MSX

# MSX ZAKBOEKJE

Wessel Akkermans


*uw* **MSX** *computer de baas*

ZAKBOEKJE
Wessel Akkermans

# MSX ZAKBOEKJE

## uw MSX computer de baas

# MSX

# ZAKBOEKJE

*uw* **MSX** *computer*
*de baas*

Wessel Akkermans

**uitgeverij STARK - TEXEL**
postbus 302 - 1794 ZG Oosterend tel. 02223 - 661

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Akkermans, Wessel

MSX zakboekje : uw MSX computer de baas / Wessel Akkermans. – Oosterend : Stark-Texel
ISBN 90-6398-888-5
SISO 365.3 UDC 681.3.06
Trefw.: microcomputers ; programmeren.

.....

1e druk 1985
ISBN 90 6398 888 5

# Voorwoord

Tekeningen maken, kleuren en geluiden produceren, en dat met hetzelfde gemak als het afdrukken van een stukje tekst op het beeldscherm. Dat was mijn eerste indruk van de MSX-computers. Na die eerste kennismaking bleven de MSX-computers mij verbazen. Een floppy disk van merk A kon zonder meer worden aangesloten op merk B. Diezelfde mate van uitwisselbaarheid gold ook voor programma's. Programma's geschreven op de ene MSX-computer, konden ongewijzigd worden afgedraaid op een andere.

De zeer uitgebreide en goed gestandaardiseerde MSX-BASIC, en de al evengoed gestandaardiseerde hardware, hebben het schrijven van dit boekje tot een werkelijk plezier gemaakt. Al werkende (misschien is het woord "spelende" beter op zijn plaats) kwam ik tot een aantal aantekeningen van al die zaken die mij in het begin niet helemaal duidelijk waren. Deze aantekeningen, gecombineerd met een aantal Z80-microprocessor gegevens en gecompleteerd met een beschrijving van de BASIC-instructie set liggen nu voor u in de vorm van een zakboekje.

Door de informatie in een aantal soorten te groeperen, hoop ik een handig naslagwerkje te hebben gecreeerd, waarmee u nog vele jaren uw voordeel kunt doen. De volgende groepen zijn opgenomen, in de volgorde waarin ze hier worden vermeld:

Algemene informatie, niet computergericht.
MSX-BASIC informatie.
Machinetaal informatie.
Hardware informatie.
Een aantal korte programmaatjes.

Mijn hartelijke dank gaat uit naar de firma "Tekelec Airtronic" uit Zoetermeer, de importeur van de Z80 microprocessor van ZILOG, voor haar hulp bij het verkrijgen van toestemming tot het overnemen van gegevens uit het Z80 CPU Manual.

februari 1985
Wessel Akkermans

# INHOUD


| | | |
|---|---|---|
| 1 | Conversies: decimaal, binair, octaal en hexadecimaal | 9 |
| 2 | Machten van 2, 8 en 16 | 12 |
| 3 | Afgeleide goniometrische functies | 13 |
| 4 | Operators | 14 |
| 5 | Kleurnummers | 15 |
| 6 | Variabelen en waardetoekenning | 16 |
| 7 | MSX-BASIC instructieset | 18 |
| 8 | Systeemboodschappen | 70 |
| 9 | Editing | 77 |
| 10 | MSX-karakterset | 79 |
| 11 | ASCII-karakterset | 80 |
| 12 | Geheugen lay-out | 81 |
| 13 | I/O-adressen | 82 |
| 14 | De programmeerbare geluidsgenerator | 83 |
| 15 | De video display processor | 89 |
| 16 | Z80 interrupt modes | 94 |
| 17 | Z80-registers | 95 |
| 18 | Symbolische omschrijving van Z80-instructies | 97 |
| 19 | Z80-instructieset op volgorde van mnemonics | 104 |
| 20 | Z80-instructieset op volgorde van hexcode | 113 |
| 21 | Z80-vlagbeinvloeding | 122 |
| 22 | Connectoren | 124 |
| 23 | Geheugen-dump | 126 |
| 24 | Inhoudsopgave van de schijf | 132 |
| | Aantekeningen | 134 |

# 1 Conversies: decimaal, binair, octaal en hexadecimaal

| dec. | binair | oct. | hex. | dec. | binair | oct. | hex. |
|---|---|---|---|---|---|---|---|
| 0 | 0000 0000 | 000 | 00 | 42 | 0010 1010 | 052 | 2A |
| 1 | 0000 0001 | 001 | 01 | 43 | 0010 1011 | 053 | 2B |
| 2 | 0000 0010 | 002 | 02 | 44 | 0010 1100 | 054 | 2C |
| 3 | 0000 0011 | 003 | 03 | 45 | 0010 1101 | 055 | 2D |
| 4 | 0000 0100 | 004 | 04 | 46 | 0010 1110 | 056 | 2E |
| 5 | 0000 0101 | 005 | 05 | 47 | 0010 1111 | 057 | 2F |
| 6 | 0000 0110 | 006 | 06 | 48 | 0011 0000 | 060 | 30 |
| 7 | 0000 0111 | 007 | 07 | 49 | 0011 0001 | 061 | 31 |
| 8 | 0000 1000 | 010 | 08 | 50 | 0011 0010 | 062 | 32 |
| 9 | 0000 1001 | 011 | 09 | 51 | 0011 0011 | 063 | 33 |
| 10 | 0000 1010 | 012 | 0A | 52 | 0011 0100 | 064 | 34 |
| 11 | 0000 1011 | 013 | 0B | 53 | 0011 0101 | 065 | 35 |
| 12 | 0000 1100 | 014 | 0C | 54 | 0011 0110 | 066 | 36 |
| 13 | 0000 1101 | 015 | 0D | 55 | 0011 0111 | 067 | 37 |
| 14 | 0000 1110 | 016 | 0E | 56 | 0011 1000 | 070 | 38 |
| 15 | 0000 1111 | 017 | 0F | 57 | 0011 1001 | 071 | 39 |
| 16 | 0001 0000 | 020 | 10 | 58 | 0011 1010 | 072 | 3A |
| 17 | 0001 0001 | 021 | 11 | 59 | 0011 1011 | 073 | 3B |
| 18 | 0001 0010 | 022 | 12 | 60 | 0011 1100 | 074 | 3C |
| 19 | 0001 0011 | 023 | 13 | 61 | 0011 1101 | 075 | 3D |
| 20 | 0001 0100 | 024 | 14 | 62 | 0011 1110 | 076 | 3E |
| 21 | 0001 0101 | 025 | 15 | 63 | 0011 1111 | 077 | 3F |
| 22 | 0001 0110 | 026 | 16 | 64 | 0100 0000 | 100 | 40 |
| 23 | 0001 0111 | 027 | 17 | 65 | 0100 0001 | 101 | 41 |
| 24 | 0001 1000 | 030 | 18 | 66 | 0100 0010 | 102 | 42 |
| 25 | 0001 1001 | 031 | 19 | 67 | 0100 0011 | 103 | 43 |
| 26 | 0001 1010 | 032 | 1A | 68 | 0100 0100 | 104 | 44 |
| 27 | 0001 1011 | 033 | 1B | 69 | 0100 0101 | 105 | 45 |
| 28 | 0001 1100 | 034 | 1C | 70 | 0100 0110 | 106 | 46 |
| 29 | 0001 1101 | 035 | 1D | 71 | 0100 0111 | 107 | 47 |
| 30 | 0001 1110 | 036 | 1E | 72 | 0100 1000 | 110 | 48 |
| 31 | 0001 1111 | 037 | 1F | 73 | 0100 1001 | 111 | 49 |
| 32 | 0010 0000 | 040 | 20 | 74 | 0100 1010 | 112 | 4A |
| 33 | 0010 0001 | 041 | 21 | 75 | 0100 1011 | 113 | 4B |
| 34 | 0010 0010 | 042 | 22 | 76 | 0100 1100 | 114 | 4C |
| 35 | 0010 0011 | 043 | 23 | 77 | 0100 1101 | 115 | 4D |
| 36 | 0010 0100 | 044 | 24 | 78 | 0100 1110 | 116 | 4E |
| 37 | 0010 0101 | 045 | 25 | 79 | 0100 1111 | 117 | 4F |
| 38 | 0010 0110 | 046 | 26 | 80 | 0101 0000 | 120 | 50 |
| 39 | 0010 0111 | 047 | 27 | 81 | 0101 0001 | 121 | 51 |
| 40 | 0010 1000 | 050 | 28 | 82 | 0101 0010 | 122 | 52 |
| 41 | 0010 1001 | 051 | 29 | 83 | 0101 0011 | 123 | 53 |

| dec. | binair | oct. | hex. | dec. | binair | oct. | hex. |
|---|---|---|---|---|---|---|---|
| 84 | 0101 0100 | 124 | 54 | 130 | 1000 0010 | 202 | 82 |
| 85 | 0101 0101 | 125 | 55 | 131 | 1000 0011 | 203 | 83 |
| 86 | 0101 0110 | 126 | 56 | 132 | 1000 0100 | 204 | 84 |
| 87 | 0101 0111 | 127 | 57 | 133 | 1000 0101 | 205 | 85 |
| 88 | 0101 1000 | 130 | 58 | 134 | 1000 0110 | 206 | 86 |
| 89 | 0101 1001 | 131 | 59 | 135 | 1000 0111 | 207 | 87 |
| 90 | 0101 1010 | 132 | 5A | 136 | 1000 1000 | 210 | 88 |
| 91 | 0101 1011 | 133 | 5B | 137 | 1000 1001 | 211 | 89 |
| 92 | 0101 1100 | 134 | 5C | 138 | 1000 1010 | 212 | 8A |
| 93 | 0101 1101 | 135 | 5D | 139 | 1000 1011 | 213 | 8B |
| 94 | 0101 1110 | 136 | 5E | 140 | 1000 1100 | 214 | 8C |
| 95 | 0101 1111 | 137 | 5F | 141 | 1000 1101 | 215 | 8D |
| 96 | 0110 0000 | 140 | 60 | 142 | 1000 1110 | 216 | 8E |
| 97 | 0110 0001 | 141 | 61 | 143 | 1000 1111 | 217 | 8F |
| 98 | 0110 0010 | 142 | 62 | 144 | 1001 0000 | 220 | 90 |
| 99 | 0110 0011 | 143 | 63 | 145 | 1001 0001 | 221 | 91 |
| 100 | 0110 0100 | 144 | 64 | 146 | 1001 0010 | 222 | 92 |
| 101 | 0110 0101 | 145 | 65 | 147 | 1001 0011 | 223 | 93 |
| 102 | 0110 0110 | 146 | 66 | 148 | 1001 0100 | 224 | 94 |
| 103 | 0110 0111 | 147 | 67 | 149 | 1001 0101 | 225 | 95 |
| 104 | 0110 1000 | 150 | 68 | 150 | 1001 0110 | 226 | 96 |
| 105 | 0110 1001 | 151 | 69 | 151 | 1001 0111 | 227 | 97 |
| 106 | 0110 1010 | 152 | 6A | 152 | 1001 1000 | 230 | 98 |
| 107 | 0110 1011 | 153 | 6B | 153 | 1001 1001 | 231 | 99 |
| 108 | 0110 1100 | 154 | 6C | 154 | 1001 1010 | 232 | 9A |
| 109 | 0110 1101 | 155 | 6D | 155 | 1001 1011 | 233 | 9B |
| 110 | 0110 1110 | 156 | 6E | 156 | 1001 1100 | 234 | 9C |
| 111 | 0110 1111 | 157 | 6F | 157 | 1001 1101 | 235 | 9D |
| 112 | 0111 0000 | 160 | 70 | 158 | 1001 1110 | 236 | 9E |
| 113 | 0111 0001 | 161 | 71 | 159 | 1001 1111 | 237 | 9F |
| 114 | 0111 0010 | 162 | 72 | 160 | 1010 0000 | 240 | A0 |
| 115 | 0111 0011 | 163 | 73 | 161 | 1010 0001 | 241 | A1 |
| 116 | 0111 0100 | 164 | 74 | 162 | 1010 0010 | 242 | A2 |
| 117 | 0111 0101 | 165 | 75 | 163 | 1010 0011 | 243 | A3 |
| 118 | 0111 0110 | 166 | 76 | 164 | 1010 0100 | 244 | A4 |
| 119 | 0111 0111 | 167 | 77 | 165 | 1010 0101 | 245 | A5 |
| 120 | 0111 1000 | 170 | 78 | 166 | 1010 0110 | 246 | A6 |
| 121 | 0111 1001 | 171 | 79 | 167 | 1010 0111 | 247 | A7 |
| 122 | 0111 1010 | 172 | 7A | 168 | 1010 1000 | 250 | A8 |
| 123 | 0111 1011 | 173 | 7B | 169 | 1010 1001 | 251 | A9 |
| 124 | 0111 1100 | 174 | 7C | 170 | 1010 1010 | 252 | AA |
| 125 | 0111 1101 | 175 | 7D | 171 | 1010 1011 | 253 | AB |
| 126 | 0111 1110 | 176 | 7E | 172 | 1010 1100 | 254 | AC |
| 127 | 0111 1111 | 177 | 7F | 173 | 1010 1101 | 255 | AD |
| 128 | 1000 0000 | 200 | 80 | 174 | 1010 1110 | 256 | AE |
| 129 | 1000 0001 | 201 | 81 | 175 | 1010 1111 | 257 | AF |

| dec. | binair | oct. | hex. | dec. | binair | oct. | hex. |
|---|---|---|---|---|---|---|---|
| 176 | 1011 0000 | 260 | B0 | 222 | 1101 1110 | 336 | DE |
| 177 | 1011 0001 | 261 | B1 | 223 | 1101 1111 | 337 | DF |
| 178 | 1011 0010 | 262 | B2 | 224 | 1110 0000 | 340 | E0 |
| 179 | 1011 0011 | 263 | B3 | 225 | 1110 0001 | 341 | E1 |
| 180 | 1011 0100 | 264 | B4 | 226 | 1110 0010 | 342 | E2 |
| 181 | 1011 0101 | 265 | B5 | 227 | 1110 0011 | 343 | E3 |
| 182 | 1011 0110 | 266 | B6 | 228 | 1110 0100 | 344 | E4 |
| 183 | 1011 0111 | 267 | B7 | 229 | 1110 0101 | 345 | E5 |
| 184 | 1011 1000 | 270 | B8 | 230 | 1110 0110 | 346 | E6 |
| 185 | 1011 1001 | 271 | B9 | 231 | 1110 0111 | 347 | E7 |
| 186 | 1011 1010 | 272 | BA | 232 | 1110 1000 | 350 | E8 |
| 187 | 1011 1011 | 273 | BB | 233 | 1110 1001 | 351 | E9 |
| 188 | 1011 1100 | 274 | BC | 234 | 1110 1010 | 352 | EA |
| 189 | 1011 1101 | 275 | BD | 235 | 1110 1011 | 353 | EB |
| 190 | 1011 1110 | 276 | BE | 236 | 1110 1100 | 354 | EC |
| 191 | 1011 1111 | 277 | BF | 237 | 1110 1101 | 355 | ED |
| 192 | 1100 0000 | 300 | C0 | 238 | 1110 1110 | 356 | EE |
| 193 | 1100 0001 | 301 | C1 | 239 | 1110 1111 | 357 | EF |
| 194 | 1100 0010 | 302 | C2 | 240 | 1111 0000 | 360 | F0 |
| 195 | 1100 0011 | 303 | C3 | 241 | 1111 0001 | 361 | F1 |
| 196 | 1100 0100 | 304 | C4 | 242 | 1111 0010 | 362 | F2 |
| 197 | 1100 0101 | 305 | C5 | 243 | 1111 0011 | 363 | F3 |
| 198 | 1100 0110 | 306 | C6 | 244 | 1111 0100 | 364 | F4 |
| 199 | 1100 0111 | 307 | C7 | 245 | 1111 0101 | 365 | F5 |
| 200 | 1100 1000 | 310 | C8 | 246 | 1111 0110 | 366 | F6 |
| 201 | 1100 1001 | 311 | C9 | 247 | 1111 0111 | 367 | F7 |
| 202 | 1100 1010 | 312 | CA | 248 | 1111 1000 | 370 | F8 |
| 203 | 1100 1011 | 313 | CB | 249 | 1111 1001 | 371 | F9 |
| 204 | 1100 1100 | 314 | CC | 250 | 1111 1010 | 372 | FA |
| 205 | 1100 1101 | 315 | CD | 251 | 1111 1011 | 373 | FB |
| 206 | 1100 1110 | 316 | CE | 252 | 1111 1100 | 374 | FC |
| 207 | 1100 1111 | 317 | CF | 253 | 1111 1101 | 375 | FD |
| 208 | 1101 0000 | 320 | D0 | 254 | 1111 1110 | 376 | FE |
| 209 | 1101 0001 | 321 | D1 | 255 | 1111 1111 | 377 | FF |
| 210 | 1101 0010 | 322 | D2 | | | | |
| 211 | 1101 0011 | 323 | D3 | | | | |
| 212 | 1101 0100 | 324 | D4 | | | | |
| 213 | 1101 0101 | 325 | D5 | | | | |
| 214 | 1101 0110 | 326 | D6 | | | | |
| 215 | 1101 0111 | 327 | D7 | | | | |
| 216 | 1101 1000 | 330 | D8 | | | | |
| 217 | 1101 1001 | 331 | D9 | | | | |
| 218 | 1101 1010 | 332 | DA | | | | |
| 219 | 1101 1011 | 333 | DB | | | | |
| 220 | 1101 1100 | 334 | DC | | | | |
| 221 | 1101 1101 | 335 | DD | | | | |

# 2 Machten van 2, 8 en 16

Machten van 2:

```
2^0  =          1
2^1  =          2
2^2  =          4
2^3  =          8
2^4  =         16
2^5  =         32
2^6  =         64
2^7  =        128
2^8  =        256
2^9  =        512
2^10 =       1024
2^11 =       2048
2^12 =       4096
2^13 =       8192
2^14 =      16384
2^15 =      32768
2^16 =      65536
2^17 =     131072
2^18 =     262144
2^19 =     524288
2^20 =    1048576
2^21 =    2097152
2^22 =    4194304
2^23 =    8388608
2^24 =   16777216
2^25 =   33554432
2^26 =   67108864
2^27 =  134217728
2^28 =  268435456
2^29 =  536870912
2^30 = 1073741824
2^31 = 2147483648
2^32 = 4294967296
```

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
```

Machten van 16:

```
16^0 =          1
16^1 =         16
16^2 =        256
16^3 =       4096
16^4 =      65536
16^5 =    1048576
16^6 =   16777216
16^7 =  268435456
16^8 = 4294967296
```

# 3 Afgeleide goniometrische functies

| Afgeleide functie | MSX-notatie |
|---|---|
| Secans | 1/COS(X) |
| Cosecans | 1/SIN(X) |
| Cotangens | 1/TAN(X) |
| Inverse sinus | ATN(X/SQR(-X*X+1)) |
| Inverse cosinus | -ATN(X/SQR(-X*X+1))+1.5708 |
| Inverse secans | ATN(X/SQR(X*X-1))+SGN(SGN(X)-1)<br>*1.5708 |
| Inverse cosecans | ATN(X/SQR(X*X-1))+(SGN(X)-1)<br>*1.5708 |
| Inverse cotangens | ATN(X)+1.5708 |
| Hyperbolische sinus | (EXP(X)-EXP(-X))/2 |
| Hyperbolische cosinus | (EXP(X)+EXP(-X))/2 |
| Hyperbolische tangens | EXP(-X)/EXP(X)+EXP(-X)*2+1 |
| Hyperbolische secans | 2/(EXP(X)+EXP(-X)) |
| Hyperbolische cosecans | 2/(EXP(X)-EXP(-X)) |
| Hyperbolische cotangens | EXP(-X)/(EXP(X)-EXP(-X))*2+1 |
| Inv. hyperb. sinus | LOG(X+SQR(X*X+1)) |
| Inv. hyperb. cosinus | LOG(X+SQR(X*X-1)) |
| Inv. hyperb. tangens | LOG((1+X)/(1-X))/2 |
| Inv. hyperb. secans | LOG((SQR(-X*X+1)+1)/X) |
| Inv. hyperb. cosecans | LOG((SGN(X)*SQR(X*X+1)+1)/X) |
| Inv. hyperb. cotangens | LOG((X+1)/(X-1))/2 |

# 4 Operators

Functionele operators

| Prioriteit | Operator | Betekenis |
|---|---|---|
| 1 | functies | Wanneer in expressies functies worden gebruikt, dan zullen deze met de hoogste prioriteit worden verwerkt. |

Rekenkundige operators

| Prioriteit | Operator | Betekenis |
|---|---|---|
| 2 | ^ | Machtsverheffen |
| 3 | * | vermenigvuldigen |
| | / | delen |
| | \ | delen van gehele getallen |
| | MOD | rest bepaling |
| 4 | + | optellen |
| | - | aftrekken |

Vergelijkende operators

| Prioriteit | Operator | Betekenis |
|---|---|---|
| 5 | = | Is gelijk aan |
| | < | Is kleiner dan |
| | > | Is groter dan |
| | <> of >< | Is ongelijk aan |
| | <= of =< | Is kleiner dan of gelijk aan |
| | >= of => | Is groter dan of gelijk aan |

Logische operators

| Prioriteit | Operator | Betekenis |
|---|---|---|
| 6 | AND | EN |
| | OR | OF |
| | NOT | NIET |
| | XOR | Exclusieve OF |
| | EQV | Gelijkwaardig met |
| | IMP | Impliceert dat |

# 5 Kleurnummers

| Kleurnummer | Kleur |
|---|---|
| 0 | transparant |
| 1 | zwart |
| 2 | groen |
| 3 | licht groen |
| 4 | donker blauw |
| 5 | licht blauw |
| 6 | donker rood |
| 7 | blauw |
| 8 | rood |
| 9 | licht rood |
| 10 | donker geel |
| 11 | licht geel |
| 12 | donker groen |
| 13 | steen rood |
| 14 | grijs |
| 15 | wit |

# 6 Variabelen en waardetoekenning

Soorten variabelen:

| soort | soortbepaling | waardetoekenning |
|---|---|---|
| Alfanumeriek | $ achter naam | LET N$="ABC" |
| Integer numeriek | % achter naam | LET A%=5<br>De toe te kennen waarde mag -32768 tot +32767 zijn, en moet een geheel getal zijn. |
| Numeriek met enkelvoudige nauwkeurigheid | ! achter naam | LET A!=123456<br>De toe te kennen waarde mag uit 6 cijfers bestaan. Daarboven wordt het getal afgerond. |
| Numeriek met dubbele nauwkeurigheid | # achter naam | LET A#=1234567890123<br>De toe te kennen waarde mag uit 14 cijfers bestaan. Daarboven wordt het getal afgerond. |

Indien een variabelenaam niet wordt gevolgd door $, %, ! of #, dan zal die variabele als een numerieke variabele met dubbele nauwkeurigheid worden beschouwd.

Soorten constanten:

| soort | omschrijving |
|---|---|
| Integer | Waarde tussen -32768 en +32767. |
| Fixed point met enkele nauwkeurigheid | Waarde wordt gevolgd door het ! teken.<br>Voorbeeld: 12.3! |

(vervolg soorten constanten)

| soort | omschrijving |
|---|---|
| Fixed point met dubbele nauwkeurigheid | Waarde wordt gevolgd door het # teken. Voorbeeld: 12.3# |
| Floating point met enkele nauwkeurigheid | Waarde wordt in wetenschappelijke notatie geschreven, waarbij E het exponentteken is. Voorbeeld: 12.3E4 |
| Floating point met dubbele nauwkeurigheid | Waarde wordt in wetenschappelijke notatie geschreven, waarbij D het exponentteken is. Voorbeeld: 12.3D9 |
| Hexadecimaal | Waarde wordt voorafgegaan door &H. Voorbeeld: &HFF (= decimaal 255). |
| Octaal | Waarde wordt voorafgegaan door &O. Voorbeeld: &O377 (= decimaal 255). |
| Binair | Waarde wordt voorafgegaan door &B. Voorbeeld: &B11111111 (= decimaal 255). |

# 7 MSX-BASIC instructieset

In dit hoofdstuk zullen alle BASIC-statements, functies en commando's in een alfabetische lijst worden behandeld. Iedere beschrijving zal uit de volgende onderdelen bestaan:

- de naam
- de categorie (functie, commando of statement)
- de syntax
- een korte omschrijving
- zonodig een voorbeeld

In de syntax-beschrijving zullen de volgende symbolen worden gebruikt:

| | |
|---|---|
| <item> | items tussen gehoekte haakjes moeten door de programmeur worden bepaald. |
| [optie] | opties staan tussen vierkante haken en mogen worden weggelaten. |
| (<item>...) | items tussen normale haakjes die door een aantal punten worden gevolgd, mogen zo vaak worden herhaald als de regellengte toestaat. |
| .,():; | deze leestekens moeten precies zoals ze in de syntax beschrijving voorkomen worden ingetikt. |
| Hoofdletters | In hoofdletters geschreven items moeten precies zo worden ingetikt als ze in de syntax-beschrijving zijn aangegeven. |
| X, Y en Z | Hiermee worden numerieke variabelen aangegeven. |
| X$, Y$ en Z$ | Hiermee worden alfanumerieke variabelen aangegeven. |

ABS functie

ABS(<X>)

Neemt de absolute waarde van het getal in variabele X. Indien in X een negatief getal staat, zal het resultaat van ABS(<X>) datzelfde getal, maar dan positief, zijn.

Voorbeeld: ABS(-512) = 512

ASC functie

ASC(<X$>)

Geeft de ASCII-karaktercode van het eerste in X$ voorkomende karakter. Indien X$ geen karakters bevat, zal de foutboodschap "Illegal function call" worden gegeven.

```
Voorbeeld:  LET X$="abc"
            PRINT ASC(X$)        --> 97
            PRINT CHR$(ASC(X$)) --> a
```

ATN functie

ATN(<X>)

Geeft de arctangens van X in radialen.

AUTO commando

AUTO [<X>[,[<Y>]]]

Na het geven van dit commando worden bij het programmeren automatisch regelnummers gegenereerd, beginnende bij regelnummer X, en met een ophoging Y.

AUTO zonder verdere parameters:
Regelnummering begint bij 10, stapgrootte is 10.

AUTO met alleen parameter X:
Regelnummering begint bij X, stapgrootte is 10.

AUTO met parameter X gevolgd door komma:
Regelnummering start bij X, stapgrootte is de laatst gebruikte stapgrootte.

AUTO met alle parameters:
Regelnummering start bij X, stapgrootte is Y.

Indien met AUTO een regelnummer wordt gegenereerd die al in het geheugen aanwezig is, zal het regelnummer onmiddellijk worden gevolgd door een "*".

Automatisch regelnummeren kan worden beeindigd, door de toetsen CONTROL & C of CONTROL & STOP gelijktijdig in te drukken.

## BASE functie

BASE(<X>)

Adresseert de Video Display Processor (VDP). Met behulp van de waarde in X kan een item uit de tabel worden geselecteerd.

X mag een van de volgende waarden hebben:

0 - Geeft het eerste adres van de namentabel in tekstmode 1.
2 - Geeft het eerste adres van de patronentabel in tekstmode 1.

5 - Geeft het eerste adres van de namentabel in tekstmode 2.
6 - Geeft het eerste adres van de kleurentabel in tekstmode 2.
7 - Geeft het eerste adres van de patronentabel in tekstmode 2.
8 - Geeft het eerste adres van de attributentabel voor sprites in tekstmode 2.
9 - Geeft het eerste adres van de patronentabel voor sprites in tekstmode 2.

10 - Geeft het eerste adres van de namentabel in grafische mode 1.
11 - Geeft het eerste adres van de kleurentabel in grafische mode 1.
12 - Geeft het eerste adres van de patronentabel in grafische mode 1.
13 - Geeft het eerste adres van de attributentabel voor sprites in grafische mode 1.
14 - Geeft het eerste adres van de patronentabel voor sprites in grafische mode 1.

15 - Geeft het eerste adres van de namentabel in grafische mode 2.
16 - Geeft het eerste adres van de kleurentabel in grafische mode 2.
17 - Geeft het eerste adres van de patronentabel in grafische mode 2.
18 - Geeft het eerste adres van de attributentabel voor sprites in grafische mode 2.
19 - Geeft het eerste adres van de patronentabel voor sprites in grafische mode 2.

Zie ook hoofdstuk 15, Video Display Processor

## BEEP — statement

BEEP

Veroorzaakt een pieptoon.

## BIN$ — functie

BIN$(<X>)

Geeft de decimale waarde X binair weer. X mag varieren van -32768 tot en met +32767. Indien X een negatief getal is, zal het resultaat in two's complement worden weergegeven. Het resultaat zal altijd een alfanumerieke string zijn.

Voorbeeld: BIN(64) = string "1000000"

BLOAD statement

```
BLOAD "<pn1>:<fnam>"[,R][,<X>]    of
BLOAD "<pn2>:<fnam>",S
```

Laadt en machinetaalprogramma of een geheugengebied van een randapparaat (pn1 of pn2) in het geheugen.
pn1 mag zijn: "CAS" voor cassette,
"A" voor diskettedrive 1,
"B" voor diskettedrive 2.
pn2 mag zijn: "A" voor diskettedrive 1,
"B" voor diskettedrive 2.

De optie R geeft aan dat het machinetaalprogramma na laden onmiddellijk wordt gestart.
De optie X geeft het geheugenadres vanaf waar de machinecode moet worden geladen. Indien X wordt weggelaten, dan zal worden geladen vanaf adres 0.

Het tweede formaat is voor het laden van een beeldscherm vanaf pn2 naar het video-RAM.

BSAVE statement

```
BSAVE "<pn1>:<fnam>",<X>,<Y>[,<Z>]    of
BSAVE "<pn2>:<fnam>",<X>,<Y>,S
```

Schrijft een machinetaalprogramma of een geheugengebied onder de naam "fnam" naar een randapparaat.

pn1 kan zijn: "CAS" voor de cassetterecorder
"A" voor diskettedrive 1
"B" voor diskettedrive 2
pn2 kan zijn: "A" voor diskettedrive 1
"B" voor diskettedrive 2

X is het startadres van het machinetaalprogramma of het geheugengebied, Y is het eindadres daarvan.
Z is het adres waarop het machinetaalprogramma moet worden gestart.

Met het tweede formaat wordt het Video-RAM naar een diskette-file geschreven.

## CALL — statement

CALL <assignment>[(<A$>[,<B$>]...)]

Roept een statement in een ROM-cartridge aan. A$ en B$ zijn alfanumerieke constanten die als argumenten voor het aangeroepen statement dienen.

## CDBL — functie

CDBL(<X>)

Geeft een getal in dubbele nauwkeurigheid aan een numerieke variabele of numerieke expressie.

Voorbeeld: LET A#=CDBL(5/3)
Het resultaat in A# zal uit 14 cijfers bestaan.

## CHR$ — functie

CHR$(<X>)

Geeft het karakter, waarvan de ASCII-code in X staat.

Voorbeeld: PRINT CHR$(53)
Het karakter S wordt afgedrukt.

## CINT — functie

CINT(<X>)

Converteert een numerieke variabele of expressie naar een integer-getal. X mag niet kleiner dan -32768 en niet groter dan +32767 zijn.

CIRCLE statement

CIRCLE [STEP](<X>,<Y>),<Z>[,<XX>][,<YY>][,<ZZ>][,<XXX>]

Maakt een cirkel met middelpunt X,Y (hor., vert.).
X mag varieren tussen 0 en 255, Y tussen 0 en 191.
Indien STEP wordt gebruikt wordt X,Y relatief ten opzichte van de huidige cursor-positie opgegeven.

Z is de straal in aantal pixels.

XX is het kleurnummer (0 t/m 15). Indien XX niet wordt opgegeven wordt kleurnummer 15 (wit) gebruikt.

YY is de hoek (in radialen) waar de cirkel wordt begonnen. Indien YY niet wordt opgegeven wordt de waarde 0 aangenomen.

ZZ is de hoek (in radialen) waar het tekenen van de cirkel wordt gestopt. Indien ZZ niet wordt opgegeven, wordt de waarde 2*pi aangenomen.

> YY en ZZ mogen waarden hebben die varieren tussen 0 en 2*pi. Indien een negatieve waarde wordt gegeven, zullen het begin- en eindstuk van de cirkelboog met het middelpunt worden verbonden.

XXX geeft de verhouding tussen horizontale en verticale straal. Indien niet opgegeven, dan zal die verhouding 1 zijn.

CLEAR statement

CLEAR [<X>[,<Y>]]

Geeft alle numerieke variabelen de waarde 0. Maakt alle alfanumerieke variabelen leeg. Sluit (CLOSE) alle files die nog open zijn.

X = de lengte van alle gedefinieerde alfanumerieke files.

Y = het hoogste geheugenadres dat door MSX-BASIC mag worden gebruikt.

CLOAD commando

CLOAD ["<prog.naam>"] of
CLOAD? ["<prog.naam>"]

Laadt een programma, dat met CSAVE op cassette was geschreven, in het geheugen. Indien <prog.naam> niet is opgegeven, wordt het eerstgevonden programma geladen.

Indien het tweede formaat (CLOAD?) wordt gebruikt, zal het in het geheugen staande programma worden vergeleken met het van cassette te lezen programma.

CLOSE statement

CLOSE [[#]<X>[,[#]<Y>...]]

Sluit de aangegeven files en maakt de bijbehorende buffers vrij. X en Y zijn de file-nummers van files die met een OPEN instructie zijn geopend. Indien geen file-nummers zijn opgegeven, zullen alle open files worden gesloten.

CLS statement

CLS

Maakt het beeldscherm schoon en zet de cursor op de eerste positie van de eerste regel.

COLOR statement

COLOR [<X>][,<Y>][,<Z>]

Definieert de schermkleuren. X is de foreground-kleur, Y is de background-kleur en Z is de kleur van de BORDER (in tekstmode 2 en grafische modes 1 en 2).

Default-waarden voor X, Y en Z zijn 15, 4 en 7.

CONT statement

CONT

Vervolgt het programma met de regel waarop het was onderbroken. Indien het programma werd onderbroken met behulp van het STOP-statement, dan zal CONT het programma voortzetten met de regel na het STOP-statement.

COPY statement

COPY "<pn>:<fnam1>" TO "<pn>:[<fnam2>]"

Copieert een diskette-file.

pn kan zijn: "A" voor diskette drive 1
"B" voor diskette drive 2

fnam1 is de naam van de file die moet worden gecopieerd fnam2 is optioneel. Indien aanwezig, dan zal de copie van de file de naam in "fnam2" krijgen. Indien niet gegeven, dan zal de gecopieerde file de naam "fnam1" krijgen.

COS functie

COS(<X>)

Berekent de cosinus van het aantal in X gegeven radialen.

CSAVE commando

CSAVE "<prog.naam>"[,<X>]

Schrijft het programma vanuit het geheugen naar cassette, onder de in <prog.naam> gegeven naam. Met de optie X kan de schrijfsnelheid worden bepaald.
X=1 --> 1200 baud. X=2 --> 2400 baud.

CSNG functie

CSNG(<X>)

Converteert een numerieke variabele of expressie naar enkele nauwkeurigheid (maximaal 6 cijfers).

CSRLIN functie

CSRLIN

Geeft het regelnummer van de regel waarop de cursor staat (van 0 t/m 23). Werkt alleen in tekstmode 1 en 2.

CVI functie

CVI(<X$>)

Converteert de 2 bytes alfanumerieke variabele X$ naar een integer waarde. Werkt alleen voor diskettesystemen.

CVS functie

CVS(<X$>)

Converteert de 4 bytes alfanumerieke variabele X$ naar enkele nauwkeurigheid.
Werkt alleen voor diskettesystemen.

CVD functie

CVD(<X$>)

Converteert de 8 bytes alfanumerieke variabele X$ naar dubbele nauwkeurigheid.
Werkt alleen voor diskettesystemen.

DATA statement

DATA <constante>[,<constante>...]

Het DATA-statement mag numerieke en alfanumerieke constanten door elkaar bevatten. DATA-regels mogen op iedere plaats in het programma worden gezet. Ze worden allemaal tezamen als een lange lijst van data-elementen beschouwd. De DATA-constanten (items) kunnen met behulp van een READ-statement worden gelezen. Indien het READ-statement een numerieke waarde verwacht, mag er niet toevallig een alfanumerieke waarde in de DATA-regel staan.

DEF FN statement

DEF FN<naam>[(<var>[,<var>...])]=<formule>

Definieert een door de programmeur samengestelde functie.

<naam> is de naam die aan de functie zal worden gegeven Indien de naam wordt gevolgd door een $-teken, is er sprake van een alfanumerieke functie.

<formule> is de door de programmeur samengestelde functie. Daarin mogen ook variabelen voorkomen.

<var> is een variabele. Met de variabelen kunnen waarden worden doorgegeven aan de formule.

```
Voorbeeld: DEF FNA(B)=B*5
           LET C=10
           LET D=FNA(C)
           D zal nu 10*5 = 50 worden.
```

DEFDBL statement

DEFDBL <l1>[,<l2>...]

Bepaalt dat alle variabelen die beginnen met de in l1, l2 enz. gegeven letters, van het type dubbele nauwkeurigheid zullen zijn. (Zie ook DEFSTR)

DEFINT statement

DEFINT <l1>[,<l2>...]

Bepaalt dat alle variabelen die beginnen met de in l1, l2 enz. gegeven letters, van het type integer zullen zijn. (Zie ook DEFSTR)

DEFSNG statement

DEFSNG <l1>[,<l2>...]

Bepaalt dat alle variabelen die beginnen met de in l1, l2 enz. gegeven letters, van het type enkele nauwkeurigheid zullen zijn. (Zie ook DEFSTR)

DEFSTR statement

DEFSTR <l1>[,<l2>...]

Bepaalt dat alle variabelen die beginnen met de in l1, l2 enz. gegeven letters van het type alfanumeriek zijn.

```
Voorbeeld: DEFSTR A,Q-Z
           LET A="xyz":Q="tekst":R="abc":etc.
```

l1 en l2 mogen zowel en enkele letter zijn als een serie letters. In dat laatste geval wordt die serie aangegeven zoals in het voorbeeld.

DEFUSER statement

DEFUSR[<X>]=<Y>

Specificeert het startadres Y van een machinetaalprogramma. X kan een nummer van 0 t/m 9 zijn.

```
Voorbeeld:  DEFUSR1=&HF000
            A=USR1(A%)
```

DELETE commando

DELETE [<X>][-<Y>]

Verwijdert alle programmaregels X t/m Y.
Indien alleen X is opgegeven, dan wordt de betreffende regel verwijderd.
Indien alleen Y is opgegeven, dan worden alle programma regels t/m Y verwijderd.
Er moet minimaal 1 parameter worden opgegeven.
Voorbeeld: DELETE 300
DELETE 100-200
DELETE -80

DIM statement

DIM <variabele>(<X>[,<Y>[,<Z>]])

Creeert ruimte in het geheugen voor een array van X elementen. Met Y kan het aantal twee dimensionale elementen worden aangegeven, met Z het aantal drie dimensionale elementen.
Wanneer een variabele niet is gedimensioneerd, kan die 11 elementen bevatten (van 0 t/m 10).

DRAW statement

DRAW <X$>

Tekent een lijn in grafische mode 1 of 2, volgens de aanwijzingen in alfanumerieke variabele X$.
In X$ kunnen de volgende subcommando's voorkomen:

[B][N]U<X> - X pixels naar boven
[B][N]D<X> - X pixels naar beneden
[B][N]L<X> - X pixels naar links
[B][N]R<X> - X pixels naar rechts
[B][N]E<X> - X pixels naar rechtsboven
[B][N]F<X> - X pixels naar rechtsonder
[B][N]G<X> - X pixels naar linksonder
[B][N]H<X> - X pixels naar linksboven
[B][N]M[+][-]<X>,[+][-]<Y>
- Van huidige cursorpositie naar X,Y.

Met het + of - teken kan worden aangegeven dat X en/of Y een relatieve verplaatsing ten opzichte van de huidige cursorpositie aangeven.

Indien voorgaande subcommando's worden voorafgegaan door een B dan wordt geen lijn getrokken, doch alleen de cursor verplaatst.
Indien voorgaande subcommando's worden voorafgegaan door een N dan zal de cursor na het trekken van de lijn teruggaan naar de oorspronkelijke positie.

C<X> - Teken met kleur X (X = 0 t/m 15)
S<X> - Teken in schaal X (X = 1 t/m 255)
De schaal is X/4. Alle afstanden (in pixels) in de voorgaande subcommando's worden met de schaalfactor vermenigvuldigd
A<X> - Voer de tekening uit onder een hoek van X graden. (X = 0, 1, 2 of 3 voor resp. 0, 90, 180 of 270 graden)
X<Z$>; - Start de uitvoering van de tekenopdracht die in Z$ is gespecificeerd.

In de subcommando's mogen in plaats van numerieke waarden ook numerieke variabelen worden gebruikt. In dat geval dient de variabele te worden voorafgegaan door een =-teken en afgesloten met een puntkomma.

Voorbeeld:
```
LET A1=50
DRAW "U=A1;"
```
tekent 50 pixels omhoog.

## DSKF — functie

DSKF(<X>)

Geeft een lijst van vrije ruimtes op diskette X.
X kan zijn: 0 = default diskette
1 = diskette drive A
2 = diskette drive B

END statement

END

Stopt de uitvoering van het programma en sluit alle open files.

EOF functie

EOF(<X>)

Controleert of het einde van de sequentiele file X is bereikt. Indien het einde van de file is bereikt zal het resultaat van deze functie -1 zijn, anders is dat 0

ERASE statement

ERASE <var1>[,<var2>...]

Verwijdert de met DIM gecreeerde arrays uit het geheugen. Deze statement moet worden gebruikt voordat een array opnieuw wordt gecreeerd met DIM.

ERL functie

ERL

Geeft het regelnummer van de regel waarin door MSX-BASIC een fout is ontdekt.
Voorbeeld:
```
100 ON ERROR GOTO 500
 -
500 PRINT ERL
```

ERR functie

ERR

Geeft de foutcode van de fout die door het MSX-BASIC is ontdekt.

ERROR statement

ERROR <X>

Definieert eigen foutcodes. X mag een waarde van 0 t/m 255 hebben. De codes van 60 t/m 255 kunnen voor eigen foutboodschappen worden gebruikt. Indien voor een code geen foutboodschap bestaat, zal "Unprintable error" worden afgedrukt.

EXP functie

EXP(<X>)

Geeft de waarde van e tot de macht X. X mag maximaal 145.06286058562 zijn.

FIX functie

FIX(<X>)

Geeft het integerdeel van X.
Voorbeeld: FIX 5.5 = 5
FIX -5.5 = -5 (INT -5.5 = -6)

FIELD statement

FIELD [#]<X>,<Y> AS <Y$>[,<Z> AS <Z$>...]

Definieert de buffer-layout voor random diskette files. FIELD moet worden gebruikt nadat de file is geopend en voordat de GET en PUT instructies kunnen worden gebruikt.

X is het file nummer waaronder de file is geopend.
Y en Z geven het aantal karakters van het veld aan.
Y$ en Z$ zijn de namen van die velden.
Het totaal aantal karakters in de buffer mag maximaal 256 zijn.

FILES commando

FILES ["<per>:<fnam>"]

FILES zonder verdere parameters geeft een lijst van alle op de "current" diskette staande files. Indien <per> en <fnam> worden gespecificeerd, dan zal gecontroleerd worden of de betreffende file op de betreffende diskette aanwezig is, en zo ja, dan wordt de filenaam op het scherm afgedrukt. Is die file niet aanwezig, dan zal de boodschap "File not found" worden afgedrukt.
<per> kan zijn: "A" voor diskette 1
"B" voor diskette 2

FOR-NEXT statement

FOR <XX>=<X> TO <Y> [STEP <Z>]
NEXT [<XX>[,<YY>]...]

<XX> is een teller die telt van <X> tot <Y>. De teller wordt opgehoogd met stapjes van 1, tenzij STEP is gespecificeerd. In dat laatste geval zal de teller worden verhoogd of verlaagd met de waarde <Z>. <Z> mag een positieve of negatieve waarde zijn. Indien <Z> een negatieve waarde is, zal <X> groter dienen te zijn dan <Y>.

Nadat de instructies tussen FOR en NEXT zijn uitgevoerd zal de teller worden bijgewerkt en wordt teruggekeerd naar de FOR-instructie.

Het is toegestaan meerdere FOR-NEXT lussen binnen elkaar te plaatsen. Indien meerdere lussen het zelfde eindpunt hebben kan worden volstaan met 1 NEXT instructie waarin verschillende terugspringadressen (<XX>,<YY>, etc.) staan.

Indien NEXT zonder verdere parameters wordt gegeven, dan zal worden teruggekeerd naar de laatste FOR-instructie.

FRE — functie

FRE(0) of
FRE("")

Met FRE(0) wordt het aantal vrije geheugen bytes voor het BASIC programma verkregen.

Met FRE("") wordt de vrije ruimte voor alfanumerieke variabelen verkregen.

GET — statement

GET [#]<X>[,<Y>]

Leest record nummer Y uit sequentiele file nummer X in het buffer. Indien Y niet is gespecificeerd zal het eerstvolgende record worden gelezen.

GOSUB-RETURN — statement

GOSUB <X>
RETURN [<Y>]

Met GOSUB wordt naar de subroutine die op regelnummer X begint gesprongen. De laatste instructie van die subroutine moet een RETURN zijn. Indien achter de RETURN geen regelnummer is opgegeven, zal worden teruggekeerd naar de regel onmiddellijk volgend op de GOSUB-regel. Door achter RETURN een regelnummer te zetten kan ook naar een andere regel worden teruggekeerd.

GOTO — statement

GOTO <X>

Springt naar regelnummer X.

HEX$ functie

HEX$(<X>)

Geeft de hexadecimale weergave van het decimale getal X. Het resultaat is alfanumeriek. X mag varieren tussen -32768 en +32767. Indien X negatief is, zal two's complement weergave worden gebruikt.

IF statement

IF <expr.> THEN <instr.> [ELSE <instr.>]
IF <expr.> GOTO <X> [ELSE <instr.>]

Indien de expressie waar is zal de instructie achter THEN worden uitgevoerd, waarna de volgende regel wordt uitgevoerd. Is de expressie niet waar, dan zal de instructie achter ELSE worden uitgevoerd, en daarna de volgende regel. Indien ELSE niet is gespecificeerd, en de expressie is niet waar, dan wordt gewoon naar de volgende regel gegaan.
Hetzelfde geldt voor het tweede formaat, waar naar regelnummer X wordt gesprongen indien de expressie waar is.

INKEY$ functie

INKEY$

Leest het karakter dat op het toetsenbord wordt ingetikt. Indien geen karakter is ingetikt wordt een "empty string" gelezen.
Voorbeeld:
```
1000 I$=INKEY$:IF I$="" THEN GOTO 1000
1010 PRINT I$
```

INP functie

INP(<X>)

Leest een byte van input-poort X. X mag varieren van 0 t/m 255.

INPUT statement

```
INPUT [<tekst>;]<var>[,<var>...]
INPUT #<X>,<var>[,<var>...]
```

Leest een of meer gegevens van het toetsenbord en kent deze toe aan de gegeven variabelen. De tekst kan worden opgenomen om op het beeldscherm te worden afgedrukt, zodat de operator weet welke gegevens hij dient in te tikken.

Indien er meerdere gegevens in 1 INPUT-statement moeten worden ingegeven, dan moeten deze gegevens door een komma van elkaar zijn gescheiden.

Indien er meer gegevens worden ingetikt dan er worden gevraagd, zal de boodschap "?Extra ignored" worden gegeven.

Indien er een numeriek gegeven wordt gevraagd, en er wordt een alfanumeriek gegeven ingetikt, dan wordt de boodschap "?Redo from start" gegeven.

Met het tweede formaat worden gegevens uit de sequentiele file X gelezen en aan de variabelen toegekend. De file moet zijn geopend voor input.

```
Voorbeeld:  100 INPUT "Hoeveel koeien en paarden";A,B
   antwoord:    20,15
```

INPUT$ functie

```
INPUT$(<X>)
INPUT$(<X>,[#]<Y>)
```

Leest X alfanumerieke karakters van het toetsenbord (formaat 1) of uit de sequentiele file Y (formaat 2). De file moet zijn geopend voor input.

De ingetoetste karakters verschijnen niet op het beeldscherm.

INSTR functie

INSTR([<X>,]<Y$>,<Z$>)

Geeft de positie binnen Y$ waar de karaktercombinatie uit Z$ wordt gevonden. X is de startpositie binnen Y$ waar moet worden begonnen met zoeken.

Indien de combinatie uit Z$ niet wordt gevonden, is het resultaat van de functie 0.

INT functie

INT(<X>)

Geeft het integer-deel van X.
Voorbeeld: INT(2.5) = 2
INT(-0.5) = -1

INTERVAL statement

INTERVAL ON
INTERVAL OFF
INTERVAL STOP

Na INTERVAL ON zal na iedere instructie worden gecontroleerd of een bepaalde tijdsinterval verstreken is. Is dit zo, dan zal de subroutine, die is aangegeven met het statement "ON INTERVAL GOSUB", worden uitgevoerd.

INTERVAL OFF maakt dat er niet meer wordt gecontroleerd op het verstrijken van de tijdsinterval.

Na INTERVAL STOP wordt nog wel gecontroleerd op het verstrijken van de tijdsinterval, doch er wordt niet direct wat mee gedaan. Zodra echter INTERVAL ON wordt gegeven, en er was een tijdsinterval verstreken, dan zal de subroutine worden uitgevoerd.

## KEY — statement

KEY<X>,<X$>

Geeft een bepaalde functie, die wordt aangegeven met X$ aan functietoets X.
X kan varieren van 1 t/m 10.
X$ mag maximaal 15 karakters lang zijn.

## KEY LIST — statement

KEY LIST

Geeft een lijst van alle aan de functietoetsen toegekende teksten, beginnende bij functietoets 1.

## KEY ON/OFF — statement

KEY ON
KEY OFF

Met KEY ON worden de functietoetsen op de 24-ste regel afgedrukt, met KEY OFF verdwijnen ze weer. Werkt alleen in tekstmode 1 of 2.

## KEY (X) — statement

KEY (<X>) ON
KEY (<X>) OFF
KEY (<X>) STOP

Na KEY (<X>) ON wordt bij iedere instructie gecontroleerd of functietoets X is ingedrukt, en zo ja, dan wordt naar de bij ON KEY GOSUB gegeven subroutine gesprongen.

Met KEY (<X>) OFF wordt er niet meer gecontroleerd.

Met KEY (<X>) STOP wordt nog wel gecontroleerd, doch er wordt pas gesprongen op het moment dat er weer een KEY (<X>) ON is gegeven.

KILL statement

```
KILL "<per>:<fnam>"

Wist de file <fnam> van diskette <per>.
<per> kan zijn:  "A" voor diskette drive 1
                 "B" voor diskette drive 2
```

LEFT$ functie

```
LEFT$(<X$>,<X>)

Geeft de X meest linkse karakters van variabele X$.
X mag varieren van 0 t/m 255.
Indien X=0 dan wordt een "empty string" gegeven.
Is X hoger dan het aantal karakters in X$ dan wordt  X$
gegeven.
Voorbeeld:  LET A$="AKKERMANS"
            PRINT LEFT$(A$,5)
            Nu wordt AKKER afgedrukt.
```

LEN functie

```
LEN(<X$>)

Geeft het aantal karakters in X$.
Voorbeeld:  LET A$="AKKERMANS"
            PRINT LEN(A$)
            Nu wordt 9 afgedrukt.
```

LET statement

```
[LET] <X[$]>=<Y[$]>

Kent de waarde Y of de alfanumerieke waarde Y$ toe  aan
de numerieke variabele X of alfanumerieke variabele X$.
Y en Y$  mogen  constanten,  variabelen  of  expressies
zijn.
```

## LINE — statement

```
LINE [[STEP](<X1>,<Y1>)]-[STEP](<X2>,<Y2>)[,<Z>][,B[F]]
```

Trekt in grafische mode 1 of 2 een lijn van startpunt X1,Y1 naar eindpunt X2,Y2 in de kleur Z.

Indien de optie B wordt gebruikt zal een rechthoek worden getekend met de gegeven lijn als diagonaal. Wordt bovendien de optie F gegeven, dan zal de rechthoek worden opgevuld met de kleur Z.

X1 mag varieren tussen 0 en 255, Y1 tussen 0 en 191. Indien ze niet worden gespecificeerd wordt de huidige coordinaat als startpunt gebruikt.

Met STEP wordt aangegeven dat de gespecificeerde coordinaten relatief ten opzicht van de "current cursor" positie worden genomen.

## LINE INPUT — statement

```
LINE INPUT [<tekst>;]<X$>   of
LINE INPUT #<X>,<X$>
```

Kent een alfanumerieke waarde toe aan de variabele X$. De alfanumerieke waarde komt van het toetsenbord (formaat 1) of vanuit een sequentiele file (formaat 2).

Met formaat 1 kunnen maximaal 254 karakters worden geaccepteerd van het toetsenbord. Indien gewenst kan de <tekst> op het beeldscherm worden afgedrukt.

Met formaat 2 wordt een heel record uit de file gelezen en aan de variabele X$ toegekend. Met X wordt aangegeven uit welke file dat record moet worden gelezen. De file moet geopend zijn voor input.

Indien een BASIC-programma met een SAVE-statement is weggeschreven, dan kan dat programma regel voor regel worden ingelezen met behulp van de LINE INPUT # statement.

LIST commando

LIST [<X>[-[<Y>]]]

Maakt een listing van het BASIC-programma op het beeldscherm.
LIST - list alle regels
LIST <X> - list alleen regel X
LIST -<Y> - list alle regels t/m <Y>
LIST <X>-<Y> - list alle regels van <X> t/m <Y>

LLIST commando

LLIST [<X>[-[<Y>]]]

Maakt een listing van het BASIC-programma op de printer
Werkt verder precies als LIST.

LOAD commando

LOAD "<per>:<prog.naam>"[,R]

Laadt een programma, dat was weggeschreven met SAVE, van cassette of diskette in het geheugen.
<per> kan zijn: CAS voor cassette
A voor diskette drive 1
B voor diskette drive 2

De optie R geeft aan dat het programma onmiddellijk na laden moet worden gestart.

LOC functie

LOC(<X>)

Geeft voor random file nummer X het laatst geschreven of gelezen record nummer, en voor sequentiele file nummer X het aantal blokken van 256 bytes dat zijn geschreven of gelezen.

LOCATE statement

LOCATE [<X>][,<Y>][,<Z>]

Stuurt de cursor naar positie X,Y op het scherm. Indien X en/of Y niet zijn gegeven, wordt de huidige cursor positie gebruikt.
<X> kan varieren van 0 t/m 39
<Y> kan varieren van 0 t/m 23
<Z> kan 0 of 1 zijn, 0 betekent dat de cursor onzichtbaar is, 1 wil zeggen dat hij zichtbaar is.

LOF functie

LOF(<X>)

Geeft de lengte van file nummer X in aantal bytes. De file moet zijn geopend voor LOF wordt gebruikt.

LOG functie

LOG(<X>)

Geeft de natuurlijke logaritme (grondtal e=2.718282) van getal X.
De Briggse logaritme (grondtal 10) kan als volgt worden berekend: 0.434294 * LOG(<X>).

LPOS functie

LPOS(0)

Geeft de printpositie binnen het printer buffer. Afhankelijk van de gebruikte printer hoeft dit geen indicatie te zijn van de werkelijk afgedrukte karakters

LPRINT statement

LPRINT [[USING <formaat>;]<expr.>...]

Print gegevens naar de printer. Werkt verder precies als het PRINT-statement.

LSET statement

LSET <X$>=<Y$>

Vult de variabele X$ van links naar rechts met de data uit de variabele of alfanumerieke expressie Y$. Moet worden gebruikt voor het vullen van het buffer voor een random diskette file.

MAXFILES statement

MAXFILES=<X>

Bepaalt het maximum aantal files (X) dat gelijktijdig geopend mag zijn in een programma.
<X> mag varieren van 0 t/m 15.

MERGE commando

MERGE "<per>:<prog.naam>"

Voegt de regels van het gespecificeerde programma toe aan het reeds in het geheugen aanwezige programma. Indien de regelnummers gelijk zijn, zullen de toegevoegde regels de reeds aanwezige regels overschrijven.
<per> kan zijn: CAS voor cassette
A voor diskette drive 1
B voor diskette drive 2

MID$ functie

MID$(<X$>,<X>[,<Y>])

Geeft de Y karakters, vanaf positie X, uit X$.
<X> en <Y> mogen varieren van 1 t/m 255.

Indien Y niet wordt gegeven, zullen alle karakters rechts van X worden genomen.
Indien X voorbij het laatste karakter van X$ wijst, zal een "empty string" worden gegeven.

MID$= statement

MID$(<X$>,<X>[,<Y>])=<Y$>

Vervangt het deel van X$, dat is aangegeven met startpositie X en lengte Y, door de alfanumerieke variabele of alfanumerieke expressie Y$.

Indien Y niet is gegeven, zullen alle karakters van Y$ worden opgenomen in X$, voorzover de lengte van X$ dat toelaat.

MKD$ functie

MKD$(<X>)

Converteert de dubbele nauwkeurigheid variabele X naar een acht bytes alfanumerieke waarde, die in het buffer voor een random diskette file kan worden geplaatst.

MKI$ functie

MKI$(<X>)

Converteert de integer variabele X naar een twee bytes alfanumerieke waarde, die in het buffer voor een random diskette file kan worden geplaatst.

MKS$ functie

MKS$(<X>)

Converteert de enkele nauwkeurigheid variabele X naar een vier bytes alfanumerieke waarde, die in het buffer voor een random diskette file kan worden geplaatst.

MOTOR statement

MOTOR [ON][OFF]

Schakelt de cassetterecordermotor aan of uit.
MOTOR zonder verdere parameter schakelt de motor uit indien die aan stond, en aan indien die uit stond.

NAME statement

NAME "<per>:<fnam1>" AS "<fnam2>"

Verandert de naam van een diskette file.
<per> kan zijn: A voor diskette drive 1
B voor diskette drive 2
<fnam1> is de bestaande file naam.
<fnam2> is de nieuwe file naam.

NEW commando

NEW

Wist het geheugen. Dient te worden gebruikt voordat een nieuw programma wordt geladen.

OCT$ functie

OCT$(<X>)

Geeft de octale representatie van het decimale getal X. X mag een waarde hebben van -32768 t/m +32767. Indien X negatief is, zal de two's complement notatie worden gebruikt.

ON ERROR statement

ON ERROR GOTO <X>
RESUME [<Y>][NEXT]

ON ERROR GOTO maakt een sprong naar de foutafhandelings routine die start op regel X, indien er een fout is opgetreden. Indien voor X de waarde 0 wordt ingevuld, zal niet worden gesprongen, doch zal de gebruikelijke foutboodschap worden gegeven.

Met RESUME kan worden teruggekeerd naar de regel waarin de fout was opgetreden. Indien RESUME NEXT wordt geprogrammeerd, zal worden teruggekeerd naar de regel die volgt op de regel waarin de fout optrad. Door voor Y een bepaald regelnummer in te vullen kan ook naar een andere regel worden teruggekeerd.

ON GOSUB statement

ON <expr.> GOSUB <X>[,<Y>...]

Springt, afhankelijk van de waarde van <expr.> naar een van de subroutines, waarvan de regelnummers waarop ze starten achter GOSUB staan gespecificeerd. Indien de waarde in <expr.> hoger is dan het aantal achter GOSUB gespecificeerde regelnummers, dan wordt gewoon door de instructie heen gezakt.
<expr.> mag varieren van 0 t/m 255.

Voorbeeld: 100 ON 2 GOSUB 200,300,400
Veroorzaakt een sprong naar de subroutine die op regel 300 begint

## ON GOTO — statement

ON <expr.> GOTO <X>[,<Y>...]

Maakt, afhankelijk van de waarde van <expr.>, een onvoorwaardelijke sprong naar een van de achter GOTO gegeven regelnummers. Indien de waarde van <expr.> hoger is dan het aantal achter GOTO gespecificeerde regelnummers, dan zal gewoon door de instructie heengezakt worden.
<expr.> mag varieren van 0 t/m 255.

Voorbeeld: 100 ON 2 GOTO 200,300,400
Veroorzaakt een onvoorwaardelijke sprong naar regel 300.

## ON INTERVAL GOSUB — statement

ON INTERVAL=<X> GOSUB <Y>

Nadat de interval timing is geactiveerd met INTERVAL ON zal met ON INTERVAL iedere X/50 seconden de subroutine, die op regel Y begint, worden uitgevoerd.

## ON KEY GOSUB — statement

ON KEY GOSUB <X>[,<Y>...]

Nadat de functietoetsen zijn geactiveerd met KEY(X) ON, kan met ON KEY GOSUB, afhankelijk van de waarde van de ingetikte functietoets, naar een van de daarna opgegeven subroutines worden gesprongen. Voor het overige werkt deze instructie net zo als de ON <expr.> GOSUB instructie.

ON SPRITE GOSUB statement

---

ON SPRITE GOSUB <X>

Nadat het detecteren van "sprite collision" is geactiveerd met SPRITE ON, wordt met ON SPRITE GOSUB naar de subroutine die op regel X begint gesprongen, zodra twee sprites met elkaar in aanraking komen.

ON STOP GOSUB statement

---

ON STOP GOSUB <X>

Springt naar de subroutine die op regel X begint, zodra de twee toetsen CONTROL en STOP tegelijkertijd worden ingedrukt.

ON STRIG GOSUB statement

---

ON STRIG GOSUB <X>[,<Y>...]

Nadat de actieknoppen van de "hand controls" zijn geactiveerd met STRIG (X) ON, kan met ON STRIG GOSUB naar een van de daarachter gespecificeerde subroutines worden gesprongen. Zie voor mogelijke waarden van STRIG de STRIG (X) ON statement.

OPEN statement

---

OPEN "<per>:<fnam>" [FOR <mode>] AS [#]<X> [LEN=<Y>]

Creeert een I/O-buffer voor een file.
<per> kan zijn: CRT voor het tekstscherm.
GRP voor het grafisch scherm.
LPT voor de printer.
CAS voor cassette.
A voor diskette drive 1.
B voor diskette drive 2.
<fnam> mag worden weggelaten voor CRT, GRP en LPT.

<mode> kan zijn: INPUT voor sequentieel lezen.
OUTPUT voor sequentieel schrijven.
APPEND voor sequentieel toevoegen aan een bestaande file.

Door de FOR-optie geheel weg te laten wordt een random file geopend. Indien de <fnam> niet op <per> wordt gevonden, zal de file onder de naam <fnam> worden gecreeerd.

OUTPUT is mogelijk voor alle devices. INPUT kan alleen voor CAS, A en B. APPEND en random gaat alleen voor A en B.

<X> is het file nummer waaronder de file met andere I/O instructies kan worden benaderd.

Met de optie LEN kan de recordlengte (in geval van een random file) worden bepaald. Indien deze optie niet wordt gebruikt, zal de recordlengte 256 zijn.

## OUT — statement

OUT <X>,<Y>

Stuurt de waarde Y naar I/O-poort X.
<X> en <Y> mogen varieren van 0 t/m 255.

## PAD — functie

PAD(<X>)

Leest de status van de "touch pads", die zijn aangesloten op de joystick-connectors. Met <X> kan een bepaalde status worden opgevraagd.

X=0 - geeft status van touch pad 1:
-1 is gebruikt
0 is niet gebruikt.
X=1 - geeft X-coordinaat van touch pad 1.
X=2 - geeft Y-coordinaat van touch pad 1.
X=3 - geeft status van de schakelaar op touch pad 1:
-1 schakelaar is ingedrukt (geweest)
0 schakelaar is niet ingedrukt (geweest).

De waarden X=4 t/m X=7 hebben dezelfde betekenis, doch dan voor touch pad 2.

## PAINT — statement

PAINT [STEP](<X>,<Y>),<Z>[,<ZZ>]

Vult een grafisch figuur (in grafische modes 1 en 2), die begint op coordiaten X,Y, met kleur Z. In grafische mode 2 kan bovendien een kleur voor de omhulling worden opgegeven (optie ZZ).
X mag varieren van 0 t/m 255.
Y mag varieren van 0 t/m 191.

De optie STEP maakt de coordinaten relatief ten opzichte van de cursor. In dat geval mogen X en Y ook negatief zijn.

## PDL — functie

PDL(<X>)

Leest een waarde van een "games paddle", die is aangesloten op een joystick interface. Met X wordt een bepaald deel van de games paddle geselecteerd, waarna een waarde tussen 0 en 255 het gelezen resultaat kan zijn.

Door X de waarden 1, 3, 5, 7, 9 of 11 te geven, wordt de paddle op connector 1 geselecteerd. De waarden 2, 4, 6, 8, 10 of 12 selecteren de paddle op connector 2.

## PEEK — functie

PEEK(<X>)

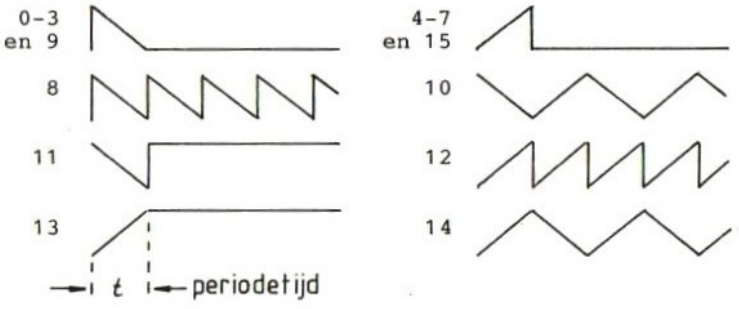
Leest de inhoud van geheugenadres X.

PLAY <X$>[,<Y$>][,<Z$>]

Speelt muziek, zoals is aangegeven in de subcommando strings X$, Y$ en Z$, die respectievelijk aan de geluidsgeneratoren 1, 2 en drie zijn toegewezen.

De subcommandostrings kunnen zijn samengesteld uit de volgende subcommando's:

T<X> Zet het tempo. <X> mag een waarde van 32 t/m 255 zijn. Het tempo is X/4 noten per minuut.
L<X> Zet de lengte van een noot. <X> mag een waarde van 1 t/m 64 zijn. De lengte van de noot is 1/X.
. Verandert de lengte van een noot. Iedere punt die een noot volgt, deelt de lengte daarvan door twee.
O<X> Selecteert een octaaf. <X> mag een waarde van 1 t/m 8 hebben.
A/G Speelt de noten A, B, C, D, E, F of G. Iedere noot kan worden gevolgd door een + of # om een verhoogde, of door een - om een verlaagde noot aan te geven.
N<X> Speelt een noot, die is aangegeven door een nummer (<X>). X mag een waarde van 0 t/m 95 hebben.
R<X> Zet een rust. <X> mag een waarde hebben van 1 t/m 64. De duur van de rust is 1/X.
V<X> Zet het volume. <X> mag een waarde hebben van 0 t/m 15.
S<X> Zet de vorm van de "sound envelope". <X> mag een waarde hebben van 0 t/m 15. Daarmee worden de volgende vormen gekozen:



M<X> Zet de periodetijd van de "sound envelope". <X> mag een waarde van 1 tvm 65535 hebben. De default waarde is 255.
"X<A$>;" voert de subcommando's uit A$ uit.
Voorbeeld: A$="CDE":PLAY "XA$;"

## PLAY() — functie

PLAY(<X>)

Geeft aan of de geluidsopdracht voor kanaal X is uitgevoerd of niet. <X> kan 0, 1, 2 of 3 zijn, voor respectievelijk alle geluidskanalen, kanaal 1, 2 of 3. Indien de opdracht nog wordt uitgevoerd zal het resultaat -1 zijn. Is de opdracht eenmaal uitgevoerd, dan is het resultaat van deze functie 0.

## POINT — functie

POINT(<X>,<Y>)

Geeft, in grafische modes 1 en 2, het kleurnummer van het pixel op coordinaat X,Y.
<X> mag varieren van 0 t/m 255.
<Y> mag varieren van 0 t/m 191.

## POKE — statement

POKE <X>,<Y>

Schrijft het byte Y naar geheugenlocatie X.
<X> mag varieren van 0 tot 65535.
<Y> mag varieren van 0 tot 255.

## POS — functie

POS(0)

Geeft, in tekstmode 1 en 2, de horizontale positie van de cursor. Het resultaat kan tussen 0 - 39 liggen.

PRESET statement

PRESET [STEP](<X>,<Y>)[,<Z>]

Geeft, in grafische mode 1 en 2, het pixel op coordinaat X,Y de kleur Z. <X> mag varieren van 0 t/m 255, <Y> van 0 t/m 191.

Met de optie STEP worden de coordinaten X,Y relatief ten opzichte van de "current" cursor positie gemaakt. In dat geval mogen X en Y ook negatief zijn.

Indien Z niet wordt gespecificeerd, wordt de laatst gezette kleur gebruikt. De default waarde voor Z is kleurnummer 4.

PRINT statement

PRINT [[USING <formaat>;]<expr.>;...]
PRINT #<X>, [USING <formaat>;]<var.>;...

Formaat 1 drukt de gegevens uit <expr.> af op het beeldscherm, indien gewenst met gebruikmaking van het opgegeven <formaat>.

Formaat 2 drukt de gegevens uit <var.> af naar de sequentiele file <X>, die moet zijn geopend voor output, indien gewenst met gebruikmaking van het opgegeven <formaat>.

De expressies kunnen van elkaar worden gescheiden, en het PRINT-statement kan worden afgesloten, door de volgende tekens:
Spatie - Het volgende gegeven wordt op de volgende regel afgedrukt.
; - Het volgende gegeven wordt onmiddellijk na het laatste gegeven afgedrukt.
, - Het volgende gegeven wordt aan het begin van de volgende kolom afgedrukt. Iedere kolom is 14 karakters breed.

Op het scherm afgedrukte getallen worden altijd gevolgd door een spatie. Positieve getallen worden ook door een spatie voorafgegaan. Negatieve getallen worden voorafgegaan door een minteken.

Wanneer naar een file wordt geprint, zal iedere PRINT-statement worden afgesloten met een CR/LF-code. Tussen de numerieke expressies worden automatisch komma's gezet. Tussen alfanumerieke expressies moet u zelf een komma zetten (als scheidingsteken).

USING <formaat>.

Alfanumerieke formaten:

USING "!" - Alleen het eerst karakter van de gegeven expressie wordt afgedrukt.

USING "\ \" - Twee + het aantal spaties tussen de \-tekens, van de gegeven expressie, worden afgedrukt.

USING "@" - Het @-teken wordt vervangen door de gegeven expressie.
A$="MSX":PRINT USING "@ BASIC";A$
Geeft als resultaat: MSX BASIC

Numerieke formaten:

USING "#" - Geeft het aantal cijfers van het getal aan. Het getal zal naar rechts uitgelijnd worden. Links wordt dan aangevuld met spaties. Zo nodig wordt het getal afgerond.

USING "." - Geeft de plaats van de decimale punt aan.

USING "," - Een komma links van de decimale punt geeft aan dat er om de drie cijfers een komma zal worden geprint.

USING "+" - Geeft aan dat een plus- of minteken aan het getal zal worden toegevoegd. Indien de + aan het begin van het formaat wordt gezet, dan wordt het teken voor het getal geprint, wordt de + aan het eind gezet, dan wordt het teken aan het eind van het getal geprint.

USING "-" - Het minteken aan het einde van een formaat maakt dat indien het getal positief is er een spatie achter wordt geprint en indien het negatief is, er een minteken achter wordt gezet.

USING "**" - Voorlopende nullen worden vervangen door asterisken i.p.v. spaties.

USING "$$" - Het getal wordt voorafgegaan door een dollarteken.

USING "**$" - Dit is een combinatie van beide voorgaande formaten.

USING "^^^^" -Het getal zal in exponentiele vorm worden afgedrukt.

Voorbeelden:

```
1000 A=1234567.89
1010 PRINT USING "##########,.###";A
resultaat:          1,234,567.890

1000 A=1.25
1010 PRINT USING "$$###.##-";A
resultaat:            $1.25

1000 A=-1.25
1010 PRINT USING "**#.##+";A
resultaat:          **1.25-
```

## PSET statement

PSET [STEP](<X>,<Y>)[,<Z>]

Geeft, in grafische modes 1 en 2, het pixel op coordinaat X,Y de kleur Z.
<X> mag varieren van 0 t/m 255.
<Y> mag varieren van 0 t/m 191.
Indien STEP wordt gespecificeerd, dan worden X en Y relatief ten opzichte van de huidige cursorpositie genomen. In dat geval mogen X en Y ook negatief zijn.

## PUT statement

PUT [#]<X>[,<Y>]

Schrijf record nummer Y van het buffer naar random file nummer X.
<Y> mag varieren van 0 t/m 32767.
Indien Y niet wordt gegeven, zal het volgende record nummer worden gebruikt.

PUT SPRITE statement

PUT SPRITE <Z>[,[STEP](<X>,<Y>)][,<XX>][,<YY>]

Plaatst SPRITE nummer YY op coordinaat X,Y op het scherm, met prioriteit Z, in kleur XX.

<X> mag varieren van -32 t/m 255.
<Y> mag varieren van -32 t/m 191, of mag de waarden 208 of 209 hebben.
208 betekent dat alle sprites met een lagere prioriteit zullen verdwijnen.
209 betekent dat alleen deze sprite van het scherm zal verdwijnen.
<Z> mag varieren van 0 t/m 31.
<XX> is de kleur (0 t/m 15).
<YY> is het sprite-nummer, zoals gegeven met SPRITE$(X) Indien YY wordt weggelaten zal het prioriteitsnummer worden genomen.

READ statement

READ <var.>[,<var.>...]

Leest gegevens uit DATA-regels in de variabele(n). Indien de variabele numeriek is, zal het gelezen gegeven ook numeriek dienen te zijn. Is de variabele alfanumeriek, dan moet ook het gelezen gegeven alfanumeriek zijn.
Indien er geen gegevens (meer) zijn om te lezen, zal de boodschap "Out of data" worden gegeven.

REM statement

REM <commentaar>

REM statements worden alleen afgedrukt bij het maken van een listing. Ze worden niet uitgevoerd. De functie van REM-statements is, commentaar in een programma listing te kunnen geven, opdat die listing beter leesbaar wordt.

RENUM commando

RENUM [[<X>][,<Y>][,<Z>]]

Hernummert alle programmaregels vanaf regel nummer Y. De nieuwe programmaregels zullen worden genummerd vanaf X, met een ophoging Z.

Indien Y wordt weggelaten, zal het hernummeren starten met de eerste regel van het bestaande programma.
Indien Z wordt weggelaten, zal de ophogingsfactor 10 zijn.
Indien helemaal geen parameters bij RENUM worden gegeven, zal het hernummeren starten bij de eerste regel van het bestaande programma, terwijl de nieuwe regels worden genummerd vanaf 10, met een ophogingsfactor van 10.

RESTORE statement

RESTORE [<X>]

Zet de "read-pointer" op het eerste element van DATA-regel X. Indien X wordt weggelaten, zal de pointer op het eerste element van de eerste DATA-regel worden gezet.

RIGHT$ functie

RIGHT$(<X$>,<X>)

Geeft de X meest rechtse karakters uit de alfanumerieke variabele <X$>.
Indien X groter is dan het aantal in X$ voorkomende karakters, wordt de hele string genomen.
Indien X gelijk is aan 0, wordt een empty string genomen.

RND functie

RND(<X>)

Geeft een nummer (tussen 0 en 1) uit een tabel. Deze tabel is, na RUN, steeds dezelfde. Met X kan de keuze van het getal uit de tabel worden gewijzigd.
X > 0 - Het volgende nummer uit de tabel.
X = 0 - Het laatste nummer uit de tabel.
X < 0 - Een nieuwe tabel, gebaseerd op de negatieve X wordt gecreeerd.

Door de waarde van de variabele "TIME" te gebruiken kan een werkelijk random getal worden verkregen:

```
A=RND(-TIME)
```

RSET statement

RSET <X$>=<Y$>

Zet de inhoud van de alfanumerieke variabele of alfanumerieke expressie Y$ in X$, beginnende vanaf de rechterkant. Deze statement dient te worden gebruikt om data in het random diskette buffer te zetten.

RUN commando

RUN [<per.>:<prog.naam>] [<X>]

Start de uitvoering van het programma <prog.naam> dat op randapparaat <per.> staat op programmaregelnummer <X>.
<per.> kan zijn: CAS voor cassette
A voor diskette drive 1
B voor diskette drive 2

Indien <X> niet is gegeven, zal de uitvoering op de eerste regel van het programma starten.

Indien geen randapparaat en programmanaam is gegeven, zal het in het geheugen staande programma worden gestart op de aangegeven regelnummer.

SAVE commando

SAVE "CAS:<prog.naam>" of
SAVE "<per.>:<prog.naam>"[,A]

Schrijft een programma vanuit het geheugen onder de naam <prog.naam> naar cassette of diskette.
Wanneer het tweede formaat wordt gebruikt zal het programma in zogenaamd "compressed binary" formaat worden weggeschreven, tenzij de optie A wordt gebruikt. In dat laatste geval zal het programma, net als bij het eerste formaat, in ASCII formaat worden weggeschreven.

SCREEN statement

SCREEN [<X>][,<Y>][,<Z>][,<XX>][,<YY>]

Zet de beeldschermmode op X, de spritegrootte op Y, de intoetsklik op Z, de baudsnelheid voor de cassetterecorder op XX en het printertype op YY.

<X> 0 = tekstmode 1 (40*24 karakters)(default)
1 = tekstmode 2 (32*24 karakters)
2 = grafische mode 1 (256*192 pixels)
3 = grafische mode 2 (64*48 pixels)
<Y> 0 = kleine sprites (8*8 pixels)(default)
1 = vergrote kleine sprites (16*16 pixels)
2 = grote sprites (16*16 pixels)
3 = vergrote grote sprites (32*32 pixels)
<Z> 0 = geen intoetsklik
1 = intoetsklik (default)
<XX> 1 = 1200 baud (default)
2 = 2400 baud
<YY> 0 = MSX-printer (default)
1 = Geen specifieke MSX-printer

SGN functie

SGN(<X>)

Bepaalt het teken van de waarde in X. Het resultaat van deze functie kan zijn: 1 = X>0, 0 = X=0 of -1 = X<0.
Voorbeeld: A=-1000: PRINT SGN(A) (-1 wordt afgedrukt)

SIN functie

SIN(<X>)

Berekent de sinus van het in X aangegeven aantal radialen.

SOUND statement

SOUND <X>,<Y>

Zet de waarde Y in register X van de programmeerbare geluidsgenerator.
<X> mag varieren van 0 t/m 15.
<Y> mag varieren van 0 t/m 255.

SPACE$ functie

SPACE$(<X>)

Geeft een alfanumerieke string die uit X spaties bestaat.
<X> mag varieren van 0 t/m 255.

SPC functie

SPC(<X>)

Drukt X spaties af.
<X> mag varieren van 0 t/m 255.
Voorbeeld: PRINT "links";SPC(20);"rechts"

SPRITE statement

SPRITE ON of SPRITE OFF of SPRITE STOP

Na SPRITE ON wordt na iedere instructie gecontroleerd of twee sprites met elkaar in botsing zijn gekomen, en zo dit het geval is, zal de subroutine die met ON

SPRITE GOSUB wordt aangeroepen, worden uitgevoerd.
Na SPRITE OFF wordt niet meer gecontroleerd.
Na SPRITE STOP wordt wel gecontroleerd op een botsing van sprites, doch de subroutine zal niet worden uitgevoerd. Pas nadat SPRITE ON is gegeven zal de subroutine onmiddellijk worden uitgevoerd.

## SPRITE$ — systeemvariabele

SPRITE$(<X>)

Bevat de definitie van een sprite.
<X> mag varieren van 0 t/m 255.

Definieren van sprites gaat als volgt:

```
100 DATA 60,66,165,129,165,153,66,60
110 LET A$="": SCREEN 2,1
120 FOR I=1 TO 8
130 READ A: LET A$=A$+CHR$(A)
140 NEXT I
150 LET SPRITE$(1)=A$
```

Gebruiken van sprites gaat als volgt:

```
200 FOR X=0 TO 255
210 PUT SPRITE 1(X,90),8,1
220 NEXT X
230 GOTO 200
```

## SQR — functie

SQR(<X>)

Geeft de vierkantswortel van X. (X >= 0)

## STICK — functie

STICK(<X>)

Geeft de positie van de joystick of de status van de cursor control toetsen. Of de cursor control toetsen of een van de joysticks zijn geselecteerd wordt bepaald

voor X. De volgende waarden van X zijn toegestaan:

0 = cursor control toetsen.
1 = joystick 1
2 = joystick 2

Het resultaat van deze functie kan een van de hierna volgende waarden zijn:

0 = neutraal (niets ingedrukt)
1 = boven
2 = rechtsboven
3 = rechts
4 = rechtsonder
5 = onder
6 = linksonder
7 = links
8 = linksboven

## STOP — statement

STOP

Onderbreekt de uitvoering van het programma met de boodschap "Break in ‹regelnummer›". Het programma kan worden voortgezet met het commando CONT.

## STOP ON/OFF/STOP — statement

STOP ON of STOP OFF of STOP STOP

Na STOP ON wordt na iedere instructie gecontroleerd of de CONTROL en STOP toetsen gelijktijdig zijn ingedrukt. Is dit het geval, dan zal de subroutine, die na ON STOP GOSUB staat aangegeven, uitgevoerd.

Na STOP OFF wordt er niet meer gecontroleerd.

Na STOP STOP wordt nog wel gecontroleerd of de CONTROL en STOP toetsen gelijktijdig zijn ingedrukt, maar wordt de subroutine niet uitgevoerd. Pas nadat weer een STOP ON statement is gegeven zal de subroutine onmiddellijk worden uitgevoerd.

## STRIG — functie

STRIG(<X>)

Geeft de status van de spatiebalk of de "hand controls", zoals aangegeven met X.

- 0 = spatiebalk
- 1 = actieknop van "hand control" 1
- 2 = actieknop van "hand control" 2
- 3 = actieknop van "hand control" 1
- 4 = actieknop van "hand control" 2

Het resultaat van deze functie kan zijn:

- -1 = De met X aangegeven actieknop of spatiebalk is ingedrukt.
- 0 = Niet ingedrukt.

## STRIG(X) — statement

STRIG(<X>) ON
STRIG(<X>) OFF
STRIG(<X>) STOP

<X> kan een van de volgende waarden hebben:

- 0 = spatiebalk
- 1 = actieknop van "hand control" 1
- 2 = actieknop van "hand control" 2
- 3 = actieknop van "hand control" 1
- 4 = actieknop van "hand control" 2

Na STRIG(<X>) ON wordt het controleren van de status geactiveerd. Is de aangegeven knop ingedrukt, dan zal de subroutine die achter ON STRIG GOSUB staat aangegeven worden uitgevoerd.

Na STRIG(<X>) OFF wordt er niet meer gecontroleerd.

Na STRIG(<X>) STOP wordt nog wel gecontroleerd, doch de subroutine wordt niet gestart, ook al blijkt de aangegeven toets ingedrukt te zijn. Zodra echter weer STRIG(<X>) ON wordt gegeven, zal de subroutine onmiddellijk worden uitgevoerd.

## STR$ — functie

STR$(<X>)

Geeft de numerieke waarde weer als een alfanumerieke string. Voorbeeld: STR$(123) = "123"

## STRING$ — functie

STRING$(<X>,<Y>)
STRING$(<X>,<Y$>)

Genereert een alfanumerieke string met een lengte van X en met als inhoud karakters van de code Y of de code van het eerste karakter van variabele Y$.
<X> mag varieren van 0 tot 255.
<Y> mag varieren van 32 tot 255.

## SWAP — statement

SWAP <var.>,<var.>

Verwisseld de inhoud van beide variabelen. Beide variabelen moeten van hetzelfde type zijn.

## TAB — functie

TAB(<X>)

Verplaatst de cursor, op dezelfde regel, naar positie X. X mag varieren van 0 tot 255. Deze functie kan alleen bij het (L)PRINT-statement worden gebruikt.

## TAN — functie

TAN(<X>)

Berekent de tangens van het in X gegeven aantal radialen.

TIME systeemvariabele

TIME

Bevat de stand van de interne teller die elke 1/50 seconde met 1 wordt verhoogd.

Gebruiksvoorbeeld:

```
1000 CLS
1010 LET TIME=0                          (reset timer)
1020 LET T=TIME
1030 LOCATE 15,10
1030 PRINT USING "###.##";T/50;
1040 GOTO 1020
```

TROFF commando

TROFF

Schakelt de trace functie uit.

TRON commando

TRON

Schakelt de trace functie in. Hierna zullen van alle programmaregels die worden uitgevoerd de regelnummers worden afgedrukt. Dit maakt het mogelijk om te zien hoe de uitvoering van het programma verloopt, en waar het eventueel fout gaat.

USR functie

USR[<X>](<Y>)

Roept een machinetaal routine <X> aan.
<X> kan varieren van 0 t/m 9 en is het nummer van de machinetaal subroutine. Indien X wordt weggelaten dan wordt nummer 0 aangenomen. Het startadres van de subroutine moet met het statement DEFUSR zijn vastgelegd.

<Y> is de waarde die aan de subroutine wordt meegegeven en kan een van de volgende soorten zijn:

% - Indien <&HF663> = 2, dan:
bevatten &HF7F8 en &HF7F9 de integer waarde.

$ - Indien <&HF663> = 3, dan:
bevatten &HF7F8 en &HF7F9 het adres van een drie bytes descriptor-blok.
Het eerste byte van dit blok geeft de lengte van de string, de volgende twee bytes geven het start-adres van de string in het RAM-geheugen.

! - Indien <&HF663> = 4, dan:
bevatten &HF7F8 en &HF7F9 de waarde met enkele nauwkeurigheid.

# - Indien <&HF663> = 8, dan:
bevatten &HF7F8 tot en met &HF7FD de waarde met dubbele nauwkeurigheid.

Waarden die door het machinetaal programma worden teruggegeven naar het BASIC-programma, moeten in dat BASIC-programma in een variabele van de juiste soort worden opgevangen ($, %, ! of #).

## VAL — functie

VAL(<X$>)

Geeft de numerieke waarde van de alfanumerieke uitdrukking X$.

## VARPTR — functie

VARPTR(<var.>)
VARPTR(#<X>)

Het eerste formaat geeft het eerste adres van de aangegeven variabele. Het tweede formaat geeft het eerste adres van het met X aangegeven file control blok.

Het resultaat van de functie zal een waarde tussen -32768 en 32767 zijn. Indien de waarde negatief is moet er 65536 worden bijgeteld om het werkelijke adres te verkrijgen.

VDP — systeemvariabele

VDP(<X>)

Bevat de inhoud van registernummer X van de video display processor.
<X> mag een waarde van 0 tot 8 hebben.
Register 8 mag alleen worden gelezen.
Zie ook hoofdstuk 15, Video Display Processor.

VPEEK — functie

VPEEK(<X>)

Leest de inhoud van het adres X uit het video geheugen.
<X> mag varieren van 0 t/m 16383.

Het resultaat zal een waarde van 0 t/m 255 zijn.

VPOKE — statement

VPOKE <X>,<Y>

Schrijft de waarde Y naar het videogeheugenadres X.
<X> mag varieren van 0 t/m 16383.
<Y> mag varieren van 0 t/m 255.

WAIT — statement

WAIT <X>,<Y>[,<Z>]

Vergelijkt de inhoud van input-poort X met de integer expressies Y en Z. Deze vergelijkingen worden als volgt uitgevoerd:

```
X XOR Y
X AND Z
```

Indien het resultaat van de vergelijking 0 is, wordt de vergelijking nogmaals uitgevoerd. Zo niet, dan wordt het programma voortgezet.

## WIDTH — statement

WIDTH <X>

Geeft aan dat het aantal karakters dat in tekstmodes 1 en 2 op een regel van het beeldscherm kan worden geschreven gelijk is aan X.

# 8 Systeemboodschappen

In dit hoofdstukje zullen de systeemboodschappen op drie manieren worden weergegeven.

1. Op alfabetische volgorde, gevolgd door het bijbehorend boodschapnummer.
2. Op volgorde van boodschapnummer, gevolgd door de boodschap zelf.
3. Op alfabetische volgorde, gevolgd door een nadere toelichting.

Mocht u tijdens het uitvoeren van een programma een systeemboodschap krijgen, dan kunt u uit de eerste lijst het bijbehorende boodschapnummer halen. Dit nummer kunt u gebruiken om in een foutafhandelingsroutine te gebruiken.

Tijdens het ontwikkelen van een programma, en speciaal tijdens het schrijven van de foutafhandelingsroutine daarin, kunt u zowel lijst 1 als lijst 2 gebruiken. Mocht u nadere informatie wensen over een systeemboodschap, dan zal lijst 3 u van dienst kunnen zijn.

Lijst 1

| Systeemboodschap | nummer | Systeemboodschap | nummer |
|---|---|---|---|
| Bad drive name | 62 | File not found | 53 |
| Bad FAT | 60 | File not open | 59 |
| Bad file mode | 61 | File still open | 64 |
| Bad file name | 56 | Illegal direct | 12 |
| Bad file number | 52 | Illegal function call | 5 |
| Bad sector number | 63 | Input past end | 55 |
| Can't continue | 17 | Internal error | 51 |
| Device I/O error | 19 | Line buffer overflow | 25 |
| Direct statement | 57 | Missing operand | 24 |
| Disk error | (geen nr) | NEXT without FOR | 1 |
| Disk full | 66 | No RESUME | 21 |
| Disk I/O error | 69 | Out of DATA | 4 |
| Disk offline | 70 | Out of memory | 7 |
| Disk write protected | 68 | Out of string space | 14 |
| Division by zero | 11 | Overflow | 6 |
| Field overflow | 50 | Redimensioned array | 10 |
| File already exists | 65 | Rename across disk | 71 |
| File already open | 54 | RESUME without error | 22 |

| Systeemboodschap | nummer | Systeemboodschap | nummer |
|---|---|---|---|
| RETURN without GOSUB | 3 | Too many files | 67 |
| Sequential I/O only | 58 | Type mismatchs | 13 |
| String formula too complex | 16 | Undefined line nr. | 8 |
| String too long | 15 | Undefined user func. | 18 |
| Subscript out of range | 9 | Unprintable errors | 23 |
| Syntax error | 2 | | 26-49 |
| | | | 72-255 |
| | | Verify error | 20 |

Lijst 2

| Nr | Systeemboodschap | Nr | Systeemboodschap |
|---|---|---|---|
| 1 | NEXT without FOR | 25 | Line buffer overflow |
| 2 | Syntax error | 26- | |
| 3 | RETURN without GOSUB | 49 | Unprintable errors |
| 4 | Out of DATA | 50 | Field overflow |
| 5 | Illegal function call | 51 | Internal error |
| 6 | Overflow | 52 | Bad file number |
| 7 | Out of memory | 53 | File not found |
| 8 | Undefined line number | 54 | File already open |
| 9 | Subscript out of range | 55 | Input past end |
| 10 | Redimensioned array | 56 | Bad file name |
| 11 | Division by zero | 57 | Direct statement |
| 12 | Illegal direct | 58 | Sequential I/O only |
| 13 | Type mismatch | 59 | File not open |
| 14 | Out of string space | 60 | Bad FAT |
| 15 | String too long | 61 | Bad file mode |
| 16 | String formula too complex | 62 | Bad srive name |
| | | 63 | Bad sector number |
| 17 | Can't continue | 64 | File still open |
| 18 | Undefined user func. | 65 | File already exists |
| 19 | Device I/O error | 66 | Disk full |
| 20 | Verify error | 67 | Too many files |
| 21 | No RESUME | 68 | Disk write protected |
| 22 | RESUME without error | 69 | Disk I/O error |
| 23 | Unprintable error | 70 | Disk off line |
| 24 | Missing operand | 71 | Rename across disk |

Aan de nummers 72 tot en met 255 zijn nog geen boodschappen toegekend. De programmeur mag aan deze nummers zelf een boodschap koppelen.

Lijst 3

Bad drive name
De in het statement opgegeven diskette drive is niet in gebruik.

Bad FAT
De disketten waarop I/O wordt gewenst, is nog niet geformateerd.

Bad file mode
De statements PUT, GET of LOF zijn gebruikt op een sequentieel bestand.
Het statement LOAD is gebruikt op een "random access" bestand.
Het bestand wordt met een niet toegestane mode geopend.

Bad file name
De bestandsaanduiding bevat syntax fouten.

Bad file number
De aangegeven file is niet onder het aangegeven file nummer geopend, of het file nummer is hoger dan volgens "MAXFILES" is toegestaan.

Bad sector number
Het in PUT of GET gebruikte bloknummer is kleiner dan 1 of groter dan 32767.

Can't continue
Er is geen programma in het geheugen, of het programma werd onderbroken door een foutboodschap, of het programma werd, nadat het was onderbroken, gewijzigd.

Device I/O error
Tijdens het lezen of schrijven van gegevens of programma's is een lees- of schrijffout ontdekt.

Direct statement
Tijdens het laden van een ASCII-file is een direct commando gezien.

Disk error
Tijdens het formateren van een diskette is een lees- of schrijffout ontdekt.

Disk full
Er zijn geen lege sectoren meer op de diskette. Het programma kan worden vervolgd door op de RETURN-toets te drukken, doch de data is nog niet naar diskette geschreven.

Disk I/O error
Tijdens het transporteren van gegevens van of naar de disk zijn fouten geconstateerd, die het transport verhinderen.

Disk off line
De disk-drive is niet aangeschakeld.

Disk write protected
De diskette kan niet worden beschreven doordat de "write protect tab" in de stand "write protect" staat.

Division by zero
In een door de computer uit te voeren berekening blijkt dat er een getal door nul moet worden gedeeld. Dit zou een oneindig groot resultaat geven, hetgeen niet door de computer kan worden verwerkt.

Field overflow
Met het FIELD statement worden meer dan 256 bytes toegekend, of het einde van het FIELD-buffer is bereikt sequential I/O naar een random file.

File already exists
De nieuwe bestandsnaam in het NAME-statement is de naam van een reeds bestaand bestand.

File already open
De file die met het OPEN-statement wordt getracht te openen blijkt al geopend te zijn.

File not found
Het bestand dat in een LOAD, KILL of OPEN statement wordt aangeduid blijkt niet op diskette te staan.

File not open
Het bestand waar naartoe wordt geschreven, of waarvan wordt gelezen is nog niet geopend.

File still open
Het bestand is nog niet afgesloten (met CLOSE of END).

Illegal direct
In de directe mode is getracht een statement uit te voeren dat niet in die mode mag worden uitgevoerd.

Illegal function call
De aangeroepen functie is op de verkeerde manier aangeroepen. Voorbeelden hiervan zijn:
- Een negatieve of te grote subscript.
- Bij de LOG-functie werd de parameter 0 of een negatieve parameter opgegeven.
- Bij de SQR-functie werd een negatieve parameter opgegeven.
- Bij MID$, LEFT$, RIGHT$, INP, OUT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR$, ON-GOTO of ON-GOSUB werd een verkeerde parameter opgegeven.

Input past end
Er wordt getracht, voorbij het einde van een bestand, nog data uit dat bestand te lezen. Dit probleem kan worden voorkomen door de EOF-functie in het programma te gebruiken.

Internal error
Er is een fout opgetreden die niet door het systeem kan worden opgelost. Er is mogelijk sprake van een hardware fout of van een fout in het MSX-software systeem.

Line buffer overflow
Er zijn te veel karakters ingetikt (>255).

Missing operand
Een statement, commando of functie heeft geen parameter die het wel zou moeten hebben.

NEXT without FOR
Er moet een NEXT-statement worden uitgevoerd, echter, er is geen (bijbehorende) FOR-statement uitgevoerd.

No RESUME
Een foutafhandelings-subroutine is gestart, doch wordt niet afgesloten met een RESUME-statement.

Out of DATA
Bij het uitvoeren van een READ-statement wijst de "pointer" voorbij het laatste DATA-element van de laatste DATA-regel. Ofwel wordt de READ-statement te vaak uitgevoerd, of het RESTORE-statement is vergeten.

Out of memory
Het programma is te groot, of bevat te veel FOR-lussen of GOSUB-statements, waardoor er "stack"-ruimte moet worden gecreeerd, terwijl daar geen ruimte meer voor is. Ook een te gecompliceerde expressie kan dit veroorzaken.

Out of string space
De voor de variabelen beschikbare geheugenruimte kan niet alle variabelen bevatten. Deze geheugenruimte kan met het CLEAR-statement worden vergroot.

Overflow
Het resultaat van een berekening is groter dan kan worden verwerkt.

Redimensioned array
Een reeds bestaande variabele, of een reeds gedimensioneerde array, mag niet (opnieuw) gedimensioneerd worden.

Rename across disk
Zowel de oude als de nieuwe bestandsnaam moet slaan op bestanden die op dezelfde disk staan. De disk mag tijdens het uitvoeren van het NAME-statement niet worden verwisseld.

RESUME without error
De computer komt, in het programma, een RESUME-statement tegen, zonder dat er een ON ERROR GOTO was uitgevoerd.

RETURN without GOSUB
De computer komt, in het programma, een RETURN-statement tegen, zonder dat er een GOSUB-statement was uitgevoerd.

Sequential I/O only
Er wordt getracht om met behulp van de GET en PUT statements, die alleen mogen worden gebruikt op random toegankelijke bestanden, gegevens uit een sequentieel bestand te lezen of er naar toe te schrijven.

String formula too complex
De string-expressie is te lang of te complex, en moet derhalve in kleinere (eenvoudiger) stukken worden opgedeeld.

String too long
Het aantal karakters dat aan een string mag worden toegekend is maximaal 255.

Subscript out of range
De subscript waarnaar wordt gerefereerd, is kleiner of groter dan de bij de array opgegeven subscripts.

Syntax error
Er is een schrijffout gemaakt in een statement, commando of functie. Check de syntaxbeschrijving van het MSX-BASIC.

Too many files
Er mogen maximaal 112 files tegelijk aanwezig zijn.

Type mismatch
Aan een numerieke variabele zijn alfanumerieke karakters toegekend, of aan een functie, die een numerieke parameter verwacht is een alfanumerieke parameter gekoppeld.

Undefined line number
Het opgegeven regelnummer komt niet in het programma voor.

Undefined user function
Er werd gerefereerd naar een user functie die niet bestaat, of die niet met DEF is gedefinieerd.

Unprintable errors
Een aantal codes hebben geen bijbehorende boodschap. In plaats daarvan wordt dan de boodschap "unprintable errors" afgedrukt.

Verify error
De gegevens die met het commando "CLOAD?" van cassette zijn gelezen, zijn niet gelijk aan de in het geheugen staande gegevens. Dit houdt in dat de gegevens op de cassette niet correct zijn. (Opnieuw wegschrijven met SAVE.)

# 9 Editing

De MSX-computer is voorzien van en full-screen editor. Dit wil zeggen dat alle op het scherm staande programma regels naar believen kunnen worden gewijzigd. Hiertoe heeft de gebruiker een aantal toetsen tot zijn beschikking. Het aantal mogelijkheden is zo groot, dat het niet doenlijk was voor iedere mogelijkheid een aparte functietoets op het toetsenbord aan te brengen. U zult dan ook voor een aantal mogelijkheden twee toetsen tegelijk dienen in te drukken, de CONTROL toets plus een lettertoets.

In de hierna volgende tabel zijn alle mogelijkheden opgenomen. Eerst de functies waarvoor aparte toetsen aanwezig zijn, daarna de gecombineerde aanslagen.

| toets | functie |
|---|---|
| BS | Back Space. Hiermee wordt het laatst ingetikte karakter gewist. |
| CLS | CLear Screen. Door indrukken van de CLS/HOME-toets samen met de SHIFT-toets wordt het beeldscherm schoongemaakt. |
| DEL | DELete. Hiermee wordt het karakter dat onder de cursor staat gewist en schuift de tekst rechts van de cursor 1 plaatsje naar links. |
| HOME | Hiermee wordt de cursor naar de eerste beeld-schermpositie verplaatst (linksbovenaan). |
| INS | INSert. Na het intikken van deze toets kunnen karakters in de tekst worden tussengevoegd. Deze karakters zullen dan op de plaats van de cursor terecht komen, terwijl alle karakters vanaf de cursor naar rechts zullen schuiven. De INS-mode wordt verlaten door het activeren van een andere functie. |
| Pijltjes | De pijltjestoetsen dienen voor het verplaatsen van de cursor. De cursor kan naar iedere gewenste plaats van het beeldscherm worden gebracht (in de richting van de pijltjes). |

| toets | functie |
|---|---|
| RETURN | Hiermee worden de op het scherm aangebrachte wijzigingen in het programmageheugen opgenomen |
| TAB | Hiermee wordt de cursor naar de volgende kolom verplaatst. Het beeldscherm is opgedeeld in kolommen van ieder 16 karakters breed. |

Nu volgt een lijst van toetsen die tesamen met de CONTROL-toets (CTRL) moeten worden ingedrukt.

| toets | functie |
|---|---|
| B | Hiermee wordt de cursor naar het begin van het vorige woord verplaatst. |
| C | Hiermee wordt de ingave onderbroken. De ingetikte wijzigingen worden niet in het programmageheugen opgenomen. |
| E | Hiermee wordt alle tekst, op de zelfde programmaregel,die de cursor volgt, gewist. |
| F | Hiermee wordt de cursor naar het begin van het volgende woord verplaatst. |
| J | Hiermee wordt de cursor naar de volgende regel verplaatst. Mocht de cursor al op de laatste regel van het beeldscherm staan, dan zal de tekst een regel omhoog "scrollen". |
| N | Hiermee wordt de cursor naar het einde van de programmaregel verplaatst. |
| U | Hiermee wordt de hele programmaregel gewist. |

# 10 MSX-karakterset

| BIT 0,1,2,3 | | | | HEX 1/2 | BIT 4,5,6,7 | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | | | | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | 0 | 0 | 0 | 0 | Null | ┼ | Spatie | θ | @ | P | ` | p | ç | É | á | Ã | | | α | ≡ |
| 0 | 0 | 0 | 1 | 1 | ☺ | ┤ | ! | 1 | A | Q | a | q | Ü | æ | í | ã | ▚ | ▞ | β | ± |
| 0 | 0 | 1 | 0 | 2 | ☻ | ├ | " | 2 | B | R | b | r | e | Æ | ó | Ĩ | ▚ | ▞ | γ | ≥ |
| 0 | 0 | 1 | 1 | 3 | ♥ | ┬ | # | 3 | C | S | c | s | ä | ð | ú | ĩ | ▀ | ▟ | π | ≤ |
| 0 | 1 | 0 | 0 | 4 | ♦ | ┴ | $ | 4 | D | T | d | t | ã | ð | ñ | Õ | ▬ | ▙ | Σ | ⌠ |
| 0 | 1 | 0 | 1 | 5 | ♣ | ┼ | % | 5 | E | U | e | u | à | ð | Ñ | õ | ▙ | ▗ | σ | ⌡ |
| 0 | 1 | 1 | 0 | 6 | ♠ | ├ | & | 6 | F | V | f | v | ã | û | a | Ũ | ▙ | ▟ | μ | ÷ |
| 0 | 1 | 1 | 1 | 7 | • | ─ | ' | 7 | G | W | g | w | ç | ù | º | ũ | ▚ | ▞ | τ | ≈ |
| 1 | 0 | 0 | 0 | 8 | ◘ | ┘ | ( | 8 | H | X | h | x | e | ÿ | ¿ | Ð | ▙ | ▽ | φ | ° |
| 1 | 0 | 0 | 1 | 9 | ○ | ┐ | ) | 9 | I | Y | i | y | e | Ö | ⌐ | ij | ▗ | ǂ | ~~θ~~ | • |
| 1 | 0 | 1 | 0 | A | ◙ | ┌ | * | : | J | Z | j | z | e | Ü | ¬ | ¼ | ▙ | ₩ | Ω | ● |
| 1 | 0 | 1 | 1 | B | ♂ | ┘ | + | ; | K | [ | k | { | ĩ | ¢ | ½ | ~ | ▞ | ▟ | δ | √ |
| 1 | 1 | 0 | 0 | C | ♀ | ┌ | , | < | L | \ | l | ¦ | î | £ | ¼ | ◇ | ▟ | ▗ | ∞ | η |
| 1 | 1 | 0 | 1 | D | ♪ | └ | - | = | M | ] | m | } | ì | ¥ | i | ‰ | ▙ | ▗ | φ | ² |
| 1 | 1 | 1 | 0 | E | ♫ | ┼ | . | > | N | ^ | n | ~ | Ä | Pt | ≪ | ¶ | ▙ | ▗ | ∈ | ■ |
| 1 | 1 | 1 | 1 | F | ☼ | ┼ | / | ? | O | _ | o | △ | Å | ƒ | ≫ | § | ▙ | ▗ | ∩ | Trans-parant |

# 11 ASCII-karakterset

De meest gebruikte Internationale gestandaardiseerde code voor karakters is de ASCII-code. ASCII is de afkorting van American Standard Code for Information Interchange. De hiernavolgende tabel kan als volgt worden gebruikt:

Opzoeken welke code bij een gegeven karakter hoort.

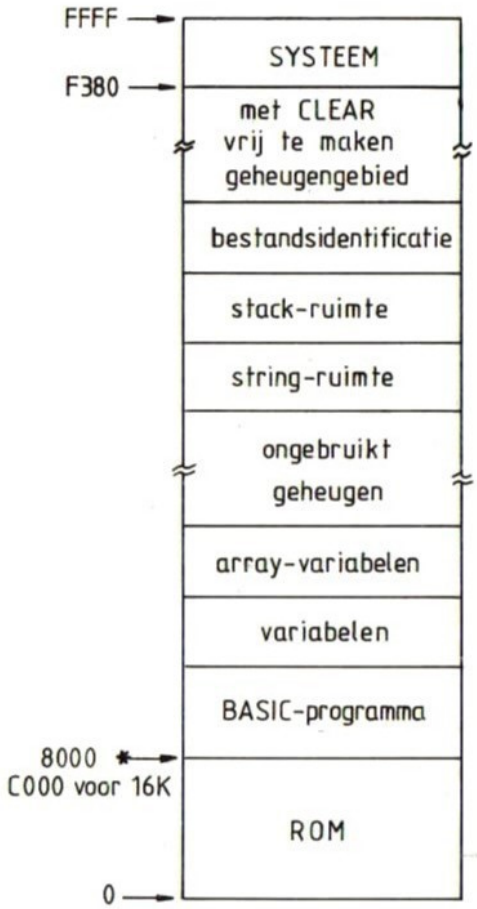
Stel u zoekt de code die bij de letter S behoort. Zoek dan de letter S op in de tabel. Kijk nu welk getal er boven de betreffende kolom staat. Dit is het meest significante deel van de code. Kijk nu welk getal er links van de betreffende kolom staat. Dit is het minst significante deel van de code. Voor de letter S vindt u zodoende de code Hex. 53, of Bin. 0101 0011.

Opzoeken welke letter bij een gegeven code hoort.

Stel u hebt de code hex. 54. Zoek eerst de kolom waarboven de waarde 5 staat (langs de bovenrand). Ga nu zover naar beneden tot u de rij hebt bereikt waarnaast (aan de linker kant) de waarde hex. 4 staat. U hebt dan het vak bereikt waarin de gezochte letter staat. In dit vak vindt u de letter T.

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

# 12 Geheugen lay-out




* 32K ROM + 16K RAM dan hier een niet gebruikte ruimte van 16K.

# 13 I/O-adressen

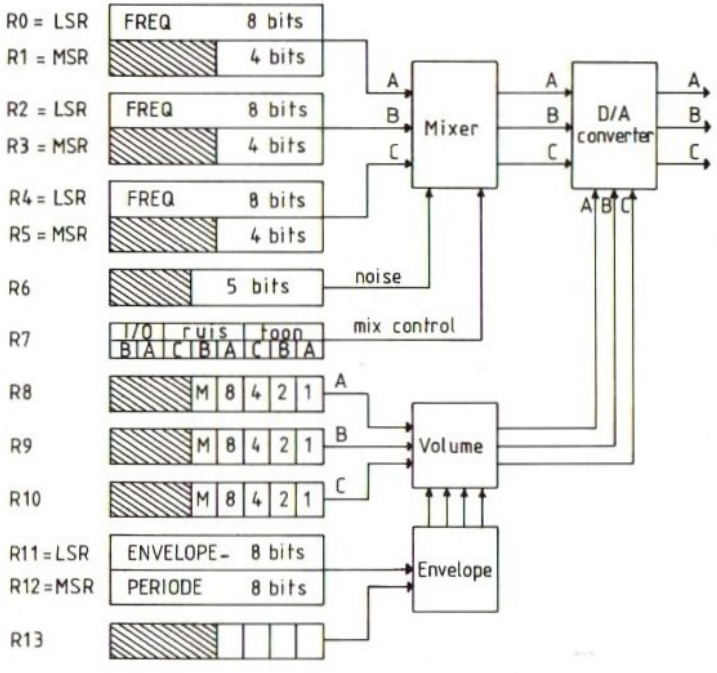
| device | Adres | R/W | Omschrijving | Opmerkingen |
|---|---|---|---|---|
| RS232C | &H80 | W | Data write | 8251A |
| | &H80 | R | Data read | (USART) |
| | &H81 | W | Mode setting | |
| | &H81 | R | Status read | |
| Printer | &H90 | W | Strobe output | Centronics |
| | &H90 | R | Status input | interface |
| | &H91 | W | Print data | |
| VDP | &H98 | W | Write data to VRAM | TMS9129NL |
| | &H98 | R | Read data from VRAM | |
| | &H99 | W | Command/Address set | |
| | &H99 | R | Status read | |
| PSG | &HA0 | W | Address latch | AY-3-8910 |
| | &HA1 | W | Data write | |
| | &HA2 | R | Data read | |
| PPI | &HA8 | W | Write data (port A) | 8255A |
| | &HA8 | R | Read data (port A) | |
| | &HA9 | W | Write data (port B) | |
| | &HA9 | R | Read data (port B) | |
| | &HAA | W | Write data (port C) | |
| | &HAA | R | Read data (port C) | |
| | &HAB | W | Mode set | |

Overige adres gebieden

```
&H00 - &H7F   Niet gedefinieerd (vrij).
&HB0 - &HCF   Gereserveerd voor het systeem.
&HD0 - &HD7   Gereserveerd voor floppy disk controller.
&HD8 - &HFF   Niet gedefinieerd (vrij).
```

# 14 De programmeerbare geluidsgenerator

In plaats van het PLAY-statement, kan men ook gebruik maken van het SOUND-statement. Hiermee zijn meer en gevarieerdere geluidseffecten te verkrijgen. Om een goed gebruik van het SOUND-statement te kunnen maken dient men echter iets meer van de geluidsgenerator te weten. De hierna volgende afbeelding laat, als het ware, de opbouw van de generator zien.



Aan de linkerkant in de afbeelding van de geluidsgenerator ziet u de te programmeren registers. Deze registers zijn genummerd van R0 tot en met R15. R14 en R15 hebben geen geluidsfunctie. Zij worden gebruikt voor de joystick-interface. Zie hiervoor het hoofdstukje over de I/O-adressen.

In de afbeelding en de hiernavolgende uitleg zult u de afkortingen RP, MSR en LSR tegenkomen. Deze afkortingen staan voor Register Paar, Meest Significant Register en Minst (Least) Significant Register.

De registers kunnen worden geladen door middel van een SOUND-statement. Register 7 dient als laatste te worden geladen. Met register 7 kunnen de verschillende geluids kanalen aan of uit worden gezet. Daarbij geldt dat, indien een van de bitjes een nul is, de betreffende functie is aangeschakeld, terwijl een bitje dat de waarde 1 krijgt de betreffen de functie uitschakelt. Hier volgt een korte verklaring van register 7.

| I/O | | NOISE | | | TOON | | |
|---|---|---|---|---|---|---|---|
| A | B | C | B | A | C | B | A |

```
128 64 32 16 8  4  2  1 <-- decimale waarde van het bit
                       |
                       |---- 0 = geluidskanaal A aan.
                             1 = geluidskanaal A uit.
                          |------- 0 = geluidskanaal B aan.
                                 1 = geluidskanaal B uit.
                             |---------- 0 = geluidskanaal C aan.
                                       1 = geluidskanaal C uit.
                          |------------- 0 = ruis op kanaal A.
                                       1 = geen ruis op kanaal A.
                       |---------------- 0 = ruis op kanaal B.
                                       1 = geen ruis op kanaal B.
                    |--------------------0 = ruis op kanaal C.
                                       1 = geen ruis op kanaal C.
                 |----------------------- 0 = I/O-poort A niet actief
                                       1 = I/O-poort A actief.
              |------------------------- 0 = I/O-poort B niet actief
                                       1 = I/O-poort B actief.
```

Om dus een toon op kanaal A te laten klinken en tegelijkertijd een ruis op kanaal B te laten klinken, zullen de bitjes 0 en 4 laag dienen te zijn. Alle andere bits mogen hoog zijn. dit zou een decimale

waarde geven van 128+64+32+8+4+2=238. Met SOUND 7,238 zouden de gewenste instellingen worden bereikt.

Het instellen van het volume van de verschillende kanalen kan worden gedaan door de registers R8, R9 en R10 met een waarde van 0 tot en met 15 te laden. Hoe hoger de waarde, hoe sterker het volume. Door de waarde 16 in te geven, wordt het bitje dat is aangeduid met M hoog gemaakt. Hieruit volgt dat het volume voor het betreffende kanaal op Maximaal staat.

Het instellen van de frequentie van de toon kan worden bereikt door de registers R0 tot en met R5 met een bepaalde waarde te laden. R0 en R1 zijn voor kanaal A, R2 en R3 voor kanaal B en R4 en R5 voor kanaal C. Om een kanaal een gewenste frequentie te laten produceren, dient men een waarde voor de betreffende registers te berekenen. De berekende waarde moet dan worden verdeeld over de beide registers. Hiervoor kunt u de volgende methode gebruiken.

```
RP  = INT (111860/<frequentie>)

MSR = INT (RP/256)

LSR = RP - 256 * MSR
```

LSR kunt u gebruiken voor de registers R0, R2 of R4. MSR laadt u dan in de bijbehorende registers R1, R3 of R5. Een voorbeeldje moge een en ander duidelijk maken. Stel we willen register B een toon van 1000 Herz laten maken. De berekening gaat dan als volgt:

```
RP  = INT (111860/1000) = 111

MSR = INT (111/256) = 0

LSR = RP - 256 * MSR = 111 - 0 = 111
```

Nu geeft u twee SOUND statements,

```
SOUND 2,111
SOUND 3,0
```

Door hierna kanaal B te activeren (R7), zal er een geluid van 1000 Hz worden geproduceerd.

Het zal u opgevallen zijn dat deze berekening zich uitstekend leent voor automatiseren. Met het volgende

BASIC-programma kunt u iedere gewenste frequentie laten omzetten naar registerinhouden voor de registers LSR en MSR. Het programma vraagt eerst naar de beginfrequentie, vervolgens naar de eindfrequentie en tenslotte naar de stapgrootte. Daarna krijgt u een tabel met registerwaarden per frequentie.

HOOFDDEEL

```
10 '*******************************
20 '*Berekenen inhoud van de toon-*
30 '*registers 0 t/m 15 van de PSG*
40 '*******************************
50 '
60 CLS
70 INPUT "DE LAAGSTE FREQUENTIE";L
80 INPUT "DE HOOGSTE FREQUENTIE";H
90 INPUT "DE STAPGROOTTE IN HZ.";S
100 PRINT
110 FOR I=L TO H STEP S
120 P=INT(1789770#/(16*I))
130 R1=INT(P/256)
140 R0=P-R1*256
```

plus

TABEL

```
150 PRINT "F=";USING"####";I;
160 PRINT "  R0=";USING "###";R0;
170 PRINT "  R1=";USING "##";R1
180 NEXT I
190 END
```

of

GELUID

```
150 SOUND 0,R0
160 SOUND 1,R1
170 SOUND 7,62
180 SOUND 8,15
190 NEXT I
200 END
```

Indien u ruis wenst, kunt u in register 6 de frequentie van die ruis opgeven. Ook daarvoor zou weer een berekening te maken zijn. Er zijn echter maar 31 mogelijkheden. De volgende tabel geeft de mogelijke register inhoud van R6 weer, met daarachter de bijbehorende frequentie.

| Inhoud R6 | Ruisfrequentie |
|---|---|
| 1 | 111860 Hz |
| 2 | 55930 Hz |
| 3 | 37286 Hz |
| 4 | 27965 Hz |
| 5 | 22372 Hz |
| 6 | 18643 Hz |
| 7 | 15980 Hz |
| 8 | 13982 Hz |
| 9 | 12428 Hz |
| 10 | 11186 Hz |
| 11 | 10169 Hz |
| 12 | 9321 Hz |
| 13 | 8604 Hz |
| 14 | 7990 Hz |
| 15 | 7457 Hz |
| 16 | 6991 Hz |
| 17 | 6580 Hz |
| 18 | 6214 Hz |
| 19 | 5887 Hz |
| 20 | 5593 Hz |
| 21 | 5326 Hz |
| 22 | 5084 Hz |
| 23 | 4863 Hz |
| 24 | 4660 Hz |
| 25 | 4474 Hz |
| 26 | 4302 Hz |
| 27 | 4142 Hz |
| 28 | 3995 Hz |
| 29 | 3857 Hz |
| 30 | 3728 Hz |
| 31 | 3608 Hz |

De inhoud van de registers waarmee de envelop-generator wordt gestuurd kan als volgt worden berekend:

RP = ‹tijd in seconden› * 6991

R12 = INT (RP/256)

R11 = RP - 256 * R12

Ook hier weer een voorbeeldje ter verduidelijking. Stel we wensen een envelop-tijd van 4 seconden. We krijgen:

```
RP = 4 * 6991 = 27964

R12 = INT (27964/256) = 109

R11 = 27964 - 109 * 256 = 27964 - 27904 = 60
```

Het zetten van de envelop-tijd wordt dan met twee SOUND statements als volgt gedaan,

```
SOUND 11,60
SOUND 12,109
```

De volgende tabel zal waarschijnlijk de meest gebruikte waarden al bevatten, zodat u niet vaak meer zult hoeven te rekenen.

| Tijd in seconden | Inhoud van R11 | en R12 |
|---|---|---|
| 9 | 199 | 245 |
| 8 | 120 | 218 |
| 7 | 41 | 191 |
| 6 | 218 | 163 |
| 5 | 139 | 136 |
| 4 | 60 | 109 |
| 3 | 237 | 81 |
| 2 | 158 | 54 |
| 1 | 79 | 27 |
| 0.9 | 147 | 24 |
| 0.8 | 216 | 21 |
| 0.7 | 29 | 19 |
| 0.6 | 98 | 16 |
| 0.5 | 167 | 13 |
| 0.4 | 236 | 10 |
| 0.3 | 49 | 8 |
| 0.2 | 118 | 5 |
| 0.1 | 187 | 2 |

# 15 De video display processor

De Video Display Processor (VDP) heeft zijn eigen 16k RAM geheugen. De manier waarop dit geheugendeel wordt gebruikt is afhankelijk van de mode waarin de computer werkt. Er zijn vier modes mogelijk, die als volgt kunnen worden bereikt:

```
SCREEN 0 = tekst mode 1
SCREEN 1 = tekst mode 2
SCREEN 2 = grafische mode 1
SCREEN 3 = grafische mode 2
```

De karakteristieken van ieder van deze modes zijn:

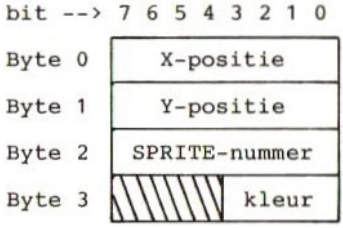
```
Tekst mode 1
Het scherm bestaat uit 24 regels van 40 posities.
BASE(0) = Het eerste adres van de NAME TABLE.
          Lengte van deze tabel is 960 bytes (24*40).
BASE(2) = Het eerste adres van de PATTERN TABLE.
          Lengte van deze tabel is 2048 bytes (256*8).

Tekst mode 2
Het scherm bestaat uit 24 regels van 32 posities.
BASE(5) = Het eerste adres van de NAME TABLE.
          Lengte van deze tabel is 768 bytes (24*32)
BASE(6) = Het eerste adres van de COLOUR TABLE.
          Lengte van deze tabel is 32 bytes (256/8)
BASE(7) = Het eerste adres van de PATTERN TABLE.
          Lengte van deze tabel is 2048 bytes (256*8)
BASE(8) = Het eerste adres van de SPRITE ATTR. TABLE.
          Lengte van deze tabel is 128 bytes (4*32).
BASE(9) = Het eerste adres van de SPRITE PATTERN TABLE.
          Lengte van deze tabel is 2048.

Grafische mode 1
Het scherm bestaat uit 3 * 8 regels van 32 posities.
BASE(10)= Het eerste adres van de NAME TABLE.
          Lengte van deze tabel is 768 bytes (3*8*32).
BASE(11)= Het eerste adres van de COLOUR TABLE.
          Lengte van deze tabel is 6144 bytes (768*8).
BASE(12)= Het eerste adres van de PATTERN TABLE.
          Lengte van deze tabel is 768 bytes (3*8*32).
BASE(13)= Het eerste adres van de SPRITE ATTR. TABLE.
          Lengte van deze tabel is 128 bytes (4*32).
BASE(14)= Het eerste adres van de SPRITE PATTERN TABLE.
          Lengte van deze tabel is 2048 bytes.
```

Grafische mode 2
Het scherm bestaat uit 24 regels van 32 posities.
Iedere positie bestaat uit 4 vierkantjes van 4*4 pixels die elk een afzonderlijke kleur kunnen hebben (er zijn 16 verschillende kleuren mogelijk).
BASE(15)= Het eerste adres van de NAME TABLE.
Lengte van deze tabel is 768 bytes (24*32).
BASE(17)= Het eerste adres van de PATTERN TABLE.
Lengte van deze tabel is 2048 bytes (8*256).
BASE(18)= Het eerste adres van de SPRITE ATTR. TABLE.
Lengte van deze tabel is 128 bytes (4*32).
BASE(19)= Het eerste adres van de SPRITE PATTERN TABLE.
Lengte van deze tabel is 2048 bytes.

Iedere SPRITE ATTRIBUTE bestaat uit vier bytes, die de volgende betekenis hebben.




Uitlezen van items uit de verschillende tabellen kan worden gedaan met de VPEEK-functie. Laden van een waarde in een van de tabellen kan worden gedaan met het VPOKE-statement. Voorbeeld:

```
1000 SCREEN 1,1:WIDTH 30
1010 A=1
1020 FOR I=0 TO 7
1030 VPOKE(BASE(9)+I),A
1040 A=A+1
1050 NEXT I
```

Hiermee wordt in tekstmode 1 een SPRITE PATTERN geladen in de SPRITE PATTERN TABLE.

```
1060 VPOKE(BASE(8)+2),0
1070 VPOKE(BASE(8)+3),1
1080 FOR I=0 TO 255 STEP 7
1090 FOR J=0 TO 191
```

```
1100 VPOKE(BASE(8)+0),J
1110 VPOKE(BASE(8)+1),I
1120 NEXT J
1130 NEXT I
1140 END
```

Hiermee wordt, door beinvloeden van de SPRITE ATTRIBUTE TABLE, de positie en de kleur van SPRITE 0 bepaald. U ziet bovendien dat het op deze manier mogelijk is om in tekstmode 2 SPRITES te gebruiken. Om dit duidelijk uit te laten komen kan nog wat tekst worden toegevoegd aan het programma, door een regel 1055 toe te voegen:

```
1055 FOR I=0 TO 23:PRINT "TEKST MODE 2":NEXT I
```

Net als bij de programmeerbare geluidsgenerator, kunnen ook bij de video processor de registers in de chip rechtstreeks worden geladen. Het toekennen van waarden aan de registers kan worden gedaan met de systeem variabele VDP. Zouden we bijvoorbeeld aan register 6 de waarde 1 willen toekennen, dan dienen we VDP(6)=1 te programmeren. Om, bijvoorbeeld register 8 uit te lezen, dienen we PRINT VDP(8) te programmeren. Hierna volgt een overzicht van de registers en de mogelijke inhoud daarvan.

| msb | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | lsb |
|---|---|---|---|---|---|---|---|---|---|
| R0 | 0 | 0 | 0 | 0 | 0 | 0 | A | D | |
| R1 | 1 | 0 | E | B | C | R | S | M | |
| R2 | 0 | 0 | 0 | 0 | NAME TABLE | | | | |
| R3 | COLOUR TABLE | | | | | | | | |
| R4 | 0 | 0 | 0 | 0 | 0 | PATTERN TABLE | | | |
| R5 | 0 | SPRITE ATTRIBUTE TABLE | | | | | | | |
| R6 | 0 | 0 | 0 | 0 | 0 | SPRITE PATTERN TABLE | | | |
| R7 | TEKSTKLEUR 1 | | | | TEKSTKLEUR 2 | | | | |
| R8 | T | N | V | SPRITE NUMMER (vijfde) | | | | | |

| A | B | C | |
|---|---|---|---|
| 0 | 0 | 0 | tekstmode 2 |
| 1 | 0 | 0 | grafische mode 1 |
| 0 | 0 | 1 | grafische mode 2 |
| 0 | 1 | 0 | tekstmode 1 |

| D | |
|---|---|
| 0 | geen externe VDP input |
| 1 | externe input |

| E | |
|---|---|
| 0 | geen VDP interrupt |
| 1 | VDP interrupt |

R gereserveerd

| S | |
|---|---|
| 0 | sprites van 8*8 |
| 1 | sprites van 16*16 |

| M | |
|---|---|
| 0 | sprites normaal |
| 1 | sprites zijn vergroot |

De velden NAME TABLE, COLOUR TABLE, PATTERN TABLE, SPRITE ATTRIBUTE TABLE en SPRITE PATTERN TABLE bevatten de beginadressen van de respectievelijke tabellen.

De velden TEKSTKLEUR 1 en 2 bevatten respectievelijk de foreground en background kleuren (in tekstmode 1).

| T | |
|---|---|
| 0 | register 8 is gelezen of de VDP is gereset. |
| 1 | de laatste rasterlijn van het scherm is geschreven. |

| N | |
|---|---|
| 0 | register 8 is gelezen of de VDP is gereset. |
| 1 | twee sprites overlappen elkaar. |

| V | |
|---|---|
| 0 | register 8 is gelezen of de VDP is gereset. |
| 1 | er bevinden zich 5 of meer sprites op een horizontale pixel-lijn. In dat geval wordt het nummer van de vijfde sprite in het veld SPRITE NUMMER gegeven. |

Indien u de startadressen van de tabellen in het video RAM een andere waarde wilt geven dan de door MSX-BASIC gegeven waarden, dan mag dat, mits de volgende regels in acht worden genomen.

Iedere tabel mag op adres 0 beginnen of op een adres dat een of meer van de hierna te geven eenheden daarop volgt:

| Tabel | eenheid |
|---|---|
| NAME TABLE | 1024 |
| COLOUR TABLE | 64 |
| PATTERN TABLE | 2048 |
| SPRITE ATTRIBUTE TABLE | 128 |
| SPRITE PATTERN TABLE | 2048 |

# 16 Z80 interrupt modes

De Z80 CPU kan in een van de volgende drie interrupt modes werken:

Mode 0
Indien de CPU in deze mode is, kan het "device", dat de interrupt genereert, iedere gewenste instructie op de databus zetten en deze instructie door de CPU laten uitvoeren.

Mode 1
In deze mode zal de CPU op een interrupt reageren door automatisch een RST-instructie uit te voeren. Deze RST instructie zal een "restart" op adres 0038 (hex.) ten gevolge hebben. De oude inhoud van de programma teller PC wordt op de stack geschreven.

Mode 2
In deze mode kan een indirecte sprong naar ieder gewenst geheugenadres worden gemaakt. De CPU vormt een adres uit de inhoud van het I-register (msb) en een byte dat door het "device", dat de interrupt genereert, wordt verstrekt (lsb). Dit gevormde adres wijst naar het eerste byte van een twee bytes veld. Dat veld wijst naar een service routine. De CPU zal deze service routine automatisch starten.

# 17 Z80-registers

| Hoofdregisterset | | Hulpregisterset | |
|---|---|---|---|
| accumulator A | vlaggen F | accumulator A' | vlaggen F' |
| B | C | B' | C' |
| D | E | D' | E' |
| H | L | H' | L' |

| Interrupt vector I | Memory refresh R |
|---|---|
| Index register IX | |
| Index register IY | |
| Program counter PC | |

Accumulator en vlagregisters.

De Z80 CPU bevat twee onafhankelijke 8-bits A-registers met bijbehorende vlagregisters. De accumulator bevat het resultaat van 8-bits rekenkundige of logische bewerkingen, terwijl het vlagregister specifieke condities voor 8 en 16 bits bewerkingen bevat, zoals het aangeven of het resultaat van een bewerking al of niet 0 is. De programmeur kan zelf kiezen welk paar (accumulator + vlagregister) hij wil gebruiken, door gebruik te maken van een exchange-instructie.

Algemene registers.

Er zijn twee sets van ieder zes 8-bits registers. Deze registers kunnen als afzonderlijke 8-bits registers of als drie paren van 16-bits registers worden gebruikt. De programmeur kan een van beide sets selecteren met behulp van exchange instructies.

Het I-register.

Het I-register wordt in interrupt mode 2 gebruikt om er de 8 meest significante adresbits, van het 16-bits indirecte adres, uit te lezen. De 8 minst significante adresbits worden door het "device" dat de interrupt veroorzaakt gegeven.

Het R-register.

Dit register werkt als een teller, die automatisch wordt verhoogd. De inhoud van de teller wordt op de minst significante adresbits gezet, samen met een refresh-control signaal. Normaliter wordt dit register niet door de programmeur gebruikt.

Index registers IX en IY.

Deze registers worden in de "Indexed addressing mode" gebruikt. Ze bevatten een 16-bits adres. De adressen in deze registers worden als basisadres van een geheugengebied gebruikt. Deze registers zijn vooral handig bij het behandelen van tabellen.

Stack Pointer.

Het 16-bits adres in dit register (SP) wijst naar de top van de stack. Deze stack heeft een LIFO-organisatie. Met behulp van de PUSH en POP-instructies kunnen gegevens op de stack worden gezet, of er van af worden gehaald.

Program Counter.

Het PC-register bevat het 16-bits adres van de instructie die uit het geheugen moet worden gelezen. Dit adres wordt na het inlezen van een instructie automatisch verhoogd naar het adres van de volgende instructie. Met behulp van sprongінstructies kan het adres ook worden veranderd.

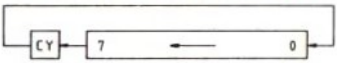
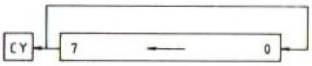
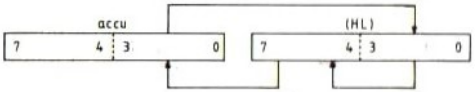
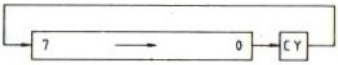
# 18 Symbolische omschrijving van Z80-instructies

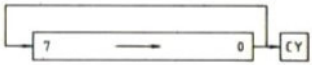
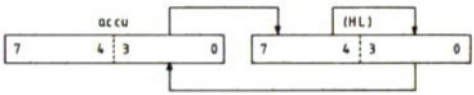
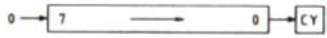
| Mnemonic | Symbolische omschrijving |
|---|---|
| ADC r | A <-- A+r+CY |
| ADC n | A <-- A+n+CY |
| ADC (HL) | A <-- A+(HL)+CY |
| ADC (IX+d) | A <-- A+(IX+d)+CY |
| ADC (IY+d) | A <-- A+(IY+d)+CY |
| ADC HL,ss | HL <-- HL+ss+CY |
| ADD r | A <-- A+r |
| ADD n | A <-- A+n |
| ADD (HL) | A <-- A+(HL) |
| ADD (IX+d) | A <-- A+(IX+d) |
| ADD (IY+d) | A <-- A+(IY+d) |
| ADD HL,ss | HL <-- HL+ss |
| ADD IX,pp | IX <-- IX+pp |
| ADD IY,rr | IY <-- IY+rr |
| AND r | A <-- A ∧ r |
| AND n | A <-- A ∧ n |
| AND (HL) | A <-- A ∧ (HL) |
| AND (IX+d) | A <-- A ∧ (IX+d) |
| AND (IY+d) | A <-- A ∧ (IY+d) |
| BIT b,r | Z <-- $\overline{rb}$ |
| BIT b,(HL) | Z <-- $\overline{(HL)b}$ |
| BIT b,(IX+d) | Z <-- $\overline{(IX+d)b}$ |
| BIT b,(IY+d) | Z <-- $\overline{(IYkd)b}$ |
| CALL cc,nn | alleen uitvoeren indien cc waar is.<br>(SP-1) <-- PCH<br>(SP-2) <-- PCL<br>PC <-- nn |
| CALL nn | (SP-1) <-- PCH<br>(SP-2) <-- PCL<br>PC <-- nn |
| CCF | CY <-- $\overline{CY}$ |
| CP r | A : r |
| CP n | A : n |
| CP (HL) | A : (HL) |
| CP (IX+d) | A : (IX+d) |
| CP (IY+d) | A : (IY+d) |
| CPD | A : (HL)<br>HL <-- HL-1<br>BC <-- BC-1 |
| CPDR | Als CPD, doch net zolang herhaald tot A = (HL) of BC = 0. |

| Mnemonic | Symbolische omschrijving |
|---|---|
| CPI | A : (HL)<br>HL <-- HL+1<br>BC <-- BC-1 |
| CPIR | Als CPI, doch net zolang herhaald tot A = (HL) of BC = 0. |
| CPL | A <-- $\overline{A}$ |
| DAA | Converteert de inhoud van reg. A naar packed BCD. |
| DEC r | r <-- r-1 |
| DEC (HL) | (HL) <-- (HL)-1 |
| DEC (IX+d) | (IX+d) <-- (IX+d)-1 |
| DEC (IY+d) | (IY+d) <-- (IY+d)+1 |
| DEC ss | ss <-- ss-1 |
| DEC IX | IX <-- IX-1 |
| DEC IY | IY <-- IY-1 |
| DI | IFF <-- 0 |
| DJNZ e | B <-- B-1<br>Indien B=0 dan volgende instructie.<br>Indien B<>0 dan:<br>PC <-- PC+e |
| EI | IFF <-- 1 |
| EX AF,AF' | AF <--> AF' |
| EX DE,HL | DE <--> HL |
| EX (SP),HL | H <--> (SP+1)<br>L <--> (SP) |
| EX (SP),IX | IXH <--> (SP+1)<br>IXL <--> (SP) |
| EX (SP),IY | IYH <--> (SP+1)<br>IYL <--> (SP) |
| EXX | BC <--> BC'<br>DE <--> DE'<br>HL <--> HL' |
| HALT | CPU wordt gestopt |
| IM 0 | Interrupt mode 0 |
| IM 1 | Interrupt mode 1 |
| IM 2 | Interrupt mode 2 |
| IN A,(n) | A <-- (n) (n naar A0-A7, A naar A8-A15) |
| IN r,(C) | r <-- (C) (C naar A0-A7, B naar A8-A15) |
| INC r | r <-- r+1 |
| INC (HL) | (HL) <-- (HL)+1 |
| INC (IX+d) | (IX+d) <-- (IX+d)+1 |
| INC (IY+d) | (IY+d) <-- (IY+d)+1 |
| INC ss | ss <-- ss+1 |

| Mnemonic | Symbolische omschrijving |
|---|---|
| INC IX | IX <-- IX+1 |
| INC IY | IY <-- IY+1 |
| IND | (HL) <-- (C)<br>B <-- B-1<br>HL <-- HL-1 |
| INDR | Als IND, totdat B=0. |
| INI | (HL) <-- (C)<br>B <-- B-1<br>HL <-- HL+1 |
| INIR | Als INI, totdat b=0. |
| JP cc,nn | Indien cc is waar, dan PC <-- nn |
| JP nn | PC <-- nn |
| JP (HL) | PC <-- HL |
| JP (IX) | PC <-- IX |
| JP (IY) | PC <-- IY |
| JR C,e | Indien Carry=1 dan PC <-- PC+e |
| JR e | PC <-- PC+e |
| JR NC,e | Indien Carry=0 dan PC <-- PC+e |
| JR NZ,e | Indien Zero=0 dan PC <-- PC+e |
| JR Z,e | Indien Zero=1 dan PC <-- PC+e |
| LD r,r' | r <-- r' |
| LD r,n | r <-- n |
| LD r,(HL) | r <-- (HL) |
| LD r,(IX+d) | r <-- (IX+d) |
| LD r,(IY+d) | r <-- (IY+d) |
| LD (HL),r | (HL) <-- r |
| LD (IX+d),r | (IX+d) <-- r |
| LD (IY+d),r | (IY+d) <-- r |
| LD (HL),n | (HL) <-- n |
| LD (IX+d),n | (IX+d) <-- n |
| LD (IY+d),n | (IY+d) <-- n |
| LD A,(BC) | A <-- (BC) |
| LD A,(DE) | A <-- (DE) |
| LD A,(nn) | A <-- (nn) |
| LD (BC),A | (BC) <-- A |
| LD (DE),A | (DE) <-- A |
| LD (nn),A | (nn) <-- A |
| LD A,I | A <-- I |
| LD A,R | A <-- R |
| LD I,A | I <-- A |
| LD R,A | R <-- A |
| LD dd,nn | dd <-- nn |
| LD IX,nn | IX <-- nn |
| LD IY,nn | IY <-- nn |

| Mnemonic | Symbolische omschrijving |
|---|---|
| LD HL,(nn) | H <-- (nn+1) |
| | L <-- (nn) |
| LD dd,(nn) | ddH <-- (nn+1) |
| | ddL <-- (nn) |
| LD IX,(nn) | IXH <-- (nn+1) |
| | IXL <-- (nn) |
| LD IY,(nn) | IYH <-- (nn+1) |
| | IYL <-- (nn) |
| LD (nn),HL | (nn+1) <-- H |
| | (nn) <-- L |
| LD (nn),dd | (nn+1) <-- ddH |
| | (nn) <-- ddL |
| LD (nn),IX | (nn+1) <-- IXH |
| | (nn) <-- IXL |
| LD (nn),IY | (nn+1) <-- IYH |
| | (nn) <-- IYL |
| LD SP,HL | SP <-- HL |
| LD SP,IX | SP <-- IX |
| LD SP,IY | SP <-- IY |
| LDD | (DE) <-- (HL) |
| | DE <-- DE-1 |
| | HL <-- HL-1 |
| | BC <-- BC-1 |
| LDDR | Als LDD, totdat BC=0 |
| LDI | (DE) <-- (HL) |
| | DE <-- DE+1 |
| | HL <-- HL+1 |
| | BC <-- B-1 |
| LDIR | Als LDI, totdat BC=0 |
| NEG | A <-- 0-A |
| NOP | Geen bewerking |
| OR r | A <-- A v r |
| OR n | A <-- A v n |
| OR (HL) | A <-- A v (HL) |
| OR (IX+d) | A <-- A v (IX+d) |
| OR (IY+d) | A <-- A v (IY+d) |
| OTDR | (C) <-- (HL) (C naar A0-A7 |
| | B <-- B-1 B naar A8-A15) |
| | HL <-- HL-1 |
| | Wordt herhaald totdat B=0 |
| OTIR | (C) <-- (HL) (C naar A0-A7 |
| | B <-- B-1 B naar A8-A15) |
| | HL <-- HL+1 |
| | Wordt herhaald totdat B=0 |

| Mnemonic | Symbolische omschrijving | |
|---|---|---|
| OUT (n),A | (n) <-- A | (n naar A0-A7<br>A naar A8-A15) |
| OUT (C),r | (C) <-- r | (C naar A0-A7<br>B naar A8-A15) |
| OUTD | Als OTDR, doch slechts eenmaal. | |
| OUTI | Als OTIR, doch slechts eenmaal. | |
| POP qq | qqH <-- (SP+1)<br>qqL <-- (SP) | |
| POP IX | IXH <-- (SP+1)<br>IXL <-- (SP) | |
| POP IY | IYH <-- (SP+1)<br>IYL <-- (SP) | |
| PUSH qq | (SP-2) <-- qqL<br>(SP-1) <-- qqH | |
| PUSH IX | (SP-2) <-- IXL<br>(SP-1) <-- IXH | |
| PUSH IY | (SP-2) <-- IYL<br>(SP-1) <-- IYH | |
| RES b,r | rb <-- 0 | |
| RES b,(HL) | (HL)b <-- 0 | |
| RES b,(IX+d) | (IX+d)b <-- 0 | |
| RES b,(IY+d) | (IY+d)b <-- 0 | |
| RET | PCL <-- (SP)<br>PCH <-- (SP+1) | |
| RET cc | Indien cc=waar, dan als RET | |
| RETI | Return from interrupt | |
| RETN | Return from non maskable interrupt | |
| RL/RLA |  | |
| RLC/RLCA |  | |
| RLD |  | |
| RR/RRA |  | |

| Mnemonic | Symbolische omschrijving |
|---|---|
| RRC/RRCA |   |
| RRD |   |
| RST p | (SP-1) <-- PCH<br>(SP-2) <-- PCL<br>PCH <-- 0<br>PCL <-- p |
| SBC r | A <-- A-r-CY |
| SBC n | A <-- A-n-CY |
| SBC (HL) | A <-- A-(HL)-CY |
| SBC (IX+d) | A <-- A-(IX+d)-CY |
| SBC (IY+d) | A <-- A-(IY+d)-CY |
| SBC HL,ss | HL <-- HL-ss-CY |
| SCF | CY <-- 1 |
| SET b,r | rb <-- 1 |
| SET b,(HL) | (HL)b <-- 1 |
| SET b,(IX+d) | (IX+d) <-- 1 |
| SET b,(IY+d) | (IY+d) <-- 1 |
| SLA |   |
| SRA |   |
| SRL |   |
| SUB r | A <-- A-r |
| SUB n | A <-- A-n |
| SUB (HL) | A <-- A-(HL) |
| SUB (IX+d) | A <-- A-(IX+d) |
| SUB (IY+d) | A <-- A-(IY+d) |
| XOR r | A <-- A ⊕ r |
| XOR n | A <-- A ⊕ n |
| XOR (HL) | A <-- A ⊕ (HL) |
| XOR (IX+d) | A <-- A ⊕ (IX+d) |
| XOR (IY+d) | A <-- A ⊕ (IY+d) |

| cc | Conditie | P | dd | qq |
|---|---|---|---|---|
| NZ | non zero | 00 hex | BC | BC |
| Z | zero | 08 hex | DE | DE |
| NC | no carry | 10 hex | HL | HL |
| C | carry | 18 hex | IX | AF |
| PO | pariteit oneven | 20 hex | IY | IX |
| PE | pariteit even | 28 hex | SP | IY |
| P | positief | 30 hex | | |
| M | negatief | 38 hex | | |

| ss | r | rr | pp | b |
|---|---|---|---|---|
| BC | B | BC | BC | 0 |
| DE | C | DE | DE | 1 |
| HL | D | IY | IX | 2 |
| SP | E | SP | SP | 3 |
| | H | | | 4 |
| | L | | | 5 |
| | A | | | 6 |
| | | | | 7 |

# 19 Z80-instructieset op volgorde van mnemonics

In de hiernavolgende tabel zijn in de hexadecimale code 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)

| 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 |

# 20 Z80-instructieset op volgorde van hexcode

In de hierna volgende tabel 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)

| 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 | hex.code | mnemonic |
|---|---|---|---|
| CB2B | SRA E | CB5F | BIT 3,A |
| CB2C | SRA H | CB60 | BIT 4,B |
| CB2D | SRA L | CB61 | BIT 4,C |
| CB2E | SRA (HL) | CB62 | BIT 4,D |
| CB2F | SRA A | CB63 | BIT 4,E |
| CB38 | SRL B | CB64 | BIT 4,H |
| CB39 | SRL C | CB65 | BIT 4,L |
| CB3A | SRL D | CB66 | BIT 4,(HL) |
| CB3B | SRL E | CB67 | BIT 4,A |
| CB3C | SRL H | CB68 | BIT 5,B |
| CB3D | SRL L | CB69 | BIT 5,C |
| CB3E | SRL (HL) | CB6A | BIT 5,D |
| CB3F | SRL A | CB6B | BIT 5,E |
| CB40 | BIT 0,B | CB6C | BIT 5,H |
| CB41 | BIT 0,C | CB6D | BIT 5,L |
| CB42 | BIT 0,D | CB6E | BIT 5,(HL) |
| CB43 | BIT 0,E | CB6F | BIT 5,A |
| CB44 | BIT 0,H | CB70 | BIT 6,B |
| CB45 | BIT 0,L | CB71 | BIT 6,C |
| CB46 | BIT 0,(HL) | CB72 | BIT 6,D |
| CB47 | BIT 0,A | CB73 | BIT 6,E |
| CB48 | BIT 1,B | CB74 | BIT 6,H |
| CB49 | BIT 1,C | CB75 | BIT 6,L |
| CB4A | BIT 1,D | CB76 | BIT 6,(HL) |
| CB4B | BIT 1,E | CB77 | BIT 6,A |
| CB4C | BIT 1,H | CB78 | BIT 7,B |
| CB4D | BIT 1,L | CB79 | BIT 7,C |
| CB4E | BIT 1,(HL) | CB7A | BIT 7,D |
| CB4F | BIT 1,A | CB7B | BIT 7,E |
| CB50 | BIT 2,B | CB7C | BIT 7,H |
| CB51 | BIT 2,C | CB7D | BIT 7,L |
| CB52 | BIT 2,D | CB7E | BIT 7,(HL) |
| CB53 | BIT 2,E | CB7F | BIT 7,A |
| CB54 | BIT 2,H | CB80 | RES 0,B |
| CB55 | BIT 2,L | CB81 | RES 0,C |
| CB56 | BIT 2,(HL) | CB82 | RES 0,D |
| CB57 | BIT 2,A | CB83 | RES 0,E |
| CB58 | BIT 3,B | CB84 | RES 0,H |
| CB59 | BIT 3,C | CB85 | RES 0,L |
| CB5A | BIT 3,D | CB86 | RES 0,(HL) |
| CB5B | BIT 3,E | CB87 | RES 0,A |
| CB5C | BIT 3,H | CB88 | RES 1,B |
| CB5D | BIT 3,L | CB89 | RES 1,C |
| CB5E | BIT 3,(HL) | CB8A | RES 1,D |

| hex.code | mnemonic | hex.code | mnemonic |
|---|---|---|---|
| CB8B | RES 1,E | CBB7 | RES 6,A |
| CB8C | RES 1,H | CBB8 | RES 7,B |
| CB8D | RES 1,L | CBB9 | RES 7,C |
| CB8E | RES 1,(HL) | CBBA | RES 7,D |
| CB8F | RES 1,A | CBBB | RES 7,E |
| CB90 | RES 2,B | CBBC | RES 7,H |
| CB91 | RES 2,C | CBBD | RES 7,L |
| CB92 | RES 2,D | CBBE | RES 7,(HL) |
| CB93 | RES 2,E | CBBF | RES 7,A |
| CB94 | RES 2,H | CBC0 | SET 0,B |
| CB95 | RES 2,L | CBC1 | SET 0,C |
| CB96 | RES 2,(HL) | CBC2 | SET 0,D |
| CB97 | RES 2,A | CBC3 | SET 0,E |
| CB98 | RES 3,B | CBC4 | SET 0,H |
| CB99 | RES 3,C | CBC5 | SET 0,L |
| CB9A | RES 3,D | CBC6 | SET 0,(HL) |
| CB9B | RES 3,E | CBC7 | SET 0,A |
| CB9C | RES 3,H | CBC8 | SET 1,B |
| CB9D | RES 3,L | CBC9 | SET 1,C |
| CB9E | RES 3,(HL) | CBCA | SET 1,D |
| CB9F | RES 3,A | CBCB | SET 1,E |
| CBA0 | RES 4,B | CBCC | SET 1,H |
| CBA1 | RES 4,C | CBCD | SET 1,L |
| CBA2 | RES 4,D | CBCE | SET 1,(HL) |
| CBA3 | RES 4,E | CBCF | SET 1,A |
| CBA4 | RES 4,H | CBD0 | SET 2,B |
| CBA5 | RES 4,L | CBD1 | SET 2,C |
| CBA6 | RES 4,(HL) | CBD2 | SET 2,D |
| CBA7 | RES 4,A | CBD3 | SET 2,E |
| CBA8 | RES 5,B | CBD4 | SET 2,H |
| CBA9 | RES 5,C | CBD5 | SET 2,L |
| CBAA | RES 5,D | CBD6 | SET 2,(HL) |
| CBAB | RES 5,E | CBD7 | SET 2,A |
| CBAC | RES 5,H | CBD8 | SET 3,B |
| CBAD | RES 5,L | CBD9 | SET 3,C |
| CBAE | RES 5,(HL) | CBDA | SET 3,D |
| CBAF | RES 5,A | CBDB | SET 3,E |
| CBB0 | RES 6,B | CBDC | SET 3,H |
| CBB1 | RES 6,C | CBDD | SET 3,L |
| CBB2 | RES 6,D | CBDE | SET 3,(HL) |
| CBB3 | RES 6,E | CBDF | SET 3,A |
| CBB4 | RES 6,H | CBE0 | SET 4,B |
| CBB5 | RES 6,L | CBE1 | SET 4,C |
| CBB6 | RES 6,(HL) | 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) |
| FD360520 | 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 |

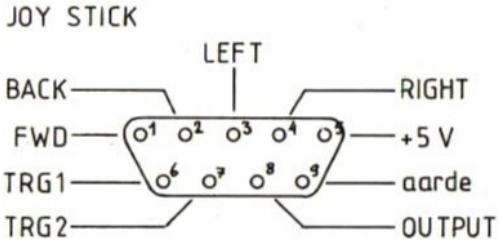
# 21 Z80-vlagbeïnvloeding

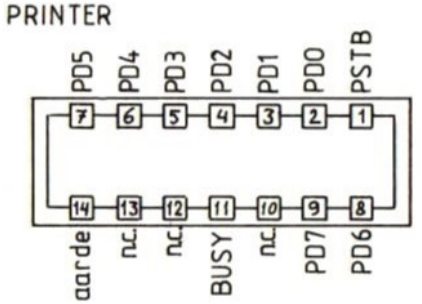
| instructies | C | Z | P - V | S | N | H | opmerkingen |
|---|---|---|---|---|---|---|---|
| ADD A,s / ADC A,s | A | A | V | A | 0 | A | 8 bits optelinstr. |
| SUB s / SBC A,s / CP s/ NEG | A | A | V | A | 1 | A | 8 bits aftrekinstr., vergelijk- en negatie-instructies. |
| AND s | 0 | A | P | A | 0 | 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 + CY |
| SBC HL,ss | A | A | V | A | 1 | X | 16 bits aftrekken + CY |
| RLA/RLCA/RRA/RRCA | A | C | C | C | 0 | 0 | roteren accu. |
| RL s/ RLC s/ RR s RRC s/SLA s/SRA s SRL 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 CY-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 B=0 |
| INIR/INDR/OTIR/ OTDR | C | 1 | X | X | 1 | X | dan Z=1, anders Z=0. |
| LDI / LDD | C | X | A | X | 0 | 0 | blok-transfer. Als b=0 |
| LDIR / LDDR | C | X | 0 | X | 0 | 0 | dan P/V=0, anders P/V=1. |
| CPI/CPIR/CPD/CPDR | C | A | A | A | 1 | X | blok-zoek. Als A=(HL) dan Z=1. Als 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. |

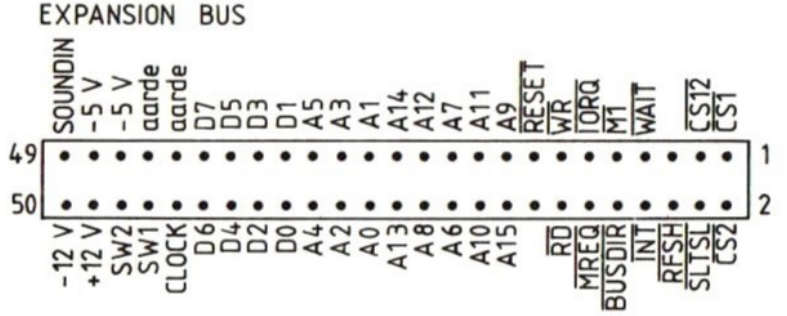
De betekenis van de in de voorgaande tabel gebruikte afkortingen wordt hieronder verklaard:

A - Vlag wordt afhankelijk van het resultaat van de bewerking gezet.
C - De vlag blijft onveranderd.
0 - De vlag wordt op 0 gezet.
1 - De vlag wordt op 1 gezet.
X - De status van de vlag is niet van belang.
V - De "overflow"-vlag wordt afhankelijk van het resultaat van de bewerking gezet.
P - De pariteitsvlag wordt afhankelijk 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).

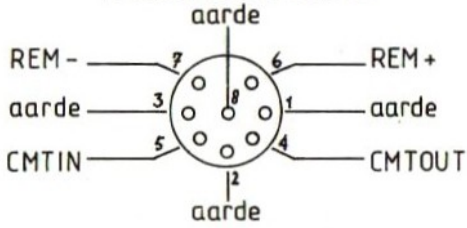
# 22 Connectoren



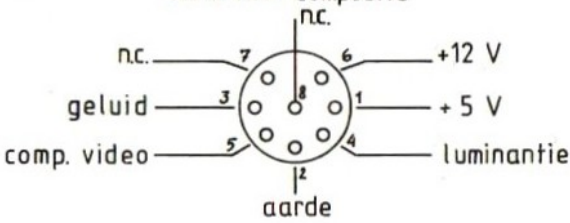



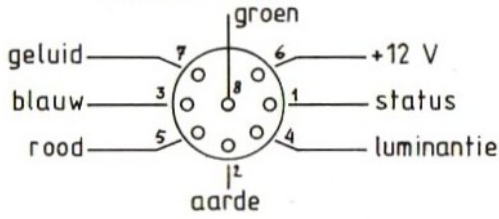
CASSETTE-RECORDER




MONITOR composite




MONITOR RGB



# 23 Geheugen-dump

Wat staat er nou eigenlijk in het geheugen? Op deze vraag krijgt u gauw antwoord, wanneer u het volgende programma intikt en uitvoert. Het laat u precies zien wat er op een door u opgegeven plaats in het geheugen staat, zowel in hexadecimale- als in ASCII-vorm.

Rekening houdend met de maximale regellengte op het beeldscherm, van 40 karakters, kwam ik tot de volgende verdeling:

```
 4 posities voor het geheugenadres (hexadecimaal).
 2 spaties.
16 posities voor de hexadecimale weergave van de inhoud
   van 8 opeenvolgende geheugenadressen.
 2  spaties.
 8 posities voor de ASCII-weergave van dezelfde 8
   opeenvolgende geheugenadressen.
```

In totaal zijn dit maar 32 karakters per regel. Het is echter, in verband met het tellen en het relateren aan geheugenadressen, handig om in groepjes van 8 te werken. Nog handiger zouden groepjes van 16 zijn, echter, dat past niet op een regel.

Met regel 80 tot en met 110 worden het begin en eind-adres van het te dumpen geheugengebied opgevraagd. Op regel 120 start een FOR-NEXT lus die voor iedere regel op het scherm eenmaal wordt doorlopen. Binnen die lus worden de volgende acties ondernomen.

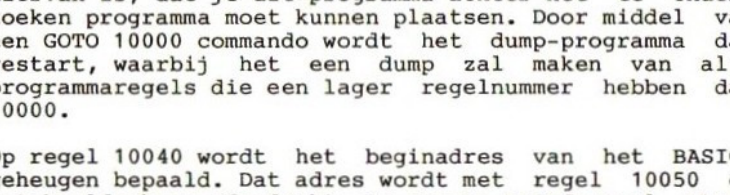
Converteer het decimale adres van de eerste, op de regel af te drukken, geheugenlocatie naar hexadecimaal en zet dit in P$ (regels 130 tot en met 190, met gebruikmaking van regel 70). Voer hierna de FOR-NEXT lus van regel 200 acht maal uit. In deze lus worden 8 geheugenlocaties uitgelezen en in hexadecimaal formaat in P$ gezet. Voer daarna de FOR-NEXT lus van regel 260 acht maal uit. In deze lus worden dezelfde geheugenlocaties nogmaals uitgelezen, doch nu worden ze als ASCII codes in P$ gezet. Hierbij dient nog te worden opgemerkt dat "unprintable" karakters worden vervangen door een "." (regel 280). Print nu de variabele P$ uit. Hiermee is een regel op het beeldscherm gezet.

```
10 '*****************************
20 '*  Afdrukken van de inhoud  *
30 '*  van het geheugen. Start  *
40 '*  en eindadres op te geven *
50 '*****************************
60 '
70 H$="0123456789ABCDEF"
80 INPUT "Startadres";S
90 IF S<0 THEN S=S+2^16
100 INPUT "Eindadres";E
110 IF E<0 THEN E=E+2^16
120 FOR I=S TO E STEP 8
130 H1=INT(I/4096)
140 R=I-H1*4096
150 H2=INT(R/256)
160 R=R-H2*256
170 H3=INT(R/16)
180 H4=R-H3*16
190 P$=MID$(H$,H1+1,1)+MID$(H$,H2+1,1
)+MID$(H$,H3+1,1)+MID$(H$,H4+1,1)+"
"
200 FOR J=1 TO 8
210 X$=HEX$(PEEK(I+J))
220 IF LEN(X$)<2 THEN X$="0"+X$
230 P$=P$+X$
240 NEXT J
250 P$=P$+"  "
260 FOR J=1 TO 8
270 X$=CHR$(PEEK(I+J))
280 IF ASC(X$)<32 THEN X$="."
290 P$=P$+X$
300 NEXT J
310 PRINT P$
320 NEXT I
330 PRINT "NOG MEER TE DUMPEN? (J/N)"
340 I$=INKEY$
350 IF I$<>"J" AND I$<>"N" AND I$<>"j
" AND I$<>"n" GOTO 340
360 IF I$="J" OR I$="j" GOTO 80
370 END
```

Met het vorige programma is al veel mogelijk. Het leek mij echter interessant om een wat georganiseerde geheugen-dump van het BASIC-geheugen te maken. Het volgende programma doet dat.

De regelnummers zijn nogal hoog gekozen. De reden hiervan is, dat je dit programma achter het te onderzoeken programma moet kunnen plaatsen. Door middel van een GOTO 10000 commando wordt het dump-programma dan gestart, waarbij het een dump zal maken van alle programmaregels die een lager regelnummer hebben dan 10000.

Op regel 10040 wordt het beginadres van het BASIC-geheugen bepaald. Dat adres wordt met regel 10050 op het beeldscherm afgedrukt. De eerste BASIC-regel start op dat adres. Iedere BASIC-regel heeft een vast formaat dat er als volgt uitziet:




De eerste positie is het minst significante byte van het adres waar de volgende BASIC-regel begint. De tweede positie is het meest significante byte van dat adres. Met regels 10060 tot en met 10080 wordt het adres van de volgende regel bepaald.

De derde en vierde positie bevatten respectievelijk het minst en meest significante byte van het regelnummer van de huidige regel. Dit regelnummer wordt op de regels 10100 tot en met 10120 bepaald. Vervolgens wordt met regel 10130 onmiddellijk gecontroleerd of de huidige regel soms regel 10000 is. In dat geval wordt het programma beeindigd.

Is de laatste regel nog niet bereikt, dan wordt het regel nummer afgedrukt, waarna met regels 10150 tot en met 10170 de inhoud van de BASIC-regel hexadecimaal wordt afgedrukt. Voordat de volgende regel wordt behandeld, zorgt regel 10190 er voor dat het adres van die volgende regel in de variabele P komt (P=pointer).

```
10000 '************************
10010 '* BASIC GEHEUGEN DUMPEN *
10020 '************************
10030 '
10040 P=2^15+1
10050 PRINT "ADRES=";P
10060 VR=PEEK(P)
10070 P=P+1
10080 VR=VR+256*PEEK(P)
10090 P=P+1
10100 RN=PEEK(P)
10110 P=P+1
10120 RN=RN+256*PEEK(P)
10130 IF RN=>10000 GOTO 10210
10140 PRINT "REGEL=";RN
10150 FOR A=P+1 TO VR-1
10160 PRINT USING "\ \";HEX$(PEEK(A));
10170 NEXT A
10180 PRINT: PRINT
10190 P=VR
10200 GOTO 10050
10210 PRINT "EINDE BASIC LISTING"
10220 END
```

Naar aanleiding van het voorgaande programma kwam het idee bij me op om, nu het formaat van een BASIC-programma bekend is, daar nog iets meer mee te doen. Door binnen de inhoud van een BASIC-regel naar het al of niet voorkomen van een bepaalde string te zoeken, kun je bijvoorbeeld eenvoudig bepalen of een bepaalde variabele of een bepaalde tekst in het programma voorkomt, en zo ja, in welke regel(s).

Zoals u in de listing ziet, zit een groot deel van het vorige programma in dit programma verwerkt. Met regel 10050 wordt de te zoeken string gevraagd en in variabele T$ gezet. Regel 10060 bepaalt de lengte van de ingegeven string.

Nu wordt het begin van de eerste BASIC-regel opgezocht. Eenmaal gevonden, wordt er met regel 10200 gekeken of het eerste karakter in de BASIC-regel gelijk is aan het eerste karakter in de opgegeven string. Is dat inderdaad het geval, dan wordt gekeken of het tweede karakter ook gelijk is aan het tweede karakter uit de opgegeven string. Dit gaat zo door tot er een ongelijkheid optreedt of totdat alle karakters van de string in de regel blijken voor te komen. In het eerste geval zal de variabele N gelijk aan 1 worden gemaakt. In het laatste geval zal de variabele N zijn in regel 10200 verkregen waarde houden.

In regel 10210 wordt vervolgens gekeken of N groter is dan het aantal karakters in de opgegeven string. Is dat zo, dan is de string dus gevonden, en wordt het regelnummer van de BASIC-regel waarin die string was gevonden afgedrukt. Hierna wordt de volgende BASIC-regel bekeken en herhaalt zich het hele proces.

Binnen dat proces zit nog regel 10160. Hiermee wordt gecontroleerd of het regelnummer van de te onderzoeken BASIC-regel wel lager is dan 10000. Net als bij het vorige programma is het ook hier weer de bedoeling om het zoekprogramma samen met het te onderzoeken programma in het geheugen te zetten. Het zoekprogramma wordt dan gestart door een GOTO-commando te geven.

Nadat de laatste regel van het te onderzoeken programma is verwerkt, geven de regels 10270 tot en met 10300 de gelegenheid om opnieuw te beginnen, eventueel met een andere string.

```
10000 '************************
10010 '* TEKST ZOEKEN IN BASIC *
10020 '************************
10030 '
10040 SCREEN 0: WIDTH 36
10050 INPUT "VAN WELKE TEKST WILT U
WETEN OF DIE IN HET PROGRAMMA VOORKO
MT";T$
10060 L=LEN(T$)
10070 P=2^15+1
10080 VR=PEEK(P)
10090 P=P+1
10100 VR=VR+256*PEEK(P)
10110 IF VR<=0 GOTO 10270
10120 P=P+1
10130 RN=PEEK(P)
10140 P=P+1
10150 RN=RN+256*PEEK(P)
10160 IF RN=10000 GOTO 10270
10170 P=P+1
10180 N=1
10190 FOR A=P TO VR-1
10200 IF MID$(T$,N,1)=CHR$(PEEK(P))
THEN N=N+1 ELSE N=1
10210 IF N>L THEN PRINT USING "#####
#";RN;
10250 P=VR
10260 GOTO 10080
10270 PRINT: PRINT "ANDERE TEKSTEN Z
OEKEN? (J/N)"
10280 I$=INKEY$
10290 IF I$<>"J" AND I$<>"j" AND I$<
>"N" AND I$<>"n" GOTO 10280
10300 IF I$="J" OR I$="j" GOTO 10050
10310 END
```

# 24 Inhoudsopgave van de schijf

Het volgende programma is weliswaar erg eenvoudig, maar laat toch een aantal leuke mogelijkheden van het MSX-BASIC zien. Bovendien is het mijn ervaring dat het nog een erg handig programmaatje is ook. Wat het namelijk doet is, dat het alle op de schijf staande bestanden en programma's op het beeldscherm afdrukt. Vervolgens geeft het aan hoeveel kilobytes vrije ruimte er nog op de schijf is. Daarna vraagt het u welk programma u wilt draaien.

Het is een goede gewoonte om BASIC programma's op schijf op te slaan onder een vrij te kiezen naam, met de toevoeging ".BAS". Hierdoor herkent u de BASIC programma's onmiddellijk, en zult u niet gauw per ongeluk een bestand proberen te starten. Aan bestandsnamen kunt u bijvoorbeeld ".DAT" toevoegen.

Op de vraag, welk programma u wilt draaien, hoeft u echter de toevoeging ".BAS" niet in te tikken. Deze toevoeging is al in het programma opgenomen (zie regel 1140. Bovendien zorgt deze geprogrammeerde toevoeging er voor dat u niets anders dan programma's met de toevoeging ".BAS" kunt starten. Hiermee worden vergissingen dus tegengegaan.

De eigenlijke inhoudsopgave van de schijf wordt met regel 1080 gemaakt. Met de functie DSKF(n) wordt de nog vrije ruimte op de schijf bepaald. De reden om een beeldschermbreedte van 38 karakters te kiezen (regel 1040), is dat er dan precies drie file-namen op een regel gaan.

Door dit programma onder de naam "AUTOEXEC.BAS" op de schijf te zetten, zal dit programma na het aanschakelen van de computer (en de schijf) automatisch worden gestart. Dit heeft dan tot gevolg dat u direct na het aanschakelen, zonder enige verdere handelingen, kunt zien welke programma's er op de schijf staan. Vervolgens kunt u dan door eenvoudigweg de programmanaam in te tikken, het gewenste programma laden.

```
1000 '**********************
1010 '*     AUTOEXEC.BAS     *
1020 '**********************
1030 '
1040 WIDTH(38)
1050 CLS
1060 PRINT "OVERZICHT VAN BESTANDEN EN
 PROGRAMMA'S"
1070 PRINT "**************************
************"
1080 FILES
1090 PRINT
1100 PRINT "**************************
************"
1110 PRINT "VRIJE RUIMTE =";DSKF(0);"K
."
1120 PRINT
1130 INPUT "WELK PROGRAMMA WILT U DRAA
IEN";P$
1140 LOAD P$+".BAS"
```

# Aantekeningen

# Aantekeningen

# Aantekeningen

# Aantekeningen

# Aantekeningen

# Aantekeningen

# Aantekeningen

# Aantekeningen

# Aantekeningen

**ENKELE MSX-uitgaven**

**serie: UW MSX COMPUTER DE BAAS**

MSX BASIC HANDBOEK voor iedereen, door A.C.J. Groeneveld
Hèt handboek voor iedere MSX computer gebruiker
ISBN 90 6398 100 7

MSX DISK HANDBOEK voor iedereen, door A.C.J. Groeneveld
ISBN 90 6398 407 3

MSX BASIC leerboek deel 1, door Wessel Akkermans en Piet den Heijer. Informatie over MSX-hardware en leren programmeren
ISBN 90 6398 649 1

MSX BASIC leerboek deel 2, door Wessel Akkermans en Piet den Heijer. Doorgaan met leren programmeren en speciale aandacht voor kleur, grafieken, geluid/muziek en joy-sticks
ISBN 90 6398 769 2

MSX DOS leerboek, door Wessel Akkermans en Piet den Heijer. Het manipuleren met strings, het bewerken en opslaan van gegevens op cassette en schijf
ISBN 90 6398 519 3

**SOFTWARE PLUS in MSX**

INTROTAPE MSX, door A.C.J. Groeneveld. Begeleid met instrukties om de computer aan te sluiten en de tape te laden, wordt MSX op een vriendelijke en onderwijzende manier vanuit nul bij de gebruiker geïntroduceerd. Na het doorwerken van deze software is de gebruiker zelf in staat MSX-basic programma's te schrijven
ISBN 90 6398 148 1

MSX-SCRIPT door Ton Weijters
Een menu-gestuurde nederlandstalige tekstverwerker
ISBN 90 6398 189 9


# MSX
## ZAKBOEKJE

Tekeningen maken, kleuren en geluiden produceren, en met hetzelfde gemak als het afdrukken van een stukje tekst op het beeldscherm. Dat was mijn eerste indruk van de MSX-computers. Na die eerste kennismaking bleven de MSX-computers mij verbazen. Een floppy disk van merk A kon zonder meer worden aangesloten op merk B. Diezelfde mate van uitwisselbaarheid gold ook voor programma's. Programma's geschreven op de ene MSX-computer, konden ongewijzigd worden afgedraaid op een andere.

Het aantal mogelijkheden, dat de zeer uitgebreide en goed gestandaardiseerde MSX-BASIC, en de al evengoed gestandaardiseerde hardware, hebben, maakt dat het wel haast onmogelijk is om al die mogelijkheden uit het hoofd te kennen. Daarnaast komen tijdens het programmeren altijd wel een aantal handige tabellen te pas.

In grote lijnen zult u in dit boek de volgende gegevens terug kunnen vinden:

Niet computergerichte tabellen,
de MSX-BASIC instructieset,
diverse tabellen die het BASIC-programmeren kunnen versnellen,
de Z80 instructieset,
hardware-gegevens (connectoren) en een aantal programmaatjes.

ISBN 90 6398 888 5