	RES	0,E
CB84	RES	0,H
CB85	RES	0,L
CB8E	RES	1,(HL)
DDCB058E	RES	1,(IX+d)
FDCB058E	RES	1,(IY+d)
CB8F	RES	1,A
CB88	RES	1,B
CB89	RES	1,C
CB8A	RES	1,D
CB8B	RES	1,E
CB8C	RES	1,H
CB8D	RES	1,L
CB96	RES	2,(HL)
DDCB0596	RES	2,(IX+d)
FDCB0596	RES	2,(IY+d)
CB97	RES	2,A
CB90	RES	2,B
CB91	RES	2,C
CB92	RES	2,D
CB93	RES	2,E
CB94	RES	2,H
CB95	RES	2,L
CB9E	RES	3,(HL)
DDCB059E	RES	3,(IX+d)
FDCB059E	RES	3,(IY+d)

Code (hex)	Assemblerbefehl	
CB9F	RES	3,A
CB98	RES	3,B
CB99	RES	3,C
CB9A	RES	3,D
CB9B	RES	3,E
CB9C	RES	3,H
CB9D	RES	3,L
CBA6	RES	4,(HL)
DDCB05A6	RES	4,(IX+d)
FDCB05A6	RES	4,(IY+d)
CBA7	RES	4,A
CBA0	RES	4,B
CBA1	RES	4,C
CBA2	RES	4,D
DBA3	RES	4,E
CBA4	RES	4,H
CBA5	RES	4,L
CBAE	RES	5,(HL)
DDCB05AE	RES	5,(IX+d)
FDCB05AE	RES	5,(IY+d)
CBAF	RES	5,A
CBA8	RES	5,B
CBA9	RES	5,C
CBAA	RES	5,D
CBAB	RES	5,E
CBAC	RES	5,H
CBAD	RES	5,L
CBB6	RES	6,(HL)
DDCB05B6	RES	6,(IX+d)
FDCB05B6	RES	6,(IY+d)
CBB7	RES	6,A
CBB0	RES	6,B
CBB1	RES	6,C
CBB2	RES	6,D
CBB3	RES	6,E
CBB4	RES	6,H
CBB5	RES	6,L
CBBE	RES	7,(HL)
DDCB05BE	RES	7,(IX+d)
FDCB05BE	RES	7,(IY+d)
CBBF	RES	7,A
CBB8	RES	7,B
CBB9	RES	7,C
CBBA	RES	7,D
CBBB	RES	7,E
CBBC	RES	7,H
CBBD	RES	7,L
C9	RET	
D8	RET	C
E8	RET	M
D0	RET	NC
C0	RET	NZ
F0	RET	P
E8	RET	PE
E8	RET	PE
E0	RET	PO
C8	RET	Z

Code (hex)	Asselblerbefehl	
ED4D	RETI	
ED45	RETN	
CB16	RL	(HL)
DDCB0516	RL	(IX+d)
FDCB0516	RL	(IY+d)
CB17	RL	A
CB10	RL	B
CB11	RL	C
CB12	RL	D
CB13	RL	E
CB14	RL	H
CB15	RL	L
17	RLA	
CB06	RLC	(HL)
DDCB0506	RLC	(IX+d)
FDCB0506	RLC	(IY+d)
CB07	RLC	A
CB00	RLC	B
CB01	RLC	C
CB02	RLC	D
CB03	RLC	E
CB04	RLC	H
CB05	RLC	L
07	RLCA	
ED6F	RLD	
CB1E	RR	(HL)
DDCB051E	RR	(IX+d)
FDCB051E	RR	(IY+d)
CB1F	RR	A
CB18	RR	B
CB19	RR	C
CB1A	RR	D
CB1B	RR	E
CB1C	RR	H
CB1D	RR	L
1F	RRA	
CB0E	RRC	(HL)
DDCB050E	RRC	(IX+d)
FDCB050E	RRC	(IY+d)
CB0F	RRC	A
CB08	RRC	B
CB09	RRC	C
CB0A	RRC	D
CB0B	RRC	E
CB0C	RRC	H
CB0D	RRC	L
OF	RRCA	
OF	RRCA	
ED67	RRD	
C7	RST	00H
CF	RST	08H
D7	RST	10H
DF	RST	18H
E7	RST	20H
EF	RST	28H
F7	RST	30H
FF	RST	38H
DE20	SBC	A,n

Code (hex)	Assemblerbefehl	
9E	SBC	A,(HL)
DD9E05	SBC	A,(IX+d)
FD9E05	SBC	A,(IY+d)
9F	SBC	A,A
98	SBC	A,B
99	SBC	A,C
9A	SBC	A,D
9B	SBC	A,E
9C	SBC	A,H
9D	SBC	A,L
ED42	SBC	HL,BC
ED52	SBC	HL,DE
ED62	SBC	HL,HL
ED72	SBC	HL,SP
37	SCF	
CBC6	SET	0,(HL)
DDCB05C6	SET	0,(IX+d)
FDCB05C6	SET	0,(IY+d)
CBC7	SET	0,A
CBC0	SET	0,B
CBC1	SET	0,C
CBC2	SET	0,D
CBC3	SET	0,E
CBC4	SET	0,H
CBC5	SET	0,L
CBCE	SET	1,(HL)
DDCB05CE	SET	1,(IX+d)
FDCB05CE	SET	1,(IY+d)
CBCF	SET	1,A
CBC8	SET	1,B
CBC9	SET	1,C
CBCA	SET	1,D
CBCB	SET	1,E
CBCC	SET	1,H
CBCD	SET	1,L
CBD6	SET	2,(HL)
DDCB05D6	SET	2,(IX+d)
FDCB05D6	SET	2,(IY+d)
CBD7	SET	2,A
CBD0	SET	2,B
CBD1	SET	2,C
CBD2	SET	2,D
CBD3	SET	2,E
CBD4	SET	2,H
CBD5	SET	2,L
CBD8	SET	3,B
CBDE	SET	3,(HL)
DDCB05DE	SET	3,(IX+d)
FDCB05DE	SET	3,(IY+d)
CBDF	SET	3,A
CBD9	SET	3,C
CBDA	SET	3,D
CBDB	SET	3,E
CBDC	SET	3,H
CBDD	SET	3,L
CBE6	SET	4,(HL)

Code (hex)	Assemblerbefehl	
DDCB05E6	SET	4,(IX+d)
FDCB05E6	SET	4,(IY+d)
CBE7	SET	4,A
CBE0	SET	4,B
CBE1	SET	4,C
CBE2	SET	4,D
CBE3	SET	4,E
CBE4	SET	4,H
CBE5	SET	4,L
CBEE	SET	5,(HL)
DDCB05EE	SET	5,(IX+d)
FDCB05EE	SET	5,(IY+d)
CBEF	SET	5,A
CBE8	SET	5,B
CBE9	SET	5,C
CBEA	SET	5,D
CBEB	SET	5,E
CBEC	SET	5,H
CBED	SET	5,L
CBF6	SET	6,(HL)
DDCB05F6	SET	6,(IX+d)
FDCB05F6	SET	6,(IY+d)
CBF7	SET	6,A
CBF0	SET	6,B
CBF1	SET	6,C
CBF2	SET	6,D
CBF3	SET	6,E
CBF4	SET	6,H
CBF5	SET	6,L
CBFE	SET	7,(HL)
DDCB05FE	SET	7,(IX+d)
FDCB05FE	SET	7,(IY+d)
CBFF	SET	7,A
CBF8	SET	7,B
CBF9	SET	7,C
CBFA	SET	7,D
CBFB	SET	7,E
CBFC	SET	7,H
CBFD	SET	7,L
CB26	SLA	(HL)
DDCB0526	SLA	(IX+d)
FDCB0526	SLA	(IY+d)
CB27	SLA	A
CB20	SLA	B
CB21	SLA	C
CB22	SLA	D
CB23	SLA	E
CB24	SLA	H
CB25	SLA	L
CB2E	SRA	(HL)
DDCB052E	SRA	(IX+d)
FDCB052E	SRA	(IY+d)
CB2F	SRA	A
CB28	SRA	B
CB29	SRA	C
CB2A	SRA	D

Code (hex)	Assemblerbefehl	
CB2B	SRA	E
CB2C	SRA	H
CB2D	SRA	L
CB3E	SRL	(HL)
DDCB053E	SRL	(IX+d)
FDCB053E	SRL	(IY+d)
CB3F	SRL	A
CB38	SRL	B
CB39	SRL	C
CB3A	SRL	D
CB3B	SRL	E
CB3C	SRL	H
CB3D	SRL	L
96	SUB	(HL)
DD9605	SUB	(IX+d)
FD9605	SUB	(IY+d)
97	SUB	A
90	SUB	B

Code (hex)	Assemblerbefehl	
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
D620	SUB	n
AE	XOR	(HL)
DDAE05	XOR	(IX+d)
FDAE05	XOR	(IY+d)
AF	XOR	A
A8	XOR	B
A9	XOR	C
AA	XOR	D
AB	XOR	E
AC	XOR	H
AD	XOR	L
EE20	XOR	n

Quelle: Zilog

Boolesche Operationen

Wahrheitstafeln:

AND (Und)

∧	0	1
0	0	0
1	0	1

OR (Oder)

∨	0	1
0	0	1
1	1	1

EOR (Exklusiv-Oder)

⊻	0	1
0	0	1
1	1	0

NOT (Negation)

–	
0	1
1	0

Speicherbelegung

Hauptspeicherverteilung

0 – 16383 0000 – 3FFF Betriebssystem ROM.
0 – 40959 4000 – BFFF BASIC RAM-Bereich.
40960 – 65535 C000 – FFFF Video RAM/Basic ROM

BASIC-Vektoren (Auszug)

Adresse dez.	Adresse hex.	Bedeutung
43904	AB80	Zeichenmatrix für die Zeichen 240 bis 255
44032	AC00	Blanks ignorieren-Flag
44033	AC01	JMP READY-Modus
44036	AC04	JMP ERROR
44039	AC07	JMP Befehl ausführen
44042	AC0A	JMP Berechnung einer Funktion
44045	AC0D	JMP holen einer Konstante
44048	AC10	JMP Zeile in Token umwandeln ...
44051	AC13	JMP Token listen
44054	AC16	JMP Ziffern umwandeln
44057	AC19	JMP Bearbeitung von Operatoren
44060	AC1C	AUTO-Modus Flag
44061	AC1D	AUTO Zeilennummer
44063	AC1F	AUTO Inkrement
44066	AC22	Eingabekanal
44071	AC27	Wert der aktuellen FOR-NEXT-Variablen
44076	AC2C	Adresse des aktuellen NEXT-Befehls
44078	AC2E	Adresse des aktuellen WEND-Befehls
44084	AC34	Adresse der Fehlerbehandlungsroutine ...
44088	AC38	SOUND-Queue #0
44100	AC44	SOUND-Queue #1
44112	AC50	SOUND-Queue #2
44196	ACA4	Eingabepuffer
44454	ADA6	Adresse der fehlerhaften Zeile
44456	ADA8	Zeiger in das Programm nach Fehler
44458	ADAA	Nummer des aktuellen Fehlers
44459	ADAB	Zeiger in das Programm nach Break
44461	ADAD	Zeiger auf die Zeile nach Break
44463	ADAF	Adresse der Fehlerbehandlungsroutine
44465	ADB1	Fehlerbehandlungsroutine aktiv
44466	ADB2	SOUND Parameter
44596	AE34	Adresse der aktuellen Anweisung
44598	AE36	Adresse der aktuellen Programmzeile
44607	AE3F	Startadresse für LOAD
44609	AE41	CHAIN-MERGE-Flag

Adresse dez.	Adresse hex.	Bedeutung
44658	AE72	CALL-Befehlsadresse
44667	AE7B	Zeiger auf die höchste BASIC-Speicheradresse
44669	AE7D	Zeiger auf das Ende des freien RAM-Bereichs
44671	AE7F	Zeiger auf den Anfang des freien RAM-Bereichs
44673	AE81	Zeiger auf den Anfang des BASIC-Programms
44675	AE83	Zeiger auf das Ende des BASIC-Programms
44677	AE85	Zeiger auf den Anfang der Variablenliste
44679	AE87	Zeiger auf den Anfang der Feldliste
44681	AE89	Zeiger auf das Ende der Feldliste
44683	AE8B	Stapel für BASIC
45195	B08B	Basic Stackpointer
45197	B08D	Zeiger auf den Anfang des Stringbereichs
45199	B08F	Zeiger auf das Ende des Stringbereichs
45210	B09A	Zeiger in den Stapel der Stringdefinitionen
45212	B09C	Start des Stringstapels
45242	B0BA	Stringbeschreibung
45249	B0C1	Aktueller Variablentyp
45250	B0C2	Wert oder Zeiger auf eine Variable
45580	B20C	Aktuelles Bildschirmfenster
45581	B20D	Parameter für Fenster 0
45596	B21C	Parameter für Fenster 1–7 (bis B285)
45701	B285	Aktuelle Cursorzeile
45702	B286	Aktuelle Cursorspalte
45703	B287	0=Ganzer Bildschirm, sonst Fenster
45704	B288	Grenzen des aktuellen Fensters (o,l,u,r)
45711	B28F	Aktuelle Zeichenfarbe
45712	B290	Aktuelle Hintergrundfarbe
45763	B2C3	Sprungtabelle für die Steuerzeichen
45864	B328	ORIGIN (x,y)
45868	B32C	X und Y Koordinaten
46321	B4F1	Joystick 1
46324	B4F4	Joystick 0
46401	B541	Zeiger auf Tastaturcodetabelle
46403	B543	Zeiger auf SHIFT-Codes-Tabelle
46405	B545	Zeiger auf Control-Codes-Tabelle
46407	B547	Zeiger auf die Tabelle der Repeatfunktion
46418	B552	Aktuelle SOUND Aktivität
46428	B55C	Parameter für SOUND Kanal A
46491	B59B	Parameter für SOUND Kanal B
46552	B5DA	Parameter für SOUND Kanal C
46602	B60A	Lautstärke Hüllkurven
46842	B6FA	Tonhüllenkurven
47104	B800	Kassettenmeldungen ausgeben ja/nein
47106	B802	Status des Kassetteneingabepuffers
47107	B803	Startadresse des Eingabepuffers
47108	B804	Zeiger in den Eingabepuffer

Adresse dez.	Adresse hex.	Bedeutung
47111	B807	Dateikopf der Eingabedatei
47175	B847	Status des Kassettenausgabepuffers
47176	B848	Startadresse des Ausgabepuffers
47178	B84A	Zeiger in den Ausgabepuffer
47180	B84C	Dateikopf der Ausgabedatei
47313	B8D1	Geschwindigkeit der Kassettenaufzeichnung
47325	B8DD	Flag für Einfügen/Überschreiben

Jumptable

Adresse dez.	Adresse hex.	Bedeutung
47872	BB00	Initialisierung der Tastatur.
47875	BB03	Reset der Tastatur.
47878	BB06	Warten auf ein Zeichen von der Tastatur.
47881	BB09	Zeichen von der Tastatur holen.
47884	BB0C	Zeichen im Tastaturpuffer speichern.
47887	BB0F	Platz für weitere Zeichen schaffen.
47890	BB12	Zeichen vom Tastaturpuffer holen.
47893	BB15	Speicher für Tastaturpuffer zuweisen.
47896	BB18	Warte auf einen Tastendruck.
47899	BB1B	Holen der Tastennummer.
47902	BB1E	Testen, ob Taste gedrückt.
47905	BB21	Teste Shift-Taste.
47908	BB24	Lese Joystick-Port.
47911	BB27	Trage Wert in Tastaturtabelle ein.
47914	BB2A	Lese Wert aus der Tastaturtabelle.
47917	BB2D	Eintragen der Shift-Codes.
47920	BB30	Lese Shift-Code.
47923	BB33	Eintragen eines Control-Codes.
47926	BB36	Lesen eines Control-Codes.
47929	BB39	Setzt die Tastenwiederholfunktion.
47932	BB3C	Prüft die Tastenwiederholfunktion.
47935	BB3F	Setzt Ansprechzeit und Frequenz der Wiederholfunktion.
47938	BB42	Liest Ansprechzeit und Frequenz.
47941	BB45	Entriegelt die Break-Taste.
47944	BB48	Verriegelt die Break-Taste.
47947	BB4B	Wenn BREAK-Taste gedrückt, Routine ausführen.
47950	BB4E	Bildschirm Initialisieren.
47953	BB51	Bildschirm rücksetzen (RESET).
47956	BB54	Bildschirmausgabe einschalten.
47959	BB57	Bildschirmausgabe abschalten.
47962	BB5A	Zeichen ausgeben (Steuerzeichen ausführen).
47965	BB5D	Zeichen auf Bildschirm ausgeben.
47968	BB60	Zeichen vom Bildschirm lesen.

Adresse dez.	Adresse hex.	Bedeutung
47971	BB63	Darstellung von Steuerzeichen ein/ausschalten.
47974	BB66	Legt die Größe des Textfensters fest.
47977	BB69	Liest die Größe des momentanen Fensters.
47980	BB6C	Löscht das momentane Fenster.
47983	BB6F	Setze Spalte.
47986	BB72	Setze Zeile.
47989	BB75	Setze Cursor.
47992	BB78	Lese Cursorposition.
47995	BB7B	Cursor entriegeln (für Benutzer).
47998	BB7E	Cursor verriegeln (für Benutzer).
48001	BB81	Cursor entriegeln (Betriebssystem).
48004	BB84	Cursor verriegeln (Betriebssystem).
48007	BB87	Teste, ob Cursor sichtbar.
48010	BB8A	Cursor einschalten.
48013	BB8D	Cursor ausschalten.
48016	BB90	Setzt die Vordergrundfarbe.
48019	BB93	Liest die Vordergrundfarbe.
48022	BB96	Setzt die Hintergrundfarbe.
48025	BB99	Liest die Hintergrundfarbe.
48028	BB9C	Vertauscht Vorder- und Hintergrundfarbe.
48031	BB9F	Schaltet Transparentmodus um.
48034	BBA2	Prüft, ob Transparentmodus gesetzt.
48037	BBA5	Adresse eines Zeichens aus Zeichensatz holen.
48040	BBA8	Adresse für Darstellung eines Zeichens setzen.
48043	BBAB	Startadresse und erstes Zeichen setzen.
48046	BBAE	Startadresse und erstes Zeichen prüfen.
48049	BBB1	Holt Adresse der Control-Jumptable.
48052	BBB4	Auswählen eines Textfensters.
48055	BBB7	Zwei Textfenster miteinander vertauschen.
48058	BBBA	Grafik initialisieren.
48061	BBBD	Grafik rücksetzen (RESET).
48064	BBC0	Grafikcursor absolut bewegen.
48067	BBC3	Grafikcursor relativ bewegen.
48070	BBC6	Aktuelle Grafikcursorposition lesen.
48073	BBC9	ORIGIN setzen.
48076	BBCC	ORIGIN lesen.
48079	BBCF	Setze Begrenzung des Grafikfensters horizontal.
48082	BBD2	Setze Begrenzung des Grafikfensters vertikal.
48085	BBD5	Lese Begrenzung des Grafikfensters horizontal.
48088	BBD8	Lese Begrenzung des Grafikfensters vertikal.
48091	BBDB	Löscht das Grafikfenster.
48094	BBDE	Setze Farbe für Zeichnen.
48097	BBE1	Liest aktuelle Farbe zum Zeichnen.
48100	BBE4	Setzt Hintergrundfarbe.
48103	BBE7	Liest die aktuelle Hintergrundfarbe.
48106	BBEA	Setzt Grafikpunkt absolut.

Adresse dez.	Adresse hex.	Bedeutung
48109	BBED	Setzt Grafikpunkt relativ.
48112	BBF0	Testet Grafikpunkt absolut.
48115	BBF3	Testet Grafikpunkt relativ.
48118	BBF6	Absolutes Zeichnen einer Linie.
48121	BBF9	Relatives Zeichnen einer Linie.
48124	BBFC	Schreibt ein Zeichen an die Grafikcursorposition.
48127	BBFF	Video-Initialisierung.
48130	BC02	Video-Reset.
48133	BC05	Setze Startadresse im Video-RAM.
48136	BC08	Setze Startadresse (Basisadresse) des Video-RAM.
48139	BC0B	Lese absolute Bildschirmstartadresse.
48142	BC0E	Setze den Bildschirmmodus.
48145	BC11	Lese den Bildschirmmodus.
48148	BC14	Lösche den Bildschirm.
48151	BC17	Zeilen- und Spaltenzahl lesen.
48154	BC1A	Char-Position.
48157	BC1D	Dot-Position.
48160	BC20	Bildschirmadresse inkrementieren.
48163	BC23	Bildschirmadresse dekrementieren.
48166	BC26	Bildschirmzeile inkrementieren.
48169	BC29	Bildschirmzeile dekrementieren.
48172	BC2C	INK ENCODE.
48175	BC2F	INK DECODE.
48178	BC32	Ordnet einer INK eine Farbe zu.
48181	BC35	Liest die Farbe einer INK.
48184	BC38	Setzt die Rahmenfarbe.
48187	BC3B	Liest die Rahmenfarbe.
48190	BC3E	Setzt die Blinkfrequenz.
48193	BC41	Liest die Blinkfrequenz.
48196	BC44	Füllt ein Bildschirmfenster mit Farbe.
48199	BC47	desgl., modusunabhängig.
48202	BC4A	Vertauscht Vorder- und Hintergrundfarbe eines Zeichens.
48205	BC4D	Hardware-Scrolling.
48208	BC50	Software-Scrolling.
48211	BC53	Vergrößert die Zeichenmatrix in Modus 0 oder 1.
48214	BC56	Zeichenmatrix verkleinern.
48217	BC59	Steuerzeichen sichtbar/unsichtbar setzen.
48220	BC5C	Setzt einen Punkt auf dem Bildschirm.
48223	BC5F	Zieht eine horizontale Linie.
48226	BC62	Zieht eine vertikale Linie.
48229	BC65	Kassette initialisieren.
48232	BC68	Setzt die Schreibgeschwindigkeit.
48235	BC6B	Schaltet Meldungen ein/aus.
48238	BC6E	Schaltet Motor ein.
48241	BC71	Schaltet Motor aus.
48244	BC74	Stellt alten Motorzustand wieder her.
48247	BC77	OPENIN.

Adresse dez.	Adresse hex.	Bedeutung
48250	BC7A	CLOSEIN.
48253	BC7D	Eingabedatei schließen (sofort).
48256	BC80	Liest ein Zeichen aus dem Puffer.
48259	BC83	Liest eine ganze Datei.
48262	BC86	Schiebt das letzte Zeichen wieder in den Puffer.
48265	BC89	Testet, ob das Dateiende (EOF) erreicht ist.
48268	BC8C	OPENOUT.
48271	BC8F	CLOSEOUT.
48274	BC92	Schließt sofort die Ausgabedatei.
48277	BC95	Schreibt ein Zeichen in den Puffer.
48280	BC98	Schreibt eine ganze Datei.
48283	BC9B	Erzeugt einen Kassettenkatalog.
48286	BC9E	Schreibt einen Block auf Kassette.
48289	BCA1	Liest einen Block von Kassette.
48292	BCA4	Führt einen Blockvergleich (Verify) durch.
48295	BCA7	Musik rücksetzen (RESET).
48298	BCAA	Reiht eine Note in die Warteschlange ein.
48301	BCAD	Prüft, ob Platz in der Warteschlange ist.
48304	BCB0	
48307	BCB3	Musik (wieder) spielen lassen.
48310	BCB6	Musik anhalten.
48313	BCB9	Angehaltene Töne weiterspielen.
48316	BCBC	Einrichten der Lautstärkenhüllkurve.
48319	BCBF	Einrichten der Tonhüllkurve.
48322	BCC2	Liest die Adresse für eine Lautstärkenhüllkurve.
48325	BCC5	Liest die Adresse für eine Tonhüllkurve.
48403	BD13	BOOT: Betriebssystem rücksetzen und JP (HL).
48406	BD16	Startet ein Programm.
48409	BD19	Wartet auf Zeilenrücklauf des Monitors.
48412	BD1C	Modus setzen.
48415	BD1F	SCREEN OFFSET.
48418	BD22	Farben (INKs) löschen.
48421	BD25	Farben setzen.
48424	BD28	Drucker zurücksetzen.
48427	BD2B	Zeichen drucken.
48430	BD2E	Prüft, ob Drucker noch arbeitet.
48433	BD31	Sendet Zeichen (wartet bis fertig).
48436	BD34	Sendet Daten an Sound-Controller.
48439	BD37	Alle Sprungvektoren initialisieren.
48442	BD3A	EDIT.
48445	BD3D	REAL Variable kopieren.
48448	BD40	Umwandlung von INT in REAL.
48451	BD43	Umwandlung eines 4-byte-Werts in REAL.
48454	BD46	Umwandlung von FLOAT nach INT.

Adresse dez.	Adresse hex.	Bedeutung
48457	BD49	
48460	BD4C	FIX
48463	BD4F	INT
48466	BD52	
48469	BD55	Komma um a Stellen nach rechts schieben.
48472	BD58	+
48475	BD5B	–
48478	BD5E	–
48481	BD61	*
48484	BD64	/
48487	BD67	*2^a
48490	BD6A	Vergleich
48493	BD6D	– (Vorzeichenwechsel)
48496	BD70	SGN
48499	BD73	DEG/RAD Umschaltung
48502	BD76	PI
48505	BD79	SQR
48508	BD7C	^ (Potenzierung)
48511	BD7F	LOG
48514	BD82	LOG10
48517	BD85	EXP
48520	BD88	SIN
48523	BD8B	COS
48526	BD8E	TAN
48529	BD91	ATN
48532	BD94	
48535	BD97	Zufallsgenerator initialisieren (RANDOMIZE).
48538	BD9A	RND-Seed setzen
48541	BD9D	RND
48544	BDA0	Holt letzten RND-Wert noch einmal.
48547	BDA3	
48550	BDA6	
48553	BDA9	
48556	BDAC	+
48559	BDAF	–
48562	BDB2	–
48565	BDB5	*
48568	BDB8	/ (Ganzzahldivision)
48571	BDBB	MOD
48574	BDBE	* (ohne Vorzeichen)
48577	BDC1	/ (ohne Vorzeichen)
48580	BDC4	Vergleich
48583	BDC7	Vorzeichenwechsel
48586	BDCA	SGN

Adresse dez.	Adresse hex.	Bedeutung
		INDIRECTIONS
48589	BDCD	Stellt Cursor auf dem Bildschirm dar.
48592	BDD0	Löscht den Cursor auf dem Bildschirm.
48595	BDD3	Gibt ein Zeichen auf dem Bildschirm aus.
48598	BDD6	Liest ein Zeichen vom Bildschirm.
48601	BDD9	Gibt ein Zeichen aus / bearbeitet Control-Code.
48604	BDDC	Setzt einen Grafikpunkt.
48607	BDDF	Prüft, ob ein Punkt gesetzt ist.
48610	BDE2	Zeichnet eine Linie.
48613	BDE5	SCR READ.
48616	BDE8	Löscht den Bildschirm mit Tinte 0.
48619	BDEB	
48622	BDEE	Prüft, ob die BREAK-Taste gedrückt ist.
48625	BDF1	Gibt ein Zeichen auf Drucker aus.

Grafik

Video Controller HD 6845/Gate Array

Beschreibung

Der Video Controller (auch Cathode Ray Tube Controller = CRTC genannt) und das Gate Array (ein Spezialbaustein für den CPC) sind für den Aufbau des Fernsehbildes zuständig. Sie verwalten die farbige Zeichenausgabe und die hochauflösende Grafik. Um die Farb- und Zeicheninformationen abzulegen, benötigen sie 16 KByte im Hauptspeicher.

Registerauflistung Gate Array

Das Gate Array wird über die Portadresse $7F00 erreicht. Dazu wird zunächst übertragen, welches interne Register beschrieben werden soll:

Bit 6	Bit 7	Adressiertes Register
0	0	Farbnummer Register
0	1	Multifunktionsregister
1	0	Farbwert-Register

Danach wird zum selben Port der Wert übertragen, der in das betreffende Register gelangen soll. Auf alle Register kann nur schreibend zugegriffen werden.

Bit	Bedeutung im Multifunktionsregister					
0–1	Bildschirmmodus					
	Bit 1	Bit 0	Modus	Text (S,Z)	Grafik (Z,S)	Farben
	0	0	0	20 x 25	160 x 200	16
	0	1	1	40 x 25	320 x 200	4
	1	0	2	80 x 25	640 x 200	2
	1	1	0	20 x 25	160 x 200	16
2	ROM $0000–$3FFF einschalten (0) oder abschalten (1)					
3	ROM $C000–$FFFF einschalten (0) oder abschalten (1)					
4	V-Sync-Zähler rücksetzen					

Registerauflistung HD 6845

Der Video Controller wird über die Portadressen $BC00 (Adressregister) und $BD00 (Datenregister) erreicht. Dazu wird zunächst über den Port $BC00 die Nummer des internen Registers übertragen und danach über Port $BD00 der zu schreibende Wert.

Register	Bedeutung		
Adresse	Port $BC00 5-bit-Register für Registerauswahl		
0	Anzahl der Zeichen pro Zeile		
1	Anzahl der darzustellenden Zeichen pro Zeile		
2	Horizontale Sync-Position		
3	Breite der Sync-Impulse im LSN		
4	Anzahl der Rasterzeilen pro Bild		
5	Bits 0–5 für Bildwiederholfrequenzfeinabgleich		
6	Anzahl der darzustellenden Rasterzeilen		
7	Vertikale Sync-Position		
8	Bits 0–1 legen den Interlace-Modus fest		
9	Bits 0–4: Anzahl der Rasterzeilen pro Zeichen		
10	Bits 0–4: Startrasterzeile für Cursor Bits 5–6: Darstellungsmodus des Cursors		
	Bit6	Bit5	Modus
	0	0	Cursor steht
	0	1	Cursor nicht sichtbar
	1	0	Cursor blinkt 3x pro Sekunde
	1	1	Cursor blinkt 3x pro zwei Sekunden
11	Bits 0–4 legen die Endrasterzeile des Cursors fest		
12	Startadresse (H) des Bildspeichers im 16K-Bereich		
13	Startadresse (L) des Bildspeichers im 16K-Bereich		
14	Startadresse (H) der momentanen Cursorposition		
15	Startadresse (L) der momentanen Cursorposition		
16	akt. Bildschirmspeicheradresse (H) für Lightpen		
17	akt. Bildschirmspeicheradresse (L) für Lightpen		

Farbtabelle

Nr.	Farbe	Nr.	Farbe
0	schwarz	13	weiß
1	blau	14	pastellblau
2	hellblau	15	orange
3	rot	16	rosa
4	magenta	17	pastellmagenta
5	hellviolett	18	hellgrün
6	hellrot	19	seegrün
7	purpur	20	helles blaugrün
8	helles magenta	21	limonengrün
9	grün	22	pastellgrün
10	blaugrün	23	pastellblaugrün
11	himmelblau	24	hellgelb
12	gelb	25	pastellgelb
		26	leuchtendweiß

Sound

Sound Generator AY-3-8912

Beschreibung

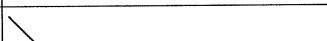
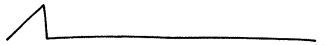
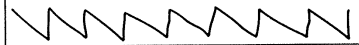
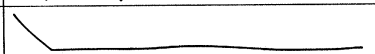
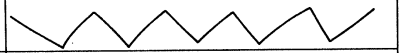
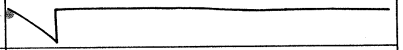
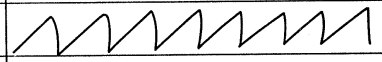
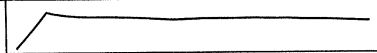
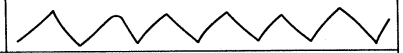
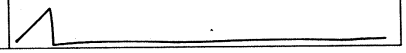
Der Sound Generator AY-3-8912 ist ein dreistimmiger Synthesizerbaustein, der für Telespiele entwickelt wurde. Er erlaubt die Erzeugung von Frequenzen in einem Bereich von acht Oktaven. Der Sound Generator ist über Port A des parallelen Interface Bausteins 8255 (Portadresse $F400) erreichbar.

Registerauflistung

Reg.	Bedeutung
0	Periodendauer des Tonsignals A (LSB)
1	Das LSN enthält den höherw. Teil der Periodendauer A
2	Periodendauer des Tonsignals B (LSB)
3	Das LSN enthält den höherw. Teil der Periodendauer B
4	Periodendauer des Tonsignals C (LSB)
5	Das LSN enthält den höherw. Teil der Periodendauer C
6	Bits 0–4: Rausch Frequenz (kleiner Wert – hohe Frequenz)
7	Multifunktionsregister
	Bit: Funktion
	0: 0 = Ton von Kanal A eingeschaltet
	1: 0 = Ton von Kanal B eingeschaltet
	2: 0 = Ton von Kanal C eingeschaltet
	3: 0 = Rauschen zu Kanal A zugeschaltet
	4: 0 = Rauschen zu Kanal B zugeschaltet
	5: 0 = Rauschen zu Kanal C zugeschaltet
	6: 0 = Port A als Eingang (1=Ausgang)
	7: 0 = Port B als Eingang (1=Ausgang)
8	LSN: Lautstärke des Signals an Kanal A Bit 4: 1 = Lautstärke wird durch Hüllkurve bestimmt
9	LSN: Lautstärke des Signals an Kanal B Bit 4: 1 = Lautstärke wird durch Hüllkurve bestimmt
10	LSN: Lautstärke des Signals an Kanal C Bit 4: 1 = Lautstärke wird durch Hüllkurve bestimmt
11,12	Periodendauer (L/H) der Hüllkurve
13	LSN Form der Hüllkurve (s. u.)

Aufbau der Hüllkurve

Hüllkurvenformen gemäß Festlegung in Register 13 (s. o.):

Bits 0	1	2	3	Hüllkurvenverlauf
0	0	x	x	
0	1	x	x	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Notentabelle

NOTE	FREQUENZ	PERIODE	
C	32.703	3822	
C#	34.648	3608	
D	36.708	3405	
D#	38.891	3214	Oktave –2
E	41.203	3034	
F	43.654	2863	
F#	46.249	2703	
G	48.999	2551	
G#	51.913	2408	
A	55.000	2273	
A#	58.270	2145	
B	61.735	2025	
NOTE	FREQUENZ	PERIODE	
C	65.406	1911	
C#	69.296	1804	
D	73.416	1703	
D#	77.782	1607	Oktave –3
E	82.407	1517	
F	87.307	1432	
F#	92.499	1351	
G	97.999	1276	
G#	103.826	1204	
A	110.000	1136	
A#	116.541	1073	
B	123.471	1012	
NOTE	FREQUENZ	PERIODE	
C	130.813	956	
C#	138.591	902	
D	146.832	851	
D#	155.564	804	
E	164.814	758	Oktave –1
F	174.614	716	
F#	184.997	676	
G	195.998	638	
G#	207.652	602	
A	220.000	568	
A#	233.082	536	
B	246.942	506	

NOTE	FREQUENZ	PERIODE	
C	261.626	478	
C#	277.183	451	
D	293.665	426	
D#	311.127	402	
E	329.628	379	Oktave 0
F	349.228	358	
F#	369.994	338	
G	391.995	319	
G#	415.305	301	Kammerton A
A	440.000	284	
A#	466.164	268	
B	493.883	253	
NOTE	FREQUENZ	PERIODE	
C	523.251	239	
C#	554.365	225	
D	587.330	213	
D#	622.254	201	
E	659.255	190	
F	698.457	179	Oktave 1
F#	739.989	169	
G	783.991	159	
G#	830.609	150	
A	880.000	142	
A#	932.328	134	
B	987.767	127	
NOTE	FREQUENZ	PERIODE	
C	1046.502	119	
C#	1108.731	113	
D	1174.659	106	
D#	1244.508	100	
E	1318.510	95	
F	1396.913	89	Oktave 2
F#	1479.978	84	
G	1567.982	80	
G#	1661.219	75	
A	1760.000	71	
A#	1864.655	67	
B	1975.533	63	

NOTE	FREQUENZ	PERIODE	
C	2093.004	60	
C#	2217.461	56	
D	2349.318	53	
D#	2489.016	50	
E	2637.021	47	
F	2793.826	45	
F#	2959.955	42	Oktave 3
G	3135.963	40	
G#	3322.438	38	
A	3520.000	36	
A#	3729.310	34	
B	3951.066	32	

NOTE	FREQUENZ	PERIODE	
C	4186.009	30	
C#	4434.922	28	
D	4698.636	27	
D#	4978.032	25	
E	5274.041	24	
F	5587.652	22	Oktave 4
F#	5919.911	21	
G	6271.927	20	
G#	6644.875	19	
A	7040.000	18	
A#	7458.621	17	
B	7902.133	16	

Diese Werte wurden vom Internationalen A aus wie folgt errechnet:

FREQUENZ = 440 * (2↑(OKTAVE + (10–N)/12))
PERIODE = ROUND (125000 / FREQUENZ)

mit N gleich 1 für C, 2 für C#, 3 für D, usw.

Quelle: Schneider

Anhang

ASCII-Tabelle

HEX	MSN	0	1	2	3	4	5	6	7
LSN	BITS	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SPACE	0	@	P	—	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	*	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	.	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	'	<	L	\	l	..
D	1101	CR	GS	—	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	←	o	DEL

Erläuterungen

NUL — Null
SOH — Start of Heading
STX — Start of Text
ETX — End of Text
EOT — End of Transmission
ENQ — Enquiry
ACK — Acknowledge
BEL — Bell
BS — Backspace
HT — Horizontal Tabulation
LF — Line Feed
VT — Vertical Tabulation
FF — Form Feed
CR — Carriage Return
SO — Shift Out
SI — Shift in

DLE — Data Link Escape
DC — Device Control
NAK — Negative Acknowledge
SYN — Synchronous Idle
ETB — End of Transmission Block
CAN — Cancel
EM — End of Medium
SUB — Substitute
ESC — Escape
FS — File Separator
GS — Group Separator
RS — Record Separator
US — Unit Separator
SP — Space (Blank)
DEL — Delete

Stichwortverzeichnis

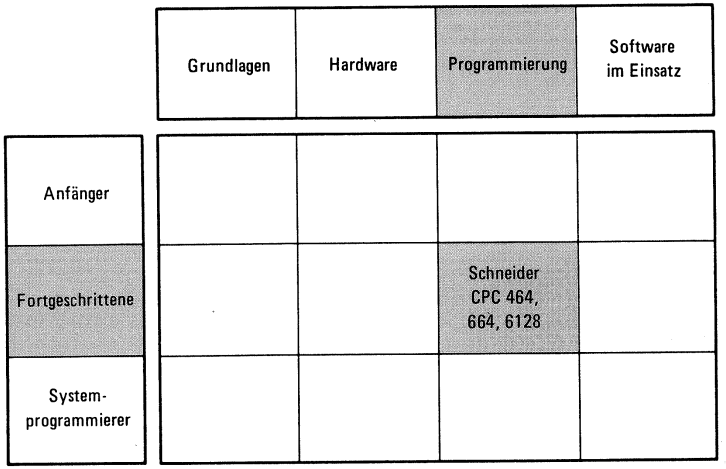
Andreas Dripke

Schneider CPC 464, 664, 6128

BASIC – Assembler – Grafik – Sound

Jeder, der einen Computer programmiert, benötigt dazu spezifische Informationen über sein jeweiliges Gerät. Der Schneider CPC-Benutzer steht dabei vor dem Problem, sich seine benötigten Informationen aus einer Flut dicker Handbücher und technischer Systemunterlagen sowie unzähliger Zeitschriftenartikel zusammenzusuchen.
Hier schafft der Programmierbegleiter Abhilfe: In einer wohlüberlegten, praxisgerechten Auswahl enthält er ohne unnützen Ballast alle Daten, die für die Programmierung der Schneider-Computer von Bedeutung sind. Das Buch berücksichtigt sämtliche Aspekte der BASIC-Programmierung über Assembler bis hin zu Grafik und Sound.

Der Autor *Andreas Dripke*, Wiesbaden, besitzt langjährige Erfahrung in der Software-Entwicklung.



ISBN 3-528-04490-X