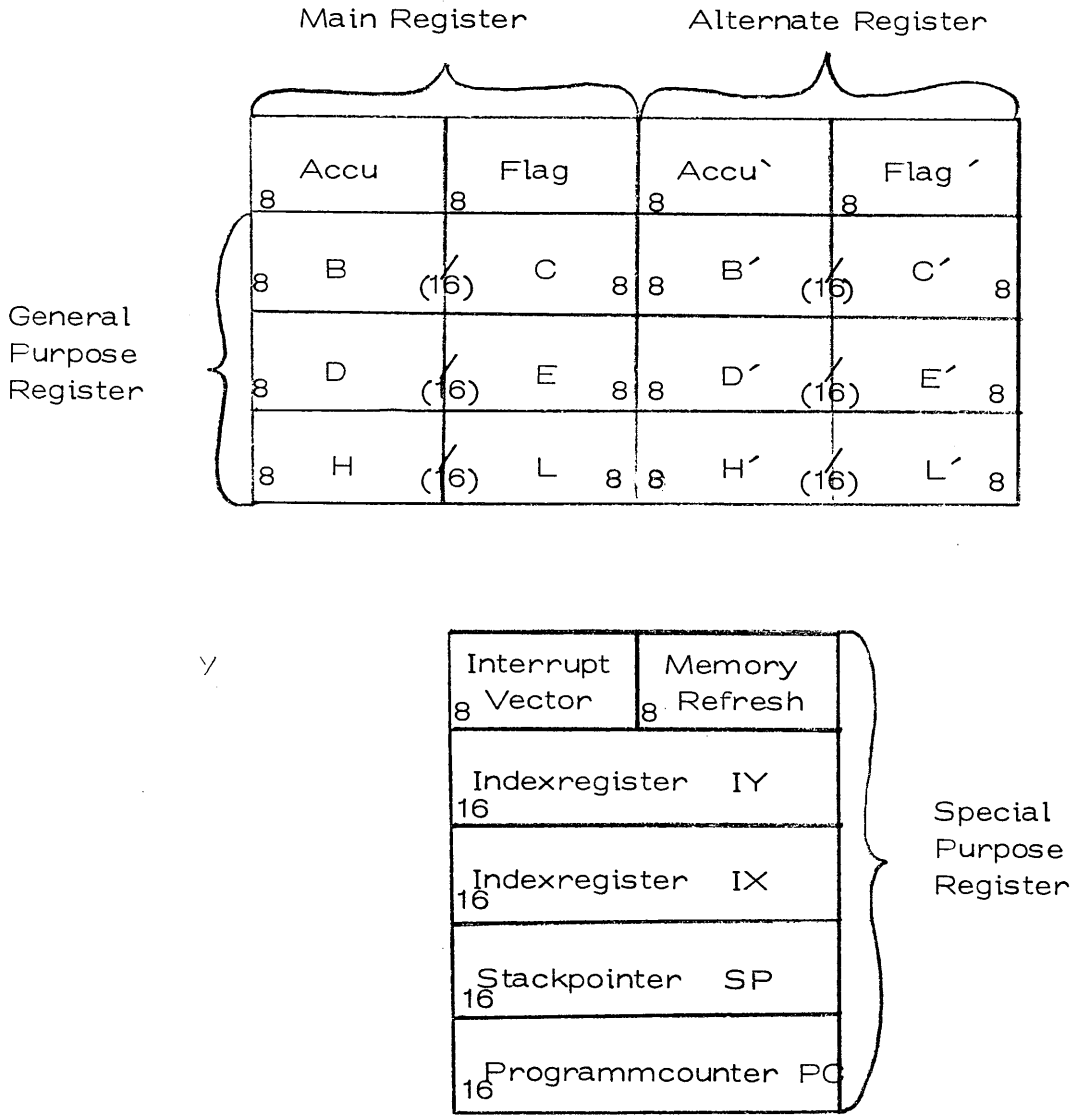
GENIE / TRS 80 - USER - CLUB

B R E M E R H A V E N

C L U B - I N F O

1. JAHRGANG
SEPTEMBER 83

Z80A - REGISTERORGANISATION



"MASCHINENSPRACHE TUT NOT !! "

Liebe Clubkameraden!

Schnell ist ein Monat vorbei und das nächste INFO liegt auf Euren Tischen.
Nicht immer ist es einfach das von mir selbst gewählte Minimum von 10 Seiten zu erreichen.
Ich tät mich freuen, wenn ihr mir bisweilen mit Texten unter die Arme greifen könntet. Auch eigene Gedanken, Erfahrungsberichte etc., ja Beteiligung an den laufenden Serien können mir nur helfen unser CLUBINFO noch interessanter zu gestalten.
Die Fortsetzung der strukturierten Programmierung, bzw.der diesbezügliche Artikel läßt leider auf sich warten. Dafür stürtzen wir uns um so mehr in die Maschinensprache: BYTE und BIT ergänzt die in der letzten Ausgabe begonnene Grundlagenerarbeitung.
Neue Roms für das COLOUR GENIE mit Verbesserungen, Fehlerkorrekturen und neuen Befehlen (!) werden die COLOUR-Besitzer freuen. Sie sind übrigens (bevor sie sich einer kauft) kostenlos im Austausch gegen die alten Chips bei der Firma Trommeschlaeger erhältlich.
Am Beispiel der APPEND-FUNKTION wird deutlich gezeigt, welche Möglichkeiten der Programmierer hat, der genaue Kenntnisse über den Basicinterpreter, Monitorunterprogramme und Kommunicationsbereich unserer GENIES besitzt.
Aus dem gleichen Grunde werden auch die ersten 5 Seiten eines "mc"-Artikels von L.Röchrath dem INFO beigefügt, damit zunächst einmal jeder von uns in Besitz der wichtigsten Fakten ist, um seinem GENIE besser zu verstehen. Die nächsten 5 Seiten gib's dann im Oktober.
Schließlich zeigt uns Michael Karnatz noch eine Version der Umlautebehandlung in Basic und mit dem Befehlssatz des Z80 alphabethisch und tabellarisch wollte ich noch die Grundlage zu ersten Gehversuchen in der Maschinensprache anbieten.
Und letztlich folgen noch einige Überlegungen zu unserer Clubbibliothek zu denen ich unbedingt Eure Meinung hören möchte.

Bis dann !

k.s.

Die Firma Trommeschläger (GENIE-Vertrieb in Deutland) gibt jetzt eine eigene Zeitung heraus, welche ich diesmal dem CLUBINFO beilege.
Wer dieses Blatt regelmäßig beziehen möchte, sollte an die auf der ersten Seite genannte Adresse wenden

"Hemmungen"

...hat augenscheinlich der Umsatz in unserer Bibliothek.
In der Zeit von der Gründung des Clubs im Januar 83 bis Heute wurde nur 13 mal auf die Bibliothek zugegriffen.
Weiterhin habe ich immer wieder Probleme mit meinem COLOUR die TRS80 bzw. GENIE I/II -Software korrekt und zeitgerecht zu copieren.
Andererseits hört die Flut an Programmen nicht auf und es ist absehbar wann die Bibliothek den Rahmen des CLUBINFOS sprengt.
Letztliche Kontakte mit anderen Clubs bestärken mich zudem in meiner Ansicht, daß die Form unseres "Programmaustausches" höchst -und wie es scheint- unnötig kompliziert ist.
Da ich nicht zu den Menschen gehöre, die wider besseren Wissens unsinnige Regeln einhalten, mache ich folgenden Verbesserungsvorschlag: Wir streichen den Abschnitt CLUBBIBLIOTHEK aus unseren Statuten und weichen auf ein in anderen Clubs übliches Verfahren aus.
Jedes Clubmitglied stellt eine Liste seiner zum Tausch angebotenen Programme auf und sendet diese Liste an mich. Auf der Liste sollten alle wichtigen Informationen über die Programme (Datenträger,Speicherplatzbedarf, Programmsprache etc.) und eine Kurzbeschreibung in geeigneter Form stehen. Die Liste sollte mit Drucker oder Schreibmaschine geschrieben sein und als Kopf Name und Anschrift des Entsenders enthalten.
Meine Arbeit wird sich darauf beschränken, die eintreffenden Listen auf ein geeignetes Format zu verkleinern und im Clubinfo zu veröffentlichen.
Will nun ein Mitglied ein Programm eines weiteren Mitglieds besitzen sendet es diesem den entsprechenden Datenträger mit Rückporto und einer (sofern auf der Liste vermerkt) Schutzgebühr/Unkostenbeitrag zu. Wobei erwähnt werden sollte, daß Programme mit fremden Urheberrechten nur zur privaten Nutzung und generell unentgeldlich weitergegeben werden dürfen! Gegen einen Unkostenbeitrag für Kopien von z.B. Handouts ist aber sicherlich nichts einzuwenden.
Dies entspricht im Übrigen der Praxis der "deutschen Clubszene" und scheint mir einem zügigen Programmaustausch äußerst zuträglich zu sein, was mir der Umfang anderer Bibliotheken, welche mir wie schon oben erwähnt kürzlich zugingen, deutlich beweist.
War sehen unsere Statuten keine Abstimmungen für den Fall ihrer Änderung vor, so sehe ich mich aber trotzdem in Eurer Pflicht und werde diese Statuten änderung nur durchführen, wenn 2/3 der Mitglieder diesem Verfahren, wie oben geschildert, bis zum 1.9.83 schriftlich zugestimmt haben.
Für einen regelmäßig überfüllten Briefkasten offend verbleibe ich wie immer

f.G. Euer Club"papa" Klaus.

Neue ROMs für das Colour-Genie

1.) Die Grafikauflösung wurde von 160 mal 96 Punkten auf 160 mal 102 Punkte erhöht. Sämtliche Grafikbefehle unterstützten diese Erweiterung.
Soll für alte Programme die alte Auflösung wiederhergestellt werden ist dies mit "POKE 17161,96:FGR" möglich.

2.) Der LGR-Modus unterstützt jetzt 25 statt bisher 24 Zeilen.
Der PRINT@ -Befehl aktzeptiert also Werte von 0 bis 999.

3.) Der "PLAY"-Befehl nimmt jetzt auch Ausdrücke an.
"PLAY(1,KEYPAD1,KEYPAD2,(JOY1X-1)AND15)"
ist also jetzt möglich.
Ferner ist Lautstärke 16 jetzt möglich, damit wird der PSG auf Hüllkurve gesetzt.
Addiert man zu den alten Notenwerten 16, werden die Töne in einer besser gestimmten C-Dur-Tonleiter gespielt.

4.) Der "PLOT"-Befehle und alle auf diesem basierende Befehle sind deutlich schneller geworden !

5.) Der "PAINT"-Befehl läuft jetzt fehlerfrei. Alle Flächen werden komplett ausgemalt. Es sind folgende 3 Formate zulässig:

```
PAINT X,Y,F
PAINT X,Y,F,B
PAINT X,Y,F,B,B
```

Dabei sind X und Y die Koordinaten, von denen aus ausgemalt wird; F ist die Farbe mit der ausgemalt wird (zugleich auch Begrenzung) und B ist eine Begrenzungsfarbe.
Ein Beispielprogramm:

```
>10 FGR:FCLS
>20 FCOLOUR 2 : CIRCLE 80,51,50
>30 FCOLOUR 3 : CIRCLE 80,51,30
>40 PAINT 90,80,4,2,3
>50 GOTO 50
```

6.) Der "FILL n"-Befehl wurde durch "FCLS n" ersetzt.
"FCLS 4" entspricht also dem ehemaligen "FILL 4" - der Grafikspeicher wird grün ausgemalt.

7.) Die Funktionstaste <SHIFT><F2> wurde mit 'SYSTEM (RETURN)', die Funktionstaste <SHIFT><F4> mit 'CSAVE "' belegt.

8.) Die neuen ROMs ermöglichen ein Autostart-Dos. Bei den alten ROMs läuft das DOS auch, aber mit "CALL"-Aufruf.

9.) Der Befehl "ERROR n" läuft jetzt fehlerfrei. Dieser Befehl gibt die Fehlermeldung aus, die Fehler Nummer n enspricht.

10.) "PRINT ERR/2+1" ergibt den Fehlercode eines aufgetretenen Fehlers und "PRINT ERL" die Zeile, in der der Fehler aufgetreten ist.

11.) Der CLS-Befehl löscht nicht mehr den Farbspeicher.
Alle Leerzeichen haben den Farbwert für Weiss.

Neue Befehle:

1.) SWAP A,B vertauscht zwei Variablen gleichen Typs.
Ein Beispiel:
A = 1 : B = 2 :SWAP A,B:PRINT A,B
ergibt:
2 1

2.) SOUND (r) ergibt den den Inhalt der Registers r des PSG. Zulässig sind Werte 0 bis 15.

3.) JOY (n) ermöglicht die Adressierung von bis zu 4 Joysticks a 2 Analoginputs mit 8 Bits Genauigkeit, d.h. es ergeben sich Werte von 0 bis 255.
Die alten Befehle "JOY1X", "JOY1Y", "JOY2X", "JOY2Y" funktionieren weiterhin, es wurden jedoch die Zeitdifferenzen in Abhängigkeit zur Joystickposition beseitigt. Die Ermittlung insgesamt wurde beschleunigt.

4.) KEYPAD(n) mit n=1,2 ermöglicht indiziertes Ansprechen der beiden Joystick-Tastaturen.
Die alten Befehle KEYPAD1, KEYPAD2 sind weiterhin korrekt.

5.) SCALE gibt den aktuellen Scale-Faktor an.
Z.B.: "SCALE 4: PRINT SCALE" ergibt 4.

6.) JOYINP, JOYOUT ermöglicht das Transfern von Basicprogrammen von einem Colour-Genie zu einem anderen Colour-Genie.
Dazu müßen die Pins 10-18 und Pin 20 des Parallelports beider Geräte miteinander verbunden werden. (Siehe auch Anhang E. Vorsicht mit den Betriebsspannungen!!!)

7.) BGRD n mit n=1...4 ermöglicht das Vorwählen eines von 4 verschiedenen Hintergrundfarben. Anschalten mit "BGRD". Dieser Befehl ist nur interessant, wenn das TCS-RGB-Modul und ein RGB-Monitor angeschlossen sind. (Beides demnächst lieferbar)

8.) SET Bitnummer, Adresse mit Bitnummer=0...7 setzt das jeweilige Bit der angegeben Adresse.

9.) RESET Bitnummer, Adresse setzt entsprechend obigem Befehl ein Bit zurück.

10.) CHECK (Bitnummer, Adresse) prüft entsprechend, ob ein Bit gesetzt ist. Ergibt -1 wenn gesetzt, 0 wenn nicht.

Bitte wenden Sie obige neue Befehle mit Bedacht an, denn Besitzer eines älteren Colour-Genie (ohne Level-Meter) können Programme mit den neuen Befehlen nicht ohne Änderungen übernehmen.

Der Maschinenzyklus

..auch Neumannscher Zyklus genannt, umschreibt den generellen (für alle Computer ähnlichen) Ablauf in einer CPU während der Ausführung eines Befehls.

1. Hole den Inhalt des aktuellen Bytes gem. PC in das Befehlsregister.
2. Überprüfe, ob zur Ausführung weitere Bytes (Daten oder Befehlsbytes) gehören. Wenn nein, springe zu 4.
3. Erhöhe den PC um eins: Hole byte: Fahre mit 3. fort bis alle zu diesem Befehl gehörenden Bytes in der CPU sind.
4. Führe den Befehl gem. interner Befehlslisten und Mikroprogrammen durch.
5. Erhöhe den PC um eins und springe zu 1.

Der tatsächliche Ablauf, daß sei hier klar gesagt, ist allerdings um einiges komplizierter, was uns aber jetzt noch nicht behindern soll.

Die Abarbeitung eines Maschinenprogramms stellt sich also als ständige Fortführung obigen Zykluses dar. Wenn wir uns jetzt den Befehlsvorrat der Z80A-CPU ansehen, so wird uns vielleicht jetzt schon klar, daß unser Computer zwar saudumm ist, aber dies so schnell machen muß, daß er uns "klug" erscheint.
So umfassen manche BASIC-Befehle hunderte von diesen Maschinenbefehlen und benötigen trotzdem nicht mehr als milli-Sekunden zur Vollendung.
Ein sogenannter Taktzyklus der Z80A-CPU benötigt zwei Perioden der Taktfrequenz. Bei 2,2Mhz Takt (T=1/f)*2 = 0,9 mycrosekunden (!).
Sehen wir uns den ersten Befehl in der Tabelle an, so lesen wir bei "LD r,r' " (Lade den Inhalt des Registers r' in das Register r, wobei r für eines der Mainregister, r' für das entsprechende Alternateregister steht.) in der vorletzten Spalte (No. of T Cycles) die Zahl 4 . D.h. unsere CPU benötigt zur Ausführung dieses Befehls 4 Taktzyklen.
Also 4*0,9=3,6mykrosekunden.
Einer der "längsten" Befehle ist INC (IY+d) (Erhöhe den Inhalt der Speicherzelle, auf die das Indexregister Y + dem Wert d zeigt, um eins.) mit 23 Taktzyklen. Er benötigt somit ca 21mykrosek. zur Ausführung. Selbst bei 1000facher Durchführung dieses Befehls kämen wir bei 21 msek gerade in den Bereich unserer Wahrnehmungsfähigkeit, - denn unser Netz arbeitet mit 50Hz (T=20msek.) und wir sind nicht in der Lage z.B. das Flimmern einer Glimmlampe auszumachen.
Soviel für diesmal! Im nächsten Info geht es mit den ersten Beispielprogrammen weiter.

BYTE, BIT und MEMORY

Mit der lakonischen Festellung -acht BIT sind ein BYTE und davon kann unsere CPU (Central Prozessing Unit), die Z80A, genau 65536 (=64KBYTE) adressieren - wäre eigentlich schon alles gesagt.
Aber ein wenig tiefer sollte man die Sache schon betrachten.
Ein BIT ist die kleinste Darstellungseinheit für Binärdaten, d.h. ein BIT kann genau zwei Zustände haben: 0 oder 1 (!).
Da uns das zu wenig ist und wir durchaus bisweilen größere Werte verarbeiten müssen, fassen wir genau acht BIT zu einem BYTE zusammen womit wir nun 256 unterschiedliche Zustände (Zahlen) beschreiben können. Wobei die Anzahl der Zustände mit N = 2↑8=256, die höchste darstellbare Zahl mit X=2↑8-1=255dez ist. Rechnet man 255dez in das DUALsystem um so erhält man 11111111dual, also genau acht Isen, womit jetzt klar sein dürfte, daß die kleinste darstellbare Zahl bei acht BIT = 00000000dual = 0dez ist!
Weiterhin erscheint uns nun die Tatsache logisch, daß alle Computer dieser Welt im DUALsystem arbeiten.
Sehen wir uns den Speicher eines solchen Computers genauer an: 65536 BYTES kann die Z80A adressieren. Daraus folgt, daß sie genau 16 duale Stellen benötigt, um jedes dieser Bytes ansprechen zu können, denn 2↑16=65536.
Weiterhin wird jetzt die Speicherorganisation deutlich: Jedes dieser 65536 Bytes hat eine 16-stellige (DUAL!) Adresse.
Der mögliche Speicher unseres Z80A besteht also aus 65535 BYTES a 8 BIT, deren achtstelliger Zustand auf genau 8 Datenleitungen (DATENBUS = DB) immer dann angezeigt wird, wenn auf den weiteren 16-Adressleitungen (ADRESSBUS = AB) die 16-stellige (DUAL) Adresse von der CPU angewählt wird.
Die CPU verwendet für die Adressierung den Programcounter (=PC).
Der Inhalt der adressierten Speicherzelle ist zumeist ein BEFEHL für die CPU, nachdem sie ihr weiteres Verhalten, nach einem internen Befehlsregister, ausrichtet.
Die Befehle sagen der CPU in welcher Weise z.B der Inhalt des ACCUMULATORS (A) mit dem Inhalt eines der anderen Register verknüpft oder verrechnet werden soll, oder ob z.B. der Inhalt ei-
Registers in eine Speicherzelle (eine von den 65536 !) transferiert werden soll, auf die ein drittes Register (dann 16-stellig) zeigt.
Wie nun aus einer Mehrzahl solcher Befehle ein Programm wird und wie die CPU sich bei jedem Befehl verhält erklärt uns anschaulich der "NEUMANSCHE ZYKLUS". (b.w.)

Befehlssatz

DC HL, ss	Add with Carry Reg. pair ss to HL
DC A, s	Add with carry operand s to Acc.
DD A, n	Add value n to Acc.
DD A, r	Add Reg. r to Acc.
DD A, (HL)	Add location (HL) to Acc.
DD A, (IX+d)	Add location (IX+d) to Acc.
DD A, (IY+d)	Add location (IY+d) to Acc.
DD HL, ss	Add Reg. pair ss to HL
DD IX, pp	Add Reg. pair pp to IX
DD IY, rr	Add Reg. pair rr to IY
ND s	Logical 'AND' of operand s and Acc.
IT b, (HL)	Test BIT b of location (HL)
IT b, (IX+d)	Test BIT b of location (IX+d)
IT b, (IY+d)	Test BIT b of location (IY+d)
IT b, r	Test BIT b of Reg. r
ALL c, nn	Call subroutine at location nn if condition cc if true
ALL nn	Unconditional call subroutine at location nn
CF	Complement carry flag
P s	Compare operand s with Acc.
PD	Compare location (HL) and Acc. decrement HL and BC
PDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0
PI	Compare location (HL) and Acc. Increment HL and decrement BC
PIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0
PL	Complement Acc. (1's comp)
A	Decimal adjust Acc.
C m	Decrement operand m
C IX	Decrement IX
DEC IY	Decrement IY
DEC ss	Decrement Reg. pair ss
DI	Disable interrupts
DJNZ e	Decrement B and Jump [illegible], 0
EI	Enable Interrupts
EX (SP), HL	Exchange the location (SP) and HL
EX (SP), IX	Exchange the location (SP) and IX
EX (SP), IY	Exchange the location (SP) and IY
EX AF, AF'	Exchange the contents of AF and AF'
EX DE, HL	Exchange the contents of DE and HL
EXX	Exchange the contents of BC, DE, HL with contents of BC', DE', HL' respectively
HALT	HALT (wait for interrupt or reset)
IM 0	Set interrupt mode 0
IM 1	Set interrupt mode 1
IM 2	Set interrupt mode 2
IN A, (n)	Load the Acc. with input from device n
IN r, (C)	Load the Reg. r with input from device (C)
INC (HL)	Increment location (HL)
INC IX	Increment IX
INC (IX+d)	Increment location (IX+d)
INC IY	Increment IY
INC (IY+d)	Increment location (IY+d)
INC r	Increment Reg. r
INC ss	Increment Reg. pair ss
IND (C)	Load location (HL) with input from port (C), decrement HL and B
INDR (C)	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0
INI (C)	Load location (HL) with input from port (C); and increment HL and decrement B
INIR (C)	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0
JP (HL)	Unconditional Jump to (HL)
JP (IX)	Unconditional Jump to (IX)
JP (IY)	Unconditonal Jump to (IY)
JP c, nn	Jump to location nn if condition cc is true
JP nn	Unconditional Jump to location nn
JP C, e	Jump relative to PC+e if carry=1
JR e	Unconditional Jump relative to PC+e
JP NC, e	Jump relative to PC+e if carry=0
JR NZ, e	Jump relative to PC+e if non zero (Z=0)
JR Z, e	Jump relative to PC+e if zero (Z=1)
LD A, (BC)	Load Acc. with location (BC)
LD A, (DE)	Load Acc. with location (DE)
LD A, I	Load Acc. with I
LD A, (nn)	Load Acc. with location nn
LD A, R	Load Acc. with Reg. R
LD (BC), A	Load location (BC) with Acc.
LD (DE), A	Load location (DE) with Acc.
LD (HL), n	Load location (HL) with value n
LD dd, nn	Load Reg. pair dd with value nn
LD HL, (nn)	Load HL with location (nn)
LD (HL), r	Load location (HL) with Reg. r
LD I, A	Load I with Acc.
LF IX, nn	Load IX with value nn
LD IX, (nn)	Load IX with location (nn)
LD (IX+d), n	Load location (IX+d) with value n
LD (IX+d), r	Load location (IX+d) with Reg. r
LD IY, nn	Load IY with value nn
LD IY, (nn)	Load IY with location (nn)
LD (IY+d), n	Load location (IY+d) with value n
LD (IY+d), r	Load location (IY+d) with Reg. r
LD (nn), A	Load location (nn) with Acc.
LD (nn), dd	Load location (nn) with Reg. pair dd
LD (nn), HL	Load location (nn) with HL
LD (nn), IX	Load location (nn) with IX
LD (nn), IY	Load location (nn) with IY
LD R, A	Load R with Acc.
LD r, (HL)	Load Reg. r with location (HL)
LD r, (IX+d)	Load Reg. r with location (IX+d)
LD r, (IY+d)	Load Reg. r with location (IY+d)
LD r, n	Load Reg. r with value n
LD r, r'	Load Reg. r with Reg. r'
LD SP, HL	Load SP with HL
LD SP, IX	Load SP with IX
LD SP, IY	Load SP with IY
LDD	Load location (DE) with location (HL), decrement DE, HL and BC
LDDR	Load location (DE) with location (HL), decrement DE, HL and BC; repeat until BC=0
LDI	Load location (DE) with location (HL), increment DE, HL, decrement BC
LDIR	Load location (DE) with location (HL), increment DE, HL, decrement BC and repeat until BC=0
NEG	Negate Acc. (2's complement)
NOP	No operation
OR s	Logical 'OR' or operand s and Acc.
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0
OUT (C), r	Load output port (C) with Reg. r
OUT (n), A	Load output port (n) with Acc.
OUTD	Load output port (C) with location (HL), decrement HL and B
OUTI	Load output port (C) with location (HL), increment HL and decrement B
POP IX	Load IX with top of stack
POP IY	Load IY with top of stack
POP qq	Load Reg. pair qq with top of stack
PUSH IX	Load IX onto stack
PUSH IY	Load IY onto stack
PUSH qq	Load Reg. pair qq onto stack
RES b, m	Reset Bit b of operand m
RET	Return from subroutine
RET c	Return from subroutine if condition cc is true
RETI	Return from interrupt
RL m	Rotate m through carry op
RLA	Rotate left Acc. through ca
RLC (HL)	Rotate location (HL) left ci
RLC (IX+d)	Rotate location (IX+d) left
RLC (IY+d)	Rotate location (IY+d) left
RLC r	Rotate Reg. r left circular
RLCA	Rotate left circular Acc.
RLD	Rotate digit left and right be Acc. and location (HL)
RR m	Rotate right through carry op
RRA	Rotate right Acc. through car
RRC m	Rotate operand m right circul
RRCA	Rotate right circular Acc.
RRD	Rotate digit right and left bet Acc. and location (HL)
RST p	Restart to location p
SBC A, s	Subtract operand s from Acc. carry
SBC HL, ss	Subtract Reg. pair ss from HL carry
SCF	Set carry flag (C=1)
SET b, (HL)	Set Bit b of location (HL)
SET b, (IX+d)	Set Bit b of location (IX+d)
SET b, (IY+d)	Set Bit b of location (IY+d)
SET b, r	Set Bit b of Reg. r
SLA m	Shift operand m left arithmetic
SRA m	Shift operand m right arithme
SRL m	Shift operand m right logical
SUB s	Subtract operand s from Acc.
XOR s	Exclusive 'OR' operand s and A

MNEMONIC	SYMBOLIC OPERATION	FLAGS C	Z	P/V	S	N	H	OP-CODE 76 543 210	NO OF T CYCLES	COMMENTS
RLC r		↕	↕	P	↕	0	0	11 001 011 00 000 r	8	Rotate left circular register r
RLC (HL)		↕	↕	P	↕	0	0	11 001 011 00 000 110	15	r: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
RLC (IX+d)		↕	↕	P	↕	0	0	11 011 101 11 001 011 · d · 00 000 110	23	
RLC (IY+d)		↕	↕	P	↕	0	0	11 111 101 11 001 011 · d · 00 000 110	23	
RL m		↕	↕	P	↕	0	0	010		Instruction format and states are as shown for RLC m. To form new OP-code replace 000 of RLC m with shown code
RRC m		↕	↕	P	↕	0	0	001		
RR m		↕	↕	P	↕	0	0	011		
SLA m		↕	↕	P	↕	0	0	100		
SRA m		↕	↕	P	↕	0	0	101		
SRL m		↕	↕	P	↕	0	0	111		
RLD		↕	↕	P	↕	0	0	11 101 101 01 101 111	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		↕	↕	P	↕	0	0	11 101 101 01 100 111	18	
DAA	Converts acc content into packed BCD following add or subtract with packed BCD operands	↕	↕	P	↕	↕	↕	00 100 111	4	Decimal adjust accumulator
CPL	$A \leftarrow \overline{A}$	•	•	•	•	1	1	00 101 111	4	Complement accumulator (one's complement)
NEG	$A \leftarrow 0 - A$	↕	↕	V	↕	1	↕	11 101 101 01 000 100	8	Negate acc. (two's complement)
CCF	$CY \leftarrow \overline{CY}$	↕	•	•	•	0	X	00 111 111	4	Complement carry flag
SCF	CY ← 1	1	•	•	•	0	0	00 110 111	4	Set carry flag
NOP	No operation	•	•	•	•	•	•	00 000 000	4	
HALT	CPU halted	•	•	•	•	•	•	01 110 110	4	
DI	IFF ← 0	•	•	•	•	•	•	11 110 011	4	
EI	IFF ← 1	•	•	•	•	•	•	11 111 011	4	
IM 0	Set interrupt mode 0	•	•	•	•	•	•	11 101 101 01 000 110	8	
IM 1	Set interrupt mode 1	•	•	•	•	•	•	11 101 101 01 010 110	8	
IM 2	Set interrupt mode 2	•	•	•	•	•	•	11 101 101 01 011 110	8	
BIT b, r	$Z \leftarrow \overline{r_b}$	•	↕	X	X	0	1	11 001 011 01 b r	8	r: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	•	↕	X	X	0	1	11 001 011 01 b 110	12	
BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)_b}$	•	↕	X	X	0	1	11 011 101 11 001 011 · d · 01 b 110	20	
BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)_b}$	•	↕	X	X	0	1	11 111 101 11 001 011 · d · 01 b 110	20	b: 000 0, 001 1, 010 2, 011 3, 100 4, 101 5, 110 6, 111 7 — Bit Tested

MNEMONIC	SYMBOLIC OPERATION	FLAGS C	Z	P/V	S	N	H	OP CODE 76 543 210	NO OF T CYCLES	COMMENTS
SET b, r	$r_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 11 b r	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 11 b 110	16	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11 011 101 11 001 011 · d · 11 b 110	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11 111 101 11 001 011 · d · 11 b 110	23	
RES b, m	$m_b \leftarrow 0$ m = r, (HL), (IX+d), (IY+d)							10		To form new OP-code replace 11 of SET b, m with 10. Flags and time states for SET instruction
JP nn	PC ← nn	•	•	•	•	•	•	11 000 011 · n · · n ·	10	
JP cc, nn	If condition is true PC ← nn, otherwise continue	•	•	•	•	•	•	11 cc 010 · n · · n ·	10	cc: 000 NZ non zero, 001 Z zero, 010 NC non carry, 011 C carry, 100 PO parity odd, 101 PE parity even, 110 P sign positive, 111 M sign negative
JR e	PC ← PC + e	•	•	•	•	•	•	00 011 000 · e-2 ·	12	
JR C, e	If C = 0, continue	•	•	•	•	•	•	00 111 000 · e-2 ·	7	If condition not met
	If C = 1, PC ← PC + e								12	If condition is met
JR NC, e	If C = 1, continue	•	•	•	•	•	•	00 110 000 · e-2 ·	7	If condition not met
	If C = 0, PC ← PC + e								12	If condition is met
JR Z, e	If Z = 0 continue	•	•	•	•	•	•	00 101 000 · e 2 ·	7	If condition not met
	If Z = 1, PC ← PC + e								12	If condition is met
JR NZ, e	If Z = 1 continue	•	•	•	•	•	•	00 100 000 · e 2 ·	7	If condition not met
	If Z = 0, PC ← PC + e								12	If condition met
JP (HL)	PC ← HL	•	•	•	•	•	•	11 101 001	4	
JP (IX)	PC ← IX	•	•	•	•	•	•	11 011 101 11 101 001	8	
JP (IY)	PC ← IY	•	•	•	•	•	•	11 111 101 11 101 001	8	
DJNZ e	B ← B-1 If B = 0, continue	•	•	•	•	•	•	00 010 000 · e-2 ·	8	If B = 0
	If B ≠ 0, PC ← PC + e								13	If B ≠ 0
CALL nn	(SP-1) ← PC_H (SP-2) ← PC_L PC ← nn	•	•	•	•	•	•	11 001 101 · n · · n ·	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	•	•	•	•	11 cc 100 · n · · n ·	10	If cc is false
									17	If cc is true
RET	$P_L \leftarrow (SP)$ $P_H \leftarrow (SP+1)$	•	•	•	•	•	•	11 001 001	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	•	•	•	•	11 cc 000	5	If cc is false
									11	If cc is true cc: 000 NZ non zero, 001 Z zero, 010 NC non carry, 011 C carry, 100 PO parity odd, 101 PE parity even, 110 P sign positive, 111 M sign negative
RETI	Return from interrupt	•	•	•	•	•	•	11 101 101 01 001 101	14	
RETN	Return from non maskable interrupt	•	•	•	•	•	•	11 101 101 01 000 101	14	

MNEMONIC	SYMBOLIC OPERATION	FLAGS C	Z	P/V	S	N	H	OP CODE 76 543 210	NO OF T CYCLES	COMMENTS
RST p	(SP-1) ← PC_H (SP-2) ← PC_L PC_H ← 0 PC_L ← P	•	•	•	•	•	•	11 t 111	11	
IN A, (n)	A ← (n)	•	•	•	•	•	•	11 011 011 · n ·	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
IN r, (C)	r ← (C) If r = 110 only the flags will be affected	•	↕ ②	P	↕	0	↕	11 101 101 01 r 000	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INI	(HL) ← (C) B ← B-1 HL ← HL + 1	•	↕ ①	X	X	1	X	11 101 101 10 100 010	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INIR	(HL) ← (C) B ← B-1 HL ← HL + 1 Repeat until B = 0	•	1 ①	X	X	1	X	11 101 101 10 110 010	21 16	
IND	(HL) ← (C) B ← B-1 HL ← HL-1	•	↕ ①	X	X	1	X	11 101 101 10 101 010	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INDR	(HL) ← (C) B ← B-1 HL ← HL-1 Repeat until B = 0	•	1 ①	X	X	1	X	11 101 101 10 111 010	21 16	
OUT (n), A	(n) ← A	•	•	•	•	•	•	11 010 011 · n ·	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
OUT (C), r	(C) ← r	•	•	•	•	•	•	11 101 101 01 r 001	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTI	(C) ← (HL) B ← B-1 HL ← HL + 1	•	↕ ③	X	X	1	X	11 101 101 10 100 011	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTIR	(C) ← (HL) B ← B-1 HL ← HL + 1 Repeat until B = 0	•	1 ③	X	X	1	X	11 101 101 10 110 011	21 16	
OUTD	(C) ← (HL) B ← B-1 HL ← HL-1	•	↕ ③	X	X	1	X	11 101 101 10 101 011	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTDR	(C) ← (HL) B ← B-1 HL ← HL-1 Repeat until B = 0	•	1 ③	X	X	1	X	11 101 101 10 111 011	21 16	

$r, r' = A \vee B \vee C \vee D \vee E \vee H \vee L$

$ss = BC \vee DE \vee HL \vee SP$

$rr = -11-$

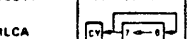
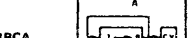
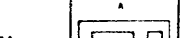
① P/V flag is 0 if the result of BC-1=0 (ow

② Z-flag is 1 if A=(HL) (ow Z=0)

③ IF the result B-1=0, the Z-flag is set(ow

• - not affected 0 - flag ist reset
1 - flag ist set

↕ flag is affected according to the result of the operation

MNEMONIC	SYMBOLIC OPERATION	FLAGS C	Z	P/V	S	N	H	OP-CODE 76 543 210	NO. OF T CYCLES	COMMENTS
LD r, r'	r ← r'	•	•	•	•	•	•	01 r r'	4	r, r' Reg.: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
LD r, n	r ← n	•	•	•	•	•	•	00 r 110 ← n →	7	
LD r, (HL)	r ← (HL)	•	•	•	•	•	•	01 r 110	7	
LD r, (IX+d)	r ← (IX+d)	•	•	•	•	•	•	11 011 101 01 r 110 ← d →	19	
LD r, (IY+d)	r ← (IY+d)	•	•	•	•	•	•	11 111 101 01 r 110 ← d →	19	
LD (HL), r	(HL) ← r	•	•	•	•	•	•	01 110 r	7	
LD (IX+d), r	(IX+d) ← r	•	•	•	•	•	•	11 011 101 01 110 r · d ·	19	
LD (IY+d), r	(IY+d) ← r	•	•	•	•	•	•	11 111 101 01 110 r · d ·	19	
LD (HL), n	(HL) ← n	•	•	•	•	•	•	00 110 110 ← n →	10	
LD (IX+d), n	(IX+d) ← n	•	•	•	•	•	•	11 011 101 00 110 110 ← d → ← n →	19	
LD (IY+d), n	(IY+d) ← n	•	•	•	•	•	•	11 111 101 00 110 110 ← d → ← n →	19	
LD A, (BC)	A ← (BC)	•	•	•	•	•	•	00 001 010	7	
LD A, (DE)	A ← (DE)	•	•	•	•	•	•	00 011 010	7	
LD A, (nn)	A ← (nn)	•	•	•	•	•	•	00 111 010 ← n → ← n →	13	
LD (BC), A	(BC) ← A	•	•	•	•	•	•	00 000 010	7	
LD (DE), A	(DE) ← A	•	•	•	•	•	•	00 010 010	7	
LD (nn), A	(nn) ← A	•	•	•	•	•	•	00 110 010 · n · · n ·	13	
LD A, I	A ← I	•	‡	IFF	‡	0	0	11 101 101 01 010 111	9	
LD A, R	A ← R	•	‡	IFF	‡	0	0	11 101 101 01 011 111	9	
LD I, A	I ← A	•	•	•	•	•	•	11 101 101 01 000 111	9	
LD R, A	R ← A	•	•	•	•	•	•	11 101 101 01 001 111	9	
LD dd, nn	dd ← nn	•	•	•	•	•	•	00 dd0 001 ← n → ← n →	10	dd Pair: 00 BC, 01 DE, 10 HL, 11 SP
LD IX, nn	IX ← nn	•	•	•	•	•	•	11 011 101 00 100 001 ← n → ← n →	14	
LD IY, nn	IY ← nn	•	•	•	•	•	•	11 111 101 00 100 001 ← n → ← n →	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	•	•	•	•	•	•	00 101 010 · n · · n ·	16	
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	•	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	•	•	•	•	11 011 101 00 101 010 ← n → ← n →	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	•	•	•	•	11 111 101 00 101 010 ← n → ← n →	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	•	•	•	•	•	•	00 100 010 ← n → ← n →	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	•	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	•	•	•	•	11 011 101 00 100 010 ← n → ← n →	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	•	•	•	•	11 111 101 00 100 010 ← n → ← n →	20	
LD SP, HL	SP ← HL	•	•	•	•	•	•	11 111 001	6	
LD SP, IX	SP ← IX	•	•	•	•	•	•	11 011 101 11 111 001	10	
LD SP, IY	SP ← IY	•	•	•	•	•	•	11 111 101 11 111 001	10	
PUSH qq	(SP-2) ← qq_L (SP-1) ← qq_H	•	•	•	•	•	•	11 qq0 101	11	qq Pair: 00 BC, 01 DE, 10 HL, 11 AF
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	•	•	•	•	•	•	11 011 101 11 100 101	15	
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	•	•	•	•	•	•	11 111 101 11 100 101	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP)	•	•	•	•	•	•	11 qq0 001	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	•	•	•	•	•	•	11 011 101 11 100 001	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	•	•	•	•	•	•	11 111 101 11 100 001	14	
EX DE, HL	DE ↔ HL	•	•	•	•	•	•	11 101 011	4	
EX AF, AF'	AF ↔ AF'	•	•	•	•	•	•	00 001 000	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	•	•	•	•	•	•	11 011 001	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP+1) L ↔ (SP)	•	•	•	•	•	•	11 100 011	19	
EX (SP), IX	IX_H ↔ (SP+1) IX_L ↔ (SP)	•	•	•	•	•	•	11 011 101 11 100 011	23	
EX (SP), IY	IY_H ↔ (SP+1) IY_L ↔ (SP)	•	•	•	•	•	•	11 111 101 11 100 011	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	•	•	①	•	0	0	11 101 101 10 100 000	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 Repeat until BC = 0	•	•	0	•	0	0	11 101 101 10 110 000	21 16	If BC ≠ 0 If BC = 0
LDD	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	•	•	①	•	0	0	11 101 101 10 101 000	16	
LDDR	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 Repeat until BC = 0	•	•	0	•	0	0	11 101 101 10 111 000	21 16	If BC ≠ 0 If BC = 0
CPI	A - (HL) HL ← HL+1 BC ← BC-1	•	①	②	①	1	①	11 101 101 10 100 001	16	
CPIR	A - (HL) HL ← HL+1 BC ← BC-1 Repeat until A = (HL) or BC = 0	•	①	②	①	1	①	11 101 101 10 110 001	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A - (HL) HL ← HL-1 BC ← BC-1	•	①	②	①	1	①	11 101 101 10 101 001	16	
CPDR	A - (HL) HL ← HL-1 BC ← BC-1 Repeat until A = (HL) or BC = 0	•	①	②	①	1	①	11 101 101 10 111 001	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
ADD A, r	A ← A + r	①	①	V	①	0	①	10 000 r	4	r Reg.: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
ADD A, n	A ← A + n	①	①	V	①	0	①	11 000 110 ← n →	7	
ADD A, (HL)	A ← A + (HL)	①	①	V	①	0	①	10 000 110	7	
ADD A, (IX+d)	A ← A + (IX+d)	①	①	V	①	0	①	11 011 101 10 000 110 · d ·	19	
ADD A, (IY+d)	A ← A+(IY+d)	①	①	V	①	0	①	11 111 101 10 000 110 · d ·	19	
ADC A,s	A ← A + s + CY	①	①	V	①	0	①	001		s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction
SUB s	A ← A - s	①	①	V	①	1	①	010		
SBC A,s	A ← A - s - CY	①	①	V	①	1	①	011		
AND s	A ← A ∧ s	0	①	P	①	1	1	100		
OR s	A ← A ∨ s	0	①	P	①	0	0	110		
XOR s	A ← A ⊕ s	0	①	P	①	0	0	101		The indicated bits replace the 000 in the ADD set above.
CP s	A - s	①	①	V	①	1	①	111		
INC r	r ← r + 1	•	①	V	①	0	①	00 r 100	4	
INC (HL)	(HL) ← (HL)+1	•	①	V	①	0	①	00 110 100	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	•	①	V	①	0	①	11 011 101 00 110 100 · d ·	23	
INC (IY+d)	(IY+d) ← (IY+d)+1	•	①	V	①	0	①	11 111 101 00 110 100 · d ·	23	
DEC m	m ← m-1	•	①	V	①	1	①	101		m is any of r, (HL), (IX+d), (IY+d) as shown for INC. Same format and states as INC. Replace 100 with 101 in OP code
ADD HL, ss	HL ← HL+ss	①	•	•	•	0	X	00 ss1 001	11	ss Reg.: 00 BC, 01 DE, 10 HL, 11 SP
ADC HL, ss	HL←HL+ss+CY	①	①	V	①	0	X	11 101 101 01 ss1 010	15	
SBC HL, ss	HL←HL-ss-CY	①	①	V	①	1	X	11 101 101 01 ss0 010	15	
ADD IX, pp	IX ← IX + pp	①	•	•	•	0	X	11 011 101 00 pp1 001	15	pp Reg.: 00 BC, 01 DE, 10 IX, 11 SP
ADD IY, rr	IY ← IY+rr	①	•	•	•	0	X	11 111 101 00 rr1 001	15	rr Reg.: 00 BC, 01 DE, 10 IY, 11 SP
INC ss	ss ← ss + 1	•	•	•	•	•	•	00 ss0 011	6	
INC IX	IX ← IX + 1	•	•	•	•	•	•	11 011 101 00 100 011	10	
INC IY	IY ← IY + 1	•	•	•	•	•	•	11 111 101 00 100 011	10	
DEC ss	ss ← ss -1	•	•	•	•	•	•	00 ss1 011	6	
DEC IX	IX ← IX -1	•	•	•	•	•	•	11 011 101 00 101 011	10	
DEC IY	IY ← IY -1	•	•	•	•	•	•	11 111 101 00 101 011	10	
RLCA		①	•	•	•	0	0	00 000 111	4	Rotate left circular accumulator
RLA		①	•	•	•	0	0	00 010 111	4	Rotate left accumulator
RRCA		①	•	•	•	0	0	00 001 111	4	Rotate right circular accumulator
RRA		①	•	•	•	0	0	00 011 111	4	Rotate right accumulator

Da beim Video Genie bei jedem CLOAD-Befehl gleichzeitig ein NEW ausgeführt und damit das alte Programm gelöscht wird, ist es nicht möglich, Programme durch Hintereinanderladen miteinander zu verketten. Mit der APPEND-Funktion kann dieses Problem beseitigt werden.

In Speicherzelle 16548/16549 ist die Adresse des Programmtextanfangs und in Speicherzelle 16633/16634 die Adresse des Variablentabellenanfangs abgespeichert. Der Programmtext endet immer mit drei Nullbytes, das heißt das erste Nullbyte steht für Zeilenende und das zweite und dritte für Programmtextende. Danach beginnt gleich die Variablentabelle. Die Adresse des Programmtextendes läßt sich nun mit der Adresse des Tabellenanfangs minus zwei (damit die beiden Nullbytes, die das Programmtextende signalisieren, überschrieben werden) ermitteln. Speichert man diese Adresse unter 16548/16549 ab, so ignoriert der Computer einfach das Programm, welches sich momentan im Arbeitsspeicher befindet. Geben Sie deshalb folgendes im Direktmodus ein:

POKE 16548,PEEK(16633)—2: POKE 16549,PEEK(16634)

(funktioniert nur, wenn 16633 größer eins ist).

Das Programm läßt sich jetzt nicht mehr listen und scheint verschwunden zu sein. Nun kann das Laden eines neuen Programms von Kassette erfolgen. Falls man weitere Programme in den Hauptspeicher laden will, muß zuvor immer die oben genannte Zeile eingegeben werden. Danach wird die Manipulation wieder rückgängig gemacht, indem man in 16548/16549 wieder den vom Basic-Interpreter vorgesehenen alten Wert (17129) mit POKE 16548,233: POKE 16549,66 eingibt. Jetzt lassen sich alle Programme listen. Es muß allerdings darauf geachtet werden, daß die Zeilennummern der verschiedenen Programme sich nicht überschneiden. Die Zeilennummern eines zu ladenden Programmes sollten deshalb immer größer sein als die des vorhergehenden Programmes, da sonst diese Zeilen beim Editieren und bei Programmsprüngen vom Interpreter nicht beachtet werden.

(Martin Aschoff)

NACHFOLGER GESUCHT

Da ich in letzter Zeit immer häufiger bemerkte, daß ich durch das Arbeiten für den Club, insbesondere am Clubinfo, kaum mehr Zeit finde selber zu programmieren- , da sich meine berufliche Belastung zusetzlich z.Z. in den Feierabend drängt -, und letztlich da meine Frau sich bei mir beschwert, ich hätte viereckige und dazu monitorgrüne Augen bekommen -

überlege ich mir z.Z. ernsthaft den Club und die Betreuung in andere Hände zu legen.
Leider fehlt hier in der Nähe ein geeigneter Nachfolger und gemäß unserer Satzung bitte ich nun Euch sofern ihr Interesse habt Euch diesbezüglich beim Club zu melden.
Aber bitte nur ernstgemeinte Zuschriften! - die Clubarbeit ist imens angeschwollen und ich möchte nicht, daß die Betreuungsadresse so schnell schon wieder wechselt.

Wer also meint ca. 20-25 Std. mtl. investieren zu können und günstig fotocopieren kann und vielleicht sogar über Floppy UND Cassettenrecorder verfügt (ein Drucker sollte schon sein!)sollte sich schnellstens melden.

Klaus Schmidt

Ausdruck von Umlauten

```
100 '
101 'Michael Karnatz
102 'Schweriner Ring 23
103 '2940 Wilhelmshaven
104 '
105 'TEL 04421 53936
106 '
107 'Ausdrucken von Umlauten  ***********************************************
108 '-------------------------------
109 '
110 'V a r i a b l e n l i s t e  ------------------------------------------------
111  'NAME$=NAME
112 ' N     =Zaehler fuer Namen
113 ' S     =Stelle im Namen
114 ' M$    =zu ueberpruefender mittlerer String
115 ' N a m e n ----------------------------------------------------------------
116 NAME$(1)="Baerbel Janssen"
117 NAME$(2)="Juergen Stoessel"
118 NAME$(3)="Joerg Weiss"
119 '
120 ' A u s d r u c k  ------------------------------------------------------------
121 '
122  FOR N=1 TO 4
123   LPRINT LEFT$(NAME$(N),1);
124    FOR S=2 TO LEN(NAME$(N))
125     M$=MID$(NAME$(N),S,2)
126     IF M$ = "ae" THEN LPRINT CHR$(123); : S=S+1 : GOTO 131
127     IF M$ = "oe" THEN LPRINT CHR$(124); : S=S+1 : GOTO 131
128     IF M$ = "ue" THEN LPRINT CHR$(125); : S=S+1 : GOTO 131
129     IF M$ = "ss" THEN LPRINT CHR$(126); : S=S+1 : GOTO 131
130     LPRINT MID$(NAME$(N),S,1);
131    NEXT S
132   LPRINT MID$(NAME$(N),(S+1),1)
133   LPRINT
134  NEXT N
135 ' E n d e  ***************************************************************
136 END
```

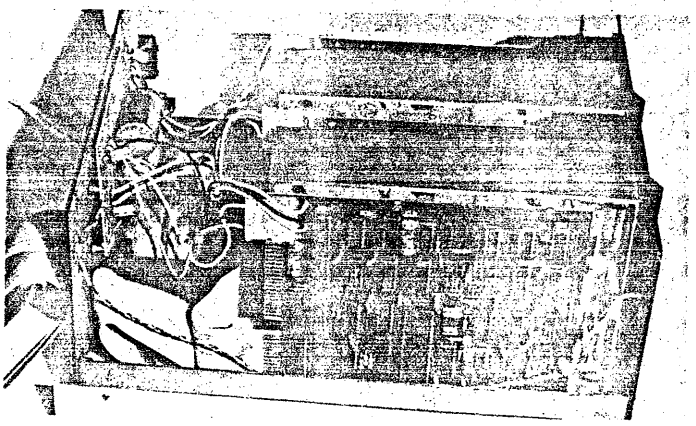
R U N :

```
Bärbel Janßen

Jürgen Stößel

Jörg Weiß
```

Durchgehende Linie

Aus einem Guß ist das in Deutschland von den Firmen Trommeschläger, Kaman und GDOS-Team entwickelte Floppy-System für alle Genie-Rechner. Das System besitzt einen Controller, der bis zu vier Laufwerke regieren kann, wobei sämtliche marktüblichen Laufwerk-Typen (bis zu 80-Track-Doppelkopf-DD!) bedient werden können. Mit dem Controller werden in einem Doppelgehäuse nach Wunsch ein oder zwei Laufwerke ausgeliefert. Grundsätzlich gehört zum Lieferumfang das GDOS, das für die Computer Genie I, II, III auf Floppy kommt und für das Colour-Genie in EPROMs (Adresse ab C000 bis DFFF).

Der Witz des Ganzen ist nun die Kompatibilität aller unter GDOS laufenden Genie-Computer. Ein auf dem Colour-Genie erarbeitetes Programm ist ohne weiteres auf den anderen Genies ladefähig. Mit anderen Worten, Programme und Daten sind jetzt unter den Geniecomputern beliebig austauschbar. Zu beachten ist jedoch, daß natürlich ein Programm, das zum Beispiel 80-Zeichendarstellung verwendet, zunächst nicht auf einem Computer läuft, der nur 40 Zeichen verwendet. Aber man kann mit einem geschickten Pokebefehl auf dem Genie III das Bildschirmformat des Colour-Genies erzeugen, so daß eine Aufwärts-Kompatibilität ganz streng gegeben ist. Idealer Einsatzzweck für Profis könnte sein, daß mit den preiswerten Colour-Genies Datenlogging betrieben wird und mit dem Genie III zum Beispiel die Verarbeitung der gesammelten Daten durchgeführt wird.



Paßt an alle Genies, das Laufwerk mit GDOS

(Trommeschläger, Postfach 2105, 5205 St. Augustin 2, ☎ 0 22 41/2 00 61)

ROM-Adressen Level-II-Basic-Interpeter
Initialisierung und Ein-/Ausgabe

					AF	BC	DE	HL
0000	0000	00000	00000	RST 0, Basic-Kaltstart				
0008	0008	00008	00008	RST 18, Basic-UP s. 1C96				
0010	0010	00016	00016	RST 10, Basic-UP s. 1D78				
0018	0018	00024	00024	RST 18, Basic-UP s. 1C90				
0020	0020	00032	00032	RST 20, Basic-UP s. 25D9				
0028	0028	00040	00040	RST 28, wird bei Drücken der Break-Taste angespr.				
002B	002B	00043	00043	INCH1-Ansprung (über DCB), Tastaturabfrage:	xxxx		4015	
				ASCII-Code gedrückter Taste in A,				
				wenn keine Taste gedrückt ist, A=0				
0030	0030	00048	00048	RST 30, unbenutzt				
0033	0033	00051	00051	OUTCH-Ansprung (über DCB),	xx		401D	
				Ausgabe des Akkuinhaltes auf den Bildschirm				
0038	0038	00056	00056	RST 38H, unbenutzt				
003B	003B	00059	00059	PRINT-Ansprung (über DCB),	xxxx		4025	
				Ausgabe des Akkuinhaltes auf den Drucker				
0049	004F	00073	00079	INCH2, wie INCH1, es wird aber gewartet,	xxxx		4015	
				bis eine Taste gedrückt wird				

					AF	BC	DE	HL
09D3	09DE	02515	02526	(DE...DE+(40AF)) (HL..)	xxxx	00	+4	+4
				(+4, wenn Single Precision Zahl in X)				
09DF	09F3	02527	02547	Vorbereitung der Argumente für Subtraktion und Addition	xxxx	xx		4125
09F4	0A0B	02548	02571	Y ⏵ X	xxxx	00	xxxx	xxxx
09FC	0A0B	02556	02571	X ⏵ Y	xxxx	00	xxxx	xxxx
0A03	0A0B	02563	02571	4121 ⏵ DE (!), 411D ⏵ DE(#)	xxxx		xxxx	
0A0C	0A25	02572	02597	Vergleich: X–BCDE (Single)	xxxx			xxxx
0A26	0A38	02598	02616	UP für Vergleich				
0A39	0A48	02617	02632	Vergleich: HL–DE (Integer)	xxxx			
0A49	0A77	02633	0269	Vergleich: X–Y (Double)	xxxx	xxxx	xxxx	xxxx
0AAF	0A77	02639	02679	Vergleich: X– (DE) (Double)	xxxx	xxxx	xxxx	xxxx
				Nach allen Vergleichen ist bei Gleichheit				
				der Operanden das Zero-Flag gesetzt				
0A7F	0AB0	02687	02736	CINT(X) ⏵ X ⏵ HL	xxxx	xxxx	xxxx	xxxx
0AB1	0ADA	02737	02778	CSNG(X) ⏵ X	xxxx	xxxx	xxxx	xxxx
0ADB	0AF3	02779	02803	CDBL(X) ⏵ X	xxxx	xxxx	xxxx	xxxx
0A9A	0AA2	02714	02722	2 ⏵ (40AF), HL ⏵ X	02			
0A9D	0AA2	02717	02722	2 ⏵ (40AF)	02			
0AEC	0AF3	02796	02803	8 ⏵ (40AF)	08	043E		
0AEF	0AF3	02799	02803	4 ⏵ (40AF)	04			
0AF4	0AFA	02804	02810	Wenn in X kein String, TM-Error	xxxx			
0AF6	0AFA	02806	02810	TM-Error				
0AFB	0B25	02811	02853	Unterprogramm für CINT, FIX und INT				
0B26	0B36	02854	02870	FIX(X) ⏵ X	xxxx	xxxx	xxxx	xxxx
0B37	0B9F	02871	02975	INT(X) ⏵ X	xxxx	xxxx	xxxx	xxxx
0BA0	0BA9	02976	02985	Unterprogramm für INT				
0BAA	0BC6	02986	03014	Unterprogramm für DIM				
0BC7	0C6F	03015	03183	Integer-Arithmetik				
0BC7	0BF1	03015	03057	HL + DE ⏵ HL ⏵ X	xxxx	xxxx	xxxx	xxxx
0BD2	0BF1	03026	03057	HL– DE ⏵ HL ⏵ X	xxxx	xxxx	xxxx	xxxx
0BF2	005A	03058	03162	HL . DE ⏵ HL ⏵ X	xxxx	xxxx	xxxx	xxxx
				Bei Overflow wird in Fließkommaformat umgewandelt				
0C5B	0C6F	03163	03183	UP für Integeraddition				
0C70	0E64	03184	03684	Double-Precision-Arithmetik				
0C70	0D32	03184	03378	X – Y ⏵ X	xxxx	xxxx	xxxx	xxxx
0C77	0D32	03191	03378	X + Y ⏵ X	xxxx	xxxx	xxxx	xxxx
0D33	0D44	03379	03396	Festkommaaddition	xxxx	00	4124	412E
0D45	0D56	03397	03414	Festkommasubtraktion	xxxx	00	4124	412E
0D57	0D68	03415	03432	Unterprogramm für Addition				
0D69	0D8F	03433	03471	(HL...HL–8) um A-Bits nach links schieben	xxxx	xx	0000	
0D90	0D96	03476	03478	(4123..411C) um 1 Bit nach links schieben	xxxx		0000	4123
0D97	0DA0	03479	03488	(HL..HL+8) um 1 Bit nach rechts schieben	xxxx	00		+8
0DA1	0DD3	03489	03439	X .Y ⏵ X	xxxx	xxxx	xxxx	xxxx
0DD4	0DDB	03440	03447	Konstante 10#				
0DD8	0DDB	03444	03447	Konstante 10!				
0DDC	0E38	03448	03640	X / 10 ⏵ X	xxxx	xxxx	xxxx	xxxx
0DE5	0E38	03557	03640	X / Y ⏵ X	xxxx	xxxx	xxxx	xxxx
0E39	0E4C	03641	03660	Unterprogramm für Multiplikation und Division				
0E4D	0E64	03661	03684	X · 10 ⏵ X	xxxx	xxxx	xxxx	xxxx

					AF	BC	DE	HL
0050	005F	00080	00095	Tabelle für die Tastaturdecodierung				
0060	0065	00096	00101	DELAY, Zeitschleife 14,6 µs × BC	0044	0000		
0066	0074	00102	00116	Reset-Ansprung (NMI-Vektor). Wenn Floppy angeschlossen, Basic-Kaltstart, sonst Warmstart				
0075	0104	00117	00260	Basic-Initialisierung:				
0075	008D	00117	00141	DCBs, RST-Vektoren und I/O-Buffer einrichten				
008E	00AB	00142	00171	Disk-Basic-Zeiger initialisieren (werden beim Laden des Disk-Basic geändert)				
00AC	00F8	00172	00248	MEM SIZE anfordern oder selbst Speicherende suchen				
00F9	0104	00249	00260	NEW, „R/S L2 BASIC" ausdrucken und Sprung zur Hauptschleife				
0105	010D	00261	00269	Text „MEM SIZE"				
010E	011B	00270	00283	Text „R/S L2 BASIC(CR)"				
011C	012C	00284	00300	Tastaturentprellung				
012D	0131	00301	00305	L3-Error				
0132	0132	00306	00306	POINT A=00 } Ansprung der drei Grafikbefehle				
0135	0135	00309	00309	SET A=80 } und Setzen des Flags (A).				
0138	0138	00312	00312	RESET A=01 }				
013A	017D	00314	00381	Aus den Koordinaten werden die Adresse im Bildschirmram und die Maske errechnet				
017E	019C	00382	00412	In Abhängigkeit von dem Flag wird ein POINT, SET oder RESET ausgeführt				
019D	01C8	00413	00456	INKE$-Funktion				
01BC	01C8	00444	00456	Leerstring nach X				
01C9	01D2	00457	00466	CLS-Befehl, Bildschirm wird gelöscht	1Fxx			
01D3	01D8	00467	00472	RANDOM-Befehl (als zufällige Größe wird das R-Register benutzt)				
01D9	01F7	00473	00503	Impuls auf Kassette ausgeben	xxxx	00		FC00
01F8	01FD	00504	00509	Kassettenrecorder abschalten	xxxx			
01FE	021D	00510	00541	Kassettenrecordernummer decodieren und Kassettenrecorder einschalten	xxxx	xxxx	xxxx	P
0215	021D	00533	00541	Kassettenrecorder einschalten	xxxx			
021E	0220	00542	00544	Bit 7 von Port (FF) zurücksetzen	xxxx			FC00
0221	022B	00545	00555	(403D) ∧ H ∨ L nach (403D) und zum Port FF	xxxx			
022C	0234	00556	00564	Stern in rechter, oberer Ecke des Bildschirms umschalten	xxxx			
0235	0260	00565	00608	Byte von Kassette lesen (wird im Akku übergeben	xxxx			
0241	0260	00577	00608	Bit (b) von Kassette lesen (2 . A + b ⧫ A)	xxxx			FC00
0261	0283	00609	00643	Byte (im Akku) zweimal auf Kassette aufzeichnen				
0264	0283	00612	00643	Byte (im Akku) auf Kassette aufzeichnen				
0284	0292	00644	00658	Kassette einschalten, 255 mal 0 und A5 aufzeichnen	xxxx	xxxx	xxxx	P
0293	02A8	00659	00680	Kassette einschalten, auf A5 warten und Sternchen auf Bildschirm setzen	2Axx	xxxx	xxxx	P
02A9	0329	00681	00809	SYSTEM-Befehl				
02B2	02B2	00690	00690	Ansprung				
02CE	0313	00718	00787	Objektfile von Kassette laden				
0314	031C	00788	00796	Wort (in HL) von Kassette laden (L, H)	xxxx			xxxx
031D	0329	00797	00809	Ansprung des Objektfile oder irgendeiner anderen Adresse				
032A	0347	00810	00839	Ausgabe des Akkuinhaltes auf Bildschirm (00), Drucker (01) oder Kassette (80) in Abhängigkeit von (409C)	xx			
033A	0347	00826	00839	OUTCH2, wie OUTCH, zusätzlich Cursorposition nach (40A6)	xx			
0348	0357	00840	00855	Position des Cursors in der Bildschirmzeile nach A	xxxx			
0358	0360	00856	00864	INCH3, wie INCH1	xxxx			
0361	0383	00865	00899	INLINE, Eingabe von max. 240 Zeichen in den I/O-Buffer mit allen Cursorfunktionen. Wenn Breack gedrückt wird, Rückkehr mit gesetzten Carry-Flag, bei Drücken von Clear wird der Bildschirm gelöscht und die Eingabe beginnt von vorne. HL enthält die Bufferanfangsadresse –1	xxxx		401D	xxxx
0384	038A	00900	00906	INCH4, wie INCH2	xxxx			
038B	039B	00907	00923	CR auf Drucker ausgeben, wenn Druckkopf nicht in Position 0, (409C) auf 0 setzen	xxxx			
039C	03C1	00924	00961	Zeichen in A auf Drucker ausgeben, Zeichenzähler (409B) incrementieren und bei 0A, 0C und 0D auf 0 setzen				
03C2	03E2	00962	00994	Routine zum Aufruf der Ein-/Ausgabe über die DCBs. (DCB-Typ in B, DCB-Adresse in DE)				
03E3	0457	00995	01111	Tastaturabfrage und Decodierung (Ansprung nur über INCH1-4, da nur dann die Register gerettet werden, siehe dort)				
0458	058C	01112	01420	Bildschirmausgabe (Ansprung nur über DCB, da nur dann Register gerettet werden und die Cursoradresse übergeben wird, siehe OUTCH)				
04B8	058C	01208	01420	Bildschirm-Steuerbefehle				

					AF	BC	DE	HL
Konvertierung für Zahlen-Ein- und Ausgabe								
0E65	0FA6	03685	04006	Umwandlung eines Strings (Zeiger: HL) in eine Zahl (in X)	xxxx	xxxx	xxxx	P
				Unterprogramme:				
0EFB	0F09	03835	03849	Wenn Z=1 ist, wird X in eine Fließkommazahl einfacher Genauigkeit umgewandelt, sonst doppelte Genauigkeit	xxxx			
0F0A	0F17	03864	03863	Typrichtige Multiplikation mit 10	xxxx	xxxx	xxxx	xxxx
0F18	0F28	03864	03880	Typrichtige Division durch 10	xxxx	xxxx		
0F89	0F93	03977	03987	X + A ⇒ X (single)	xxxx	xxxx	00xx	xxxx
0FA7	0FBC	04007	04028	Text „–in–"und HL dezimal ausdrucken	xxxx	xxxx	00xx	xxxx
0FAF	0FBC	04015	04028	HL dezimal ausdrucken	xxxx	xxxx	00xx	xxxx
0FBD	1363	04029	04963	Zahl in X in String umwandeln	xxxx	xxxx	xxxx	xxxx
				Dieser wird in (4130) ff. abgelegt und mit 0 beendet				
1364	13E1	04964	05089	Daten für Umwandlung von Zahlen in Strings				
1364	136B	04964	04971	Konstante $1{\cdot}10^{10}$ #				
136C	1373	04972	04979	Konstante $1{\cdot}10^{15}$ #				
1374	137B	04980	04987	Konstante $1{\cdot}10^{16}$ #				
137C	1383	04988	04995	Konstante ·5 #				
1380	1383	04992	04995	Konstante ·5 !				
1384	138B	04996	05003	Konstante $1{\cdot}10^{16}$ #				
Single Precision Funktionen								
13E2	13E6	05090	05094	UP für SQR und TAN, bewirkt Multiplikation des Ergebnisses mit −1.				xxxx
13E7	1478	05095	05240	SQR(X) ⇒ X (nach SQR(X)=EXP(0.5.LOG(X))	xxxx	xxxx	xxxx	xxxx
13F2	1478	05106	05240	(SP) ⇐ X ⇒ X	xxxx	xxxx	xxxx	xxxx
1439	1478	05177	05240	EXP(X) ⇒ X	xxxx	xxxx	xxxx	xxxx
1479	1499	95241	05273	Konstanten für EXP-Reihe				
149A	14C8	05274	05320	Reihe: $y = c_2x+c_3x^3+c_4x^5$... berechnen HL zeigt auf die Koeffizienten	xxxx	xxxx	xxxx	xxxx
14A9	14C8	05289	05320	Reihe: $y = c_1+c_2x+c_3x^2$... berechnen	xxxx	xxxx	xxxx	xxxx
14C9	1540	05321	05440	RND(X) ⇒ X	xxxx	xxxx	xxxx	xxxx
14F0	1540	05360	05440	Erzeugung einer reellen Zufallszahl (in X)	xxxx	xxxx	xxxx	xxxx
1541	158A	05441	05514	COS(X) ⇒ X	xxxx	xxxx	xxxx	xxxx
1547	158A	05447	05514	SIN(X) ⇒ X	xxxx	xxxx	xxxx	xxxx
158B	158E	05515	05518	Konstante π/2!				
158F	1592	05519	95522	Konstante 0.25!				
1593	15A7	05523	06643	Konstanten für SIN, COS-Reihe				
15A8	15BC	05544	05564	TAN(X) ⇒ X	xxxx	xxxx	xxxx	xxxx
15BD	15E2	05565	05602	ATN(X) ⇒ X	xxxx	xxxx	xxxx	xxxx
15E3	1607	05603	05639	Konstanten für ATN-Reihe				
Tabellen								
1608	164F	05640	05711	Sprungtabelle für Funktionen (Zwischencodes D7-FA)				
1650	1821	05712	06177	Tabelle der Basic-Keywords. Der erste Buchstabe ist durch das gesetzte Bit 7 gekennzeichnet. Die Keywords sind nach aufsteigenden Zwischencodes (80-FA) sortiert				
1822	1899	06178	06297	Sprungtabelle für Befehle (Zwischencodes 80-BB				
189A	18A0	06298	06304	Prioritätscodes für Operatoren				
18A1	18AA	06305	06314	Sprungtabelle für Typanpassung				
18AB	18C8	06315	06344	Sprungtabelle für Grundrechenarten und Vergleich. Für jeden der drei numerischen Datentypen sind diese 5 Adressen enthalten				
18C9	18F6	06345	06390	Tabelle der Fehlerabkürzungen nach aufsteigenden Fehlercodes sortiert				
18F7	191C	06391	06428	Daten, die bei Initialisierung ins RAM übertragen werden (4080–40A6)				
Programmeingabe und -ausführung								
191D	1923	06429	06435	Text „Error"				
1924	1928	06436	06440	Text „–in–"				
1929	192F	06441	06447	Text „READY(CR)"				
1930	1935	06448	06453	Text „Break"				
1936	1954	0654	06484	Unterprogramm für FOR und GOSUB usw. (holt Daten vom Stack zurück)				

					AF	BC	E	HL
1955	1962	06485	06498	Unterprogramm, macht Platz für einzufügende Programmzeile oder Variable. Vorher wird getestet, ob noch genügend Platz frei ist.				
1963	197D	06499	06525	Unterprogramm, testet ob noch 2.C Bytes Speicher frei. Wenn nicht, OM-Error	xxxx	00		
197A	197D	06522	06525	OM-Error				
197E	1990	06526	06544	Implizites END (Erreichen des Programmendes ohne END-Anweisung). Nur bei dieser Programmbeendung wird der NR-Error erkannt!				
1991	1A18	06545	06680	Fehlerbehandlung				
1991	1991	06545	06545	SN-Error in DATA-Zeile				
1997	1997	06551	06551	SN-Error				
199A	199A	06554	06554	/O-Error				
199D	199D	06557	06557	NF-Error				
19A0	19A0	06560	06560	RW-Error				
19A2	19A2	06562	06562	Fehlercode in E (zwischen 0 und 2C)				
1A19	1AF7	06681	06903	Hauptschleife:				
1A3F	1A75	06719	06773	Programmeingabe unter AUTO				
1A76	1AA6	06774	06822	Programmeingabe und Zwischencodeerzeugung				
1AA7	1AF7	06823	06903	Neue Zeile in Programmtext einfügen				
1AF8	1B0F	06904	06927	Zeiger im ganzen Programmtext erneuern	xxxx		xxx	xxxx
1AFC	1B0F	06908	06927	Zeiger im Programmtext ab DE erneuern	xxxx		xxx	xxxx
1B10	1B48	06928	06984	Argumente von LIST, DELETE usw. analysieren Die Adresse des Anfangs der ersten Zeile ist nachher in BC, die zweite Zeilennummer in (SP)	xxxx	xxxx	xxx	P
1B2C	1B48	06956	06984	Sucht Zeile mit Nummer DE im Programmtest. Ist diese Zeile vorhanden, sind beim Rücksprung C und Z gesetzt und die Adresse der Zeile ist in BC. Ist die Zeile nicht vorhanden, enthält BC die Adresse der nächsten Zeile bzw. des Programmendes	xxxx	xxxx		xxxx
1B49	1BB2	06985	07090	NEW-Befehl:				
1B49	1B49	06985	06985	alles löschen				
1B61	1B61	07009	07009	nur Variablen löschen				
1B9A	1B9A	07066	07066	Stack neu initialisieren				
1BB3	1BBF	07091	07103	„–?–" ausdrucken und INLINE (siehe 0361)				
1BC0	1C8F	07104	07311	Text in I/O-Buffer in Zwischencode umwandeln (von HL an). Bei der Rückkehr ist in BC die Länge des Zwischencodes +5, in HL der Anfang des Buffers-3. (Der Zwischencode beginnt 2 Bytes vor Anfang des I/O-Buffers.)	xxxx	xxxx	xxxx	xxxx
1C90	1C95	07312	07317	RST 18H UP: Vergleich HL-DE, die Flagbeeinflussung entspricht den normalen Vergleichsbefehlen	xxxx			
1C96	1CA0	07318	07328	RST 8H UP: (HL) wird mit dem dem RST 8 folgenden Byte verglichen. Bei Gleichheit wird der RST 10 angesprungen, sonst wird ein SN-Error erkannt	xxxx			P
1CA1	1D1D	07329	07453	FOR-TO-STEP-Schleife auswerten				
1D1E	1D90	07454	07568	Steuerung der Programmausführung: Syntaxprüfung, Ansprung der Befehle über Sprungtabelle				
1D78	1D90	07544	07568	RST 10H UP: analysiert Programmtext. HL dient als Zeiger und wird nachgestellt. Ist (HL+1) eine Zahl, so ist bei Rückkehr das Carry-Flag gesetzt, bei 3A und 00 das Zero-Flag. Space und LF werden übergangen	xxxx			P
1D91	1D9A	07569	07578	RESTORE-Befehl				
1D9B	1DAD	07579	07597	Behandlung von Tastendrücken während Programmausführung oder Auflistung				
1DAE	1DE3	07598	07652	Explizites END (siehe auch oben 197E)				
1DB4	1DE3	07604	07651	Programmunterbrechung (Break)				
1DE4	1DF6	07652	07670	CONT-Befehl				
1DF7	1DFF	07671	07679	TRON-Befehl (TRACE-Flag wird auf AF gesetzt)				
1DF8	1DFF	07672	07679	TROFF-Befehl (TRACE-Flag wird auf 0 gesetzt)				
1E00	1E3C	07680	07640	DEFSTR-Befehl				
1E03	1E3C	07683	07640	DETINT-Befehl				
1E06	1E3C	07686	07640	DEFSNG-Befehl				
1E09	1E3C	07689	07640	DEFDBL-Befehl Alle 4 Befehle manipulieren die Typencodetabel e (siehe 4101–411A)				
1E3D	1E44	07741	048	UP, wenn in (HL) ein Buchstabe ist, ist das nich Carry gesetzt	xxxx			
1E45	1E4E	07749	07758	Ganzzahligen Wert (<= 32767) des Ausdruckes ermitteln (in DE), (bei Überschreiten dieser Grenze FC-Error)	xxxx	xxxx	xxxx	P
1E4A	1E4E	07754	07758	FC-Error				
1E4F	1E79	07759	07801	String in Zahl umwandeln. Nur für positive ganze Zahlen bis 65530 (Zeilennummern). Ergebnis in DE	xxxx		xxxx	P
1E7A	1EA2	07802	07842	CLEAR-Befehl				
1EA3	1EDD	07843	07901	RUN-Befehl				

					AF	BC	DE	HL
04B8	04BC	01208	01212	Cursor ON	(0E)			
04BD	04BF	01213	01215	Cursor OFF	(0F)			
04C0	04CD	01216	01229	Cursor HOME (64 cpl)	(1C)			
04CE	04D9	01230	01241	Cursor BACKSPACE	(08)			
04DA	04EB	01242	01259	Cursor ↑	(18)			
04E7	04EB	01255	01259	Cursor ↓	(1A)			
04EC	04F5	01260	01269	Cursor →	(19)			
04F1	04F5	01265	01269	Cursor ←	(1B)			
04F6	0505	01270	01285	32cpl	(17)			
0506	0540	01286	01344	Auswertung der Steuerbefehle				
0541	0563	01345	01379	Zeichen auf Bildschirm setzen und Scroll, wenn nötig				
0564	0572	01380	01394	New-Line	(0A-0D)			
0573	058C	01395	01420	Clear to end of line	(1E)			
057C	058C	01404	01420	Clear to end of frame	(1F)			
058D	05D0	01421	01488	Drucker-Treiber				
05D1	05D8	01489	01496	Drucker bereit?	xxxx			
				j,Z=1.				
05D9	0673	01497	01651	Unterprogramm für INLINE wie INLINE, maximale Anzahl	xxxx	xxxx	401D	
				der Zeichen in B, Bufferanfangsadresse in HL				
				Bei Rückkehr Anzahl der eingegebenen Zeichen in B				
0674	06CB	01652	01739	Wenn Breack gedrückt ist, Sprung zur Basic-				
				Initialisierung, sonst Testen ob Floppy vorhanden				
				Wenn ja, DOS-Laden und starten				
06CC	06CC	01740	01740	Adresse für Rücksprung von SYSTEM-Programmen zum Basic				
				(Basic-Warmstart)				
06D2	0707	01746	01799	RST-Vektoren, DCBs usw. (werden bei Initialisierung				
				ins RAM übertragen (4000–4035)				
Arithmetik								
0708	07F7	01800	02039	X + 1 ↓ X	xxxx	xxxx	xxxx	xxxx
0710	07F7	01808	02039	(HL...) + X ↓ X	xxxx	xxxx	xxxx	xxxx
0713	07F7	01811	02039	BCDE − X ↓ X	xxxx	xxxx	xxxx	xxxx
0716	07F7	01814	02039	BCDE + X ↓ X	xxxx	xxxx	xxxx	xxxx
0778	077C	01912	01916	0 ↓ X	0044			
07A8	07B1	01960	01969	Rundung				
07B2	07B6	01970	01974	OV-Error				
07B7	07C2	01975	01986	Festkommaaddition:	xxxx	xx	xxxx	+2
				(HL...) + CDE ↓ CDE				
07D7	07F7	02007	02039	CDE um A-Bitpositionen	00xx	xxxx	xxxx	xx
				nach rechts schieben				
07F8	07FB	02040	02043	Konstante 1!				
07FC	0808	02044	02056	Konstanten für LOG-Reihe				
0809	0896	02057	02189	LOG(X) ↓ X	xxxx	xxxx	xxxx	xxxx
0841	0896	02057	02189	X . ln2 ↓ X	xxxx	xxxx	xxxx	xxxx
0847	0896	02119	02189	X . BCDE ↓ X	xxxx	xxxx	xxxx	xxxx
0897	0906	02199	02311	X / 10 ↓ X	xxxx	xxxx	xxxx	xxxx
08A2	0906	02211	02310	BCDE / X ↓ X	xxxx	xxxx	xxxx	xxxx
0907	093D	02311	02365	Vorbehandlung der Exponenten und Vorzeichen				
				bei Multiplikation und Division				
093E	0954	02366	02388	X . 10 ↓ X	xxxx	xxxx	xxxx	xxxx
0955	0963	02389	02403	Text X (für Single und Double Presision)	xxxx			
				Wenn X = 0, Z,P				
				Wenn X < 0, C,S				
				Wenn X > 0, keine				
0964	0976	00404	02422	Akku in Fließkommazahl (in X) umformen	xxxx	xxxx	xxxx	xxxx
0977	0989	02423	02441	ABS(X) ↓ X	xxxx	xxxx	xxxx	xxxx
0982	0989	02434	02441	− X ↓ X	xxxx			4123
098A	0993	02442	02451	SGN(X) ↓ X	xxxx			xxxx
0994	09A3	02452	02467	Test X, auch für Integer (siehe oben 0955)	xxxx			xxxx
				bei String in X TM-Error				
09A4	09B0	02468	02480	X ↓ (SP)			xxxx	
09B1	09BE	02481	02494	(HL...) ↓ BCDE ↓ X		xxxx	xxxx	+4
09B4	09BE	02484	02494	BCDE ↓ X			xxxx	
09BF	09CA	02495	02506	X ↓ BCDE		xxxx	xxxx	4125
09C2	09CA	02498	02506	(HL...) ↓ BCDE		xxxx	xxxx	+4
09CB	09DE	02507	02526	X ↓ (HL..)	xxxx	00	4125	+4
09D2	09DE	02514	02526	(HL...HL+(40AF)) ↓ (DE..)	xxxx	00	+4	+4

Fortsetzung folgt