Ondanks zijn bescheiden afmetingen, is de ZX Spectrum groot in mogelijkheden. Omdat het voor de gebruiker praktisch onmogelijk is om alle mogelijkheden uit het hoofd te kennen, is dit zakboekje samengesteld met een opsomming van zo goed als alle belangrijke gegevens.
In grote lijnen kan men het volgende vinden:

- Een beschrijving van het BASIC-systeem, de instructies, de functies en de systeemboodschappen.
- Het gebruik van het geheugen toegelicht met voorbeelden.
- De functies en het gebruik van registers.
- ROM-routines met startadressen.
- Voor machinetaalprogrammeurs: de volledige Z80-instructieset met bovendien een aantal niet officiële Z80-functies.

Het boekje wordt afgesloten met een aantal programma's die, hoewel bedoeld als voorbeeld van het praktisch toepassen van de behandelde theorie, direct bruikbaar zijn als handige 'utilities'.

ISBN 90 201 17068

zakboekje voor de

ZX Spectrum



instructiesets
tabellen
conversielijsten
programmabeschrijvingen

wessel akkermans

kluwer technische boeken

Zakboekje voor de ZX Spectrum

Wessel Akkermans

Zakboekje voor de

ZX Spectrum

Instructiesets
tabellen
conversielijsten
programmabeschrijvingen



Kluwer Technische Boeken b.v. – Deventer – Antwerpen

Omslagontwerp: Peter Michels

ISBN 90 201 1706 8
D/1983/0108/220

© 1983 Kluwer Technische Boeken B.V. Deventer

1e druk 1983

Niets uit deze uitgave mag worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm of op welke andere wijze ook, zonder voorafgaande schriftelijke toestemming van de uitgever.

No part of this book may be reproduced in any form, by print, photoprint, microfilm or any other means without written permission from the publisher.

Ondanks alle aan de samenstelling van de tekst bestede zorg kan noch de redactie noch de uitgever aansprakelijkheid aanvaarden voor eventuele schade die zou kunnen voortvloeien uit enige fout die in deze uitgave zou kunnen voorkomen.

Voorwoord

Reeds lang voordat de ZX Spectrum in Nederland op de markt verscheen voelde ik mij tot deze machine aangetrokken. Dit kwam waarschijnlijk voort uit mijn ervaringen met de ZX81, een computertje met onverwachte mogelijkheden.
Ik hoopte dat de ZX Spectrum ook meer mogelijkheden zou hebben dan je op het eerste gezicht zou denken. Wat dat betreft zijn mijn stoutste verwachtingen uitgekomen.
Hopelijk geeft dit boekje een goed verslag van al die onverwachte mogelijkheden van de ZX Spectrum. De onderwerpen in dit boekje heb ik als volgt gerangschikt:

- Algemene zaken,
- BASIC-systeem,
- Geheugenbeschrijvingen,
- Z80 en hardware gegevens,
- Voorbeeldprogramma's.

Voor de in dit boekje opgenomen Z80-informatie heb ik met toestemming van ZILOG in Engeland gebruik mogen maken van het Z80-CPU manual en een Z80 reference card.
Z80 is een geregistreerd handelsmerk van ZILOG.
Mijn hartelijke dank gaat dan ook uit naar ZILOG en haar Nederlandse importeur Tekelec Airtronic in Zoetermeer.

Ook de Nederlandse importeur van de ZX Spectrum, de firma Compac in Kortenhoef, wil ik bij deze hartelijk danken voor hun verleende medewerking.

Hopelijk heeft u minstens evenveel plezier in het gebruik van dit boek als ik had bij het schrijven ervan.

Wessel Akkermans

Inhoud

Conversietabel decimaal, hexadecimaal, binair en octaal 8
Machten van 2, 8 en 16 12
BASIC-instructies 14
BASIC-functies 33
Bewerkingssymbolen met hun functie 41
Tabel van de ZX Spectrum-karakterset 43
Systeemboodschappen 44
Geheugenindeling 56
Beeldschermgeheugen 58
Attributengeheugen 61
Printer-buffer 62
Systeemvariabelen:
op volgorde van adres 63
op alfabetische volgorde van naam 65
omschrijving van de systeemvariabelen 67
Microdrive maps 80
Channel-informatie 81
Programmageheugen 82
Variabelengeheugen 84
EDIT-geheugen 89
Werkgeheugen 90
Reken-stack 91
Z80-stack 92
GOSUB-stack 93
UDG-geheugen 94
Aanschakelroutine 95
I/O-poortadressen 96
Cassettelabel 100
Tips 103
Het gebruik van ROM-routines 105
Programmabeschrijvingen:
VU-CALC 108
VU-FILE 111
Z80-CPU met signaalbeschrijving 115
Uitbreidingsconnector 118
Z80-registers 120
Z80-instructieset op volgorde van mnemonics 124
Z80-instructieset op volgorde van hex.code 133
Z80-vlagbeinvloeding 142
Symbolische omschrijving van Z80-instructies 144
Niet officiele Z80-instructies 149
Programma: DECHEX-conversie 151
Hexadecimale geheugendump 153
Hexlader 155
ROM-test 158
UDG-grootte 160
VARLIST 162
Aantekeningen 165

Conversietabel decimaal, hexadecimaal, binair en octaal

dec	hex	bin	oct	dec	hex	bin	oct
0	0	0000 0000	0	36	24	0010 0100	44
1	1	0000 0001	1	37	25	0010 0101	45
2	2	0000 0010	2	38	26	0010 0110	46
3	3	0000 0011	3	39	27	0010 0111	47
4	4	0000 0100	4	40	28	0010 1000	50
5	5	0000 0101	5	41	29	0010 1001	51
6	6	0000 0110	6	42	2A	0010 1010	52
7	7	0000 0111	7	43	2B	0010 1011	53
8	8	0000 1000	10	44	2C	0010 1100	54
9	9	0000 1001	11	45	2D	0010 1101	55
10	A	0000 1010	12	46	2E	0010 1110	56
11	B	0000 1011	13	47	2F	0010 1111	57
12	C	0000 1100	14	48	30	0011 0000	60
13	D	0000 1101	15	49	31	0011 0001	61
14	E	0000 1110	16	50	32	0011 0010	62
15	F	0000 1111	17	51	33	0011 0011	63
16	10	0001 0000	20	52	34	0011 0100	64
17	11	0001 0001	21	53	35	0011 0101	65
18	12	0001 0010	22	54	36	0011 0110	66
19	13	0001 0011	23	55	37	0011 0111	67
20	14	0001 0100	24	56	38	0011 1000	70
21	15	0001 0101	25	57	39	0011 1001	71
22	16	0001 0110	26	58	3A	0011 1010	72
23	17	0001 0111	27	59	3B	0011 1011	73
24	18	0001 1000	30	60	3C	0011 1100	74
25	19	0001 1001	31	61	3D	0011 1101	75
26	1A	0001 1010	32	62	3E	0011 1110	76
27	1B	0001 1011	33	63	3F	0011 1111	77
28	1C	0001 1100	34	64	40	0100 0000	100
29	1D	0001 1101	35	65	41	0100 0001	101
30	1E	0001 1110	36	66	42	0100 0010	102
31	1F	0001 1111	37	67	43	0100 0011	103
32	20	0010 0000	40	68	44	0100 0100	104
33	21	0010 0001	41	69	45	0100 0101	105
34	22	0010 0010	42	70	46	0100 0110	106
35	23	0010 0011	43	71	47	0100 0111	107

dec	hex	bin	oct	dec	hex	bin	oct
72	48	0100 1000	110	116	74	0111 0100	164
73	49	0100 1001	111	117	75	0111 0101	165
74	4A	0100 1010	112	118	76	0111 0110	166
75	4B	0100 1011	113	119	77	0111 0111	167
76	4C	0100 1100	114	120	78	0111 1000	170
77	4D	0100 1101	115	121	79	0111 1001	171
78	4E	0100 1110	116	122	7A	0111 1010	172
79	4F	0100 1111	117	123	7B	0111 1011	173
80	50	0101 0000	120	124	7C	0111 1100	174
81	51	0101 0001	121	125	7D	0111 1101	175
82	52	0101 0010	122	126	7E	0111 1110	176
83	53	0101 0011	123	127	7F	0111 1111	177
84	54	0101 0100	124	128	80	1000 0000	200
85	55	0101 0101	125	129	81	1000 0001	201
86	56	0101 0110	126	130	82	1000 0010	202
87	57	0101 0111	127	131	83	1000 0011	203
88	58	0101 1000	130	132	84	1000 0100	204
89	59	0101 1001	131	133	85	1000 0101	205
90	5A	0101 1010	132	134	86	1000 0110	206
91	5B	0101 1011	133	135	87	1000 0111	207
92	5C	0101 1100	134	136	88	1000 1000	210
93	5D	0101 1101	135	137	89	1000 1001	211
94	5E	0101 1110	136	138	8A	1000 1010	212
95	5F	0101 1111	137	139	8B	1000 1011	213
96	60	0110 0000	140	140	8C	1000 1100	214
97	61	0110 0001	141	141	8D	1000 1101	215
98	62	0110 0010	142	142	8E	1000 1110	216
99	63	0110 0011	143	143	8F	1000 1111	217
100	64	0110 0100	144	144	90	1001 0000	220
101	65	0110 0101	145	145	91	1001 0001	221
102	66	0110 0110	146	146	92	1001 0010	222
103	67	0110 0111	147	147	93	1001 0011	223
104	68	0110 1000	150	148	94	1001 0100	224
105	69	0110 1001	151	149	95	1001 0101	225
106	6A	0110 1010	152	150	96	1001 0110	226
107	6B	0110 1011	153	151	97	1001 0111	227
108	6C	0110 1100	154	152	98	1001 1000	230
109	6D	0110 1101	155	153	99	1001 1001	231
110	6E	0110 1110	156	154	9A	1001 1010	232
111	6F	0110 1111	157	155	9B	1001 1011	233
112	70	0111 0000	160	156	9C	1001 1100	234
113	71	0111 0001	161	157	9D	1001 1101	235
114	72	0111 0010	162	158	9E	1001 1110	236
115	73	0111 0011	163	159	9F	1001 1111	237

dec	hex	bin	oct	dec	hex	bin	oct
160	A0	1010 0000	240	204	CC	1100 1100	314
161	A1	1010 0001	241	205	CD	1100 1101	315
162	A2	1010 0010	242	206	CE	1100 1110	316
163	A3	1010 0011	243	207	CF	1100 1111	317
164	A4	1010 0100	244	208	D0	1101 0000	320
165	A5	1010 0101	245	209	D1	1101 0001	321
166	A6	1010 0110	246	210	D2	1101 0010	322
167	A7	1010 0111	247	211	D3	1101 0011	323
168	A8	1010 1000	250	212	D4	1101 0100	324
169	A9	1010 1001	251	213	D5	1101 0101	325
170	AA	1010 1010	252	214	D6	1101 0110	326
171	AB	1010 1011	253	215	D7	1101 0111	327
172	AC	1010 1100	254	216	D8	1101 1000	330
173	AD	1010 1101	255	217	D9	1101 1001	331
174	AE	1010 1110	256	218	DA	1101 1010	332
175	AF	1010 1111	257	219	DB	1101 1011	333
176	B0	1011 0000	260	220	DC	1101 1100	334
177	B1	1011 0001	261	221	DD	1101 1101	335
178	B2	1011 0010	262	222	DE	1101 1110	336
179	B3	1011 0011	263	223	DF	1101 1111	337
180	B4	1011 0100	264	224	E0	1110 0000	340
181	B5	1011 0101	265	225	E1	1110 0001	341
182	B6	1011 0110	266	226	E2	1110 0010	342
183	B7	1011 0111	267	227	E3	1110 0011	343
184	B8	1011 1000	270	228	E4	1110 0100	344
185	B9	1011 1001	271	229	E5	1110 0101	345
186	BA	1011 1010	272	230	E6	1110 0110	346
187	BB	1011 1011	273	231	E7	1110 0111	347
188	BC	1011 1100	274	232	E8	1110 1000	350
189	BD	1011 1101	275	233	E9	1110 1001	351
190	BE	1011 1110	276	234	EA	1110 1010	352
191	BF	1011 1111	277	235	EB	1110 1011	353
192	C0	1100 0000	300	236	EC	1110 1100	354
193	C1	1100 0001	301	237	ED	1110 1101	355
194	C2	1100 0010	302	238	EE	1110 1110	356
195	C3	1100 0011	303	239	EF	1110 1111	357
196	C4	1100 0100	304	240	F0	1111 0000	360
197	C5	1100 0101	305	241	F1	1111 0001	361
198	C6	1100 0110	306	242	F2	1111 0010	362
199	C7	1100 0111	307	243	F3	1111 0011	363
200	C8	1100 1000	310	244	F4	1111 0100	364
201	C9	1100 1001	311	245	F5	1111 0101	365
202	CA	1100 1010	312	246	F6	1111 0110	366
203	CB	1100 1011	313	247	F7	1111 0111	367

dec	hex	bin	oct
248	F8	1111 1000	370
249	F9	1111 1001	371
250	FA	1111 1010	372
251	FB	1111 1011	373
252	FC	1111 1100	374
253	FD	1111 1101	375
254	FE	1111 1110	376
255	FF	1111 1111	377
256	100	10000 0000	400

Hierna volgen nog enkele conversies van grotere getallen, die veel worden gebruikt bij de aanduiding van geheugengroottes. Daar bij deze aanduidingen in praktisch alle literatuur het hexadecimale getallenstelsel wordt gebruikt, zullen in de volgende tabel alleen de decimale en hexadecimale waarden worden gegeven.

	aantal bytes		hoogste adres
	dec	hex	
1K	1024	04 00	03 FF
2K	2048	08 00	07 FF
4K	4096	10 00	0F FF
8K	8192	20 00	1F FF
16K	16384	40 00	3F FF
24K	24576	60 00	5F FF
32K	32768	80 00	7F FF
40K	40960	A0 00	9F FF
48K	49152	C0 00	BF FF
56K	57344	E0 00	DF FF
64K	65536	1 00 00	FF FF

Machten van 2, 8 en 16

Machten van 2:

2^0 = 1	2^23 = 8388608
2^1 = 2	2^24 = 16777216
2^2 = 4	2^25 = 33554432
2^3 = 8	2^26 = 67108864
2^4 = 16	2^27 = 134217728
2^5 = 32	2^28 = 268435456
2^6 = 64	2^29 = 536870912
2^7 = 128	2^30 = 1073741824
2^8 = 256	2^31 = 2147483648
2^9 = 512	2^32 = 4294967296
2^10 = 1024	2^33 = 8589934592
2^11 = 2048	2^34 = 17179869184
2^12 = 4096	2^35 = 34359738368
2^13 = 8192	2^36 = 68719476736
2^14 = 16384	2^37 = 137438953472
2^15 = 32768	2^38 = 274877906944
2^16 = 65536	2^39 = 549755813888
2^17 = 131072	2^40 = 1099511627776
2^18 = 262144	2^41 = 2199023255552
2^19 = 524288	2^42 = 4398046511104
2^20 = 1048576	2^43 = 8796093022208
2^21 = 2097152	2^44 = 17592186044416
2^22 = 4194304	2^45 = 35184372088832

Machten van 8:

8^0 = 1
8^1 = 8
8^2 = 64
8^3 = 512
8^4 = 4096
8^5 = 32768
8^6 = 262144
8^7 = 2097152
8^8 = 16777216
8^9 = 134217728
8^10 = 1073741824
8^11 = 8589934592
8^12 = 68719476736
8^13 = 549755813888
8^14 = 4398046511104
8^15 = 35184372088832

Machten van 16:

16^0 = 1
16^1 = 16
16^2 = 256
16^3 = 4096
16^4 = 65536 (64 K)
16^5 = 1048576 (1 M)
16^6 = 16777216
16^7 = 268435456
16^8 = 4294967296
16^9 = 68719476736
16^10 = 1099511627776
16^11 = 17592186044416

BASIC-instructies

BEEP

BEEP <tijdsduur>,<toonhoogte>

Geeft een toon van de aangegeven tijdsduur en toonhoogte op de ingebouwde luidspreker.
0 <= tijdsduur < 10,5
-60 <= toonhoogte < 70

BORDER

BORDER <kleurnummer>

Hiermee wordt de 'Borderkleur' samen met de papierkleur van het 'lower-part' van het scherm de aangegeven kleur toegekend. Alleen de kleurnummers 0 tot en met 7 zijn toegestaan. Getallen achter de komma worden op een half naar boven of beneden afgerond.

BRIGHT

BRIGHT <mode>

Zet de helderheid van de karakterposities waarop wordt ge-'print' op:
0 = niet helder
1 = helder
8 = de karakterpositie behoudt de helderheid die het al had.
Andere waarden voor <mode> zijn niet toegestaan. Getallen met cijfers achter de komma worden op een half naar boven of beneden afgerond.

CAT

CAT

Op de standaard ZX Spectrum, dus zonder expansion interface, geeft het gebruik van deze instructie de systeemboodschap 0.

CIRCLE

CIRCLE <X-coordinaat>,<Y-coordinaat>,<straal>

Tekent een cirkel met als middelpunt de beeldpuntcoordinaten X,Y en met een straal ter lengte van het aantal in <straal> aangegeven beeldpunten.
0 <= X-coordinaat <= 255
0 <= Y-coordinaat <= 175
<straal> moet gelijk zijn aan of kleiner zijn dan het verschil tussen 0 en de X-coordinaat, 255 en de X-coordinaat, 0 en de Y-coordinaat, 175 en de Y-coordinaat.

CLEAR

CLEAR <RAMTOP>

Indien <RAMTOP> wordt weggelaten, dan heeft CLEAR de volgende functie:
- Wissen van variabelengeheugen en GOSUB-stack.
- Uitvoeren van een RESTORE en een CLS.
- Resetten van de PLOT-positie op 0,0.
Indien wel een <RAMTOP> wordt gespecificeerd, dan heeft CLEAR dezelfde gevolgen als hierboven, doch bovendien wordt dan de systeemvariabele RAMTOP op de aangegeven waarde gezet en wordt de GOSUB-stack op deze nieuwe plaats in het geheugen gezet.

CLOSE#

CLOSE# <streamnummer>

Deze instructie is bedoeld om bij de microdrives te worden gebruikt. Gebruik van deze instructie met een streamnummer hoger dan 3 op een

standaard ZX Spectrum kan een systeem-hangup veroorzaken. Hier is alleen door uit en aanschakelen weer uit te komen.
0 <= streamnummer <= 15

CLS

CLS

Het hele beeldschermgeheugen wordt gewist en de attributen worden gezet op de waarden zoals die op dat moment voorkomen in de systeemvariabelen ATTR-P en MASK-P. De attributen voor het onderste deel van het scherm behouden hun laatste waarde.

CONTINUE

CONTINUE

Gaat verder met de volgende instructie indien er was gestopt met de systeemboodschap 9 of L. Herhaalt de zelfde regel indien er met een andere systeemboodschap was gestopt.

COPY

COPY

Stuurt het bovenste deel van het beeldscherm (max. 22 regels) naar de printer. Indien geen printer is aangesloten doet deze instructie niets.

DATA

DATA $\left[\begin{matrix}n\\1\end{matrix}\{\left\lfloor\begin{matrix}\langle num\rangle\\\langle var\rangle\\\text{"}\langle str\rangle\text{"}\end{matrix}\right\rfloor,\}\right]$

$\langle num\rangle$ = numerieke waarde
$\langle var\rangle$ = numerieke- of stringvariabele
$\langle str\rangle$ = string

Met DATA-instructie(s) kan een DATA-list worden samengesteld. Een DATA-list mag uit meerdere DATA-instructies bestaan. Deze

DATA-instructies hoeven niet allemaal bij elkaar te staan. Iedere DATA-instructie kan van 1 tot n DATA-gegevens bevatten. Deze gegevens mogen zowel numeriek als alfanumeriek zijn. Alle gegevens dienen van elkaar te zijn gescheiden door een komma. Het laatste gegeven mag niet meer gevolgd worden door een komma. De grootte van n is afhankelijk van de lengte van de afzonderlijke gegevens, het INPUT-buffer en de geheugengrootte.

DEF FN

DEF FN $\langle letter\rangle[\$][(\langle arg\rangle)]=\langle expr.\rangle$

$\langle arg\rangle = \left[\begin{matrix}n\\1\end{matrix}\{\langle letter\rangle[\$],\}\right]\langle letter\rangle[\$]$

Met DEF FN kan men zelf een functie definieren. Deze functie geeft men dan een naam van 1 letter, eventueel gevolgd door $ indien het een stringfunctie betreft. Daarna kunnen aan die naam een aantal argumenten worden toegevoegd, die tussen haakjes moeten staan. Indien deze argumentnamen identiek zijn aan de variabelenamen in de achter het = teken staande expressie, dan kunnen met de functie FN waarden aan die expressie worden toegekend (dit gaat dan automatisch via de DEF FN).
Indien geen argumenten worden gespecificeerd, dan zullen de in de expressie voorkomende variabelen als normale variabelen worden beschouwd, en zullen ze dus ergens in het programma moeten zijn gespecificeerd.
Het maximum aantal argumenten bij een functie is:
26 numerieke- + 26 stringargumenten, die door elkaar mogen staan.
Zie verder de BASIC-functie FN.

DIM

DIM <letter>(<arrayindex> $\left[\begin{matrix}n\\1\end{matrix}\right.$ {,<arrayindex>} $\left.\right]$)

Creeert ruimte in het variabelengeheugen voor een numerieke array met een grootte die afhangt van de tussen haakjes gegeven dimensies. Zie ook de beschrijving van het variabelengeheugen.

DIM

DIM <letter>$(<arrayindex> $\left[\begin{matrix}n\\1\end{matrix}\right.$ {,<arrayindex>} $\left.\right]$)

Creëert ruimte in het variabelengeheugen voor een string array met een grootte die afhangt van de tussen haakjes opgegeven dimensies.

DRAW

DRAW <hor>,<vert>[,<hoek>]

DRAW trekt een lijn vanaf de 'current cursor'-positie naar een punt dat <hor> beeldpunten horizontaal en <vert> beeldpunten verticaal van die 'current cursor' positie is verwijderd. Na aanschakelen, NEW of CLS wordt de 'current cursor'-positie gereset naar de linker benedenhoek van het beeldscherm.
Indien <hoek> niet is gespecificeerd, dan wordt er een rechte lijn getrokken. Dit is ook het geval indien de opgegeven hoek 0, 2*PI of een veelvoud daarvan is.
Heeft de parameter <hoek> een andere waarde, dan zal een cirkelboog worden getekend. Het eindpunt daarvan zijn de 'current cursor'-positie en het punt met de verplaatsing <hor>,<vert>. De grootte van de boog hangt af van de

waarde in <hoek>. (PI zou bijvoorbeeld een halve cirkel tekenen.)
-255 <= hor <= 255
-175 <= vert <= 175
-255 <= hoek <= 255
(Getallen groter dan 2*PI of kleiner dan -2*PI geven meerdere cirkels over elkaar.)

ERASE

ERASE

Op de standaard ZX Spectrum, dus zonder expansion interface, veroorzaakt deze instructie de systeemboodschap 0.

FLASH

FLASH <mode>

Hiermee wordt aangegeven of karakters moeten knipperen of niet. <mode> mag een waarde 0, 1 of 8 hebben. Deze waarde kan in de vorm van een numerieke expressie worden toegekend.
0 = niet knipperen
1 = knipperen
8 = geen wijziging (een PRINT zonder FLASH 8 zou een knipperend tekstveld weer niet-knipperend maken.)

FOR

FOR <letter>=<num> TO <num>[STEP <num>]

<letter> is de 'FOR..NEXT'-variabelenaam. (Zie ook de beschrijving van het variabelengeheugen.)

<num> is een numerieke uitdrukking (waarde of variabele), die zowel positief als negatief kan zijn.

Alle instructies die tussen de FOR-instructie en de NEXT-instructie met dezelfde variabele (<letter>) staan worden uitgevoerd. De NEXT-instructie controleert of <num> achter TO al is

bereikt, zoniet dan wordt <num> verhoogd met de STEP-waarde, en worden de instructies na de FOR opnieuw uitgevoerd. Indien de STEP-waarde negatief is wordt de waarde dus verlaagd. Was <num> achter TO al wel bereikt dan wordt verder gegaan met de instructies na de NEXT. Indien STEP wordt weggelaten, dan neemt het systeem aan dat de STEP-waarde 1 is.

FORMAT

FORMAT

Op de standaard ZX Spectrum, dus zonder expansion interface, resulteert deze instructie altijd in de systeemboodschap 0.

GOSUB

GOSUB <regelnummer>

<regelnummer> is een numerieke expressie.

De GOSUB springt naar het aangegeven regelnummer, doch onthoudt het volgende regelnummer. Zodra een RETURN-instructie wordt uitgevoerd, wordt naar het onthouden regelnummer teruggesprongen.

GOTO

GOTO <regelnummer>

<regelnummer> is een numerieke expressie. De GOTO-instructie veroorzaakt een sprong naar het aangegeven regelnummer.

IF..THEN

IF <uitdrukking> THEN <instructie>

Indien de uitdrukking waar is, dan zal de instructie achter THEN worden uitgevoerd. Is de uitdrukking echter niet waar, dan wordt de instructie op de volgende programmaregel uitgevoerd.

INK

INK <nummer>

<nummer> is een numerieke expressie. Hiermee kan worden aangegeven met welke kleur de karakters moeten worden geschreven. De toegestane waarden van <nummer> zijn:

0 t/m 7 = De boven die toetsen aangegeven kleuren.

8 = Geen wijziging ten opzichte van de reeds bestaande kleur.

9 = Het systeem zal een kleur kiezen die goed contrasteert met de papierkleur.

-0.5 <= nummer < 9.5 .
Getallen achter de komma worden op een half naar boven of naar beneden afgerond.

INPUT

INPUT ⌈⌈⌊<func>|"<tekst>"|<var>⌈$⌋⌋ⁿ₁(⌊',|;⌋)⌈⌊<func>|"<tekst>"|<var>⌈$⌋⌋

Hiermee kunnen via het toetsenbord waarden aan variabelen worden toegekend.

<func> = Een van de functies AT, TAB, INK, PAPER, FLASH, BRIGHT, INVERSE of OVER.

<tekst>= De tekst die op het onderste deel van het beeldscherm moet worden afgedrukt. Bij INPUT is

het niet mogelijk de inhoud van een variabele te laten afdrukken.

‹var› = De variabele waarin de in te geven data zal worden opgeslagen. Indien ‹var› door het $- teken wordt gevolgd, dan betreft het een string-variabele, zoniet dan betreft het een numerieke variabele. Ook numerieke variabelen die namen hebben van meer dan een letter zijn toegestaan.

Daar de te printen tekst op het onderste deel van het beeldscherm moet worden afgedrukt, zal bij gebruik van meer dan twee regels de reactie van het systeem anders zijn dan bij een normale PRINT-instructie.

INVERSE

INVERSE ‹nummer›

‹nummer› is een numerieke expressie. Met INVERSE kunnen de kleuren van inkt en papier onderling worden verwisseld in de daarna af te drukken tekst.
-0.5 <= nummer < 1.5 .
Het resultaat van de numerieke expressie wordt op een half naar boven of beneden afgerond. Alleen de afgeronde waarden 0 en 1 zijn geldig. Alle andere waarden geven een foutboodschap.
De waarde 0 = normaal.
De waarde 1 = invers.

LET

LET ‹var› = ‹expressie›

Hiermee wordt de waarde van de ‹expressie› toegekend aan de variabele met de in ‹var› aangegeven naam. Indien ‹var› een gedimensioneerde string-variabele of een "sliced"

string-variabele is, dan is de lengte vast. Indien de expressie te lang is zal het rechter deel van die stringexpressie verloren gaan. Indien de expressie korter is dan de variabele-lengte dan zal het rechter deel van de variabele worden opgevuld met spaties.

LIST

LIST ‹regelnummer›

‹regelnummer› is een numerieke expressie. 0 <= regelnummer <= 65535. Indien de numerieke expressie wordt weggelaten, dan neemt het systeem aan dat regelnummer 0 wordt bedoeld.
Met LIST kan een listing van het programma worden gemaakt vanaf het aangegeven regelnummer of vanaf de eerste regel volgende op het aangegeven regelnummer. De listing zal op het beeldscherm worden gemaakt. Zodra het beeldscherm vol is stopt de listing en wordt u gevraagd of er ge-"scrolled" moet worden. Hoewel in deze instructie het maximum toegestane regelnummer 65535 is, kan een programma nooit meer dan 9999 regels bevatten.

LLIST

LLIST ‹regelnummer›

Heeft dezelfde functie als LIST, doch maakt die listing op de printer. Er zal dan ook niet worden gevraagd of er moet worden ge-"scrolled".

LOAD

LOAD "[‹naam›]"[,DATA ‹letter›[$]()|CODE[‹start›[,‹lengte›]]|SCREEN$]

Terugladen van programma's, strings en arrays, bytes en beeldschermgegevens, die met SAVE op cassette waren gezet. Indien geen ‹naam› wordt

gespecificeerd, zal het eerste programma, de eerste string of array of de eerste bytes die worden gevonden, in het geheugen worden geladen. Dit natuurlijk op voorwaarde dat het de met DATA, CODE en SCREEN$ aangegeven type data is.
Indien na de naam geen verdere parameters worden opgegeven, dan wordt aangenomen dat er een programma moet worden geladen.
Wordt DATA gespecificeerd, dan zal de array of string met de in <letter> gegeven naam worden geladen.
Met CODE kunnen bytes in het geheugen worden geladen. Deze bytes worden vanaf het adres <start> in het geheugen gezet en er vindt controle plaats op de juiste <lengte> van het aantal bytes.
Indien alleen <start> was gespecificeerd, dan wordt niet op lengte gecontroleerd. Indien zowel <start> als <lengte> worden weggelaten, dan worden de aangegeven bytes op dezelfde plaats teruggezet in het geheugen waar ze met SAVE vandaan werden gehaald.
Wordt SCREEN$ gespecificeerd, dan wordt de betreffende file op cassette opgezocht en in het beeldschermgeheugen geladen.

LPRINT

LPRINT . . . .

Zie voor de werking van deze instructie de beschrijving van de PRINT-instructie. LPRINT doet hetzelfde, doch dan op de printer.

MERGE

MERGE <letter>$|"⌈<naam>⌉"

Met MERGE kan een programma of een programmadeel van cassette worden geladen en in het programmageheugen

worden gezet. De met MERGE geladen programmaregels worden op hun regelnummer tussen de reeds in het geheugen staande programmaregels gevoegd. Zijn de regelnummers van het reeds in het geheugen staande programma gelijk aan de met MERGE geladen programmaregelnummers, dan wordt het oude programma overschreven.

MOVE

MOVE ⌊<letter>$|"⌈<naam>⌉"⌋ ⌊<letter>$|"⌈<naam>⌉"⌋

Op de standaard ZX Spectrum, dus zonder expansion interface, is het resultaat van deze instructie de systeemboodschap O.

NEW

NEW

Hiermee wordt het systeem opnieuw geinitialiseerd. In grote lijnen doet NEW het zelfde als de aanschakel routine, alleen behouden nu de systeemvariabelen RAMTOP, P-RAMT, RASP, PIP en UDG hun oude waarden.

NEXT

NEXT <letter>

<letter> is de variabelenaam van de FOR..NEXT-control variabele. NEXT veroorzaakt een sprong terug naar de FOR-instructie met dezelfde controlvariabelenaam en verhoogt of verlaagt de met <letter> aangegeven variabele (afhankelijk van de STEP-waarde).

OPEN#

OPEN#<stream nr>,"<channel>"

Deze instructie is bedoeld om te worden gebruikt op de ZX Spectrum met de expansion interface. Hij werkt echter

ook op de standaard ZX Spectrum, alleen mag het streamnummer niet hoger zijn dan 3.

OUT

OUT ‹poort›,‹waarde›

‹poort› wordt in het Z80-registerpaar BC geladen.
‹waarde› wordt in de Z80-accumulator (register A) geladen.
Hierna wordt de machine-instructie OUT (c),a uitgevoerd.
(Voorbeeld: OUT 0,3 maakt de border magenta, doch deze kleur is niet blijvend. Bij het eerstvolgende beeldschermgebruik krijgt de border weer de kleur die hij had voor de OUT werd uitgevoerd.)

OVER

OVER ‹nummer›

‹nummer› is een numerieke expressie.
Met OVER kunnen twee karakters over elkaar heen worden geschreven. De beperking van 1 inktkleur en 1 papierkleur per karakterpositie blijft echter onverminderd van kracht.
‹nummer› mag alleen 0 of 1 zijn.
0 = Print het nieuwe karakter in de plaats van het oude karakter.
1 = Print het nieuwe karakter over het oude heen zonder het oude te wissen. Hierbij zullen inkt beeldpunten die over elkaar heen vallen de papier kleur krijgen.
-0.5 <= nummer < 1.5 .
De getallen achter de komma worden op een half naar boven of naar beneden afgerond.

PAPER

PAPER ‹nummer›

‹nummer› is een numerieke expressie.
De kleur van het papier kan hiermee

worden aangegeven. De waarde van ‹nummer› mag 0 t/m 7, 8 of 9 zijn.
0 t/m 7 = De boven de toets aangegeven kleur.
8 = Geen wijziging ten opzichte van de oude kleur.
9 = Het systeem kiest een kleur die goed met de kleur van de inkt contrasteerd.
-0.5 <= nummer < 9.5 .
Getallen achter de komma worden op een half naar boven of naar beneden afgerond.

PAUSE

PAUSE ‹nummer›

‹nummer› is een numerieke expressie.
De uitvoering van het programma wordt voor een periode van (‹nummer› * 1/50) seconden gestopt. Indien ‹nummer› de waarde 0 heeft, dan wordt het programma voor onbepaalde tijd gestopt. Door het indrukken van een willekeurige toets wordt het programma doorgestart, ongeacht of de aangegeven tijd is verstreken of niet.

PLOT

PLOT [‹func›;]‹hor›,‹vert›

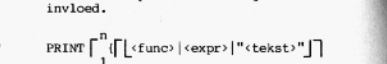
‹hor› en ‹vert› zijn numerieke expressies.
Het beeldpunt op de in ‹hor› en ‹vert› aangegeven coordinaten wordt in de op dat moment geldende kleuren van papier en inkt afgedrukt, tenzij met ‹func› iets anders is aangegeven.
‹func› = Een van de functies INK, PAPER, FLASH, BRIGHT, OVER of INVERSE.
0 <= hor <= 255
0 <= vert <= 175
(linksonder is 0,0; rechtsboven is 255,175)

POKE

POKE <adres>,<waarde>

<adres> en <waarde> zijn numerieke expressies.
De aangegeven <waarde> wordt naar het <adres> geschreven.
0 <= adres <= 65535
0 <= waarde <= 255
Daar de eerste 16 K adressen ROM zijn heeft POKE op die adressen geen invloed.

PRINT



Met PRINT wordt de aangegeven tekst in het bovenste deel van het beeldscherm geschreven.

<func> = Een van de functies AT, TAB, PAPER, INK, FLASH, BRIGHT, INVERSE of OVER.
<expr> = Een numerieke- of string-expressie, waarin weer diverse functies en operatoren zijn toegestaan.
<tekst> = Een of meer karakters die moeten worden afgedrukt zoals ze tussen de aanhalingstekens zijn weergegeven.
' = Een regel opschuiven.
; = Karakterpositie niet wijzigen.
, = Karakterpositie naar de volgende kolom. (Het scherm heeft twee kolommen, positie 0 en 16)

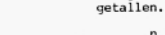
Indien aan het einde van de instructie geen apostrophe, puntkomma of komma wordt ingegeven, dan zal het effect zijn dat de karakterpositie naar de volgende regel wordt verplaatst.

RANDOMIZE



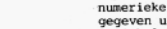
<nummer> is een numerieke expressie.
0 <= <nummer> <= 65535.
De systeemvariabele SEED wordt geladen met de waarde in <nummer>. SEED wordt door de functie RND gebruikt om een serie "pseudo-random"-getallen op een bepaalde plaats te beginnen.
Indien <nummer> wordt weggelaten of de waarde 0 heeft, dan kijkt de functie RND in de systeemvariabele FRAMES om de SEED met een waarde te laden. FRAMES telt het aantal gegenereerde TV-beelden en zal dus een redelijke mate van willekeurigheid tot gevolg hebben bij het genereren van willekeurige getallen.

READ



READ kent de waarden uit de DATA-list toe aan de gespecificeerde variabelen. De eerste variabele uit de eerste READ-instructie krijgt de eerste waarde uit de DATA-list. De tweede variabele uit de eerste READ-instructie krijgt de tweede waarde uit de DATA-list, enz..
Indien in de READ-instructie een numerieke variabele staat, dan moet het gegeven uit de DATA-list dat in die variabele moet worden gezet ook numeriek zijn. Dit geldt natuurlijk ook voor string-variabelen.

REM



Hiermee kan men commentaar aan het programma toevoegen. Alle karakters, behalve ENTER, zijn toegestaan. Daar ook de dubbelepunt als commentaar wordt beschouwd, kan een REM-instructie niet op de zelfde regel worden gevolgd door een andere instructie.

RESTORE RESTORE [<nummer>]

<nummer> is een numerieke expressie.
0 <= nummer <= 9999 .
Nadat met READ een waarde uit de DATA-list is gelezen wijst een pointer naar de volgende waarde uit de DATA list. Op deze manier kan men, zonder zich er zorgen over te moeten maken of een bepaalde waarde niet per ongeluk twee maal is gelezen, alle waarden uit de DATA-list een voor een lezen. Wil men echter de waarden uit de DATA-list nog eens lezen dan moet de pointer gereset worden. Dit nu kan met RESTORE. Een DATA-list kan uit meerdere regels bestaan. Het nummer achter RESTORE is een regelnummer. De pointer zal worden gereset naar het eerste DATA-item dat op de aangegeven regel voor komt.
Indien het gespecificeerde regelnummer geen DATA-instructie bevat, dan zal de eerstvolgende DATA-regel worden gezocht.
Indien een hoger regelnummer dan 9999 wordt gegeven, dan kan het systeem zodanig in de war raken dat er aan en uit geschakeld moet worden. Wees hier dus voorzichtig mee.

RETURN RETURN

Gaat terug naar de programmaregel onmiddellijk volgend op de regel waarin de laatste GOSUB instructie werd uitgevoerd.

RUN RUN [<nummer>]

<nummer> is een numerieke expressie.
Wist het variabelengeheugen, de GOSUB-stack en het beeldschermgeheugen. Reset de plot-positie op 0,0 en voert RESTORE uit. Pas hierna wordt het programma gestart op de in <nummer>

aangegeven regel.
Indien geen <nummer> was gespecificeerd dan zal het programma op het laagste regelnummer worden gestart.

SAVE SAVE "<naam>"[,LINE <nummer>|
DATA <letter>[$]()|
CODE <start>,<lengte>|SCREEN$]

<naam> = De naam waaronder de te bewaren gegevens op cassette zullen worden weggeschreven. Deze naam mag maximaal 10 karakters lang zijn. Alle karakters, inclusief graphics, zijn toegestaan. Een instructie, bijv. RESTORE, geldt slechts voor 1 karakter.
Met LINE geeft men aan dat het programma automatisch gestart moet worden na het met LOAD terugladen van het programma in het geheugen.
<nummer> geeft het regelnummer aan waarop gestart moet worden.
Met DATA kan men een numerieke- of string-variabele of een array naar cassette wegschrijven.
<letter> is de variabelenaam.
Met CODE kan een deel van het geheugen naar cassette worden weggeschreven, beginnende vanaf het adres <start> en met een lengte als aangegeven in <lengte>.
Met SCREEN$ doet men in feite hetzelfde als met CODE, alleen hoeft men nu geen start en lengte op te geven, omdat het systeem als startadres het begin van het beeldschermgeheugen neemt en de lengte zo kiest dat ook nog het attributen geheugen wordt weggeschreven.

SCREEN$ = CODE 16384,6912.

STOP STOP

Stopt de uitvoering van het programma en geeft de systeemboodschap 9. Indien hierna op CONTINUE wordt gedrukt, wordt de uitvoering van het programma voort gezet met de eerstvolgende instructie.

VERIFY VERIFY "[<naam>]"[DATA <letter>[$]() | CODE[<start>[,<lengte>]] | SCREEN$]

Werkt net als de LOAD, doch in plaats van de gelezen data in het geheugen te zetten wordt die data vergeleken met de reeds in het geheugen staande data. Indien een ongelijkheid wordt geconstateerd, dan wordt de systeemboodschap R gegeven.

Op diverse plaatsen in de voorgaande instructie-beschrijvingen was sprake van variabelen. In die beschrijvingen werd geen syntaxbeschrijving gegeven van variabelen, omdat dit de leesbaarheid niet ten goede zou komen. Voor de volledigheid volgt hier nog een syntaxomschrijving van een variabele.

<var> = <letter>[$_{1}^{n}$ ([<letter> | <cijfer>]) | $]
[(<index> [$_{1}^{n}$ (, <index>)])]

BASIC-functies

ABS ABS <numerieke expressie>

Geeft de absolute waarde van de numerieke expressie. Stel dat de waarde -8.4 was, dan zal ABS -8.4 gelijk zijn aan 8.4. Het minteken is dus verdwenen.

ACS ACS <numerieke expressie>

Geeft de arccosinus van de numerieke expressie in radialen. De numerieke expressie moet aan de volgende eisen voldoen: -1 <= numerieke expressie <= +1.

AND [<num. expr.> | <string expr.>]
AND <num.expr.>

Indien de linker en de rechter expressie beide numeriek zijn, dan zal het resultaat de waarde van de linker expressie zijn, op voorwaarde dat de rechter expressie ongelijk aan 0 is. Is de rechter expressie 0, dan zal het resultaat ook 0 zijn. Indien de linker expressie een string is, dan zal het resultaat de string van de linker expressie zijn, op voorwaarde dat de rechter expressie ongelijk aan 0 is. Is de rechter expressie echter een 0, dan zal het resultaat een "empty string" zijn.

ASN ASN <numerieke expressie>

Geeft de arcsinus van de numerieke expressie in radialen. De numerieke expressie moet aan de volgende eisen voldoen: -1 <= numerieke expressie <= +1.

AT

AT <regelnummer>,<karakterpositie>

Zowel regelnummer als karakterpositie zijn numerieke expressies.
AT kan als functie worden gebruikt bij de INPUT en PRINT statements. In beide gevallen kan op de gespecificeerde plaats worden geschreven. De volgende regels dienen in acht te worden genomen:
0 <= karakterpositie <= 31
Bij PRINT : 0 <= regelnummer <= 22
Bij INPUT : 0 <= regelnummer <= 21

ATN

ATN <numerieke expressie>

Geeft de arctangent van de numerieke expressie in radialen.

ATTR

ATTR (<regelnummer>,<karakterpositie>)

Hiermee kunnen de attributengegevens van een beeldscherm positie worden opgevraagd. (Zie ook attributengeheugen indeling.) <regelnummer> en <karakterpositie> werken beide modulo 32. Dus als men ATTR (32,0) vraagt, is dit hetzelfde als wanneer men ATTR (0,0) had gevraagd. De volgende regels dienen in acht te worden genomen:
0 <= regelnummer <= 255
0 <= karakterpositie <= 255
Ieder attribuut heeft de volgende indeling:

bit nr.	7	6	5	4	3	2	1	0
waarde	128	64	32	16	8	4	2	1
	FLASH	BRIGHT	PAPIERKLEUR			INKTKLEUR		

BIN

BIN $\left(\begin{matrix}n\\1\end{matrix}\right)\left\{\left\lfloor 0|1|\ \right\rfloor\right\}$

Met behulp van BIN kan een getal binair worden ingegeven. De maximum waarde van dat getal mag 65535 zijn. De nullen en

enen mogen van elkaar worden gescheiden door een spatie.

CHR$

CHR$ <numerieke expressie>

De waarde van de numerieke expressie is de karaktercode. De numerieke expressie moet aan de volgende voorwaarden voldoen: 0 <= numerieke expressie <= 255. De numerieke expressie wordt op een half naar boven of beneden afgerond indien het niet een geheel getal is.
Voorbeeld: CHR$ 64.6 = CHR$ 65 = A.

CODE

CODE $\left\lfloor\right.$"<string>"|<string variabele>$\left.\right\rfloor$

Met deze functie wordt het eerste in de string voorkomende karakter geconverteerd naar de bij dat karakter behorende code.
Voorbeeld: CODE "abc" = 97 (ASCII-code van a).

COS

COS <numerieke expressie>

Berekent de cosinus van het in de numerieke expressie gegeven aantal radialen.

EXP

EXP <numerieke expressie>

Geeft de waarde van e tot de in de numerieke expressie aangegeven macht.
(e = 2.7182818)

FN

FN <letter>$\left\lceil\$\right\rceil\left(\left\lceil\right.\right.$<argument>$\left.\left.\right\rceil\right)$

<argument> = $\left\lceil\begin{matrix}n\\1\end{matrix}\right.\{$<letter>$\left\lceil\$\right\rceil,\}\left.\right\rceil$<letter>$\left\lceil\$\right\rceil$

Hiermee wordt de met DEF FN gedefinieerde functie aangeroepen. Denk er wel om dat de haakjes om het argument altijd moeten

worden gegeven, ook als er geen argument is.

IN

IN ‹numerieke expressie›

Geeft de inhoud van het met de numerieke expressie aangegeven I/O-poortnummer. Er zijn in feite 65536 I/O-poorten. De Spectrum gebruikt hiervan slechts enkele. (Zie I/O-poorten)

INKEY$

INKEY$

Geeft de ingedrukte toetscode. Indien geen toets wordt ingedrukt, dan zal het resultaat een "empty string" zijn.

INT

INT ‹numerieke expressie›

Geeft het gehele deel van een niet geheel getal.
Voorbeeld: INT 2.7 = 2 .
Om op een half naar boven of beneden te kunnen afronden moet eerst 0.5 worden bijgeteld.
Voorbeeld: INT 2.7 = 2
INT (2.7+0.5) = INT 3.2 = 3 .

LEN

LEN ⌊"‹string›"|‹string variabele›⌋

Geeft de lengte van de gegeven string.
Voorbeeld: LEN "Spectrum" = 8.

LN

LN ‹numerieke expressie›

Geeft de natuurlijke logaritme van de numerieke expressie. De numerieke expressie moet voldoen aan de voorwaarde:
0 < numerieke expressie.

NOT

NOT ⌊‹num. expr.›| ‹vergelijkende expr.›⌋

NOT kan worden gevolgd door een getal of door een vergelijkende uitdrukking. Indien NOT wordt gevolgd door een getal dan geldt:
NOT ‹numerieke expressie› = 0 als de numerieke expressie 1 of meer is.
NOT ‹numerieke expressie› = 1 als de numerieke expressie 0 is.
Indien NOT wordt gevolgd door een vergelijking dan geldt:
NOT ‹vergelijkende expressie› = 1 indien die expressie niet waar is.
NOT ‹vergelijkende expressie› = 0 indien die expressie wel waar is.

OR

‹expressie› OR ‹expressie›

‹expressie› is hier een numerieke- of vergelijkende expressie.
Indien sprake is van een numerieke expressie dan kan deze of 0 zijn of ongelijk aan 0 (dit is dan 1).
Indien sprake is van een vergelijkende expressie dan kan deze of waar of niet waar zijn (waar=1; niet waar=0).
Indien nu een van de expressies (voor of na OR) 1 is, dan zal het resultaat 1 zijn.

PEEK

PEEK ‹numerieke expressie›

Leest de inhoud van het in ‹numerieke expressie› gegeven adres. De numerieke expressie moet voldoen aan:
0 <= numerieke expressie <= 65535.

PI

PI

Vertegenwoordigt de waarde 3.141592653558.

POINT POINT (<hor.pos.>,<vert.pos.>)

<hor.pos.> en <vert.pos.> zijn beide numerieke expressies. Met POINT kan van het aangegeven "pixel" worden opgevraagd of het de papier- dan wel de inktkleur heeft. Indien het resultaat 0 is, dan had het pixel de papierkleur. Indien het resultaat 1 is, dan had het pixel de inktkleur. Aan de volgende voorwaarden moet worden voldaan:
0 <= hor.pos. <= 255; 0 <= vert.pos. <= 175

RND RND

Geeft een pseudo-random-getal. Dit getal zal varieren tussen 0 en 1 doch zal nooit 1 zijn. Met de statement RANDOMIZE kan de reeks pseudo-random getallen op een bepaalde plaats worden gestart.

SCREEN$ SCREEN$ (<regel>,<karakterpositie>)

<regel> en <karakterpositie> zijn numerieke expressies. Deze numerieke expressies moeten voldoen aan:
0 <= regel <= 23; 0 <= karakterpositie <= 31.
Er wordt pas een foutboodschap gegeven wanneer het regelnummer of de karakterpositie meer dan 255 is. Met SCREEN$ wordt het karakter op de aangegeven positie van het scherm gelezen. Ook indien dit karakter invers op het scherm staat zal het resultaat het normale karakter zijn. Indien op de aangegeven positie niet een standaard karakter staat, dan zal het resultaat een "empty string" zijn.

Zie verder de LOAD en SAVE statements voor een ander soort gebruik van SCREEN$.

SGN SGN <numerieke expressie>

Geeft tekeninformatie over de in de numerieke expressie aangegeven waarde.
Een negatieve waarde geeft als resultaat -1.
De waarde 0 geeft als resultaat 0.
Een positieve waarde geeft als resultaat +1.

SIN SIN <numerieke expressie>

Berekent de sinus van het in <numerieke expressie> aangegeven aantal radialen.

SQR SQR <numerieke expressie>

Berekent de vierkantswortel uit het in <numerieke expressie> gegeven getal.

STR$ STR$ <numerieke expressie>

Geeft de string karakters die nodig zouden zijn om de waarde van de <numerieke expressie> af te drukken.
Voorbeeld: STR$ 87 = "87"
(STR$ 987654)(2 TO 4) = "876"

TAB TAB <numerieke expressie>

Verplaatst de printpositie het in <numerieke expressie> aangegeven aantal plaatsen. De numerieke expressie wordt door 32 gedeeld en de rest van die deling geeft de verplaatsing aan.
De numerieke expressie moet voldoen aan:
0 <= numerieke expressie <= 65535.

TAN TAN <numerieke expressie>

Berekent de tangens van het aantal in <numerieke expressie> aangegeven radialen.

USR USR [<numerieke expressie>|<string expressie>]

Indien er een numerieke expressie achter USR wordt gespecificeerd dan zal USR een "call" naar het aangegeven adres uitvoeren. Er wordt dan op dat adres een machinetaalroutine verwacht. Indien de machinetaal routine normaal wordt beeindigd, dan zal het resultaat van USR de inhoud van Z80-registerpaar BC zijn. Indien er een string expressie achter USR wordt gespecificeerd, dan wordt hiermee het adres van een User-defined-graphic aangeduid. De string moet dan ook slechts 1 karakter lang zijn. De toegestane karakters zijn "a" tot en met "u" of een user defined graphic. Voorbeeld: USR "a"+0 = adres 65368 (Dit is dus de eerste bitrij van UDG "a").

VAL VAL <string expressie>

Geeft de numerieke waarde van de in de string aangegeven expressie.
Voorbeeld: VAL "3+2*5" = 13

VAL$ VAL$ <string expressie>

Geeft de string waarde van de in de string aangegeven expressie.
Voorbeeld:

```
VAL$ """abc""" = abc

INPUT a$          (abc ingegeven)
VAL$ "a$"       = abc
```


Bewerkingssymbolen met hun functie

Er zijn drie soorten bewerkingstekens (operators), te weten:

1 - Rekenkundige operators.
2 - Relatie operators.
3 - Logische operators.

1. Rekenkundige operators:

^	machtsverheffen
*	vermenigvuldigen
/	delen
+	optellen
-	aftrekken

2. Relatie operators:

=	is gelijk aan
>	is groter dan
<	is kleiner dan
<=	is gelijk aan of kleiner dan
>=	is gelijk aan of groter dan
<>	is ongelijk aan

3. Logische operators:

AND	logische EN
OR	logische OF
NOT	logische NIET

Bij het evalueren van bewerkingen in uitdrukkingen hebben die bewerkingen ieder een bepaalde prioriteit. Indien twee bewerkingen dezelfde prioriteit hebben, dan zal de meest linkse bewerking als eerste worden uitgevoerd.

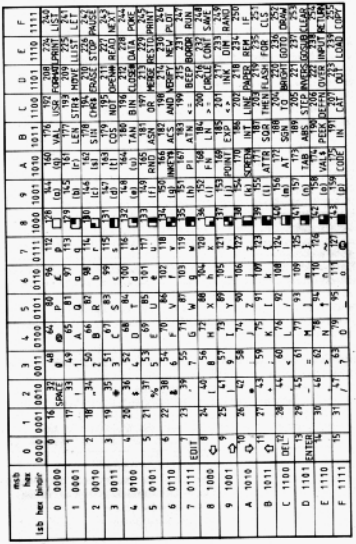
Indien men de prioriteit van bewerkingen wil beinvloeden, dan kan men dit doen door gebruik te maken van haakjes. De termen tussen haakjes worden eerst geevalueerd, geheel volgens de

prioriteitsregels, daarna worden de overblijvende resultaten verder geevalueerd, weer volgens dezelfde prioriteitsregels. In de volgende tabel zijn de prioriteiten van de verschillende operators gegeven:

bewerking (operator/functie)	prioriteit
het bepalen van een element uit een array	12
slicing	12
alle functies behalve NOT, AND, OR en - (unary minus)	11
^ (machtsverheffen)	10
- (unary minus)	9
*, / (vermenigvuldigen en delen)	8
+, - (optellen en aftrekken)	6
=, >, <, <=, >=, <> (relatie operators)	5
NOT	4
AND	3
OR	2

Zoals u ziet staan in bovenstaande tabel niet alleen operators, maar ook functies. Een beschrijving van deze functies is gegeven onder de titel "BASIC-functies" elders in dit boek.

Tabel van de ZX Spectrum-karakterset



Systeemboodschappen

Wanneer de ZX Spectrum u iets wil laten weten, dan stuurt het een boodschap naar het onderste deel van het beeldscherm. Meestal betreffen deze boodschappen foutmeldingen.
Deze foutmeldingen kunnen ontstaan tijdens het intypen van een programma of tijdens het uitvoeren daarvan. Steeds nadat men een BASIC-regel heeft ingegeven, wordt eerst de syntax (de spelling) gecontroleerd. Ontdekt de ZX Spectrum een fout, dan zal het daar melding van maken.
Een programma dat zonder syntax-fouten is opgeslagen in het geheugen, kan tijdens het uitvoeren toch nog fouten blijken te bevatten. Dit betreft dan logische fouten (denkfouten van de programmeur). Een voorbeeld hiervan zou kunnen zijn dat u een nog niet gedefinieerde variabele gebruikt.
Een enkele keer kan het voorkomen dat wat tijdens de syntax controle goed werd bevonden tijdens het uitvoeren toch fout is. Het is dus niet altijd waar dat er na het intypen en opslaan van een programma geen syntaxfouten meer in een programma kunnen staan.

Het formaat waarin de systeemboodschappen worden gegeven is als volgt:

‹code› ‹boodschap›,‹regelnummer›:‹statementnummer›

‹code› = Een van de cijfers 0 tot en met 9 of een van de letters A tot en met R.

‹boodschap› = Een bij de daarvoor gegeven code behorende verklarende boodschap.

‹regelnummer› = Het regelnummer van de programmaregel die deze systeemboodschap veroorzaakte. Indien de systeemboodschap werd veroorzaakt door een direct commando, dan zal het regelnummer 0 zijn.

‹statementnummer›= Binnen een programmaregel kunnen 1 of meer statements, van elkaar gescheiden door ‹:›, voor komen. Met het statementnummer wordt aangegeven het hoeveelste statement van het aangegeven regelnummer de systeemboodschap veroorzaakte.

In de hierna volgende tabel staan alle mogelijke systeemboodschappen op volgorde van hun code. Per boodschap staan op alfabetische volgorde van de veroorzakende statement de mogelijke oorzaken aangegeven.

code	statement	tekst/betekenis
0		OK Normale programmabeeindiging. De laatste programmaregel is goed uitgevoerd. Er is geen programmaregel meer met een hoger regelnummer. Dit kan ook gebeuren wanneer een sprong wordt uigevoerd naar een regelnummer dat hoger is dan het hoogste bestaande regelnummer.
1		NEXT without FOR
	NEXT	De control-variabele ontbreekt, doch er is wel een andere variabele met dezelfde naam.

code	statement	tekst/betekenis
2		Variable not found Dit kan voorkomen bij alle statements die variabelen gebruiken, of waarin waarden van variabelen worden toegekend aan andere variabelen die nog niet bestaan. Dit kan dus bij praktisch alle statements voorkomen.
3		Subscript wrong
	DIM	0 ‹ array-index ‹ 65491. Indien de array-index niet aan deze waarden voldoet wordt boodschap 3 gegeven.
	allerlei	Indien een gedimensioneerde variabele in een andere statement wordt gebruikt en de dimensies zijn daarin 0 of groter dan in de DIM-statement was aangegeven.
	slicing	‹begin› of ‹eind› van de slice is kleiner of groter dan de string waarvan een slice wordt afgenomen.
4		Out of memory
	DIM	De te creeren array past niet in het geheugen.
	FOR	Er is onvoldoende geheugenruimte vrij om de voor de FOR...NEXT-lus benodigde stack in op te bouwen.
	GOSUB	Er is onvoldoende geheugenruimte vrij om de voor het aanroepen van een subroutine benodigde stack in op te bouwen.

code	statement	tekst/betekenis
	INPUT	Er is onvoldoende geheugenruimte vrij om de "workspace" (het INPUT-buffer) verder te vergroten.
	LET	Er is onvoldoende geheugenruimte vrij om het variabelengeheugen met nog een variabele uit te breiden.
	LOAD	Het te laden programma, de variabele of de bytes passen niet in de beschikbare vrije geheugenruimte.
	MERGE	De nieuw bij te voegen programmaregels of variabelen passen niet in de beschikbare vrije geheugenruimte.
	expressie-evaluatie	Voor het uitwerken van de expressie moeten tussenresultaten op de stack worden geplaatst, doch de vrije geheugenruimte is niet groot genoeg. (Probeer wat haakjes weg te werken.)
5		Out of screen
	INPUT	Er worden meer dan 23 regels gebruikt voor het onderste deel van het scherm.
	INPUT AT	De startprintpositie plus de tekstlengte valt buiten het bereik van het scherm. (bijv. INPUT AT 22,31;"ab")
	PRINT AT	Het gespecificeerde regelnummer is 22. Dit mag maximaal 21 zijn. Indien regelnummer 23 of hoger wordt gespecificeerd dan wordt boodschap 8 gegeven.

code	statement	tekst/betekenis
6		Number too big
	allerlei	Het resultaat van een berekening is te groot. Dit kan zowel een positief als negatief resultaat betreffen. (-2^126 <= resultaat <= 2^127)
7		RETURN without GOSUB
	RETURN	Er wordt een RETURN-statement uitgevoerd, doch de GOSUB-stack blijkt leeg te zijn. Er waren dus blijkbaar meer RETURNs dan GOSUBs.
8		Voor toekomstige uitbreiding.
9		STOP statement
	STOP	De laatst uitgevoerde instructie was een STOP-statement. Met CONTINUE kan de direct op de STOP statement volgende instructie worden uitgevoerd.
A		Invalid argument
	ACS	De gebruikte numerieke expressie mag alleen een waarde vam -1 tot +1 hebben.
	ASN	De gebruikte numerieke expressie mag alleen een waarde van -1 tot +1 hebben.
	LN	De gebruikte numerieke expressie mag niet 0 of negatief zijn.
	SQR	De gebruikte numerieke expressie mag niet 0 zijn.

code	statement	tekst/betekenis
	USR	De string-expressie is langer dan 1 letter, of die letter is niet een van de karakters a tot en met u, of het is geen user defined graphic.
B		Integer out of range
	BEEP	De tijdsduur moet van 0 tot 10 seconden zijn en de toonhoogte moet tussen -60 en +69 liggen.
	BRIGHT	De mode mag niet negatief of groter dan 255 zijn.
	CHR$	De numerieke expressie mag alleen varieren van 0 tot en met 255.
	CIRCLE	De straal mag niet groter zijn dan de x of y coordinaat. De straal mag niet groter zijn dan 255 - x of 175 - y.
	CLEAR	De RAMTOP mag niet groter dan 65535 zijn.
	DIM	De array-index mag niet groter dan 65535 zijn.
	DRAW	De numerieke expressie mag alleen varieren tussen de waarden -255 en +255.
	GOSUB	De numerieke expressie mag niet negatief of groter dan 61439 zijn.
	GOTO	De numerieke expressie mag niet negatief of groter dan 61439 zijn.
	INPUT AT	Het regelnummer mag niet groter dan 22 zijn.

code	statement	tekst/betekenis
	INPUT AT	Het regelnummer mag niet groter dan 22 zijn.
	INK	Het kleurnummer mag niet groter dan 255 zijn.
	LIST	Het regelnummer mag niet groter zijn dan 65535.
	LLIST	Het regelnummer mag niet groter zijn dan 65535.
	PAUSE	De numerieke expressie mag niet groter zijn dan 65535.
	PEEK	Het adres mag niet groter zijn dan 65535.
	PLOT	De horizontale positie mag niet groter zijn dan 255 of de verticale positie mag niet groter zijn dan 175.
	POINT	De horizontale positie mag niet groter zijn dan 255 of de verticale positie mag niet groter zijn dan 175.
	POKE	Het adres mag niet negatief of groter dan 65535 zijn. De waarde mag niet lager dan -255 of hoger dan +255 zijn.
	RANDOMIZE	De numerieke expressie mag niet negatief of groter dan 65535 zijn.
	RUN	Het regelnummer mag miet negatief of groter dan 61439 zijn.
	slicing	De begin- of eindindicatie van de slice mag niet negatief of groter dan 65535 zijn.

code	statement	tekst/betekenis
	USR	Het adres mag niet hoger dan 65535 zijn.
C		Nonsense in BASIC
	MERGE	MERGE <naam> CODE mag niet. MERGE <naam> DATA mag niet. MERGE <naam> SCREEN$ mag niet.
	SAVE	SAVE <naam> CODE is alleen toegestaan indien ook het startadres en de lengte van de te bewaren bytes wordt gespecificeerd.
	overige	Er mogen niet meer dan 127 statements in 1 programmaregel staan. Er staat geen BASIC-statement op een plaats waar dat wel werd verwacht. Er zit een syntaxfout in de aangegeven regel.
D		BREAK - CONTINUE repeats
	COPY/LLIST LPRINT VERIFY MERGE/SAVE	Tijdens het uitvoeren van een van deze statements werd op de BREAK-toets gedrukt. (Al deze statements sturen een randapparaat aan.) De uitvoering van de statement werd gestopt. Met CONTINUE kan dezelfde statement opnieuw worden gestart.
	scroll?	Op de vraag "scroll?" werd n, STOP, SPACE of BREAK ingedrukt.
E		Out of data
	READ	Het laatste data-element uit de laatste DATA-statement was al gelezen.

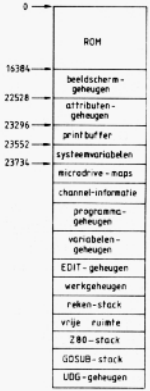
code	statement	tekst/betekenis
F		Invalid file name
	OPEN#	De gespecificeerde file-naam is een "empty string" of is niet een van de letters K, S of P. (wellicht mogen op de Spectrum met de expansion interface meerdere letters worden gebruikt.)
	SAVE	De file-naam is langer dan 10 karakters.
G		No room for line
	ENTER	Tijdens het invoeren van een programma via het toetsenbord blijkt er niet voldoende geheugenruimte te zijn om de ingegeven programmaregel in op te slaan.
H		STOP in INPUT
	INPUT	In plaats van de verwachte numerieke ingave werd een STOP-statement gegeven, of in plaats van de verwachte string werd het linker aanhalingsteken gewist en vervangen door een STOP-statement.
I		FOR without NEXT
	FOR	Er is geen NEXT-statement gevonden die de zelfde control-variabele heeft als in de FOR-statement werd gebruikt.

code	statement	tekst/betekenis
J		Invalid I/O device
		Dit bericht zal pas gegeven worden wanneer de "expansion unit" met de microdrives wordt gebruikt.
K		Invalid colour
	BORDER	Alleen de kleuren 0 tot en met 7 zijn toegestaan.
	BRIGHT	De gespecificeerde waarde moet 0, 1 of 8 zijn.
	FLASH	De gespecificeerde waarde moet 0, 1 of 8 zijn.
	INK	Alleen de kleuren 0 tot en met 9 zijn toegestaan.
	INVERSE	De gespecificeerde waarde moet 0 of 1 zijn.
	OVER	De gespecificeerde waarde moet 0 of 1 zijn.
	PAPER	Alleen de kleuren 0 tot en met 9 zijn toegestaan.
L		BREAK into program
	allerlei	Steeds nadat een BASIC-statement is uitgevoerd controleert het systeem of de BREAK toets was ingedrukt. Indien dit het geval is, dan wordt de verdere uitvoering van het programma gestopt. Met CONTINUE kan het programma worden doorgestart met de volgende statement.

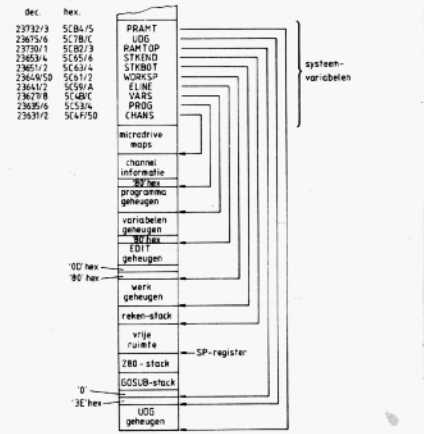
code	statement	tekst/betekenis
M		RAMTOP no good
	CLEAR	RAMTOP is te hoog of te laag gedefinieerd. Te laag: Het is mogelijk RAMTOP zo te kiezen dat dit bericht niet komt, terwijl het toch niet mogelijk is nog statements of commando's in te geven. (Probeer maar eens CLEAR 23860) Te hoog: Het is niet toegestaan RAMTOP hoger te kiezen dan het in P-RAMT aangegeven adres. (Het physieke RAM-geheugen.)
N		Statement lost
	CONTINUE	Tijdens het uitvoeren van een direct commando werd een foutboodschap gegeven voor het derde of daaropvolgende statement in die commandoregel.
	GOSUB	Het regelnummer waar naartoe moet worden gesprongen is niet te vinden. (hoger dan 32767 doch lager dan 61440)
	GOTO	Het regelnummer waar naartoe moet worden gesprongen is niet te vinden. (hoger dan 32767 doch lager dan 61440)
	NEXT	De FOR-statement bestaat niet meer of het regelnummer van de bijbehorende FOR-statement is gewijzigd.
	RETURN	De oorspronkelijke GOSUB-statement bestaat niet meer of het regelnummer is gewijzigd.

code	statement	tekst/betekenis
O		Invalid stream
	OPEN#	Het stream-nummer mag niet groter dan 15 zijn.
	CLOSE#	Het stream-nummer mag niet groter dan 15 zijn.
	CAT/FORMAT ERASE/MOVE	Deze statements mogen op de standaard ZX Spectrum niet worden gebruikt.
P		FN without DEF
	FN	De bijbehorende DEF FN-statement kan niet worden gevonden.
Q		Parameter error
	FN	Het argument is numeriek terwijl een string werd verwacht of omgekeerd. Het argument in FN komt niet overeen met het argument in DEF FN.
R		Tape loading error
	LOAD	Er blijkt meer data te zijn dan er in de parameter ‹lengte› werd gespecificeerd, of er werd een leesfout geconstateerd.
	MERGE	Er werd een leesfout geconstateerd.
	VERIFY	De in het geheugen aanwezige data is niet gelijk aan de van cassette gelezen data. Er staan meer bytes op de cassette dan er in de statement werden gespecificeerd. Er werd een leesfout geconstateerd.

Geheugenindeling



Alleen de geheugengebieden ROM, beeldschermgeheugen, attributengeheugen, printbuffer en systeemvariabelen hebben een vaste lengte.
Hierdoor zijn de startadressen van deze gebieden en van het direct daaropvolgende gebied, de microdrive-maps, altijd hetzelfde.
Deze startadressen zijn in de bijgaande tekening aan de linkerkant gegeven.
De startadressen van alle andere geheugengebieden zijn variabel. Deze adressen zijn daarom opgenomen in systeemvariabelen.
Deze systeemvariabelen worden door het systeem zelf geladen, met waarden die afhankelijk zijn van de manier waarop het systeem wordt gebruikt.
De tekening op de volgende pagina geeft een meer gedetailleerd overzicht van het variabele deel van het geheugen.



Beeldschermgeheugen

Op het beeldscherm kunnen 24 regels van ieder 32 karakters worden afgedrukt. Ieder karakter wordt in een 8*8 matrix afgedrukt. Dit houdt in dat er in totaal 8*8*24*32 = 49152 beeldpunten (pixels) zijn.
Voor ieder beeldpunt is 1 bit in het beeldschermgeheugen nodig. De grootte van dit geheugen dient dan ook 49152 bits = 6144 bytes te zijn. De organisatie van het beeldschermgeheugen is als volgt:

Het totale geheugen van 6 K is opgedeeld in drie blokken van ieder 2 K. Ieder blok van 2 K bevat 8 regels van ieder 32 karakters. Iedere regel bestaat uit 8 beeldlijnen. 8 regels bestaan dus uit 8*8 = 64 beeldlijnen.
Deze beeldlijnen zijn niet netjes opeenvolgend in het beeldschermgeheugen opgeslagen. De eerste lijn van de eerste regel uit het eerste blok wordt gevolgd door de eerste lijn van de tweede regel uit het eerste blok. Daarna volgt de eerste lijn van de derde regel, enzovoorts.
Nadat de eerste lijnen van alle acht regels uit het eerste blok zijn opgeslagen, volgende de tweede lijnen van alle acht regels uit het eerste blok, enzovoorts.
Pas wanneer alle lijnen van alle acht regels op de hiervoorbeschreven manier zijn opgeslagen, volgen de lijnen voor de regels van het tweede blok van acht regels.
Als laatste volgen dan nog de lijnen voor de laatste acht regels.

De beeldpunten kunnen alleen aan of uit worden gezet (INK of PAPER). De kleur van INK en PAPER wordt er pas in het attributen geheugen aan toegekend.

Het is mogelijk om met behulp van PEEK en POKE het beeldscherm uit te lezen of er rechtstreeks iets in te schrijven. De vreemde organisatie van het beeldschermgeheugen maakt het echter niet zo gemakkelijk om de juiste plaats te vinden.
De volgende formule kan ons daar echter bij behulpzaam zijn:

```
adr = 16384+blk*2048+reg*32+kar+lyn*256
```

Hierin is:

adr - Het PEEK- of POKE-adres
blk - Het gewenste bloknummer (0 t/m 2)
reg - Het regelnummer binnen dat blok (0 t/m 7)
kar - De karakterpositie binnen die regel (0 t/m 31)
lyn - Het beeldlijnnummer binnen het karakter (0 t/m 7)

In het hiernavolgende voorbeeldprogramma is van deze formule gebruik gemaakt om een bepaalde plaats op het beeldscherm te beschrijven met een bepaald patroon.

```
1000 INPUT "Regelnummer? (0 t/m 23)"; reg
1010 LET blk=INT (reg/8)
1020 LET reg=reg-8*blk
1030 INPUT "Karakternummer? (0 t/m 31)"; kar
1040 FOR l=0 TO 7
1050 POKE 16384+blk*2048+reg*32+kar+l*256,85
1060 NEXT l
1070 PAUSE 0
1080 PRINT "Regel ";reg;" van blok ";blk;", karak
ter ";kar
1090 FOR l=0 TO 7
1100 PRINT "adres ";16384+blk*2048+reg*32+kar+l*2
56
1110 NEXT l
1120 PAUSE 0
1130 CLS
1140 GO TO 1000
```

Wees met dit programma erg voorzichtig. Geef nooit een te hoog regelnummer in, want dat zal tot gevolg hebben dat er geheugenplaatsen boven het

beeldschermgeheugen zullen worden overschreven. Zolang dit alleen nog maar het attributengeheugen is, zal dat geen grote problemen opleveren. Overschrijft u echter de systeemvariabelen, dan is een systeem hang up niet denkbeeldig.
Mocht u voorgaande manier van beschrijven van het beeldscherm werkelijk willen toepassen, bouwt u dan voor de zekerheid een controle op de waarden van blk, reg en kar in.
In het voorbeeldprogramma zou u bijvoorbeeld de volgende toevoegingen kunnen maken:

```
1005 IF reg>23 THEN GO TO 1000

1035 IF kar>31 THEN GO TO 1030
```

De programmaregels met PAUSE 0 laten het systeem net zo lang wachten tot u een toets hebt ingedrukt. U hebt daardoor rustig de tijd om het resultaat van het programma te bekijken voordat u het programma weer verder laat draaien.

Attributengeheugen

Het beeldscherm kan 24 regels van ieder 32 karakters bevatten. In het attributengeheugen wordt voor ieder van die karakters een byte gebruikt om de kleurgegevens voor het karakter uit het beeldschermgeheugen aan te geven.
De organisatie van het attributengeheugen is recht toe recht aan. De attributenbytes staan in de zelfde volgorde als waarin wij de karakters van het beeldscherm lezen, van links naar rechts en van boven naar beneden.
Het eerste attributenbyte slaat dus op het eerste karakter uit de eerste regel van het beeldscherm.

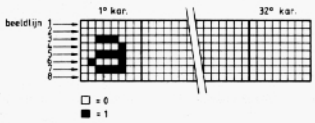
Iedere byte in het attributengeheugen heeft de volgende vorm:

waarde	128	64	32	16	8	4	2	1
bitnummer	7	6	5	4	3	2	1	0
betekenis	FLASH	BRIGHT	PAPER			INK		

Tenslotte dient nog te worden opgemerkt dat de attributen op een hele karakterpositie slaan. Dit wil dus zeggen dat er per 64 pixels (van dat karakter) nooit meer dan twee verschillende kleuren mogelijk zijn. Dit geeft enige beperking in de grafische mogelijkheden, doch u hoeft maar te kijken naar een aantal programma's voor de Spectrum om te zien dat er desondanks toch hele mooie plaatjes mee te maken zijn.

Printer-buffer

Het printbuffer is 256 bytes groot en start op adres 23296. Deze grootte komt overeen met 8*32 bytes. De organisatie van dit geheugen is dan ook zo dat de 8 beeldpuntlijnen van de af te drukken regel van 32 karakters er in opgeslagen kunnen worden. De eerste 32 bytes van het printbuffer bevatten de eerste beeldpuntlijn, de volgende 32 bytes de tweede beeldpuntlijn, en zo verder tot alle acht beeldpuntlijnen zijn opgeslagen.



Systeemvariabelen

Op volgorde van adres

dec.adres	hex.adres	naam
23552/9	5C00/7	KSTATE
23560	5C08	LASTK
23561	5C09	REPDEL
23562	5C0A	REPPER
23563/4	5C0B/C	DEFADD
23565	5C0D	KDATA
23566/7	5C0E/F	TVDATA
23568/605	5C10/35	STRMS
23606/7	5C36/7	CHARS
23608	5C38	RASP
23609	5C39	PIP
23610	5C3A	ERRNR
23611	5C3B	FLAGS
23612	5C3C	TVFLAG
23613/4	5C3D/E	ERRSP
23615/6	5C3F/40	LISTSP
23617	5C41	MODE
23618/9	5C42/3	NEWPPC
23620	5C44	NSPPC
23621/2	5C45/6	PPC
23623	5C47	SUBPPC
23624	5C48	BORDCR
23625/6	5C49/A	EPPC
23627/8	5C4B/C	VARS
23629/30	5C4D/E	DEST
23631/2	5C4F/50	CHANS
23633/4	5C51/2	CURCHL
23635/6	5C53/4	PROG
23637/8	5C55/6	NXTLIN
23639/40	5C57/8	DATADD
23641/2	5C59/A	ELINE
23643/4	5C5B/C	KCUR
23645/6	5C5D/E	CHADD
23647/8	5C5F/60	XPTR

dec.adres	hex.adres	naam
23649/50	5C61/2	WORKSP
23651/2	5C63/4	STKBOT
23653/4	5C65/6	STKEND
23655	5C67	BREG
23656/7	5C68/9	MEM
23658	5C6A	FLAGS2
23659	5C6B	DFSZ
23660/1	5C6C/D	S-TOP
23662/3	5C6E/F	OLDPPC
23664	5C70	OSPPC
23665	5C71	FLAGX
23666/7	5C72/3	STRLEN
23668/9	5C74/5	TADDR
23670/1	5C76/7	SEED
23672/4	5C78/A	FRAMES
23675/6	5C7B/C	UDG
23677	5C7D	COORDX
23678	5C7E	COORDY
23679	5C7F	PPOSN
23680	5C80	PRCC
23682/3	5C82/3	ECHOE
23684/5	5C84/5	DFCC
23686/7	5C86/7	DFCCL
23688	5C88	SPOSUC
23689	5C89	SPOSUL
23690	5C8A	SPOSLC
23691	5C8B	SPOSLL
23692	5C8C	SCRCT
23693	5C8D	ATTRP
23694	5C8E	MASKP
23695	5C8F	ATTRT
23696	5C90	MASKT
23697	5C91	PFLAG
23698/727	5C92/AF	MEMBOT
23728/9	5CB0/1	NONAME
23730/1	5CB2/3	RAMTOP
23732/3	5CB4/5	PRAMT

Op alfabetische volgorde van naam

naam	dec.adres	hex.adres
ATTRP	23693	5C8D
ATTRT	23695	5C8F
BORDCR	23624	5C48
BREG	23655	5C67
CHADD	23645/6	5C5D/E
CHANS	23631/2	5C4F/50
CHARS	23606/7	5C36/7
COORDX	23677	5C7D
COORDY	23678	5C7E
CURCHL	23633/4	5C51/2
DATADD	23639/40	5C57/8
DEFADD	23563/4	5C0B/C
DEST	23629/30	5C4D/E
DFCC	23684/5	5C84/5
DFCCL	23686/7	5C86/7
DFSZ	23659	5C6B
ECHOE	23682/3	5C82/3
ELINE	23641/2	5C59/A
EPPC	23625/6	5C49/A
ERRNR	23610	5C3A
ERRSP	23613/4	5C3D/E
FLAGS	23611	5C3B
FLAGS2	23658	5C6A
FLAGX	23665	5C71
FRAMES	23672/4	5C78/A
KCUR	23643/4	5C5B/C
KDATA	23565	5C0D
KSTATE	23552/9	5C00/7
LASTK	23560	5C08
LISTSP	23615/6	5C3F/40
MASKP	23694	5C8E
MASKT	23696	5C90
MEM	23656/7	5C68/9
MEMBOT	23698/727	5C92/AF
MODE	23617	5C41
NEWPPC	23618/9	5C42/3
NONAME	23728/9	5CB0/1
NSPPC	23620	5C44
NXTLIN	23637/8	5C55/6
OLDPPC	23662/3	5C6E/F
OSPPC	23664	5C70
PFLAG	23697	5C91
PIP	23609	5C39

naam	dec.adres	hex.adres
PPC	23621/2	5C45/6
PPOSN	23679	5C7F
PRAMT	23732/3	5CB4/5
PRCC	23680	5C80
PROG	23635/6	5C53/4
RAMTOP	23730/1	5CB2/3
RASP	23608	5C38
REPDEL	23561	5C09
REPPER	23562	5C0A
SCRCT	23692	5C8C
SEED	23670/1	5C76/7
SPOSLC	23690	5C8A
SPOSLL	23691	5C8B
SPOSUC	23688	5C88
SPOSUL	23689	5C89
STKBOT	23651/2	5C63/4
STKEND	23653/4	5C65/6
S-TOP	23660/1	5C6C/D
STRLEN	23666/7	5C72/3
STRMS	23568/605	5C10/35
SUBPPC	23623	5C47
TADDR	23668/9	5C74/5
TVDATA	23566/7	5C0E/F
TVFLAG	23612	5C3C
UDG	23675/6	5C7B/C
VARS	23627/8	5C4B/C
WORKSP	23649/50	5C61/2
XPTR	23647/8	5C5F/60

Omschrijving van de systeemvariabelen

ATTRP De permanente attributen worden tijdens de aanschakelroutine als volgt gezet:

Inkt = 0 (zwart),
papier = 7 (wit),
helderheid = normaal (BRIGHT 0),
FLASH = niet knipperend (FLASH 0).

Deze waarden kunnen met de instructies INK, PAPER, FLASH en BRIGHT worden overschreven, mits deze instructies op zichzelfstaand worden gebruikt en niet als functie van een PRINT- of INPUT-instructie.

De indeling van de systeemvariabele ATTRP is als volgt:

waarde	128	64	32	16	8	4	2	1
bitnummer	7	6	5	4	3	2	1	0
betekenis	FLASH	BRIGHT	PAPER			INK		

ATTRT De tijdelijke attributen worden tijdens de aanschakelroutine gelijk gemaakt aan de permanente attributen. De systeemvariabele ATTRT heeft dan ook dezelfde indeling als ATTRP en kan ook overschreven worden met de instructies INK, PAPER, FLASH en BRIGHT, mits deze instructies als functie in een output-instructie zijn opgenomen.

BORDCR Deze variabele bevat informatie voor de Border-kleur en de attributen voor het onderste deel van het scherm. De papierkleur van het onderste deel van het scherm is gelijk aan de Border-kleur, de inktkleur contrasteert hiermee. FLASH en BRIGHT worden normaal op 0 gezet.

De indeling is als volgt:

bits 0, 1 en 2 - inktkleur van het onderste deel van het scherm.
bits 3, 4 en 5 - border-kleur (tevens papierkleur van het onderste deel van het scherm).
bit 6 - BRIGHT
bit 7 - FLASH

BREG Het B-register van de rekeneenheid.

CHADD Het karakteradres van het eerstvolgende karakter dat moet worden geïnterpreteerd.

CHANS Het startadres van het geheugengebied dat de "channel-informatie" bevat.

CHARS Het ‹startadres - 256› van de karakterset. In deze karakterset zijn per karakter de bitpatronen voor de acht beeldlijnen van 8 punten opgenomen. De karakterset bevat de bitpatronen voor alle karakters van code 32 tot en met 127 (decimaal), ofwel 20 tot en met 7F (hexadecimaal).
Deze systeemvariabele wijst naar een adres in het ROM, doch u kunt een eigen karakterset definieren in RAM en met behulp van deze variabele daar naartoe verwijzen.

Ter verduidelijking volgt hier nog een voorbeeld van hoe een karakter in de karakterset is opgeslagen. Het karakter met de decimale code 33 is het uitroepteken dat met behulp van acht bytes in de karakterset is weer- gegeven. Hieronder volgen deze acht bytes met hun ROM- adres:

adres	inhoud	bit:76543210
15624	0	byte 1 00000000
15625	32	byte 2 00100000
15626	32	byte 3 00100000
15627	32	byte 4 00100000
15628	32	byte 5 00100000
15629	0	byte 6 00000000
15630	32	byte 7 00100000
15631	0	byte 8 00000000

COORDX De X-coordinaat van het laatst geplotte beeldpunt. Onmiddellijk na aanschakelen is deze 0.

COORDY De Y-coordinaat van het laatst geplotte beeldpunt. Onmiddellijk na aanschakelen is deze 0.

CURCHL Het I/O-routine-adres voor het in gebruik zijnde channel.

DATADD Dit adres wijst naar het geheugenadres waar het laatst gelezen DATA-item staat. Door RUN wordt deze pointer naar het begin van het geheugen gezet. Met RESTORE kan deze pointer ook worden teruggezet.

DEFADD Gedurende de tijd dat er een user defined function wordt geevalueerd, staat in deze variabele het adres van de argumenten van die functie.

DEST Indien een variabele nog niet bestaat, dan wijst DEST naar de eerste letter van de variabelenaam, terwijl bit 1 van FLAGX is gezet. Indien de variabele al bestond, dan wordt bit 1 van FLAGX teruggezet en wijst DEST ingeval van een numerieke variabele naar het punt vlak voor de 5-bytes waarde en ingeval van een string variabele naar de eerste byte van de oude string.

DFCC Hiermee wordt de positie van de eerste beeldpuntlijn waar het volgende karakter moet worden geprint, binnen het bovenste deel van het beeldscherm, weergegeven. (Zie ook de indeling van het beeldschermgeheugen.)

DFCCL Deze variabele heeft de zelfde functie als DFCC, doch is alleen voor het onderste deel van het beeldscherm.

DFSZ Hierin staat het aantal regels waaruit het onderste deel van het beeldscherm bestaat aangegeven. In de aanschakelroutine wordt hier de waarde 2 in gezet.

ECHOE Hierin staat de grootte van het inputbuffer in het onderste deel van het scherm aangegeven. De eerste byte bevat de karakterpositie, de tweede het regelnummer.

ELINE Het startadres van het edit-geheugen. In dit geheugen wordt de BASIC-regel die wordt ingegeven of die met EDIT wordt gewijzigd tijdelijk opgeslagen. Zodra de regel met behulp van ENTER in het programma wordt gezet wordt het edit-geheugen weer gewist.

EPPC Deze variabele bevat het regelnummer van de BASIC-regel uit een listing waarin de cursor voorkomt (de current line).

ERRNR Hier wordt de code voor de selectie van een systeemboodschap - 1 gezet. In de aanschakelroutine wordt hier de waarde 255 (hex FF) ingezet, hetgeen overeenkomt met de code 0 (= OK).

ERRSP Voordat een routine wordt uitgevoerd kan een returnadres op de machine-stack worden gezet. Indien er tijdens het uitvoeren van die routine iets fout gaat, dan kan met behulp van het op de stack bewaarde adres opnieuw worden begonnen. ERRSP bevat het adres binnen de stack waar het returnadres is opgeslagen.

FLAGS De vlagbits in deze byte hebben de volgende betekenis:

bit 0 - Leading space wel of niet toegestaan.
bit 1 - Tekst moet naar printer of beeldscherm.
bit 2 - Het volgende karakter dient in K- of L-mode te worden geschreven.
bit 3 - Huidige mode is K of L.
bit 4 -
bit 5 - Er is een toets ingedrukt maar nog niet verwerkt of het systeem is klaar voor een volgende toets.
bit 6 - Een numeriek- of string-resultaat.
bit 7 - Syntaxchecking of programma-uitvoering.

FLAGS2 De vlagbits in deze byte hebben de volgende betekenis:

bit 0 - Scherm is wel of niet leeg.
bit 1 - Printerbuffer is wel of niet leeg.
bit 2 - Data staat wel of niet tussen aanhalingstekens.
bit 3 - Wel of niet in de C-mode.
bit 4 - Channel K of een ander channel in gebruik.
bit 5 -
bit 6 -
bit 7 -

FLAGX De vlagbits in deze byte hebben de volgende betekenis:

bit 0 - Oude string moet wel of niet worden gewist.
bit 1 - Wel of niet een nieuwe variabele.
bit 2 -
bit 3 -
bit 4 -
bit 5 - EDIT- of INPUT-mode.
bit 6 - Numerieke of string variabele.
bit 7 - Wel of niet INPUT LINE.

FRAMES Dit is de drie bytes TV-frames teller. De byte op adres 23672 is de minst significante. Indien RANDOMIZE zonder argument of met argument=0 wordt gegeven, dan zullen de twee minst significante bytes van FRAMES naar de systeemvariabele SEED worden gecopieerd.

KCUR Hierin wordt het adres van de cursor binnen de EDIT-area opgeslagen.

KDATA In key data wordt het kleurnummer opgeslagen (0 tot en met 7).

KSTATE De key status bestaat uit twee sets van ieder vier bytes, KSTATE0-3 en KSTATE4-7. Bij het lezen van het toetsenbord wordt een van deze twee sets gebruikt. Bij de repeat-functie kan een volgend karakter al zijn ingedrukt terwijl het laatste repeat-karakter nog niet is verwerkt. Doordat er twee sets KSTATE zijn behoeft dit karakter niet verloren te gaan.

LASTK De code van de laatst ingedrukte toets wordt hierin opgeslagen. Zolang bit 5 van FLAGS gelijk is aan 1 is dit karakter nog niet verwerkt en kan er nog geen volgende toetsaanslag worden geaccepteerd.

LISTSP Het adres in deze systeemvariabele verwijst naar de plaats in de stack waar het return-adres na een automatische listing kan worden gevonden.

MASKP Hiermee kunnen, wanneer bij het schrijven de permanente waarden worden gebruikt, de attributen gelijk worden gehouden aan wat ze al waren in het attributengeheugen, ongeacht de waarde van ATTRP. De bits in MASKP hebben de volgende betekenis:

bits 0 t/m 2 - INK 8
bits 3 t/m 5 - PAPER 8
bit 6 - BRIGHT 8
bit 7 - FLASH 8

MASKT Bij het schrijven met tijdelijke kleurwaarden kan met behulp van dit masker de waarde uit het attributengeheugen worden behouden, in plaats van te worden overschreven met de tijdelijke waarde. De bits in MASKT hebben de volgende betekenis:

bits 0 t/m 2 - INK 8
bits 3 t/m 5 - PAPER 8
bit 6 - BRIGHT 8
bit 7 - FLASH 8

MEM Dit adres wijst naar het geheugengebied dat door de rekenroutines van het BASIC-systeem wordt gebruikt. Meestal zal dit het gebied MEMBOT zijn.

MEMBOT Dit is het geheugengebied dat door de rekenroutines van het BASIC-systeem wordt gebruikt. Het rekenen gebeurt normaal met 5-bytes getallen, die in de reken-stack worden opgeslagen (het gebied tussen STKBOT en STKEND). De resultaten die moeten worden afgedrukt kunnen echter niet

in de stack worden opgeslagen. Daarvoor wordt dan MEMBOT gebruikt. Ook waarden uit STR$ worden eerst in MEMBOT opgeslagen, bewerkt en pas daarna in de reken-stack gezet.

MODE Hierin worden de modes aangegeven. De volgende modes zijn mogelijk:
K - Keywords
L - Letters
C - Hoofdletters
E - Extended mode
G - Grafische mode

NEWPPC Hierin wordt het regelnummer waar met een GO TO naar toe moet worden gesprongen opgeslagen.

NONAME Alleen wanneer deze variabele de waarde 0 heeft zal bij het activeren van de NMI-lijn een system-reset worden uitgevoerd (er wordt dan naar adres 0 gesprongen). Is de variabele ongelijk aan 0 dan gebeurt er niets. In de standaard ZX Spectrum wordt deze variabele niet gebruikt.

NSPPC Hierin wordt het instructienummer waar met een GO TO naar toe moet worden gesprongen, uit de in NEWPPC aangegeven regel, opgeslagen.

NXTLIN Deze systeemvariabele bevat het adres van de volgende regelnummer in het programmageheugen.

OLDPPC Hierin wordt het regelnummer waar met CONTINUE mee moet worden doorgegaan opgeslagen.

OSPPC Hierin wordt het instructienummer uit de in OLDPPC aangegeven regel, waar met CONTINUE moet worden verder gegaan, opgeslagen.

PFLAG De vlagbits in deze byte hebben de volgende betekenis:

bit 0 - OVER (tijdelijk)
bit 1 - OVER (permanent)
bit 2 - INVERSE (tijdelijk)
bit 3 - INVERSE (permanent)
bit 4 - INK 9 (tijdelijk)
bit 5 - INK 9 (permanent)
bit 6 - PAPER 9 (tijdelijk)
bit 7 - PAPER 9 (permanent)

PIP Deze variabele geeft de lengte van de keyboard-klik. Door hier na het aanschakelen een waarde 15 in te zetten met behulp van POKE ontstaat een duidelijk hoorbare klik bij het intypen van uw programma. PIP wordt gebruikt door de EDIT-routines, doch niet door INPUT-instructies.

PPC Hierin wordt het regelnummer opgeslagen van de regel waarin de instructie staat die momenteel wordt uitgevoerd.

PPOSN De printpositie voor de printer (33 posities mogelijk, 32 karakters en CR).

PRAMT Het adres van het laatste byte van het physieke RAM. Dit adres wordt tijdens het aanschakelen gevonden en ingevuld. Indien geen geheugenfouten werden gevonden dan zal hier voor de 16k Spectrum 32767 en voor de 48k Spectrum 65535 in staan. Indien er wel een geheugenfout werd gevonden dan geeft deze variabele het laatste goede adres voor het adres dat de fout gaf aan.

PROC Het adres binnen het printbuffer waar het volgende karakter naartoe moet worden geschreven.

PROG Het startadres van het BASIC-programmageheugen.

RAMTOP Het adres van het laatste voor het BASIC-systeem toegankelijke byte. Veranderen van RAMTOP met een POKE-instructie werkt niet goed, omdat ook de GOSOB-stack en de machine-stack opnieuw moeten worden geinitialiseerd. Verander daarom RAMTOP altijd met een CLEAR-instructie.

RASP Hierin staat de tijdsduur van de waarschuwingszoemer aangegeven.

REPDEL De tijd die een toets moet zijn ingedrukt voordat de repeat-functie wordt geactiveerd. Iedere eenheid staat voor 1/50 seconde. Tijdens het uitvoeren van de aanschakelroutine wordt hier 35 (= 0,7 sec.) in gezet.

REPPER De tijd die tussen het genereren van twee toetsaanslagen in de repeat-mode zit. Iedere eenheid staat voor 1/50 seconde. Tijdens het uitvoeren van de aanschakelroutine wordt hier 5 (= 0,1 sec.) in gezet.

SCRCT Het aantal uit te voeren scrolls+1 dat moet worden uitgevoerd voordat met de boodschap "scroll?" wordt gestopt.

SEED Het getal in deze variabele zal door RND worden gebruikt om het begin van de reeks pseudo-willekeurige getallen te vinden. SEED kan worden geladen met behulp van de functie RANDOMIZE.

SPOSLC De karakterpositie binnen de in SPOSLL aangegeven regel. Deze variabele wordt voor het onderste deel van het beeldscherm gebruikt.

SPOSLL Het regelnummer voor het onderste deel van het beeldscherm.

SPOSUC Als SPOSLC doch nu voor het bovenste deel van het beeldscherm.

SPOSUL Als SPOSLL doch nu voor het bovenste deel van het beeldscherm.

STKBOT Het adres van het begin van de reken-stack. Deze stack wordt door de rekenroutines in de ROM gebruikt voor het opslaan van floatingpoint-nummers, 5-byte integers en strings (in delen van 5 bytes).

STKEND Het eerste vrije adres na de reken-stack. Zie ook STKBOT. Dit adres kan ook worden gezien als het beginadres van de vrije geheugenruimte.

S-TOP Het hoogste regelnummer van het programma waarvan een listing moet worden gemaakt.

STRLEN Voor alle numerieke variabelen en voor nieuwe string variabelen bevat het byte van STRLEN met het laagste adres de letter van de variabelenaam. Voor bestaande stringvariabelen bevat STRLEN de lengte van die variabelen.

STRMS Adressen van channels die aan een stream zijn gekoppeld.

SUBPPC Hierin wordt het nummer van de instructie binnen de in PPC gegeven regel (die momenteel wordt uitgevoerd) opgeslagen.

TADDR Dit is een pointer naar een item in de parametertabel. In deze tabel staan gegevens betreffende de commando klasse, scheidingstekens en startadressen van de verschillende commando-routines.

TVDATA Het minst significante byte bevat de code van een van de control-karakters INK, PAPER, FLASH, BRIGHT, INVERSE of OVER (die ieder 1 parameter hebben), of van AT of TAB (die ieder twee parameters hebben). Het meest significante byte bevat voor de control-karakters AT of TAB de eerste parameter. De parameters voor INK t/m OVER en de tweede parameters voor AT en TAB staan in het A-register.

TVFLAG De vlagbits in deze byte hebben de volgende betekenis:

bit 0 - Onderste of bovenste deel van het beeldscherm is in gebruik.
bit 1 -
bit 2 -
bit 3 - De mode is wel of niet gewijzigd.
bit 4 - Er wordt wel of niet een automatic listing gemaakt.
bit 5 - Het onderste deel van het beeldscherm moet worden gewist.
bit 6 -
bit 7 -

UDG Deze pointer wijst naar het eerste byte van het geheugengebied waarin de user defined graphics staan. Er kunnen zonder verdere aanpassing 21 karakters worden gedefinieerd (= 8*21 bytes).

VARS Dit adres wijst naar het begin van het variabelengeheugen.

WORKSP Dit adres wijst naar het begin van het werkgeheugen. Dit deel van het geheugen wordt zo klein mogelijk gehouden door het systeem. Het zal dan ook vaak een lengt 0 hebben. Bovendien verplaatst dit deel van het geheugen zich vaak.

XPTR Hierin wordt tijdens programma-invoer via het toetsenbord het adres van het karakter na de syntaxfoutindicator opgeslagen. Tijdens LOAD en MERGE wordt XPTR gebruikt voor het tijdelijke opslaan van diverse adressen.

Microdrive maps

In de standaard ZX Spectrum heeft dit geheugendeel een lengte van 0 bytes, ofwel het bestaat niet. Door echter de systeemvariabele CHANS aan te passen kan er wel ruimte worden gecreeerd.

Channel-informatie

Dit geheugendeel begint op het adres dat is aangegeven in de systeemvariabele CHANS. De lengte hangt af van het gebruikte systeem. De standaard ZX Spectrum heeft vier kanalen. Daar er per kanaal 5 bytes nodig zijn is de lengte van dit geheugengebied in de standaard ZX Spectrum 4 * 5 = 20 bytes + 1 byte die het einde van dit gebied aangeeft. Deze laatste byte bevat altijd de code 128 (decimaal) ofwel 80 (hexadecimaal). Hierna volgt de indeling van dit geheugengebied voor de standaard ZX Spectrum.

2	2	1	: aantal bytes
OUTPUT	INPUT	K	toetsenbord/scherm (systeem)
OUTPUT	BOODSCH	S	scherm (programma)
OUTPUT	BOODSCH	R	werkgeheugen
OUTPUT	BOODSCH	P	printer
128	eindindicator		

Hierin is:

OUTPUT - Het startadres van de voor dit kanaal bestemde outputroutine (in de ROM).

INPUT - Het startadres van de voor dit kanaal bestemde input routine.

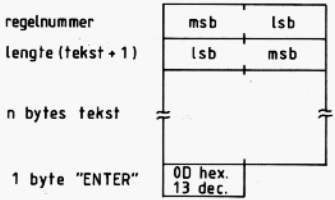
BOODSCHAP - Het adres van de routine waarmee de systeemboodschap "J - Invalid I/O device" kan worden afgedrukt.

Programmageheugen

Het programmageheugen bevat de BASIC-programma-regels.

Dit geheugendeel start op het adres dat is aangegeven in de systeemvariabele PROG en heeft een variabele lengte. Deze lengte is afhankelijk van het aantal BASIC-regels en de lengte van iedere regel.

Alle programmaregels worden direct achter elkaar in het programmageheugen opgeslagen, waarbij iedere programmaregel een verdere onderverdeling kent, zoals in de volgende afbeelding is weergegeven:



Hierin is: msb - most significant byte
lsb - least significant byte.

Het regelnummer is het door u in het programma aangegeven regelnummer.

Hierna volgt een voorbeeld van de manier waarop een programmaregel wordt opgeslagen. Dit voorbeeld is gemaakt aan de hand van de programmaregel:

60 PRINT "VOORBEELD"

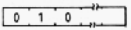
Merk hierbij op dat het lengteveld de lengte geeft van alle bytes die na het lengteveld nog in de regel staan, inclusief het ENTER-karakter, doch exclusief het regelnummer en het lengteveld zelf.

regelnummer 60		00	60
lengte 13 bytes		13	00
PRINT	"	245	34
V	O	86	79
O	R	79	82
B	E	66	69
E	L	69	76
D	"	68	34
ENTER		13	

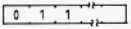
Ook blijkt uit dit voorbeeld duidelijk dat de significantie van de bytes in het regelnummerveld en het lengteveld elkaars tegenovergestelde zijn. Bij het gebruik van deze velden in machinetaalprogramma's of in PEEK en POKE instructies dient u hiermee dan ook rekening te houden.

Variabelengeheugen

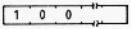
Het variabelengeheugen begint op het adres dat is aangegeven in de systeemvariabele VARS en wordt afgesloten met een eindindicator ('80' hexadecimaal = '128' decimaal).
De lengte van het variabelengeheugen is variabel en hangt af van het aantal variabelen dat er in is opgenomen en de lengte van ieder van die variabelen.
Er zijn zes verschillende soorten variabelen. De eerste drie bits van iedere variabele geven de soort aan. De 5 daaropvolgende bits geven de code van de (eerste) letter van de variabelenaam. Daar de lettercode eigenlijk hoger is dan er met 5 bits kan worden aangegeven, moet er bij deze 5 bits code nog een waarde van '60' hexadecimaal (= 96 decimaal) worden opgeteld om de werkelijke karaktercode te verkrijgen.
De 6 soorten zijn hieronder aangegeven:



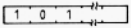
stringvariabele (LET a$="abc")



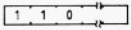
1-letter numerieke variabele
(LET a=123)



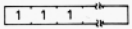
numerieke array (DIM a(4,5))



meerletter numerieke variabele
(LET teller=123)

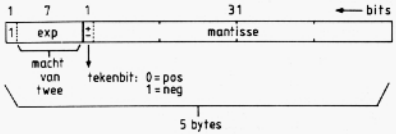


string array (DIM a$(4,5))



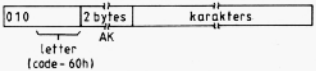
control variabele
(FOR a=1 TO 123
NEXT a)

In alle numerieke variabelen worden de waarden door middel van een veld van 5 bytes weergegeven. Dit veld van 5 bytes heeft het volgende formaat:



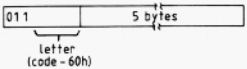
In de formaten van de hiernavolgende numerieke variabelen zullen de numerieke waarden steeds worden aangegeven met "5 bytes".
Op volgorde van de waarde van de eerste drie bits van iedere variabele volgen nu de 6 verschillende formaten.

Stringvariabele



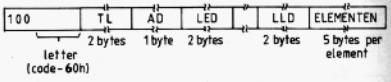
AK - In dit twee bytes veld wordt de lengte van het veld "KARAKTERS" gegeven.
KARAKTERS - Dit zijn de karakters van de gespecificeerde string.

1-letter numerieke variabele



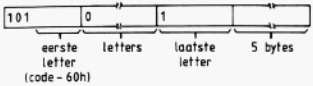
5 bytes - Het formaat van dit veld is hiervoor al besproken.

Numerieke array



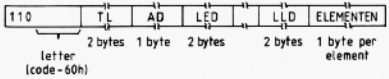
TL - De totale lengte van de array, dat wil zeggen de lengte van AD + LED + LLD + ELEMENTEN.
AD - Het aantal dimensies.
LED - De lengte van de eerste dimensie. Dit is het aantal in die dimensie voorkomende rijen.
LLD - De lengte van de laatste dimensie. Dit is ook weer het aantal in die dimensie voorkomende rijen.
ELEMENTEN - Ieder element bestaat uit een 5 bytes veld, zoals hiervoor al werd beschreven.

Meerletter numerieke variabele



Merk op dat van de eerste en de laatste letter van de naam van dit soort variabele de meest significante bit 1 is. Alle tussenliggende bytes beginnen met een bit met een waarde 0, gevolgd door de ASCII karaktercode. Het 5 bytes veld is al eerder besproken.

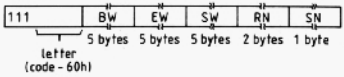
String array



ELEMENTEN - Ieder element is een karakter uit de array en is dus 1 byte lang.

De overige velden zijn hiervoor reeds beschreven bij de numerieke array.

Control variabele



BW - Beginwaarde.
EW - Eindwaarde.
SW - Stapwaarde.
RN - Het regelnummer van de regel waar met de NEXT-instructie naar terug moet worden gesprongen. De linker byte bevat het lsb-deel, het rechter byte het msb-deel.
SN - Het statement-nummer binnen de regel uit RN. SN is nodig omdat de ZX Spectrum multi-statement regels toestaat.

De hiervoor gegeven afkortingen passen als volgt in een FOR-instructie:

<RN> FOR A= <BW> TO <EW> STEP <SW>

EDIT-geheugen

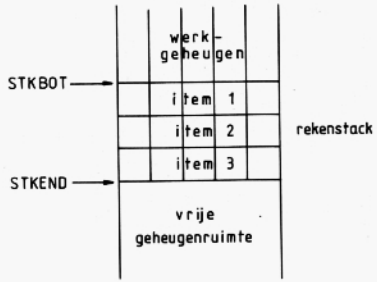
Het EDIT-geheugen heeft een variabele lengte die af hangt van de ingetypte hoeveelheid karakters of de lengte van de met de EDIT-toets geladen BASIC-regels.
Dit geheugendeel start op het adres dat is aangegeven in de systeemvariabele ELINE en wordt ten alle tijde afgesloten met een ENTER-code en een eindindicator (hex '80' = dec '128').

Werkgeheugen

Het werkgeheugen start op het adres dat is aangegeven in de systeemvariabele WORKSP en is variabel van lengte.
Het werkgeheugen wordt onder meer gebruikt voor het opslaan van INPUT-gegevens en cassettelabelinformatie. Ook wordt het werkgeheugen gebruikt voor het samenvoegen van stings (concatination).
Zodra het werkgeheugen niet meer nodig is wordt het gewist en wordt de lengte 0.

Reken-stack

De reken-stack is variabel van lengte en start op het adres dat is aangegeven in de systeemvariabele STKBOT.
De systeemvariabele STKEND wijst naar het eerste vrije byte na de reken-stack. Dit is dan tevens het begin van de nog niet in gebruik zijnde geheugenruimte.
De reken-stack wordt gebruikt door de rekenroutines van de BASIC-ROM, voor het opslaan van floating-point-nummers, vijf-byte-integers en vijf-bytes delen van strings die nog moeten worden geevalueerd (STR$).
Deze stack heeft een LIFO-organisatie (Last In First Out). Indien er geen items (meer) in de stack staan is de lengte 0 bytes.



Z80-stack

De Z80-stack is de stack die door de microprocessor zelf wordt gebruikt.

Hoe deze stack wordt gebruikt en welke informatie er in wordt gezet is de verantwoordelijkheid van de ROM-programmeur. Er worden behalve adressen ook andere gegevens in de stack bewaard.
Alle data wordt altijd per twee bytes op de stack gezet of van de stack teruggelezen. Het Z80 register SP geeft aan waar de laatste data is weggeschreven.

De organisatie van deze stack is Last In First Out (LIFO).

GOSUB-stack

Deze stack bevat altijd de waarden 0 en hex '3E' (= dec '62').
Zodra er een GO SUB-instructie wordt uitgevoerd wordt de stack uitgebreid met een stack-item dat het volgende formaat heeft:



Hierin is:

RNM - Het meest significante byte van het twee bytes regelnummer waar naartoe moet worden teruggesprongen.
RNL - Het minst significante byte van het twee bytes regelnummer waar naartoe moet worden teruggesprongen.
SN - Het statement-nummer binnen de in RNM/RNL aangegeven regel waar naartoe moet worden teruggesprongen.

Indien in het BASIC-programma meer RETURN-instructies staan dan er return-adressen in de GO SUB-stack staan, dan zullen de altijd in de stack staande waarden 0 en hex '3E' er voor zorgen dat de systeemboodschap "7 - RETURN without GO SUB" wordt afgedrukt.

De systeemvariabele RAMTOP wijst altijd naar het hoogste adres in de GO SUB-stack (het byte met de waarde hex '3E'). Dit is het laatste byte waartoe het BASIC-systeem toegang heeft.

UDG-geheugen

Na aanschakelen worden de laatste 21 * 8 = 168 bytes van het RAM toegekend aan het UDG-geheugen. Hierin worden dan ook standaard de bitpatronen voor de letters a t/m u gezet.

U kunt het startadres van dit geheugengebied, dat in de systeemvariabele UDG staat met een POKE-instructie wijzigen.

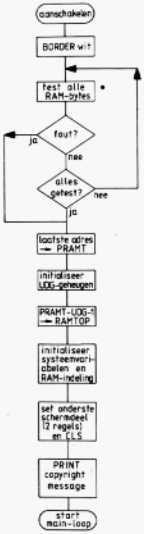
Om te voorkomen dat de user defined graphics door het BASIC-systeem worden overschreven verdient het aanbeveling dit startadres altijd boven RAMTOP te houden.

Om zelf user defined graphics te definieren kan zonder kennis van de organisatie van dit geheugendeel gebruik worden gemaakt van de procedures die in het bij de machine geleverde handboek zijn beschreven.

Wilt u echter graphics definieren met behulp van POKE-instructies rechtstreeks naar geheugen adressen dan volgt hier de (eenvoudige) organisatie van dit geheugendeel:

Voor ieder van de 21 graphics zijn acht bytes opgenomen.
De eerste beeldpuntlijn van 8 bits staat in de eerste byte, de tweede beeldpuntlijn van 8 bits staat in de tweede byte, en zo verder tot alle acht beeldpuntlijnen van het eerste user defined graphic zijn geweest. Daarna volgt de eerste beeldpuntlijn van de tweede user defined graphic in het negende byte, de tweede beeldpuntlijn in het tiende byte, enzovoorts.
Hiermee komt de achtste beeldpuntlijn van de 21ste user defined graphic dus in het 168ste byte te staan.

Aanschakelroutine



* Tijdens dit deel van de aanschakelroutine wordt het beeld gedurende enige tijd zwart. De lengte van de tijd dat het beeld zwart blijft is afhankelijk van de hoeveelheid RAM die getest wordt. Is het beeld plotseling veel korter zwart dan normaal het geval is, dan kan het zijn dat er een fout in het geheugen zit. Controleer dan voor de zekerheid of RAMTOP wel de normale waarde heeft.

I/O-poortadressen

De BASIC-instructies IN en OUT komen in feite overeen met de Z80-machinetaalinstructies IN en OUT. Het enige wat de BASIC-instructies extra doen is het controleren van de bij de instructies opgegeven parameters. Zijn die parameters te hoog of te laag, dan zal een systeemboodschap worden gegeven. Zijn ze goed, dan wordt voor de BASIC-functie IN de machinetaalinstructie IN A(C) en voor de BASIC-instructie OUT de machinetaal-instructie OUT (C),A uitgevoerd, nadat het poort-adres in register C is ingevuld.
In de ROM vindt u deze machinetaalinstructies als volgt terug:

adres hex	adres dec	mnemonic	machinetaalinstructie hex	machinetaalinstructie dec
1E7D	7805	OUT (C),A	ED 79	237 121
34A8	13480	IN A,(C)	ED 78	237 120

Met de OUT-instructie kunnen de volgende I/O-apparaten worden aangeroepen:

OUT 254,A (schrijf de waarde in A naar poort 254)

Poort 254 dient om te schrijven naar de BORDER, de cassette en de luidspreker. De bits van A hebben de volgende functie:

bits 0, 1 en 2 (waarde 0 tot en met 7) geven de BORDER-kleur aan.
bit 3 (waarde 8) geeft output naar de MIC-plug aan.
bit 4 (waarde 16) geeft output naar de luidspreker aan.

OUT 251,A (schrijf de waarde in A naar poort 251)

Poort 251 dient om te schrijven naar de printer. De bits van A hebben nu de volgende functie:

bit 1 = 0 --> motor snel
= 1 --> motor langzaam (waarde 2)
bit 2 = 0 --> start motor
= 1 --> stop motor (waarde 4)
bit 7 = 0 --> pixel is papierkleur
= 1 --> pixel is inktkleur (waarde 128)

IN A,254 (lees de waarde van poort 254 in A)

Poort 254 dient om het toetsenbord en de cassette uit te lezen. De gelezen bits in A hebben de volgende functie:

bits 0 t/m 4 (waarden 0 t/m 31) lezen de 5 keyboard lijnen. Zie voor verdere uitleg "Keyboard I/O" hierna.
bit 6 (waarde 64) leest de EAR-plug uit.

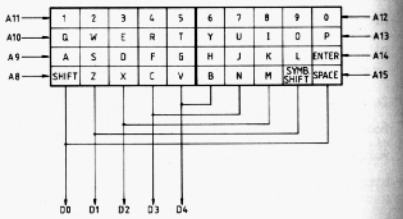
IN A,251 (lees de waarde van poort 251 in A)

Poort 251 dient om de status van de printer uit te lezen. De gelezen bits in A hebben de volgende functie:

bit 0 = 0 --> klaar voor ontvangst van het volgende pixel
= 1 --> wacht met versturen van het volgende pixel.

Keyboard I/O

Het toetsenbord is een speciaal geval van I/O. Om dit te kunnen begrijpen dient u het toetsenbord als volgt te zien:



In register C komt het poortnummer (254) te staan. In register B komt een adressering van een van de halve rijen van 5 toetsen te staan (Een van de lijnen A15 t/m A8 wordt geactiveerd door hem 0 te maken). Tijdens het uitvoeren van de machine-instructie IN wordt namelijk register B op die adreslijnen gecopieerd.
In onderstaande tabel zijn de 8 rijen van 5 toetsen allemaal gedecodeerd.

register B A15 . . . . . A8	register C poortnummer 254	decimaal
1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 0	= 65278
1 1 1 1 1 1 0 1	1 1 1 1 1 1 1 0	= 65022
1 1 1 1 1 0 1 1	1 1 1 1 1 1 1 0	= 64510
1 1 1 1 0 1 1 1	1 1 1 1 1 1 1 0	= 63486
1 1 1 0 1 1 1 1	1 1 1 1 1 1 1 0	= 61438
1 1 0 1 1 1 1 1	1 1 1 1 1 1 1 0	= 57342
1 0 1 1 1 1 1 1	1 1 1 1 1 1 1 0	= 49150
0 1 1 1 1 1 1 1	1 1 1 1 1 1 1 0	= 32766

Als u de waarden van de tabel uitzet tegen het schematje van het toetsenbord dan zult u zien dat het volgende waar is:

IN 65278 leest de rij toetsen van CAPS SHIFT tot V
IN 65022 leest de rij toetsen van A tot G
IN 64510 leest de rij toetsen van Q tot T
IN 63486 leest de rij toetsen van 1 tot 5
IN 61438 leest de rij toetsen van 0 tot 6
IN 57342 leest de rij toetsen van P tot Y
IN 49150 leest de rij toetsen van ENTER tot H
IN 32766 leest de rij toetsen van SPACE tot B

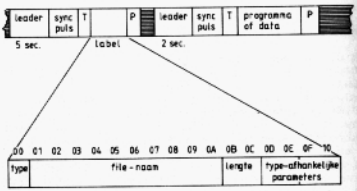
Het zal u nu duidelijk zijn dat de ingelezen waarden geen enkele relatie heeft tot een ASCII-code. Deze relatie wordt in het microprogramma in de ROM gelegd. Daarin wordt ook gekeken of er soms meer dan een toets tegelijk wordt ingedrukt.

Beeldscherm I/O

Ook dit is een speciaal geval van I/O.
Het schrijven van data naar het beeldscherm wordt namelijk continue en automatisch door de Spectrum hardware gedaan, rechtstreeks vanuit het beeldschermgeheugen. Er is geen I/O-poort voor aanwezig, er kan dus ook niet met een machinetaal-instructie naar het beeldscherm worden geschreven. De enige manier om min of meer rechtstreeks naar het beeldscherm te schrijven is door gebruik te maken van een POKE-instructie, waarbij pixels in het beeldschermgeheugen worden geschreven.
Hierbij dient u wel rekening te houden met de speciale organisatie van dat beeldschermgeheugen. (Zie ook het betreffende hoofdstuk)

Cassettelabel

Alle gegevens die naar cassette worden geschreven, worden door een label voorafgegaan. Dit label heeft als functie de naam en het type van het daaropvolgende datablok tijdens het terugladen te kunnen controleren. Het label maakt deel uit van de "header". De header gaat het datablok vooraf. Een en ander is in de hiernavolgende afbeelding weergegeven.



Leader 5 sec
Dit is de fluittoon van ongeveer 5 seconden die deel uitmaakt van de header. Cassetterecorders met automatische opnamevolumeregeling gebruiken dit signaal om zich tijdens het opnemen op de juiste versterking in te stellen. Bij het terugladen van programma's en gegevens kunt u dit signaal gebruiken om de volumeregelaar op de cassette recorder op de juiste sterkte af te stellen (donkere en lichte balken van gelijke dikte aan de beeldrand).

Sync
Dit is een synchronisatiepuls die het begin van de werkelijke gegevens, label of datablok, aangeeft.

T
De inhoud van dit veld van 1 byte geeft aan of de direct hierop volgende data een label (00) is of dat dit een datablok (FF) is.

Label
Het label wordt tijdens het wegschrijven van gegevens naar de cassette in de "workspace" van het RAM-geheugen samengesteld. De inhoud van het label wordt bepaald door de bij het SAVE-commando opgegeven parameters. De indeling is altijd volgens het in de afbeelding gegeven patroon. Ook voor een LOAD-commando wordt in de workspace een label aangemaakt. Bovendien wordt er in diezelfde workspace ruimte gereserveerd voor het van een cassette te lezen label. Zodra een label van de cassette is ingelezen, worden beide in de workspace aanwezige labels met elkaar vergeleken en beslist het systeem of het de gegevens uit het volgende datablok in het RAM zal laden.

Type:
00 - Een cassette-statement wordt gevolgd door LINE, of wordt niet gevolgd door verdere parameters.
01 - Na de parameter DATA is een numerieke variabele gegeven.
02 - Na de parameter DATA is een string variabele gespecificeerd.
03 - De cassette-statement wordt gevolgd door een van de parameters SCREEN$ of CODE.

File-naam
Een veld van 10 bytes, met daarin de naam van het datablok of, indien er geen naam werd gespecificeerd bij een LOAD, de hexadecimale waarde "FF" op de eerste positie daarvan.

Lengte
In deze twee bytes wordt de lengte van het datablok gegeven. Afhankelijk van de opgegeven parameters heeft dit veld de volgende betekenis:

CODE - De bij CODE opgegeven lengte.
SCREEN$ - De totale lengte van het beeldscherm + attributen geheugen. (hexadecimaal "1B00" = decimaal 6912).
DATA - De lengte van de opgegeven variabele.

Type-afh.
Afhankelijk van de bij de statements geplaatste parameters kunnen hier de volgende gegevens voorkomen:

CODE - Startadres van de CODE-data op posities 0D en 0E.
DATA - De variabelenaam op positie 0E.
LINE - Het gespecificeerde regelnummer op de posities 0D en 0E.
SCREEN$ - Het startadres van het beeldschermgeheugen op de posities 0D en 0E.

Indien geen parameters bij de statement werden opgegeven, dan zal op positie 0E de hexadecimale waarde "80" (decimaal 128) worden geplaatst.

P
Dit is het pariteits-byte, dat tijdens SAVE wordt gegenereerd.
Tijdens een LOAD, MERGE of VERIFY wordt ook een pariteits-byte gegenereerd. De pariteits-byte die van de cassette wordt gelezen, wordt vergeleken met de zojuist gegenereerde pariteits-byte. Alleen als beide bytes identiek zijn is de data goed geladen.

Leader 2 sec
Dit is de fluittoon die aan het datablok vooraf gaat. De functie is dezelfde als voor "Leader 5 sec".

Datablok
Het datablok kan het BASIC-programma, de variabele of de bytes bevatten die men wil wegschrijven, teruglezen of verifieren.

Tips

In dit hoofdstuk heb ik een aantal zaken opgenomen die niet binnen andere hoofdstukken pasten, maar die toch belangrijk genoeg waren om te vermelden.

Het opvragen van de nog vrije geheugenruimte kan als volgt worden gedaan:

```
PRINT 65535-USR 7962
```

Op adres 7962 (decimaal) start een routine die de vrije geheugenruimte uitzoekt. Met bovenstaand commando start u die routine. Daar deze routine het resultaat (de vrije geheugenruimte) in register BC zet krijgt u die informatie met dit commando netjes op uw scherm afgedrukt.

De klik die u tijdens het indrukken van een toets hoort is erg zwak. Wilt u een duidelijker klik horen, geef dan even het commando POKE 23609,10 in.
Adres 23609 is namelijk het adres van de systeem variabele PIP. Deze systeemvariabele wordt door het systeem gebruikt om de lengte van de klik te bepalen.

Wilt u voorkomen dat onbevoegden uw programma draaien of uitlezen, dan is er, mits die ander dat niet door heeft, een zekere beveiliging mogelijk.
Begin uw programma met de regels:

```
1 PRINT "wachtwoord?"
2 POKE 23693,63
3 INPUT a$
4 IF a$<>"<uw wachtwoord>" THEN NEW
5 POKE 23693,56
```

SAVE deze routine samen met uw programma op cassette met het commando SAVE <prognaam> LINE 1. Hierdoor zal bij laden dit programma automatisch op regel nummer 1 worden gestart. In regel 2 worden de attributen zo gezet dat, mocht men proberen het programma te "listen", er niets te zien is. Wie het verkeerde wachtwoord geeft zal het commando NEW veroorzaken. Wie het wachtwoord echter kent, die komt in regel 5, waarin de attributen weer zo worden gezet dat de listing te lezen is.

Indien u in uw programma een onderbreking van onbepaalde tijdsduur wenst, maakt u dan gebruik van de instructie PAUSE 0.
Door na PAUSE een 0 te specificeren geeft u de Spectrum namelijk opdracht net zo lang te wachten tot er een toets wordt ingedrukt, en pas daarna weer door te gaan met het programma. Het is daarbij niet belangrijk welke toets er wordt ingedrukt.

Het resetten van de Spectrum kan met behulp van het aan en uitschakelen van de spanning. Hiertoe trekt men de voedingsplug uit de Spectrum en vervolgens steekt men die er weer in. In geval van een hang up is dit inderdaad noodzakelijk, tenzij men een schakelaar heeft aangesloten op de reset- of op de NMI-ingang op de parallelplug.
Heeft men echter geen hang up en wil men toch resetten (net als na aanschakelen), dan kan dit door het commando RANDOMIZE USR 0 te geven.

Het gebruik van ROM-routines

Bij het programmeren in machinetaal is het goed er aan te denken dat het vaak eenvoudig is om voor bepaalde functies een ROM-routine aan te roepen, in plaats van zelf een routine voor die functie te gaan schrijven. Voor een compleet overzicht van alle ROM-routines is dit boekje te klein. Er zijn andere boeken die enkel en alleen over dat onderwerp handelen. Ter illustratie heb ik toch gemeend er goed aan te doen hier enkele voorbeelden te geven.
In deze voorbeelden zult u zien dat er slechts weinig moeite behoeft te worden gedaan. Men moet alleen weten welke informatie de ROM-routines verwachten en in welke registers zij dit verwachten. Vervolgens laadt men die registers met de gewenste informatie. Nu kan de ROM-routine worden aangeroepen en uitgevoerd. Na uitvoering zal de ROM-routine bepaalde registers met bepaalde informatie hebben geladen. Om deze informatie te kunnen gebruiken dient men dus te weten welke informatie in welke registers is geplaatst. Zoals u ziet is het erg eenvoudig. Het enige probleem is de soort informatie per register per ROM-routine uit te zoeken en het startadres van die ROM-routine te vinden. Dit is een karweitje dat veel tijd in beslag kan nemen. Voor de hierna volgende routines behoeft u dat zoek werk alvast niet meer te doen.

```
BEEP
startadres       : 03B5
registergebruik  : DE = tijdsduur
                   HL = toonhoogte
                   mnemonic:              hex.code:
machinetaalprog  : LD   DE,0001           11 01 00
                   LD   HL,0005           21 05 00
                   CALL BEEP              CD B5 03
                   RET                    C9
```

```
CLS
startadres      : 0DAF
registergebruik : -
                  mnemonic:          hex.code:
machinetaalprog : CALL CLS           CD AF 0D
                  RET                C9
```

```
CLEAR LINE(S)
startadres      : 0E44
registergebruik : B = aantal te wissen regels
                  mnemonic:          hex.code:
machinetaalprog : LD B,10            06 0A
                  CALL CL            CD 44 0E
                  RET                C9
```

Ter illustratie een klein BASIC-programmatje van waaruit bovenstaand machinetaalroutinetje wordt aangeroepen. Er is aangenomen dat het routinetje start op adres 64000.
Hiermee heeft u de mogelijkheid om een kop bovenaan het scherm te laten staan terwijl u de rest van het scherm wist.

```
10 FOR a=0 TO 10
20 PRINT "dit blijft staan"
30 NEXT a
40 INPUT "tekst? ";a$
50 PRINT AT 20,0;a$
60 PAUSE 50
70 RANDOMIZE USR 64000
80 GO TO 40
```

Gebruikt u voor het invoeren van de machinetaalroutine het programma "hexlader".

```
SCROLL LINES
startadres      : 0E00
registergebruik : B = aantal te scrollen regels
                  mnemonic:          hex.code:
machinetaalprog : LD   B,10          06 0A
                  CALL SL            CD 00 0E
                  RET                C9
```


Ook voor het scrollen van regels volgt een illustratief BASIC-programma, opdat u de smaak te pakken zult krijgen.
Weer is aangenomen dat het machinetaalprogramma vanaf adres 64000 is geladen.

```
 10 FOR a=0 TO 21
 20 PRINT "deze regels blijven stil staan"
 30 NEXT a
 40 LET a$="abcdefghijklmnopqrstuvwxyz"
 50 PRINT AT 21,0;a$
 60 LET b$=a$(26)
 70 LET a$=b$+a$(TO 25)
 80 RANDOMIZE USR 64000
 90 PAUSE 0
100 GO TO 50
```

Regel 90 vereist dat u voor iedere "loop" een toets indrukt. U kunt daardoor rustig de resultaten bekijken. Weglaten van regel 90 geeft echter een prachtig effect. Probeert u het maar eens.

Programmabeschrijvingen

In dit hoofdstuk zullen van de moeilijkst te bedienen en tegelijkertijd meest verkochte programma's een korte beschrijving worden gegeven. Deze beschrijving heeft alleen tot doel het geheugen wat op te frissen. De ervaring leert dat, wanneer deze programma's langere tijd niet zijn gedraaid, men de functies en de mogelijkheden van die functies vergeet.
Voor een dergelijke beschrijving komen twee programma's in aanmerking, VU-CALC en VU-FILE.

VU-CALC

Na het laden van het programma kan men naar de commandomode, de tekstmode of naar de gegevens/formule-mode gaan.
= naar commandomode.
" = naar tekstmode.
getallen of formules kunnen zonder meer worden ingegeven.
VU-CALC maakt geen onderscheid tussen hoofd- en kleine letters, men mag ze door elkaar gebruiken.

Wil men een bestaande "spreadsheet" van cassette laden, dan dient men #l in te geven.

Commando's:

#B - Wist het "current" veld met de eventueel daarbij behorende formule.

#C - Herberekent alle velden in het hele rooster waaraan een formule is toegekend.

#E - De formule uit het "current" veld kan worden vervangen door een nieuwe.

#F,k,f,j - Formateren van een kolom of van alle kolommen.
k = cijfer --> nummer van de te formateren kolom.
k = A --> Alle kolommen moeten worden geformateerd.
f = i --> getallen worden als integer afgedrukt.
f = $ --> getallen krijgen twee cijfers achter komma.
f = G --> algemeen formaat.
j = L --> links uitgelijnd.
j = R --> rechts uitgelijnd.

#G,rk - Verplaatst de cursor naar de in rk aangegeven rij en kolom.

#L - Laden van een op cassette bewaarde VU-CALC file.

#P - Drukt de op het scherm staande informatie af op de printer.

#Q - Beeindigt VU-CALC. Men krijgt een menu waarmee men kan kiezen uit stoppen, wissen van de spreadsheet of laden van een andere spreadsheet.

#R,rk,e:l - De inhoud van het met rk (rij/kolom) aangegeven veld kan worden gecopieerd naar alle velden tussen e en l.
e = eerste rij en kolom (linksboven)
l = laatste rij en kolom (rechtsonder)
e en l moeten worden gezien als coordinaten in een rechthoek.
Dit commando kan ook worden gebruikt als wis-commando. Door rk te laten wijzen naar een leeg veld, kan men het blok tussen e en l wissen.

#S - Schrijf de inhoud van de spreadsheet naar cassette.

#T,r,r' - Maak een copie van rij (horizontaal) r naar rij r'.
#T,k,k' - Maak een copie van kolom (verticaal) k naar kolom k'.

In de formules mag men van de volgende bewerkingen gebruik maken:

+ voor optellen
- voor aftrekken
* voor vermenigvuldigen
/ voor delen

Getallen die men in de formules wil gebruiken mogen zowel directe getallen als verwijzingen naar velden waarin getallen staan zijn.
Verwijst men naar een veld waarin geen getal staat dan zal dit een fout opleveren.
Naast de hiervoor gegeven bewerkingen is er nog een bewerking mogelijk, de som-faciliteit.

& = som-faciliteit

Voorbeeld:

&A2:C3

Dit zal de som van de velden A2, A3, B2, B3, C2 en C3 in het veld zetten waaraan de formule was toegekend.

Daar het programma VU-FILE een veelverkocht programma is en daar het niet zo eenvoudig te bedienen is volgt hier een korte beschrijving van de mogelijkheden van dit programma.

COMMANDO-mode

In deze mode hebt u de volgende mogelijkheden:

A (lter) Wijzigen van het record dat op het scherm staat. "ENTER" wist het dataveld waarop de cursor staat. Met de cursor besturingstoetsen kunt u naar datavelden springen zonder deze te wissen. Ingegeven data afsluiten met "ENTER".

B (ack) Terug naar het vorige record.

C (opy) Kopieert het op het beeldscherm staande record naar de printer.

D (elete) Wist het op het beeldscherm staande record uit het bestand.

E (nter) Toevoegen van nieuwe records aan het bestand.

F (orward) Door naar het volgende record.

I (nform) Geeft informatie over uw file en de gebruikte geheugenruimte. Door op ongeacht welke toets te drukken komt u terug in de commando-mode.

L (ist) Alle records uit het bestand worden op het beeldscherm afgedrukt. Ieder record blijft ongeveer 1 seconde staan. Door indrukken van ongeacht welke toets stopt u de listing.

O (rder) Sorteren van het bestand op alfabetische volgorde van het dataveld waar de cursor op staat. U kunt de cursor naar een ander dataveld verplaatsen door ongeacht welke toets

in te drukken. Met "ENTER" maakt u het dataveld waarop de cursor staat het veld waarop zal worden gesorteerd.

P (rint) Hiermee worden de records op een in de "printer layout-mode" vastgestelde manier op de printer afgedrukt.

Q (uit) Hiermee beeindigt u de commando-mode. U kunt nu een functie uit het volgende menu kiezen:

1. Enter VU-file (naar commando-mode)
2. Set record layout (naar record layout mode)
3. Set printer layout (naar printer layout mode)
4. Save datafile (bestand naar cassette schrijven)
5. Load new datafile (lees bestand van cassette)
6. Erase current file (wis bestand uit geheugen)

R (eset) Terug naar het eerste record uit het bestand.

S (elect) Zoeken naar een op te geven string in (een item van) de records.
"ENTER" = Controleer het hele record op het voorkomen van de op te geven string. Door een andere toets in te drukken wordt de cursor op een dataveld binnen het record geplaatst. Bij indrukken van "ENTER" zal nu alleen het dataveld waar de cursor op staat worden gecontroleerd op het voorkomen van de op te geven string.
Hierna kunt u de zoek-string ingeven.
"STOP" beeindigt het zoeken.
Wordt een record met de opgegeven zoek-string gevonden, dan wordt u gevraagd of deze string actief dient te blijven. Antwoordt u "y", dan zullen hierna alleen geselecteerde records worden ge-"print" of ge-"list".

DATAFIELD mode

Op uw scherm ziet u de door u opgegeven itemnamen. Met de cursor besturingstoetsen kunt u nu naar de

plaats gaan waar u de inhoud van de items wenst te hebben. Daar aangekomen kunt u die plaats vastleggen door op "ENTER" te drukken. De inhoud hoeft niet op dezelfde regel te staan als de itemnaam. Nadat u de plaats van de item-inhoud hebt opgegeven kunt u de kleur van inkt en papier aangeven.
U dient er rekening mee te houden dat een item nooit langer kan zijn dan 1 regel. Het is wel mogelijk meerdere regels onder een itemnaam te plaatsen, doch later bij het zoeken en sorteren zal slechts maximaal 1 regel in aanmerking worden genomen.
Hebt u alle plaatsen waar u de gegevens, behorende tot het record, wilde opslaan aangegeven, dan kunt u met "STOP" naar de "enter record mode" gaan.

ENTER RECORD mode

U ziet nu de itemnamen op het scherm, terwijl de cursor op het eerste door u gedefinieerde dataveld staat. U kunt nu de inhoud van dit item intikken. Met "ENTER" gaat u naar het volgende dataveld. Hebt u alle datavelden van het record ingevuld, dan verschijnt de cursor weer op het eerste dataveld, dat nu echter van het volgende record is. Tijdens de "Enter record mode" kunt u door indrukken van "STOP" naar de commando mode gaan.

PRINTER LAYOUT mode

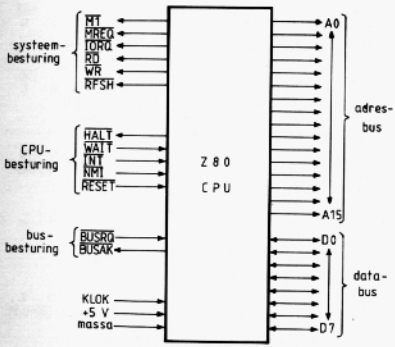
Indien u dat wenst kunt u namen van af te drukken informatie ingeven. Dit behoeven niet dezelfde namen te zijn die u voor de record layout mode had ingegeven. Ook hoeven de namen niet op dezelfde regel te staan. De ingegeven namen zullen op de printer worden afgedrukt, samen met de inhoud van de records.
"STOP" beeindigt de printer layout mode, ook als u geen enkele naam hebt ingegeven.
U komt dan in de Datafield mode. Daarin kunt u de inhoud van de items verplaatsen naar een door u gewenste plaats.
Opm.: Het is verstandig de eerste paar regels niet te beschrijven. Op die manier krijgt u een

aantal regels tussenruimte tussen de op uw printer afgedrukte records.
Voordat u gaat printen kunt u controleren of u de juiste layout hebt gekozen door te printen zonder aangesloten printer.

RECORD LAYOUT mode

Het bestand zal uit een aantal records bestaan. Ieder record kan zijn onderverdeeld in een aantal items (datavelden). In de record layout mode kunnen aan ieder van die items namen worden toegekend. Deze namen kunt u intikken op de plaats waar u dat wilt. Met de cursorbesturingstoetsen kunt u naar die plaats gaan. Met "EDIT" kunt u de kleur van inkt en papier, en het al of niet BRIGHT of FLASHING afdrukken van de itemnamen aangeven. Wanneer u alle gewenste namen op de gewenste plaats (er mogen meer dan 1 namen per regel worden gespecificeerd) hebt gezet, kunt u met "STOP" naar de DATAFIELD mode gaan.

Z80-CPU met signaalbeschrijving



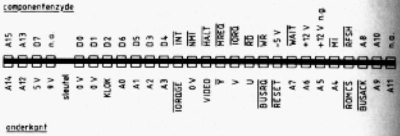
De ZX Spectrum gebruikt als centrale processor een Z80-CPU. De signalen die door de Z80 worden gebruikt en gegenereerd komen allemaal voor op de parallelpoort aan de achterzijde van de ZX Spectrum. Hier volgt een beschrijving van die signalen.

signaal	omschrijving	I/O	H/L
A0-A15	adresbus (3-status)	O	H
D0-D7	databus (3-status)	I/O	H

signaal	omschrijving	I/O	H/L
$\overline{M1}$	machinecyclus 1 Dit signaal is actief gedurende het ophalen van de OPCODE-bytes.	O	L
MREQ	memory request (3-status) Geeft aan dat de adresbus een geheugenadres voor lezen of schrijven bevat.	O	L
$\overline{IORQ}$	input/output request (3-status) Geeft aan dat de adreslijnen A0 t/m A7 een I/O-device-adres voor lezen of schrijven bevatten.	O	L
$\overline{RD}$	read (3-status) Geeft aan dat de Z80 wil lezen uit het geheugen of van het geadresseerde I/O-device.	O	L
$\overline{WR}$	write (3-status) Geeft aan dat de databuslijnen D0 t/m D7 data bevatten die naar het geheugen of naar een I/O-device moeten worden geschreven.	O	L
$\overline{RFSH}$	refresh Geeft aan dat de adreslijnen A0 t/m A6 een refresh-adres voor het dynamisch RAM bevatten. De refresh-cyclus kan worden gestart met het signaal MREQ.	O	L
$\overline{HALT}$	stop Geeft aan dat de Z80-CPU zogenaamde NOP-instruc- ties uitvoert, om een refresh-cyclus mogelijk te maken. Door middel van een interrupt kan de Z80 weer worden doorgestart.	O	L
$\overline{WAIT}$	wacht Geeft aan de Z80 te kennen dat het geadresseerde geheugenadres of de geadresseerde I/O-poort niet direct	I	L

signaal	omschrijving	I/O	H/L
	data kan leveren. De Z80 wacht zolang WAIT actief blijft.		
$\overline{INT}$	interrupt request Hiermee kan een I/O-device de Z80 onderbreken. Indien de Z80 de interrupt accepteert, zet het IORQ.	I	L
$\overline{NMI}$	niet te maskeren interrupt Dit is een interrupt request met een hogere prioriteit dan INT. NMI veroorzaakt een restart op adres 0066(hex). Alleen BUSRQ heeft een nog hogere prioriteit.	I	*
$\overline{RESET}$	reset Zet de volgende CPU-registers op 0: PC (= program counter) I (= interrupt register) R (= refresh register) Gedurende het resetten is de CPU niet ontvankelijk voor interrupts. Na het resetten begint de CPU de instructie op het in PC staande adres (= 0000) uit te voeren.	I	L
$\overline{BUSRQ}$	bus request De Z80 zal bij het zien van dit signaal alle 3-status signalen op hoge impedantie zetten, om zodoende deze signalen vrij te maken voor gebruik door een ander device (bijvoorbeeld de DMA-chip).	I	L
$\overline{BUSAK}$	bus acknowledge Hiermee geeft de Z80 aan dat alle 3-statussignalen op hoge impedantie zijn gezet en dus dat ze vrij zijn voor gebruik door andere devices.	O	L
KLOK	kloksignaal Het kloksignaal moet van TTL-niveau zijn en heeft een "pull-up"-weerstand van 330 ohm nodig.	I	

Uitbreidingsconnector



n.a. = niet aangesloten
n.g. = niet gestabiliseerd

De meeste signalen op deze connector worden door de Z80 gegenereerd en zijn dan ook bij de beschrijving van de Z80-CPU gegeven.
De overige signalen en de voedingsspanningen worden hierna beschreven.

IORQGE - Door dit signaal met +5 V te verbinden kan men voorkomen dat de ULA (Uncommitted Logic Array), de speciaal door Sinclair ontworpen chip die alle I/O verzorgt, het signaal IORQ ontvangt. Men verhindert dan dat er I/O wordt gedaan.

ROMCS - Dit is het chip-select signaal voor de ROM. Door dit signaal met de +5 V te verbinden schakelt men in feite de in de Spectrum aanwezige ROM uit. Indien men tegelijkertijd een vervangende ROM (of RAM) zou inschakelen, zou men een eigen operating systeem of een deel daarvan kunnen uitvoeren. Waarschijnlijk wordt deze mogelijkheid bij de microdrives voor speciale routines gebruikt.

u - Kleurinformatie "blauwgeel"

v - Kleurinformatie "roodgeel"

VIDEO - Dit is de video uitgang waarop het composite video signaal staat dat door een kleurenvideomonitor met een overeenkomstige videoingang kan worden aangesloten.
Dit signaal is niet werkelijk doorverbonden. Er moet een draadbrugje worden gelegd in de Spectrum om dit signaal te activeren.

y - Video sync. + luminantie.

De signalen u, v en y worden toegevoerd aan de LM1889N, die van deze signalen het VIDEO signaal maakt dat via een draadbrug naar de connector gaat en dat direct naar de video-modulator gaat. Uit de video modulator komt dan het TV-signaal dat naar de antenne-ingang van uw TV gaat.

+5 V - Deze voedingsspanning kan gebruikt worden om enkele externe schakelingen van spanning te voorzien. Deze externe schakelingen mogen niet te veel stroom trekken. In de 16 K Spectrum max. 300 mA, in de 48 K Spectrum enkele tientallen milli-amperes.

-5 V - Hiervan kan slechts een zeer lage stroom worden afgenomen, daar deze spanning al spoedig in elkaar zakt. Dit zou tot gevolg hebben dat u uw geheugeninhoud verliest.

+9 V - Dit is de 9 V gelijkspanning die uit het voedingsapparaat bij de Spectrum komt. Men kan hieruit een 5 V spanning via een stabilisatieschakeling genereren. Men mag dan tot maximaal 500 mA stroom trekken. Hogere stromen belasten de transformator te zwaar.

+12 V - Zie de beschrijving van de -5 V.

Z80-registers

De Z80-CPU bevat veertien achtbitsregisters plus vier zestien- bitsregisters. De functies van deze registers wordt hierna beschreven.

A	F	A'	F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'
I	R		
indexregister IX			
indexregister IY			
stackpointer SP			
programcounter PC			

Register A (accumulator)

De accumulator wordt voor achtbits berekeningen en logische functies gebruikt. Het resultaat van die berekening of logische bewerking blijft in de accumulator bewaard.

Register F (vlagregister)

Het vlagregister geeft de condities aan, die het gevolg zijn van acht- of zestienbits bewerkingen. Het F-register heeft de volgende indeling:

S	Z	-	H	-	P/V	N	C

S = sign
S=1 indien het meest significante bit van het resultaat 1 is, anders is S=0.

Z = zero
Z=1 betekent dat het resultaat van de bewerking nul is. Indien het resultaat ongelijk aan nul is, wordt Z=0.

H = halve carry
H=1 wil zeggen dat er een carry was naar bit 4 van de accumulator. Was er geen carry dan is H=0.

P/V = pariteit/overflow
Na logische instructies bevat dit bit informatie over de pariteit. Na rekeninstructies wordt met dit bit aangegeven of er een overflow is opgetreden.

N = optel/aftrek
Indien N=0 dan is er opgeteld. Is N=1 dan is er afgetrokken.

C = carry
C=1 wil zeggen dat er een carry is opgetreden, C=0 betekent dat er geen carry was.

Registerparen BC, DE en HL

Deze registerparen kunnen ieder afzonderlijk worden gebruikt, er zijn dan 6 achtbitsregisters, doch ze kunnen ook per paar worden gebruikt, er zijn dan drie zestienbitsregisters. Het gebruik is vrij te kiezen door de programmeur.

Alternatieve registers

Deze registers hebben dezelfde functies als de hiervoor beschreven overeenkomstige registers. Met behulp van de exchange-instructies kan de programmeur kiezen uit de door hem gewenste register-set. BC, DE en HL kunnen allemaal tesamen met een instructie worden omgewisseld met de registers BC', DE' en HL'.

I-register

Indien de Z80 CPU in interruptmode 2 is, dan zal bij het optreden van en interrupt een sprong worden uitgevoerd naar een adres dat is samengesteld uit de acht bits van het I-register (dit is het meest significante deel) en de acht bits die door het interrumperende randapparaat worden gegenereerd.

R-register

Dit register is een automatische teller voor de "refresh" van het dynamisch RAM. Alleen ten behoeve van en test mag de programmeur dit register met een waarde laden. Onder normale omstandigheden wordt diit register alleen door de Z80 zelf gebruikt.

IX- en IY-registers

In deze indexregisters kan een basisadres worden bewaard, die bij geindexeerd adresseren kan worden gebruikt. Instructies waarmee geindexeerd kan worden geadresseerd gebruiken naast de waarde uit het indexregister nog een extra byte, waarmee de verplaatsing ten opzichte van het adres uit het indexregister wordt aangegeven.

SP (stack pointer)

In dit register wordt het topadres van de stack bewaard. De stack kan op iedere willekeurige

plaats in het geheugen staan. (zie geheugenindeling.) De organisatie is LIPO (last in first out). Met PUSH zet men data op de stack en met POP haalt men die data er weer af.

PC (program counter)

Hierin bevindt zich het adres van de volgende uit te voeren instructie.

Z80-instructieset op volgorde van mnemonics

mnemonic	hex.code	mnemonic	hex.code
ADC A,(HL)	8E	ADD IY,DE	FD19
ADC A,(IX+d)	DD8E05	ADD IY,IY	FD29
ADC A,(IY+d)	FD8E05	ADD IY,SP	FD39
ADC A,A	8F	AND (HL)	A6
ADC A,B	88	AND (IX+d)	DDA605
ADC A,C	89	AND (IY+d)	FDA605
ADC A,D	8A	AND A	A7
ADC A,E	8B	AND B	A0
ADC A,H	8C	AND C	A1
ADC A,L	8D	AND D	A2
ADC A,n	CE20	AND E	A3
ADC HL,BC	ED4A	AND H	A4
ADC HL,DE	ED5A	AND L	A5
ADC HL,HL	ED6A	AND n	E620
ADC HL,SP	ED7A	BIT 0,(HL)	CB46
ADD A,(HL)	86	BIT 0,(IX+d)	DDCB0546
ADD A,(IX+d)	DD8605	BIT 0,(IY+d)	FDCB0546
ADD A,(IY+d)	FD8605	BIT 0,A	CB47
ADD A,A	87	BIT 0,B	CB40
ADD A,B	80	BIT 0,C	CB41
ADD A,C	81	BIT 0,D	CB42
ADD A,D	82	BIT 0,E	CB43
ADD A,E	83	BIT 0,H	CB44
ADD A,H	84	BIT 0,L	CB45
ADD A,L	85	BIT 1,(HL)	CB4E
ADD A,n	C620	BIT 1,(IX+d)	DDCB054E
ADD HL,BC	09	BIT 1,(IY+d)	FDCB054E
ADD HL,DE	19	BIT 1,A	CB4F
ADD HL,HL	29	BIT 1,B	CB48
ADD HL,SP	39	BIT 1,C	CB49
ADD IX,BC	DD09	BIT 1,D	CB4A
ADD IX,DE	DD19	BIT 1,E	CB4B
ADD IX,IX	DD29	BIT 1,H	CB4C
ADD IX,SP	DD39	BIT 1,L	CB4D
ADD IY,BC	FD09	BIT 2,(HL)	CB56

mnemonic	hex.code	mnemonic	hex.code
BIT 2,(IX+d)	DDCB0556	BIT 6,C	CB71
BIT 2,(IY+d)	FDCB0556	BIT 6,D	CB72
BIT 2,A	CB57	BIT 6,E	CB73
BIT 2,B	CB50	BIT 6,H	CB74
BIT 2,C	CB51	BIT 6,L	CB75
BIT 2,D	CB52	BIT 7,(HL)	CB7E
BIT 2,E	CB53	BIT 7,(IX+d)	DDCB057E
BIT 2,H	CB54	BIT 7,(IY+d)	FDCB057E
BIT 2,L	CB55	BIT 7,A	CB7F
BIT 3,(HL)	CB5E	BIT 7,B	CB78
BIT 3,(IX+d)	DDCB055E	BIT 7,C	CB79
BIT 3,(IY+d)	FDCB055E	BIT 7,D	CB7A
BIT 3,A	CB5F	BIT 7,E	CB7B
BIT 3,B	CB58	BIT 7,H	CB7C
BIT 3,C	CB59	BIT 7,L	CB7D
BIT 3,D	CB5A	CALL C,nn	DC8405
BIT 3,E	CB5B	CALL M,nn	FC8405
BIT 3,H	CB5C	CALL NC,nn	D48405
BIT 3,L	CB5D	CALL NZ,nn	C48405
BIT 4,(HL)	CB66	CALL P,nn	F48405
BIT 4,(IX+d)	DDCB0566	CALL PE,nn	EC8405
BIT 4,(IY+d)	FDCB0566	CALL PO,nn	E48405
BIT 4,A	CB67	CALL Z,nn	CC8405
BIT 4,B	CB60	CALL nn	CD8405
BIT 4,C	CB61	CCF	3F
BIT 4,D	CB62	CP (HL)	BE
BIT 4,E	CB63	CP (IX+d)	DDBE05
BIT 4,H	CB64	CP (IY+d)	FDBE05
BIT 4,L	CB65	CP A	BF
BIT 5,(HL)	CB6E	CP B	B8
BIT 5,(IX+d)	DDCB056E	CP C	B9
BIT 5,(IY+d)	FDCB056E	CP D	BA
BIT 5,A	CB6F	CP E	BB
BIT 5,B	CB68	CP H	BC
BIT 5,C	CB69	CP L	BD
BIT 5,D	CB6A	CP n	FE20
BIT 5,E	CB6B	CPD	EDA9
BIT 5,H	CB6C	CPDR	ED89
BIT 5,L	CB6D	CPI	EDA1
BIT 6,(HL)	CB76	CPIR	ED81
BIT 6,(IX+d)	DDCB0576	CPL	2F
BIT 6,(IY+d)	FDCB0576	DAA	27
BIT 6,A	CB77	DEC (HL)	35
BIT 6,B	CB70	DEC (IX+d)	DD3505

mnemonic	hex.code	mnemonic	hex.code
DEC (IY+d)	FD3505	INC H	24
DEC A	3D	INC HL	23
DEC B	05	INC IX	DD23
DEC BC	0B	INC IY	FD23
DEC C	0D	INC L	2C
DEC D	15	INC SP	33
DEC DE	1B	IN A,(n)	DB20
DEC E	1D	IND	EDAA
DEC H	25	INDR	EDBA
DEC HL	2B	INI	EDA2
DEC IX	DD2B	INIR	EDB2
DEC IY	FD2B	JP nn	C38405
DEC L	2D	JP (HL)	E9
DEC SP	3B	JP (IX)	DDE9
DI	F3	JP (IY)	FDE9
DJNZ e	102E	JP C,nn	DA8405
EI	FB	JP M,nn	FA8405
EX (SP),HL	E3	JP NC,nn	D28405
EX (SP),IX	DDE3	JP NZ,nn	C28405
EX (SP),IY	FDE3	JP P,nn	F28405
EX AF,AF'	08	JP PE,nn	EA8405
EX DE,HL	EB	JP PO,nn	E28405
EXX	D9	JP Z,nn	CA8405
HALT	76	JR C,e	382E
IM 0	ED46	JR NC,e	302E
IM 1	ED56	JR NZ,e	202E
IM 2	ED5E	JR Z,e	282E
IN A,(C)	ED78	JR e	182E
IN B,(C)	ED40	LD (BC),A	02
IN C,(C)	ED48	LD (DE),A	12
IN D,(C)	ED50	LD (HL),A	77
IN E,(C)	ED58	LD (HL),B	70
IN H,(C)	ED60	LD (HL),C	71
IN L,(C)	ED68	LD (HL),D	72
INC (HL)	34	LD (HL),E	73
INC (IX+d)	DD3405	LD (HL),H	74
INC (IY+d)	FD3405	LD (HL),L	75
INC A	3C	LD (HL),n	3620
INC B	04	LD (IX+d),A	DD7705
INC BC	03	LD (IX+d),B	DD7005
INC C	0C	LD (IX+d),C	DD7105
INC D	14	LD (IX+d),D	DD7205
INC DE	13	LD (IX+d),E	DD7305
INC E	1C	LD (IX+d),H	DD7405

mnemonic	hex.code	mnemonic	hex.code
LD (IX+d),L	DD7505	LD BC,(nn)	ED4B8405
LD (IX+d),n	DD360520	LD BC,nn	018405
LD (IY+d),A	FD7705	LD C,(HL)	4E
LD (IY+d),B	FD7005	LD C,(IX+d)	DD4E05
LD (IY+d),C	FD7105	LD C,(IY+d)	FD4E05
LD (IY+d),D	FD7205	LD C,A	4F
LD (IY+d),E	FD7305	LD C,B	48
LD (IY+d),H	FD7405	LD C,C	49
LD (IY+d),L	FD7505	LD C,D	4A
LD (IY+d),n	FD360520	LD C,E	4B
LD (nn),A	328405	LD C,H	4C
LD (nn),BC	ED438405	LD C,L	4D
LD (nn),DE	ED538405	LD C,n	0E20
LD (nn),HL	228405	LD D,(HL)	56
LD (nn),IX	DD228405	LD D,(IX+d)	DD5605
LD (nn),IY	FD228405	LD D,(IY+d)	FD5605
LD (nn),SP	ED738405	LD D,A	57
LD A,(BC)	0A	LD D,B	50
LD A,(DE)	1A	LD D,C	51
LD A,(HL)	7E	LD D,D	52
LD A,(IX+d)	DD7E05	LD D,E	53
LD A,(IY+d)	FD7E05	LD D,H	54
LD A,(nn)	3A8405	LD D,L	55
LD A,A	7F	LD D,n	1620
LD A,B	78	LD DE,(nn)	ED5B8405
LD A,C	79	LD DE,nn	118405
LD A,D	7A	LD E,(HL)	5E
LD A,E	7B	LD E,(IX+d)	DD5E05
LD A,H	7C	LD E,(IY+d)	FD5E05
LD A,I	ED57	LD E,A	5F
LD A,L	7D	LD E,B	58
LD A,n	3E20	LD E,C	59
LD A,R	ED5F	LD E,D	5A
LD B,(HL)	46	LD E,E	5B
LD B,(IX+d)	DD4605	LD E,H	5C
LD B,(IY+d)	FD4605	LD E,L	5D
LD B,A	47	LD E,n	1E20
LD B,B	40	LD H,(HL)	66
LD B,C	41	LD H,(IX+d)	DD6605
LD B,D	42	LD H,(IY+d)	FD6605
LD B,E	43	LD H,A	67
LD B,H	44	LD H,B	60
LD B,L	45	LD H,C	61
LD B,n	0620	LD H,D	62

mnemonic		hex.code	mnemonic		hex.code
LD	H,E	63	OR	n	F620
LD	H,H	64	OTDR		ED8B
LD	H,L	65	OTIR		EDB3
LD	H,n	2620	OUT	(C),A	ED79
LD	HL,(nn)	2A8405	OUT	(C),B	ED41
LD	HL,nn	218405	OUT	(C),C	ED49
LD	I,A	ED47	OUT	(C),D	ED51
LD	IX,(nn)	DD2A8405	OUT	(C),E	ED59
LD	IX,nn	DD218405	OUT	(C),H	ED61
LD	IY,(nn)	FD2A8405	OUT	(C),L	ED69
LD	IY,nn	FD218405	OUT	(n),A	D320
LD	L,(HL)	6E	OUTD		EDAB
LD	L,(IX+d)	DD6E05	OUTI		EDA3
LD	L,(IY+d)	FD6E05	POP	AF	F1
LD	L,A	6F	POP	BC	C1
LD	L,B	68	POP	DE	D1
LD	L,C	69	POP	HL	E1
LD	L,D	6A	POP	IX	DDE1
LD	L,E	6B	POP	IY	FDE1
LD	L,H	6C	PUSH	AF	F5
LD	L,L	6D	PUSH	BC	C5
LD	L,n	2E20	PUSH	DE	D5
LD	R,A	ED4F	PUSH	HL	E5
LD	SP,(nn)	ED7B8405	PUSH	IX	DDE5
LD	SP,HL	F9	PUSH	IY	FDE5
LD	SP,IX	DDF9	RES	0,(HL)	CB86
LD	SP,IY	FDF9	RES	0,(IX+d)	DDCB0586
LD	SP,nn	318405	RES	0,(IY+d)	FDCB0586
LDD		EDA8	RES	0,A	CB87
LDDR		EDB8	RES	0,B	CB80
LDI		EDA0	RES	0,C	CB81
LDIR		EDB0	RES	0,D	CB82
NEG		ED44	RES	0,E	CB83
NOP		00	RES	0,H	CB84
OR	(HL)	B6	RES	0,L	CB85
OR	(IX+d)	DDB605	RES	1,(HL)	CB8E
OR	(IY+d)	FDB605	RES	1,(IX+d)	DDCB058E
OR	A	B7	RES	1,(IY+d)	FDCB058E
OR	B	B0	RES	1,A	CB8F
OR	C	B1	RES	1,B	CB88
OR	D	B2	RES	1,C	CB89
OR	E	B3	RES	1,D	CB8A
OR	H	B4	RES	1,E	CB8B
OR	L	B5	RES	1,H	CB8C

mnemonic		hex.code	mnemonic		hex.code
RES	1,L	CB8D	RES	6,A	CBB7
RES	2,(HL)	CB96	RES	6,B	CBB0
RES	2,(IX+d)	DDCB0596	RES	6,C	CBB1
RES	2,(IY+d)	FDCB0596	RES	6,D	CBB2
RES	2,A	CB97	RES	6,E	CBB3
RES	2,B	CB90	RES	6,H	CBB4
RES	2,C	CB91	RES	6,L	CBB5
RES	2,D	CB92	RES	7,(HL)	CBBE
RES	2,E	CB93	RES	7,(IX+d)	DDCB05BE
RES	2,H	CB94	RES	7,(IY+d)	FDCB05BE
RES	2,L	CB95	RES	7,A	CBBF
RES	3,(HL)	CB9E	RES	7,B	CBB8
RES	3,(IX+d)	DDCB059E	RES	7,C	CBB9
RES	3,(IY+d)	FDCB059E	RES	7,D	CBBA
RES	3,A	CB9F	RES	7,E	CBBB
RES	3,B	CB98	RES	7,H	CBBC
RES	3,C	CB99	RES	7,L	CBBD
RES	3,D	CB9A	RET		C9
RES	3,E	CB9B	RET	C	D8
RES	3,H	CB9C	RET	M	F8
RES	3,L	CB9D	RET	NC	D0
RES	4,(HL)	CBA6	RET	NZ	C0
RES	4,(IX+d)	DDCB05A6	RET	P	F0
RES	4,(IY+d)	FDCB05A6	RET	PE	E8
RES	4,A	CBA7	RET	PO	E0
RES	4,B	CBA0	RET	Z	C8
RES	4,C	CBA1	RETI		ED4D
RES	4,D	CBA2	RETN		ED45
RES	4,E	CBA3	RL	(HL)	CB16
RES	4,H	CBA4	RL	(IX+d)	DDCB0516
RES	4,L	CBA5	RL	(IY+d)	FDCB0516
RES	5,(HL)	CBAE	RL	A	CB17
RES	5,(IX+d)	DDCB05AE	RL	B	CB10
RES	5,(IY+d)	FDCB05AE	RL	C	CB11
RES	5,A	CBAF	RL	D	CB12
RES	5,B	CBA8	RL	E	CB13
RES	5,C	CBA9	RL	H	CB14
RES	5,D	CBAA	RL	L	CB15
RES	5,E	CBAB	RLA		17
RES	5,H	CBAC	RLC	(HL)	CB06
RES	5,L	CBAD	RLC	(IX+d)	DDCB0506
RES	6,(HL)	CBB6	RLC	(IY+d)	FDCB0506
RES	6,(IX+d)	DDCB05B6	RLC	A	CB07
RES	6,(IY+d)	FDCB05B6	RLC	B	CB00

mnemonic		hex.code	mnemonic		hex.code
RLC	C	CB01	SBC	A,C	99
RLC	D	CB02	SBC	A,D	9A
RLC	E	CB03	SBC	A,E	9B
RLC	H	CB04	SBC	A,H	9C
RLC	L	CB05	SBC	A,L	9D
RLCA		07	SBC	HL,BC	ED42
RLD		ED6F	SBC	HL,DE	ED52
RR	(HL)	CB1E	SBC	HL,HL	ED62
RR	(IX+d)	DDCB051E	SBC	HL,SP	ED72
RR	(IY+d)	FDCB051E	SCF		37
RR	A	CB1F	SET	0,(HL)	CBC6
RR	B	CB18	SET	0,(IX+d)	DDCB05C6
RR	C	CB19	SET	0,(IY+d)	FDCB05C6
RR	D	CB1A	SET	0,A	CBC7
RR	E	CB1B	SET	0,B	CBC0
RR	H	CB1C	SET	0,C	CBC1
RR	L	CB1D	SET	0,D	CBC2
RRA		1F	SET	0,E	CBC3
RRC	(HL)	CB0E	SET	0,H	CBC4
RRC	(IX+d)	DDCB050E	SET	0,L	CBC5
RRC	(IY+d)	FDCB050E	SET	1,(HL)	CBCE
RRC	A	CB0F	SET	1,(IX+d)	DDCB05CE
RRC	B	CB08	SET	1,(IY+d)	FDCB05CE
RRC	C	CB09	SET	1,A	CBCF
RRC	D	CB0A	SET	1,B	CBC8
RRC	E	CB0B	SET	1,C	CBC9
RRC	H	CB0C	SET	1,D	CBCA
RRC	L	CB0D	SET	1,E	CBCB
RRCA		0F	SET	1,H	CBCC
RRD		ED67	SET	1,L	CBCD
RST	00H	C7	SET	2,(HL)	CBD6
RST	08H	CF	SET	2,(IX+d)	DDCB05D6
RST	10H	D7	SET	2,(IY+d)	FDCB05D6
RST	18H	DF	SET	2,A	CBD7
RST	20H	E7	SET	2,B	CBD0
RST	28H	EF	SET	2,C	CBD1
RST	30H	F7	SET	2,D	CBD2
RST	38H	FF	SET	2,E	CBD3
SBC	A,n	DE20	SET	2,H	CBD4
SBC	A,(HL)	9E	SET	2,L	CBD5
SBC	A,(IX+d)	DD9E05	SET	3,(HL)	CBDE
SBC	A,(IY+d)	FD9E05	SET	3,(IX+d)	DDCB05DE
SBC	A,A	9F	SET	3,(IY+d)	FDCB05DE
SBC	A,B	98	SET	3,A	CBDF

mnemonic		hex.code	mnemonic		hex.code
SET	3,B	CBD8	SET	7,H	CBFC
SET	3,C	CBD9	SET	7,L	CBFD
SET	3,D	CBDA	SLA	(HL)	CB26
SET	3,E	CBDB	SLA	(IX+d)	DDCB0526
SET	3,H	CBDC	SLA	(IY+d)	FDCB0526
SET	3,L	CBDD	SLA	A	CB27
SET	4,(HL)	CBE6	SLA	B	CB20
SET	4,(IX+d)	DDCB05E6	SLA	C	CB21
SET	4,(IY+d)	FDCB05E6	SLA	D	CB22
SET	4,A	CBE7	SLA	E	CB23
SET	4,B	CBE0	SLA	H	CB24
SET	4,C	CBE1	SLA	L	CB25
SET	4,D	CBE2	SRA	(HL)	CB2E
SET	4,E	CBE3	SRA	(IX+d)	DDCB052E
SET	4,H	CBE4	SRA	(IY+d)	FDCB052E
SET	4,L	CBE5	SRA	A	CB2F
SET	5,(HL)	CBEE	SRA	B	CB28
SET	5,(IX+d)	DDCB05EE	SRA	C	CB29
SET	5,(IY+d)	FDCB05EE	SRA	D	CB2A
SET	5,A	CBEF	SRA	E	CB2B
SET	5,B	CBE8	SRA	H	CB2C
SET	5,C	CBE9	SRA	L	CB2D
SET	5,D	CBEA	SRL	(HL)	CB3E
SET	5,E	CBEB	SRL	(IX+d)	DDCB053E
SET	5,H	CBEC	SRL	(IY+d)	FDCB053E
SET	5,L	CBED	SRL	A	CB3F
SET	6,(HL)	CBF6	SRL	B	CB38
SET	6,(IX+d)	DDCB05F6	SRL	C	CB39
SET	6,(IY+d)	FDCB05F6	SRL	D	CB3A
SET	6,A	CBF7	SRL	E	CB3B
SET	6,B	CBF0	SRL	H	CB3C
SET	6,C	CBF1	SRL	L	CB3D
SET	6,D	CBF2	SUB	(HL)	96
SET	6,E	CBF3	SUB	(IX+d)	DD9605
SET	6,H	CBF4	SUB	(IY+d)	FD9605
SET	6,L	CBF5	SUB	A	97
SET	7,(HL)	CBFE	SUB	B	90
SET	7,(IX+d)	DDCB05FE	SUB	C	91
SET	7,(IY+d)	FDCB05FE	SUB	D	92
SET	7,A	CBFF	SUB	E	93
SET	7,B	CBF8	SUB	H	94
SET	7,C	CBF9	SUB	L	95
SET	7,D	CBFA	SUB	n	D620
SET	7,E	CBFB	XOR	(HL)	AE

mnemonic	hex.code
XOR (IX+d)	DDAE05
XOR (IY+d)	FDAE05
XOR A	AF
XOR B	A8
XOR C	A9
XOR D	AA
XOR E	AB
XOR H	AC
XOR L	AD
XOR n	EE20

In de hiervoor gegeven tabel van Z80-instructies op alfabetische volgorde van mnemonics zijn in de hexadecimale codes van de instructies de volgende voorbeeldwaarden opgenomen:

nn - 0584H (dit is in de instructie 8405)

d - 5 (dit is in de instructie 05)

n - 20H (dit is in de instructie 20)

e - 2EH (dit is in de instructie 2E)

Dit heeft tot gevolg dat de in deze tabel gegeven instructies een reele lengte hebben. Bij het in machinetaal programmeren behoeft u alleen de door u gewenste waarden voor nn, d, n en e in te vullen.

Z80-instructieset op volgorde van hex.code

hex.code	mnemonic	hex.code	mnemonic
00	NOP	23	INC HL
018405	LD BC,nn	24	INC H
02	LD (BC),A	25	DEC H
03	INC BC	2620	LD H,n
04	INC B	27	DAA
05	DEC B	282E	JR Z,e
0620	LD B,n	29	ADD HL,HL
07	RLCA	2A8405	LD HL,(nn)
08	EX AF,AF'	2B	DEC HL
09	ADD HL,BC	2C	INC L
0A	LD A,(BC)	2D	DEC L
0B	DEC BC	2E20	LD L,n
0C	INC C	2F	CPL
0D	DEC C	302E	JR NC,e
0E20	LD C,n	318405	LD SP,nn
0F	RRCA	328405	LD (nn),A
102E	DJNZ e	33	INC SP
118405	LD DE,nn	34	INC (HL)
12	LD (DE),A	35	DEC (HL)
13	INC DE	3620	LD (HL),n
14	INC D	37	SCF
15	DEC D	382E	JR C,e
1620	LD D,n	39	ADD HL,SP
17	RLA	3A8405	LD A,(nn)
182E	JR e	3B	DEC SP
19	ADD HL,DE	3C	INC A
1A	LD A,(DE)	3D	DEC A
1B	DEC DE	3E20	LD A,n
1C	INC E	3F	CCF
1D	DEC E	40	LD B,B
1E20	LD E,n	41	LD B,C
1F	RRA	42	LD B,D
202E	JR NZ,e	43	LD B,E
218405	LD HL,nn	44	LD B,H
228405	LD (nn),HL	45	LD B,L

hex.code	mnemonic		hex.code	mnemonic	
46	LD	B,(HL)	72	LD	(HL),D
47	LD	B,A	73	LD	(HL),E
48	LD	C,B	74	LD	(HL),H
49	LD	C,C	75	LD	(HL),L
4A	LD	C,D	76	HALT	
4B	LD	C,E	77	LD	(HL),A
4C	LD	C,H	78	LD	A,B
4D	LD	C,L	79	LD	A,C
4E	LD	C,(HL)	7A	LD	A,D
4F	LD	C,A	7B	LD	A,E
50	LD	D,B	7C	LD	A,H
51	LD	D,C	7D	LD	A,L
52	LD	D,D	7E	LD	A,(HL)
53	LD	D,E	7F	LD	A,A
54	LD	D,H	80	ADD	A,B
55	LD	D,L	81	ADD	A,C
56	LD	D,(HL)	82	ADD	A,D
57	LD	D,A	83	ADD	A,E
58	LD	E,B	84	ADD	A,H
59	LD	E,C	85	ADD	A,L
5A	LD	E,D	86	ADD	A,(HL)
5B	LD	E,E	87	ADD	A,A
5C	LD	E,H	88	ADC	A,B
5D	LD	E,L	89	ADC	A,C
5E	LD	E,(HL)	8A	ADC	A,D
5F	LD	E,A	8B	ADC	A,E
60	LD	H,B	8C	ADC	A,H
61	LD	H,C	8D	ADC	A,L
62	LD	H,D	8E	ADC	A,(HL)
63	LD	H,E	8F	ADC	A,A
64	LD	H,H	90	SUB	B
65	LD	H,L	91	SUB	C
66	LD	H,(HL)	92	SUB	D
67	LD	H,A	93	SUB	E
68	LD	L,B	94	SUB	H
69	LD	L,C	95	SUB	L
6A	LD	L,D	96	SUB	(HL)
6B	LD	L,E	97	SUB	A
6C	LD	L,H	98	SBC	A,B
6D	LD	L,L	99	SBC	A,C
6E	LD	L,(HL)	9A	SBC	A,D
6F	LD	L,A	9B	SBC	A,E
70	LD	(HL),B	9C	SBC	A,H
71	LD	(HL),C	9D	SBC	A,L

hex.code	mnemonic		hex.code	mnemonic	
9E	SBC	A,(HL)	CA8405	JP	Z,nn
9F	SBC	A,A	CB00	RLC	B
A0	AND	B	CB01	RLC	C
A1	AND	C	CB02	RLC	D
A2	AND	D	CB03	RLC	E
A3	AND	E	CB04	RLC	H
A4	AND	H	CB05	RLC	L
A5	AND	L	CB06	RLC	(HL)
A6	AND	(HL)	CB07	RLC	A
A7	AND	A	CB08	RRC	B
A8	XOR	B	CB09	RRC	C
A9	XOR	C	CB0A	RRC	D
AA	XOR	D	CB0B	RRC	E
AB	XOR	E	CB0C	RRC	H
AC	XOR	H	CB0D	RRC	L
AD	XOR	L	CB0E	RRC	(HL)
AE	XOR	(HL)	CB0F	RRC	A
AF	XOR	A	CB10	RL	B
B0	OR	B	CB11	RL	C
B1	OR	C	CB12	RL	D
B2	OR	D	CB13	RL	E
B3	OR	E	CB14	RL	H
B4	OR	H	CB15	RL	L
B5	OR	L	CB16	RL	(HL)
B6	OR	(HL)	CB17	RL	A
B7	OR	A	CB18	RR	B
B8	CP	B	CB19	RR	C
B9	CP	C	CB1A	RR	D
BA	CP	D	CB1B	RR	E
BB	CP	E	CB1C	RR	H
BC	CP	H	CB1D	RR	L
BD	CP	L	CB1E	RR	(HL)
BE	CP	(HL)	CB1F	RR	A
BF	CP	A	CB20	SLA	B
C0	RET	NZ	CB21	SLA	C
C1	POP	BC	CB22	SLA	D
C28405	JP	NZ,nn	CB23	SLA	E
C38405	JP	nn	CB24	SLA	H
C48405	CALL	NZ,nn	CB25	SLA	L
C5	PUSH	BC	CB26	SLA	(HL)
C620	ADD	A,n	CB27	SLA	A
C7	RST	0	CB28	SRA	B
C8	RET	Z	CB29	SRA	C
C9	RET		CB2A	SRA	D

hex.code	mnemonic
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB2E	SRA (HL)
CB2F	SRA A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
CB3E	SRL (HL)
CB3F	SRL 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
CB46	BIT 0,(HL)
CB47	BIT 0,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
CB4E	BIT 1,(HL)
CB4F	BIT 1,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
CB56	BIT 2,(HL)
CB57	BIT 2,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
CB5E	BIT 3,(HL)
CB5F	BIT 3,A
CB60	BIT 4,B
CB61	BIT 4,C
CB62	BIT 4,D
CB63	BIT 4,E
CB64	BIT 4,H
CB65	BIT 4,L
CB66	BIT 4,(HL)
CB67	BIT 4,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
CB6E	BIT 5,(HL)
CB6F	BIT 5,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
CB76	BIT 6,(HL)
CB77	BIT 6,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
CB7E	BIT 7,(HL)
CB7F	BIT 7,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
CB86	RES 0,(HL)
CB87	RES 0,A
CB88	RES 1,B
CB89	RES 1,C
CB8A	RES 1,D

hex.code	mnemonic
CB8B	RES 1,E
CB8C	RES 1,H
CB8D	RES 1,L
CB8E	RES 1,(HL)
CB8F	RES 1,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
CB96	RES 2,(HL)
CB97	RES 2,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
CB9E	RES 3,(HL)
CB9F	RES 3,A
CBA0	RES 4,B
CBA1	RES 4,C
CBA2	RES 4,D
CBA3	RES 4,E
CBA4	RES 4,H
CBA5	RES 4,L
CBA6	RES 4,(HL)
CBA7	RES 4,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
CBAE	RES 5,(HL)
CBAF	RES 5,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
CBB6	RES 6,(HL)
CBB7	RES 6,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
CBBE	RES 7,(HL)
CBBF	RES 7,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
CBC6	SET 0,(HL)
CBC7	SET 0,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
CBCE	SET 1,(HL)
CBCF	SET 1,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
CBD6	SET 2,(HL)
CBD7	SET 2,A
CBD8	SET 3,B
CBD9	SET 3,C
CBDA	SET 3,D
CBDB	SET 3,E
CBDC	SET 3,H
CBDD	SET 3,L
CBDE	SET 3,(HL)
CBDF	SET 3,A
CBE0	SET 4,B
CBE1	SET 4,C
CBE2	SET 4,D

hex.code	mnemonic		hex.code	mnemonic	
CBE3	SET	4,E	DB20	IN	A,n
CBE4	SET	4,H	DC8405	CALL	C,nn
CBE5	SET	4,L	DD09	ADD	IX,BC
CBE6	SET	4,(HL)	DD19	ADD	IX,DE
CBE7	SET	4,A	DD218405	LD	IX,nn
CBE8	SET	5,B	DD228405	LD	(nn),IX
CBE9	SET	5,C	DD23	INC	IX
CBEA	SET	5,D	DD29	ADD	IX,IX
CBEB	SET	5,E	DD2A8405	LD	IX,(nn)
CBEC	SET	5,H	DD2B	DEC	IX
CBED	SET	5,L	DD3405	INC	(IX+d)
CBEE	SET	5,(HL)	DD3505	DEC	(IX+d)
CBEF	SET	5,A	DD360520	LD	(IX+d),n
CBF0	SET	6,B	DD39	ADD	IX,SP
CBF1	SET	6,C	DD4605	LD	B,(IX+d)
CBF2	SET	6,D	DD4E05	LD	C,(IX+d)
CBF3	SET	6,E	DD5605	LD	D,(IX+d)
CBF4	SET	6,H	DD5E05	LD	E,(IX+d)
CBF5	SET	6,L	DD6605	LD	H,(IX+d)
CBF6	SET	6,(HL)	DD6E05	LD	L,(IX+d)
CBF7	SET	6,A	DD7005	LD	(IX+d),B
CBF8	SET	7,B	DD7105	LD	(IX+d),C
CBF9	SET	7,C	DD7205	LD	(IX+d),D
CBFA	SET	7,D	DD7305	LD	(IX+d),E
CBFB	SET	7,E	DD7405	LD	(IX+d),H
CBFC	SET	7,H	DD7505	LD	(IX+d),L
CBFD	SET	7,L	DD7705	LD	(IX+d),A
CBFE	SET	7,(HL)	DD7E05	LD	A,(IX+d)
CBFF	SET	7,A	DD8605	ADD	A,(IX+d)
CC8405	CALL	Z,nn	DD8E05	ADC	A,(IX+d)
CD8405	CALL	nn	DD9605	SUB	(IX+d)
CE20	ADC	A,n	DD9E05	SBC	A,(IX+d)
CF	RST	8	DDA605	AND	(IX+d)
D0	RET	NC	DDAE05	XOR	(IX+d)
D1	POP	DE	DDB605	OR	(IX+d)
D28405	JP	NC,nn	DDBE05	CP	(IX+d)
D320	OUT	n,A	DDCB0506	RLC	(IX+d)
D48405	CALL	NC,nn	DDCB050E	RLC	(IX+d)
D5	PUSH	DE	DDCB0516	RL	(IX+d)
D620	SUB	n	DDCB051E	RR	(IX+d)
D7	RST	10	DDCB0526	SLA	(IX+d)
D8	RET	C	DDCB052E	SRA	(IX+d)
D9	EXX		DDCB053E	SRL	(IX+d)
DA8405	JP	C,nn	DDCB0546	BIT	0,(IX+d)

hex.code	mnemonic		hex.code	mnemonic	
DDCB054E	BIT	1,(IX+d)	ED41	OUT	(C),B
DDCB0556	BIT	2,(IX+d)	ED42	SBC	HL,BC
DDCB055E	BIT	3,(IX+d)	ED438405	LD	(nn),BC
DDCB0566	BIT	4,(IX+d)	ED44	NEG	
DDCB056E	BIT	5,(IX+d)	ED45	RETN	
DDCB0576	BIT	6,(IX+d)	ED46	IM	0
DDCB057E	BIT	7,(IX+d)	ED47	LD	I,A
DDCB0586	RES	0,(IX+d)	ED48	IN	C,(C)
DDCB058E	RES	1,(IX+d)	ED49	OUT	(C),C
DDCB0596	RES	2,(IX+d)	ED4A	ADC	HL,BC
DDCB059E	RES	3,(IX+d)	ED4B8405	LD	BC,(nn)
DDCB05A6	RES	4,(IX+d)	ED4D	RETI	
DDCB05AE	RES	5,(IX+d)	ED50	IN	D,(C)
DDCB05B6	RES	6,(IX+d)	ED51	OUT	(C),D
DDCB05BE	RES	7,(IX+d)	ED52	SBC	HL,DE
DDCB05C6	SET	0,(IX+d)	ED538405	LD	(nn),DE
DDCB05CE	SET	1,(IX+d)	ED56	IM	1
DDCB05D6	SET	2,(IX+d)	ED57	LD	A,I
DDCB05DE	SET	3,(IX+d)	ED58	IN	E,(C)
DDCB05E6	SET	4,(IX+d)	ED59	OUT	(C),E
DDCB05EE	SET	5,(IX+d)	ED5A	ADC	HL,DE
DDCB05F6	SET	6,(IX+d)	ED5B8405	LD	DE,(nn)
DDCB05FE	SET	7,(IX+d)	ED5E	IM	2
DDE1	POP	IX	ED60	IN	H,(C)
DDE3	EX	(SP),IX	ED61	OUT	(C),H
DDE5	PUSH	IX	ED62	SBC	HL,HL
DDE9	JP	(IX)	ED67	RRD	
DDF9	LD	SP,IX	ED68	IN	L,(C)
DD20	SBC	A,n	ED69	OUT	(C),L
DF	RST	18	ED6A	ADC	HL,HL
E0	RET	PO	ED6F	RLD	
E1	POP	HL	ED72	SBC	HL,SP
E28405	JP	PO,nn	ED738405	LD	(nn),SP
E3	EX	(SP),HL	ED78	IN	A,(C)
E48405	CALL	PO,nn	ED79	OUT	(C),A
E5	PUSH	HL	ED7A	ADC	HL,SP
E620	AND	n	ED7B8405	LD	SP,(nn)
E7	RST	20	EDA0	LDI	
E8	RET	PE	EDA1	CPI	
E9	JP	(HL)	EDA2	INI	
EA8405	JP	PE,nn	EDA3	OUTI	
EB	EX	DE,nn	EDA8	LDD	
EC8405	CALL	PE,nn	EDA9	CPD	
ED40	IN	B,(C)	EDAA	IND	

hex.code	mnemonic		hex.code	mnemonic	
EDAB	OUTD		FD7205	LD	(IY+d),D
EDB0	LDIR		FD7305	LD	(IY+d),E
EDB1	CPIR		FD7405	LD	(IY+d),H
EDB2	INIR		FD7505	LD	(IY+d),L
EDB3	OTIR		FD7705	LD	(IY+d),A
EDB8	LDDR		FD7E05	LD	A,(IY+d)
EDB9	CPDR		FD8605	ADD	A,(IY+d)
EDBA	INDR		FD8E05	ADC	A,(IY+d)
EDBB	OTDR		FD9605	SUB	(IY+d)
EE20	XOR	n	FD9E05	SBC	A,(IY+d)
EF	RST	28	FDA605	AND	(IY+d)
F0	RET	P	FDAE05	XOR	(IY+d)
F1	POP	AF	FDB605	OR	(IY+d)
F28405	JP	P,nn	FDBE05	CP	(IY+d)
F3	DI		FDCB0506	RLC	(IY+d)
F48405	CALL	P,nn	FDCB050E	RRC	(IY+d)
F5	PUSH	AF	FDCB0516	RL	(IY+d)
F620	OR	n	FDCB051E	RR	(IY+d)
F7	RST	30	FDCB0526	SLA	(IY+d)
F8	RET	M	FDCB052E	SRA	(IY+d)
F9	LD	SP,HL	FDCB053E	SRL	(IY+d)
FA8405	JP	M,nn	FDCB0546	BIT	0,(IY+d)
FB	EI		FDCB054E	BIT	1,(IY+d)
FC8405	CALL	M,nn	FDCB0556	BIT	2,(IY+d)
FD09	ADD	IY,BC	FDCB055E	BIT	3,(IY+d)
FD19	ADD	IY,DE	FDCB0566	BIT	4,(IY+d)
FD218405	LD	IY,nn	FDCB056E	BIT	5,(IY+d)
FD228405	LD	(nn),IY	FDCB0576	BIT	6,(IY+d)
FD23	INC	IY	FDCB057E	BIT	7,(IY+d)
FD29	ADD	IY,IY	FDCB0586	RES	0,(IY+d)
FD2A8405	LD	IY,(nn)	FDCB058E	RES	1,(IY+d)
FD2B	DEC	IY	FDCB0596	RES	2,(IY+d)
FD3405	INC	(IY+d)	FDCB059E	RES	3,(IY+d)
FD3505	DEC	(IY+d)	FDCB05A6	RES	4,(IY+d)
FD3660520	LD	(IY+d),n	FDCB05AE	RES	5,(IY+d)
FD39	ADD	IY,SP	FDCB05B6	RES	6,(IY+d)
FD4605	LD	B,(IY+d)	FDCB05BE	RES	7,(IY+d)
FD4E05	LD	C,(IY+d)	FDCB05C6	SET	0,(IY+d)
FD5605	LD	D,(IY+d)	FDCB05CE	SET	1,(IY+d)
FD5E05	LD	E,(IY+d)	FDCB05D6	SET	2,(IY+d)
FD6605	LD	H,(IY+d)	FDCB05DE	SET	3,(IY+d)
FD6E05	LD	L,(IY+d)	FDCB05E6	SET	4,(IY+d)
FD7005	LD	(IY+d),B	FDCB05EE	SET	5,(IY+d)
FD7105	LD	(IY+d),C	FDCB05F6	SET	6,(IY+d)

hex.code	mnemonic	
FDCB05FE	SET	7,(IY+d)
FDE1	POP	IY
FDE3	EX	(SP),IY
FDE5	PUSH	IY
FDE9	JP	(IY)
FDF9	LD	SP,IY
FE20	CP	n
FF	RST	38

In de hiervoor gegeven tabel van Z80-instructies op volgorde van hun hexadecimale code zijn in de hexadecimale code van die instructies de volgende voorbeeldwaarden opgenomen:

nn - 584H (dit is in de instructie 8405)

d - 5 (dit is in de instructie 05)

n - 20H (dit is in de instructie 20)

e - 2EH (dit is in de instructie 2E)

Dit heeft tot gevolg dat de in deze tabel opgenomen instructies een reele lengte hebben. Bij het uitlezen van een stuk machinetaalprogramma zult u echter naar alle waarschijnlijkheid andere waarden tegenkomen.

Z80-vlagbeïnvloeding

instructies	C	Z	P/V	-	S	N	H	opmerkingen
ADD A,s / ADC A,s	A	A	V	A	0	A		8 bits optel-instructies.
SUB s / SBC A,s / CP s / NEG	A	A	V	A	1	A		8 bits aftrek-vergelijk- en negatie-instructies.
AND s	0	A	P	A	0	1		} logische
OR s / XOR s	0	A	P	A	0	0		} bewerkingen.
INC s	C	A	V	A	0	A		8 bits increment.
DEC s	C	A	V	A	1	A		8 bits decrement.
ADD dd,ss	A	C	C	C	0	X		16 bits optellen.
ADC HL,ss	A	A	V	A	0	X		16 bits optellen + carry.
SBC HL,ss	A	A	V	A	1	X		16 bits aftrekken + carry.
RLA/RLCA/RRA/RRCA	A	C	C	C	0	0		Roteren accu.
RL s/ RLC s/ RR s RRC s/ SLA s/ SRL s/SRA s	A	A	P	A	0	0		Roteren en schuiven van locatie s.
RLD / RRD	C	A	P	A	0	0		Roteren tetrade.
DAA	A	A	P	A	C	A		Decimal adjust accu.
CPL	C	C	C	C	1	1		Complementeren accu.
SCF	1	C	C	C	0	0		Zet carry-vlag.
CCF	A	C	C	C	0	X		Complementeer carry-vlag.
IN r,(C)	C	A	P	A	0	0		Input in register.
INI/IND/OUTI/OUTD	C	A	X	X	1	X		Blok-I/O. Indien
INIR/INDR/OTIR/ OTDR	C	1	X	X	1	X		B=0 dan Z=1, anders Z=0.
LDI / LDD	C	X	A	X	0	0		Blok-transfer.

instructies	C	Z	P/V	-	S	N	H	opmerkingen
LDIR / LDDR	C	X	0	X	0	0		Indien B=0 dan P/V=0, anders P/V=1.
CPI/CPIR/CPD/CPDR	C	A	A	A	1	X		Blok-zoek. Indien A=(HL) dan Z=1. Indien BC=0 dan P/V=0.
LD A,I / LD A,R	C	A	F	A	0	0		IFF naar P/V-vlag.
BIT b,s	C	A	X	X	0	1		Complement van bit b uit locatie s naar Z-vlag.

De betekenis van de in de voorgaande tabel gebruikte afkortingen wordt hieronder verklaard:

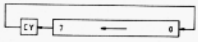
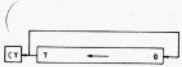
A - Vlag wordt afhankelijk van het resultaat van de bewerking gezet.
C - De vlag blijft onveranderd.
0 - Vlag wordt op 0 gezet.
1 - Vlag wordt op 1 gezet.
X - Status van de vlag is niet belangrijk.
V - De "overflow"-vlag wordt al naar gelang van het resultaat van de bewerking gezet.
P - De pariteitsvlag wordt al naar gelang van het resultaat van de bewerking gezet.
r - Een van de CPU-registers A, B, C, D, E, H of L.
s - Een 8 bits geheugenlocatie.
ss - Een 16 bits geheugenlocatie.
n - Een 8 bits waarde (0 t/m 255).
nn - Een 16 bits waarde (0 t/m 65535).

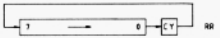
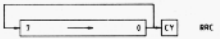
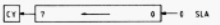
Symbolische omschrijving van Z80-instructies

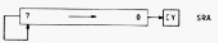
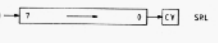
In de hiernavolgende tabel zijn op alfabetische volgorde van mnemonic een aantal aantekeningen opgenomen. Alleen die insrtuctie die meer doen of anders doen dan hun mnemonic doet vermoeden, zijn in de tabel opgenomen.

instructie	symbolische omschrijving
BIT b,t	Z <-- $\overline{b}$ (test bit b op locatie t)
CALL cc,nn	Indien cc=waar dan doorgaan, anders: (SP-1) <-- PCh (SP-2) <-- PCl PC <-- nn
CALL nn	(SP-1) <-- PCh (SP-2) <-- PCl PC <-- nn
CCF	CY <-- CY
CP s	A-s (A blijft ongewijzigd. Alleen de vlaggen worden gezet.)
CPD	A-(HL) HL <-- HL-1 BC <-- BC-1
CPDR	Als CPD, doch wordt herhaald tot A=(HL) of tot BC=0.
CPI	A-(HL) HL <-- HL+1 BC <-- BC-1
CPIR	Als CPI, doch wordt herhaald tot A=(HL) of tot BC=0.
CPL	A <-- $\overline{A}$
DAA	Converteert de inhoud van A naar "packed BCD" indien volgende op een optel- of aftrekinstructie met een "packed BCD"-operand.

instructie	symbolische omschrijving
DI	IFF <-- 0 (IFF = Interrupt Flip Flop.)
DJNZ e	B <-- B-1 Indien B gelijk is aan 0, dan doorgaan. Indien B ongelijk is aan 0, dan PC <-- PC+e.
EI	IFF <-- 1 (IFF = Interrupt Flip Flop.)
EX (SP),HL	H <-- (SP+1) L <-- (SP)
EX (SP),IX	IXh <-- (SP+1) IXl <-- (SP)
IN A,(n)	A <-- (n) n op adreslijnen A0 t/m A7 A op adreslijnen A8 t/m A15
IN r,(C)	r <-- (C) register C op adreslijnen A0 t/m A7 register B op adreslijnen A8 t/m A15 Indien r=110 dan worden alleen vlaggen gezet.
IND	(HL) <-- (C) B <-- B-1 HL <-- HL-1
INDR	Als IND, doch wordt herhaald tot B=0.
INI	(HL) <-- (C) B <-- B-1 HL <-- HL+1
INIR	Als INI, doch wordt herhaald tot B=0.
JP nn	PC <-- nn
JP cc,nn	Indien cc = waar, dan PC <-- nn, anders doorgaan met volgende instructie.
JR e	PC <-- PC+e
JR cc,e	Indien cc niet waar is, dan doorgaan met volgende instructie, anders: PC <-- PC+e (voor deze instructie kan cc alleen C, NC, Z of NZ zijn.)
LD (nn),dd	(nn+1) <-- ddh (nn) <-- ddl

instructie	symbolische omschrijving
LD dd,(nn)	ddh <-- (nn+1) ddl <-- (nn)
LDD	(DE) <-- (HL) DE <-- DE-1 HL <-- HL-1 BC <-- BC-1
LDDR	Als LDD, doch wordt herhaald tot BC=0
LDI	(DE) <-- (HL) DE <-- DE+1 HL <-- HL+1 BC <-- BC-1
LDIR	Als LDI, doch wordt herhaald tot BC=0
NEG	A <-- 0-A
OUTD	(C) <-- (HL) B <-- B-1 HL <-- HL-1
OTDR	Als OUTD, doch wordt herhaald tot B=0.
OUTI	(C) <-- (HL) B <-- B-1 HL <-- HL+1
OTIR	Als OUTI, doch wordt herhaald tot B=0.
POP qq	qqh <-- (SP+1) qql <-- (SP)
PUSH qq	(SP-2) <-- qql (SP-1) <-- qqh
RET	PCl <-- (SP) PCh <-- (SP+1)
RET cc	Indien cc = waar, dan als RET. Anders doorgaan met de volgende instructie.
RL/RLA	
RLC/RLCA	

instructie	symbolische omschrijving
RLD	
RR/RRA	
RRC/RRCA	
RRD	
RST p	(SP-1) <-- PCh (SP-2) <-- PCl PCh <-- 0 PCl <-- p
SCF	CY <-- 1
SLA	

instructie	symbolische omschrijving
SRA	
SRL	

In de voorgaande tabel komen een aantal symbolen voor, waarvan de betekenis hier volgt:

cc	conditie	P	dd	qq	s
000	NZ - non zero	00 hex	BC	BC	r
001	Z - zero	08 hex	DE	DE	n
010	NC - no carry	10 hex	HL	HL	(HL)
011	C - carry	18 hex	IX	AF	(IX+d)
100	PO - pariteit oneven	20 hex	IY	IX	(IY+d)
101	PE - pariteit even	28 hex	SP	IY	
110	P - positief	30 hex			
111	M - negatief	38 hex			

Niet officiële Z80-instructies

De Z80 kent een aantal instructies die niet door de fabrikant worden ondersteund. Het is dan ook niet met enige zekerheid te zeggen of deze instructies op alle Z80 microprocessors dezelfde resultaten geven. Het is zelfs denkbaar dat ze op sommige Z80 microprocessors helemaal niet werken. Daar er echter enkele erg interessante instructies bij zijn, heb ik besloten ze hier toch op te nemen.

hex.code	mnemonic	opmerkingen
CB30	SLL B	bit 0 wordt 1
CB31	SLL C	bit 0 wordt 1
CB32	SLL D	bit 0 wordt 1
CB33	SLL E	bit 0 wordt 1
CB34	SLL H	bit 0 wordt 1
CB35	SLL L	bit 0 wordt 1
CB36	SLL (HL)	bit 0 wordt 1
CB37	SLL A	bit 0 wordt 1
DD24	INC IXh	msb van IX
DD25	DEC IXh	msb van IX
DD2620	LD IXh,n	msb van IX
DD2C	INC IXl	lsb van IX
DD2D	DEC IXl	lsb van IX
DD2E20	LD IXl,n	lsb van IX
DD44	LD B,IXh	msb van IX
DD45	LD B,IXl	lsb van IX
DD4C	LD C,IXh	msb van IX
DD4D	LD C,IXl	lsb van IX
DD54	LD D,IXh	msb van IX
DD55	LD D,IXl	lsb van IX
DD5C	LD E,IXh	msb van IX
DD5D	LD E,IXl	lsb van IX
DD60	LD IXh,B	msb van IX
DD61	LD IXh,C	msb van IX
DD62	LD IXh,D	msb van IX

hex.code	mnemonic		opmerkingen
DD63	LD	IXh,E	msb van IX
DD67	LD	IXh,A	msb van IX
DD68	LD	IXl,B	lsb van IX
DD69	LD	IXl,C	lsb van IX
DD6A	LD	IXl,D	lsb van IX
DD6B	LD	IXl,E	lsb van IX
DD6F	LD	IXl,A	lsb van IX
DD7C	LD	A,IXh	msb van IX
DD7D	LD	A,IXl	lsb van IX
DD84	ADD	A,IXh	msb van IX
DD85	ADD	A,IXl	lsb van IX
DD8C	ADC	A,IXh	msb van IX
DD8D	ADC	A,IXl	lsb van IX
DD94	SUB	A,IXh	msb van IX
DD95	SUB	A,IXl	lsb van IX
DD9C	SBC	A,IXh	msb van IX
DD9D	SBC	A,IXl	lsb van IX
DDA4	AND	A,IXh	msb van IX
DDA5	AND	A,IXl	lsb van IX
DDAC	XOR	A,IXh	msb van IX
DDAD	XOR	A,IXl	lsb van IX
DDB4	OR	A,IXh	msb van IX
DDB5	OR	A,IXl	lsb van IX
DDBC	CP	A,IXh	msb van IX
DDBD	CP	A,IXl	lsb van IX
DDCB0536	SLL	(IX+d)	bit 0 wordt 1
ED70	IN	(HL),(C)	?
ED71	OUT	(C),(HL)	?

Evenals in de officiele lijsten zijn ook hier waarden ingevuld in de hexadecimale code. Voor d is de waarde 05 ingevuld en voor n de waarde 20. Hierdoor krijgen de instructies in deze tabel de werkelijke lengte.
Wees wel voorzichtig met het gebruik van deze instructies. Ze zouden een hang-up kunnen veroorzaken, en daarmee veel intikwerk teniet kunnen doen. Vergeet dus vooral niet uw ingetypte machinetaalroutine eerst naar cassette te schrijven en het daarna pas uit te voeren.

Programma: DECHEX-conversie

Decimale getallen van 0 tot en met 65535 zullen bij conversie een hexadecimaal getal van 4 cijfers opleveren, van 0000 tot en met FFFF.
In dat hexadecimale getal kan ieder cijfer 16 verschillende waarden aannemen. De twee rechter cijfers samen kunnen 16*16=256 verschillende waarden aannemen, terwijl de drie rechter cijfers samen 16*16*16=4096 verschillende waarden kunnen aannemen. Voor het begrijpen van de conversiemethode in het volgende programma is dit belangrijk.
De gevonden waarden worden als volgt in een conversieberekening, waarmee we het decimale getal 35783 willen omzetten naar een hexadecimaal getal, toegepast:

```
35783 : 4096 =  8 rest 3015
 3015 :  256 = 11 rest  199
  199 :   16 = 12 rest    7
```

Het hexadecimale resultaat is dus:

8 11 12 7 = 8B C7.

In regel 40 tot en met 90 van het programma wordt de conversie volgens bovenstaand voorbeeld uitgevoerd. De verkregen waarden staan echter nog in decimale notatie. Ze zijn echter vrij eenvoudig in de hexadecimale notatie om te zetten, al zorgt de ASCII-code er voor dat er nog een extra controle nodig is op de hoogte van het verkregen resultaat.
Om de juiste ASCII-code te krijgen moet er voor de cijfers 48 bij de verkregen waarde worden opgeteld, terwijl er voor de letters 55 bij moet worden opgeteld. Dit wordt in regels 110 en 120 gedaan.

```
  1 REM ************************
  2 REM * Decimale getallen   *
  3 REM * van maximaal 65535  *
  4 REM * worden omgezet naar *
  5 REM * hexadecimale waarden*
  6 REM ************************
  7 REM DECHEXCONV
  8 REM copyright 1983
  9 REM
 10 DIM h(4)
 20 INPUT "Geef decimaal getal
";g
 30 PRINT "dec. ";g;" is","hex.
";
 40 LET h(1)=INT (g/4096)
 50 LET g=g-4096*h(1)
 60 LET h(2)=INT (g/256)
 70 LET g=g-256*h(2)
 80 LET h(3)=INT (g/16)
 90 LET h(4)=g-16*h(3)
100 FOR a=1 TO 4
110 IF h(a)<10 THEN  LET h(a)=h
(a)+48: GO TO 130
120 IF h(a)>=10 THEN  LET h(a)=
h(a)+55
130 PRINT CHR$ h(a);
140 NEXT a
150 PRINT
160 INPUT "Doorgaan? (j/n)";a$
170 IF a$="j" THEN  GO TO 20
180 STOP
```

VOORBEELD

```
dec. 0 is        hex. 0000
dec. 65535 is    hex. FFFF
dec. 127 is      hex. 007F
dec. 128 is      hex. 0080
dec. 384 is      hex. 0180
dec. 23456 is    hex. 5BA0
dec. 12345 is    hex. 3039
```


Programma: Hexadecimale geheugen-dump

Met dit programma kunt u het geheugen van de Spectrum hexadecimaal uitlezen.
Het programma zal u eerst vragen naar het adres van waar af de dump moet beginnen, vervolgens zal het 16 regels van ieder acht geheugenlocaties afdrukken. Iedere regel wordt voorafgegaan door het geheugenadres van de eerste geheugenlocatie uit die regel. Dit adres is decimaal weergegeven. Na 16 regels te hebben afgedrukt vraagt het programma u of het door moet gaan. Indien u met j antwoordt, zullen de 16 volgende regels worden afgedrukt.

Mocht u de adressen liever in hexadecimaal zien afgedrukt, dan raad ik u aan de conversie routine uit het programma "DECHEXCONV" in dit programma op te nemen.

VOORBEELD

```
adres/inhoud
0       F3 AF 11 FF FF C3 CB 11
8       2A 5D 5C 22 5F 5C 18 43
16      C3 F2 15 FF FF FF FF FF
24      2A 5D 5C 7E CD 7D 00 D0
32      CD 74 00 18 F7 FF FF FF
40      C3 5B 33 FF FF FF FF FF
48      C5 2A 61 5C E5 C3 9E 16
56      F5 E5 2A 78 5C 23 22 78
64      5C 7C B5 20 03 FD 34 40
72      C5 D5 CD BF 02 D1 C1 E1
80      F1 FB C9 E1 6E FD 75 00
88      ED 7B 3D 5C C3 C5 16 FF
96      FF FF FF FF FF FF F5 E5
104     2A B0 5C 7C B5 20 01 E9
112     E1 F1 ED 45 2A 5D 5C 23
120     22 5D 5C 7E C9 FE 21 D0
```

```
  1 REM ************************
  2 REM * hexadecimale geheu- *
  3 REM * gendump. U kunt het *
  4 REM * startadres ingeven. *
  5 REM ************************
  6 REM HEXDUMP
  7 REM copyright 1983
  8 REM
  9 REM
 10 DIM h(2)
 20 INPUT "Geef startadres ";s
 30 CLS
 40 PRINT "adres/inhoud"
 50 FOR k=1 TO 16
 60 PRINT s;
 70 FOR r=0 TO 7
 80 LET h(1)=INT (PEEK (s+r)/16
)
 90 LET h(2)=INT (PEEK (s+r))-1
6*h(1)
100 FOR a=1 TO 2
110 IF h(a)<10 THEN  LET h(a)=h
(a)+48: GO TO 130
120 IF h(a)>=10 THEN  LET h(a)=
h(a)+55
130 PRINT AT k,r*3+5+a;CHR$ h(a
)
140 NEXT a
150 NEXT r
160 LET s=s+8
170 NEXT k
180 INPUT "doorgaan? j/n ";a$
190 IF a$="j" THEN  GO TO 30
200 STOP
```


Programma: Hexlader

Dit programma geeft u de mogelijkheid om machinetaalprogramma's hexadecimaal in te geven.
Het vraagt u vanaf welk adres u het machinetaalprogramma geladen wilt hebben en zet vervolgens de RAMTOP op een waarde die juist onder het begin van uw machinetaalprogramma ligt, zodat u niet de kans loopt uw programma vanuit het BASIC te overschrijven.

De letters die u tijdens het ingeven van de hexadecimale getallen gebruikt mogen zowel kleine als hoofdletters zijn. Het programma zorgt er voor dat de juiste waarde naar het geheugen wordt geschreven.
Wilt u de ingave stoppen, geeft u dan in plaats van een normale hexadecimale waarde "**" in. De reactie van het programma zal zijn dat het u vraagt of het ingegeven programma moet worden uitgevoerd.
Een klein foutje in een machinetaalprogramma kan grote gevolgen hebben. De kans op een "hang up" is erg groot. Zet uw machinetaalprogramma altijd eerst op cassette voordat u het gaat uitvoeren.
Op de vraag "programma uitvoeren?" dient u dus pas met j te antwoorden als uw routine op cassette staat.

```
1 REM ************************
2 REM * hexadecimaal laden  *
3 REM * van machinetaal.    *
4 REM * Zowel hoofd- als    *
5 REM * kleine letters mag. *
6 REM * Stoppen met "**".   *
7 REM * RAMTOP := startadres*
8 REM ************************
```

```
  9 REM hexlader
 10 REM copyright 1983
 11 REM
 12 REM
 20 INPUT "Geef startadres ";s
 30 LET RAMTOP=s-1
 40 CLEAR RAMTOP
 50 DIM h$(2): CLS
 60 LET s=PEEK 23730+PEEK 23731
*256+1
 70 FOR a=s TO 65535
 80 PRINT "byte op adres ";a;"
= ";
 90 INPUT h$
100 IF h$="**" THEN  GO TO 230
110 LET msb=CODE h$(1)
120 LET lsb=CODE h$(2)
130 IF msb>47 AND msb<58 THEN
LET msb=msb-48
140 IF msb>64 AND msb<71 THEN
LET msb=msb-55
150 IF msb>96 AND msb<103 THEN
LET msb=msb-87
160 IF lsb>47 AND lsb<58 THEN
LET lsb=lsb-48
170 IF lsb>64 AND lsb<71 THEN
LET lsb=lsb-55
180 IF lsb>96 AND lsb<103 THEN
LET lsb=lsb-87
190 IF lsb>15 OR msb>15 THEN  G
O TO 90
200 POKE a,msb*16+lsb
210 PRINT h$
220 NEXT a
230 INPUT "programma uitvoeren?
";a$
240 IF a$="j" THEN  PRINT 'USR
s
250 IF a$<>"j" THEN  STOP
260 PRINT '"Machinetaalroutine
uitgevoerd."
270 INPUT "nog eens uitvoeren?"
;a$
280 GO TO 240
```


VOORBEELD

```
byte op adres 40000 = 00
byte op adres 40001 = 00
byte op adres 40002 = 11
byte op adres 40003 = 22
byte op adres 40004 = 33
byte op adres 40005 = 44
byte op adres 40006 = 55
byte op adres 40007 = 66
byte op adres 40008 = 77
byte op adres 40009 = 88
byte op adres 40010 = 99
byte op adres 40011 = Aa
byte op adres 40012 = bB
byte op adres 40013 = Cc
byte op adres 40014 = dD
byte op adres 40015 = Ee
byte op adres 40016 = fF
byte op adres 40017 =
```

Programma: ROM-test

Mocht u uw Spectrum er van verdenken niet helemaal goed te werken, dan is er voor het testen van de ROM, waarin de BASIC-interpreter en het operating-systeem zich bevinden, een zeer eenvoudige test mogelijk.
Het hierna volgende programma telt gewoon de inhoud van alle geheugenadressen van de ROM bij elkaar op en controleert het verkregen resultaat met een waarde die zou moeten worden verkregen als de ROM in orde is.
Er zijn verschillende versies van de Spectrum in omloop. Bovendien is het niet de gewoonte van Sinclair om bekend te maken of er wijzigingen in de geproduceerde apparatuur zijn aangebracht. Het zou dus mogelijk kunnen zijn dat de in dit programma opgenomen testwaarde niet dezelfde is als die van uw eigen Spectrum.
Mocht dit programma bij de eerste keer dat u het draait geen goed resultaat geven, verandert u dan regel 50 in:

```
50 PRINT totaal
```

Indien het resultaat daarvan niet 1926175 is dan is het mogelijk dat u een ouder of nieuwer type ZX Spectrum heeft dan de machine waarop het totaal uit dit programma was verkregen.


```
  1 REM **********************
  2 REM * ROMTEST-programma   *
  3 REM * Deze test duurt     *
  4 REM * ongeveer 1 minuut   *
  5 REM * per 16K geheugen    *
  6 REM **********************
  7 REM ROMTEST
  8 REM copyright 1983
  9 REM
 10 LET totaal=0
 20 FOR a=0 TO 16383
 30 LET totaal=totaal+PEEK a
 40 NEXT a
 50 IF totaal=1926175 THEN  PRI
NT "standaard Spectrum ROM OK"
 60 STOP
```

Programma: UDG-grootte

De ZX Spectrum heeft slechts 21 user defined graphics. Deze karakters staan in RAM, onmiddellijk volgend op RAMTOP. Zou men echter meer of minder user defined graphics wensen, dan is dit vrij eenvoudig te realiseren.

De systeemvariabele UDG geeft het adres van het begin van het eerste user defined graphic aan. Door UDG te wijzigen kan men meer of minder RAM reserveren voor user defined graphics.

Om te voorkomen dat tijdens het werken met een vergroot UDG-geheugen de user defined graphics door het BASIC-systeem worden overschreven, doet men er goed aan de RAMTOP vlak voor het gebied met de user defined graphics te zetten.

Het volgende programma laat zien hoe het aantal user defined graphics kan worden gewijzigd. Nadat het programma u heeft gevraagd hoeveel graphics u wenst wordt de systeemvariabele UDG overeenkomstig uw wens gezet. Daarna laat het programma u de oude RAMTOP en de bij het door u opgegeven aantal graphics gewenste nieuwe RAMTOP.

Nu kunt u beslissen of u de RAMTOP wilt aanpassen. Het programma zal uw beslissing uitvoeren en het resultaat aan u laten zien.

```
  1 REM ************************
  2 REM * Wijzigen van de user*
  3 REM * defined graphics     *
  4 REM * area + automatisch   *
  5 REM * aanpassen van RAMTOP*
  6 REM ************************
  7 REM UDG
  8 REM copyright 1983
  9 REM
 10 PRINT "Gewenst aantal UDG's
? ";
 20 INPUT n
 30 PRINT n
 40 LET UDG=65536-n*8
 50 POKE 23676,INT (UDG/256)
 60 POKE 23675,UDG-INT (UDG/256
)*256
 70 LET RAMTOP=UDG-1
 80 PRINT "Oude RAMTOP = ";PEEK
 23730+256*PEEK 23731
 90 PRINT "RAMTOP zou     ";RAMT
OP;" moeten worden."''"wijzigen?
(j/n)"
100 INPUT a$
110 IF a$<>"j" THEN  GO TO 130
120 CLEAR RAMTOP
130 PRINT "Nieuwe RAMTOP = ";PE
EK 23730+256*PEEK 23731
140 PRINT "Nieuwe UDG    = ";PE
EK 23675+256*PEEK 23676
150 STOP
```

Programma: VARLIST

Dit programma gebruikt zelf programmaregels vanaf regelnummer 9900. Indien u er voor zorgt dat uw eigen programma's geen regelnummers van 9900 of hoger gebruiken, dan kunt u dit programma met behulp van MERGE met uw eigen programma samenvoegen. U kunt dan op elk moment dat u dat wenst een overzicht verkrijgen van de in uw programma gebruikte variabelen.

Het programma VARLIST maakt namelijk een lijst van de gebruikte variabelen vanuit het variabelengeheugen. In deze lijst komen behalve de variabelenamen ook de daarbij behorende typen voor. Hierdoor bent u in staat om snel te zien of u een variabelenaam voor een bepaald type variabele al had gebruikt of nog niet.

Daarnaast is dit programma een voorbeeld van het in de praktijk gebruiken van de informatie die in het hoofdstuk over het variabelengeheugen werd gegeven.

In programmaregel 9957 wordt het beginadres van het variabelen geheugen uit de systeemvariabele VARS gelezen.

In de subroutine die op regel 9967 begint wordt uitgezocht van wat voor type de variabele is. Afhankelijk van het gevonden type wordt daarna de gehele naam en de totale lengte van de variabele uitgezocht. Is de lengt eenmaal gevonden, dan is ook het begin van de volgende variabele gevonden. Indien op dat beginadres van de volgende variabele echter een decimale waarde 128 (eindindicator) staat, dan is het eind van het variabelen geheugen gevonden.


```
9940 REM ************************
9941 REM * type 1 = string var *
9942 REM * type 2 = num.var 11 *
9943 REM * type 3 = num.array  *
9944 REM * type 4 = num.var >11*
9945 REM * type 5 = str. array *
9946 REM * type 6 = FORNEXT-var*
9947 REM ************************
9948 REM
9949 REM copyright 1983
9950 REM programma "varlist"
9951 REM
9952 REM
9957 LET pointer=PEEK 23627+256*
PEEK 23628
9958 LET var=PEEK pointer
9959 GO SUB 9967
9960 IF type=1 THEN  GO SUB 9985
9961 IF type=2 THEN  GO SUB 9990
9962 IF type=3 THEN  GO SUB 9977
9963 IF type=4 THEN  GO SUB 9993
9964 IF type=5 THEN  GO SUB 9975
9965 IF type=6 THEN  GO SUB 9982
9966 GO TO 9958
9967 IF var<=95 THEN  LET type=1
9968 IF var>95 AND var<=127 THEN
  LET type=2
9969 IF var>127 AND var<=159 THE
N  LET type=3
9970 IF var>159 AND var<=191 THE
N  LET type=4
9971 IF var>191 AND var<=223 THE
N  LET type=5
9972 IF var>223 THEN  LET type=6
9973 IF var=128 THEN  PRINT '"ei
nde variabelengeheugen": STOP
9974 RETURN
9975 PRINT "string array",CHR$ (
var-96);"$"
9976 GO TO 9978
9977 PRINT "num array",CHR$ (var
-32)
9978 LET pointer=pointer+1
9979 LET len=PEEK pointer+256*PE
EK (pointer+1)
```

```
9980 LET pointer=pointer+len+2
9981 RETURN
9982 PRINT "FOR NEXT var",CHR$ (
var-128)
9983 LET pointer=pointer+19
9984 RETURN
9985 PRINT "string var",CHR$ (va
r+32);"$"
9986 LET pointer=pointer+1
9987 LET len=PEEK pointer+256*PE
EK (pointer+1)
9988 LET pointer=pointer+len+2
9989 RETURN
9990 PRINT "num var ",CHR$ var
9991 LET pointer=pointer+6
9992 RETURN
9993 PRINT "num var ",CHR$ (var-
64);
9994 LET pointer=pointer+1
9995 LET var=PEEK pointer
9996 IF var<128 THEN  PRINT CHR$
 var;: GO TO 9994
9997 IF var>=128 THEN  PRINT CHR
$ (var-128)
9998 LET pointer=pointer+6
9999 RETURN
```

VOORBEELD

```
num var          x
num var          z
string var       a$
FOR NEXT var     b
string array     c$
num array        d
num var          pointer
num var          var
num var          type
num var          len

einde variabelengeheugen
```


Aantekeningen

Aantekeningen