MEGA 65

Naslaggids BASIC 65



MUSEUM OF ELECTRONIC GAMES & ART

MEGA65-TEAM

Paul Gardner-Stephen, prof.
(highlander)
Oprichter
Software- en virtuele hardwarearchitect
Woordvoerder en hoofdwetenschapper

Martin Streit
(seriously)
Video- en fotoproductie
Fiscaal en organisatie
Sociale media

Dan Sanderson
(dddaaannn)
Media en documentatie
MEGA65.ROM

Edilbert Kirk, dr.
(Bit Shifter)
MEGA65.ROM
Handleiding en gereedschappen

Gábor Lénárt
(LGB)
Emulator

Farai Aschwanden
(Tayger)
Bestands-hosting en gereedschappen
Financieel adviseur

Falk Rehwagen
(bluewaysw)
GEOS

Robert Steffens
(kibo)
Netwerktechnologie
Jacht op bugs in onze cores

Detlef Hastik
(deft)
Mede-oprichter
Projectaansturing
Marketing en verkoop

Oliver Graf
(lydon)
Releasemanagement
VHDL- en platformverbeteringen

Antti Lukats
(antti-brain)
Host-hardwareontwerp en -productie

Dieter Penner
(doubleflash)
Host-hardwareondersteuning

Mirko H.
(sy2002)
Aanvullende platforms en advies

Gürçe Işıkyıldız
(gurce)
Gereedschappen en verbeteringen

Daniel England
(Mew Pokémon)
Aanvullende code en gereedschappen

Hernán Di Pietro
(indiocolifa)
Aanvullende emulatie en gereedschappen

Roman Standzikowski
(FeralChild)
Open ROMs

Anton Schneider-Michallek
(adtbm)
Presentatie en ondersteuning

Fouten en onvolkomenheden melden

Dit boek wordt voortdurend verbeterd door de MEGA65-gemeenschap. De versie van deze editie is:

```
commit c598652ac7f860872b8be62911f59ad5ad752169
date: Sat Oct 24 17:02:30 2026 +0200
```

We willen dat dit boek zo goed mogelijk wordt. Als u fouten ontdekt, iets mist of graag meer informatie wilt, meld dit dan via het issue-tracker van de MEGA65 User's Guide:

https://github.com/mega65/mega65-user-guide/issues

U kunt daar ook controleren of iemand anders al een vergelijkbaar probleem heeft gemeld, terwijl u wacht op een update van dit boek.

Ten slotte kunt u altijd de nieuwste versies van onze boekenserie downloaden via:

- https://mega65.org/mega65-book
- https://mega65.org/user-guide
- https://mega65.org/developer-guide
- https://mega65.org/basic65-ref
- https://mega65.org/chipset-ref
- https://mega65.org/docs

MEGA65 BASIC 65 NASLAGGIDS

Uitgegeven door
het MEGA Museum of Electronic Games & Art e.V., Duitsland.

IN AANBOUW

Copyright ©2019 - 2026 Paul Gardner-Stephen, het MEGA Museum of Electronic Games & Art e.V. en medewerkers.

Dit boek is beschikbaar onder de GNU Free Documentation License v1.3 of latere versie. Dit betekent dat u dit boek vrij mag wijzigen, vermenigvuldigen en herdistribueren, onder bepaalde voorwaarden. De volledige tekst van de GNU Free Documentation License v1.3 is te vinden op https://www.gnu.org/licenses/fdl-1.3.en.html. Een Nederlandse vertaling van de licentie is te vinden op https://meta.wikimedia.org/wiki/GNU_Free_Documentation_License/nl.

Deze copyrightlicentie omvat ook de toestemming om dit document geheel of gedeeltelijk te kopiëren en/of te verspreiden voor gebruik in onderwijsomgevingen. Wij willen het onderwijs van toekomstige generaties ondersteunen, dus als u vragen of bezwaren heeft, neem dan contact met ons op.

oktober 29, 2026

Inhoudsopgave

1 Inleiding **1**

Welkom bij de MEGA65! 3

Overige boeken in deze reeks 4

Doe mee! 4

2 Naslag BASIC65-commando's **5**

Commando's, functies en operatoren 7

Alfabetisch overzicht BASIC-commando's 21

3 Schermcodes **303**

Schermcodes 305

4 PETSCII-codes **307**

PETSCII-codes 309

5 Toetsen van de schermeditor **313**

Toetsen van de schermeditor 315

Control-codes 315

Shiftcodes 318

Escape-sequenties 318

6 Systeempalet **325**

Systeempalet 327

7 Steuners & donateurs **329**

Organisaties 331

Bijdragers 332

Steuners 333

Index **343**

HOOFDSTUK 1

Inleiding

- Welkom bij de MEGA65!
- Overige boeken in deze reeks
- Doe mee!

WELKOM BIJ DE MEGA65!

Van harte gefeliciteerd met de aanschaf van één van de meest langverwachte computers uit de geschiedenis van het computergebruik! De MEGA65 is door de community ontworpen en is gebaseerd op de nooit uitgebrachte Commodore® 65[1]-computer; een computer die in 1989 werd ontworpen en in 1990 voor het publiek verkrijgbaar zou worden. Inmiddels zijn er decennia verstreken en wij hebben ons ingespannen om herinneringen op te roepen aan een vroegere tijd waarin computers eenvoudig en vriendelijk waren. Zij waren niet alleen eenvoudig te bedienen en te begrijpen, doch ook vriendelijk en toegankelijk voor nieuwe gebruikers.

Deze computers uit de jaren tachtig inspireerden veel van hun eigenaren tot de opwindende en dankbare technologiecarrières die zij vandaag de dag hebben. Stelt u zich eens de opwinding voor die deze computerpioniers beleefden, toen zij ontdekten dat zij hun nieuwe computer konden gebruiken om problemen op te lossen, een brief te schrijven, belastingaangifte te doen, nieuwe dingen uit te vinden, te ontdekken hoe het heelal werkt en misschien zelfs een spannend spelletje of twee te spelen! Wij willen datzelfde niveau van opwinding weer aanwakkeren (dat helaas in het moderne computergebruik niet meer te vinden is) en daarom hebben wij de **MEGA65** gemaakt.

Het MEGA65-team is van mening dat het bezitten van een computer vergelijkbaar is met het bezitten van een huis. U gebruikt een huis niet alleen maar; u verandert dingen, groot en klein, om er uw eigen persoonlijke leefruimte van te maken. Na verloop van tijd, wanneer u zich er eenmaal thuis voelt, kunt u besluiten uw huis te verbouwen of uit te breiden om het comfortabeler of praktischer te maken. Beschouw de MEGA65 als uw eigen "computerthuis".

Deze handleiding leert u meer dan alleen plaatjes aan de muur te hangen; hij laat u zien hoe u uw droomhuis bouwt. Terwijl u deze handleiding leest, leert u de MEGA65 bedienen, programma's schrijven, extra programmatuur toevoegen en de mogelijkheden van de hardware uitbreiden. Wat niet meteen duidelijk zal zijn, is dat u onderweg ook meer leert over de geschiedenis van het computergebruik, terwijl u de vele facetten van BASIC-versie 65 en de commando's van het besturingssysteem verkent.

Computergrafiek en muziek maken het computergebruik leuker, en wij hebben de MEGA65 ontworpen om leuk te zijn! In deze handleiding leert u programma's schrijven die gebruikmaken van de ingebouwde **grafische** en **geluids**mogelijkheden van de MEGA65. U hoeft echter geen programmeur te zijn om plezier met de MEGA65 te beleven. Omdat de MEGA65 een complete Commodore® 64™[2] bevat, kan hij ook duizenden bestaande spellen, hulpprogramma's en zakelijke programmapakketten uitvoeren, evenals nieuwe programma's die op dit moment door liefhebbers van de Commodore-computer worden geschreven. Het enthousiasme voor de MEGA65 zal groeien naarmate wij getuige zijn van de programmeerwonderen die onze MEGA65-community voortbrengt, terwijl zij (en u!) de krachtige mogelijkheden van deze moderne herschepping van een Commodore-computer ontdekken en beheersen. Samen

[1] Commodore is een handelsmerk van C= Holdings
[2] Commodore 64 is een handelsmerk van C= Holdings

met u kunnen wij een nieuwe "homebrew"-community opbouwen, barstensvol proram-matuur en projecten die de mogelijkheden van de MEGA65 oprekken tot ver voorbij wat iemand ooit voor mogelijk had gehouden.

Wij heten u welkom op deze reis! Dank u dat u deel uitmaakt van de MEGA65-community van gebruikers, programmeurs en enthousiastelingen!

OVERIGE BOEKEN IN DEZE REEKS

Dit boek is één van de boeken uit de MEGA65-documentatiereeks. De reeks omvat:

- **De Mega65-gebruikershandleiding**
  Biedt een inleiding tot de MEGA65 en een beknopt overzicht van de BASIC 65-commando's
- **De Mega65 naslaggids BASIC65**
  Uitgebreide documentatie van alle BASIC 65-commando's, functies en opera-tors
- **De Mega65-chipset naslaggids**
  Gedetailleerde documentatie over de eigen chips van de MEGA65 en de C65
- **De Mega65 programmeergids**
  Informatie voor ontwikkelaars die programma's voor de MEGA65 willen schrijven
- **Alles over de Mega65**
  (Ook bekend als **Het grote Mega65-boek**)
  Alle delen in één enkel omvangrijk pdf-bestand voor eenvoudig zoeken. 1200 bladzijden en nog steeds groeiend!

DOE MEE!

Raak betrokken, leer meer over uw MEGA65 en sluit u online bij ons aan op:

- https://mega65.org/chat
- https://mega65.org/forum

HOOFDSTUK 2

Naslag BASIC65-commando's

- Commando's, functies en operatoren
- Alfabetisch overzicht BASIC-commando's

COMMANDO'S, FUNCTIES EN OPERATOREN

In deze bijlage worden de commando's, functies en andere aanroepbare elementen van BASIC 65 beschreven. BASIC 65 is een uitgebreide versie van BASIC 10. Een aantal van deze elementen kan één of meer argumenten aannemen: dit zijn gegevens die u als onderdeel van het commando of de functieaanroep opgeeft om te beschrijven wat u wilt bereiken. Sommige vereisen bovendien het gebruik van speciale trefwoorden.

Hieronder volgt een voorbeeld van hoe commando's, functies en operatoren (welke alle ook wel trefwoorden worden genoemd) in dit bijlage worden beschreven.

KEY getal**,** tekenreeks

Hier is **KEY** een *trefwoord*. Trefwoorden zijn speciale woorden die BASIC begrijpt. In deze handleiding worden trefwoorden altijd in **VETTE HOOFDLETTERS** geschreven, zodat u ze eenvoudig kunt herkennen.

De "getal" en "tekenreeks" (in niet-vette tekst) zijn voorbeelden van *argumenten*. Deze vervangt u door waarden of algebraïsche expressies (*expressies*) die de gegevens weergeven die het gedrag van het commando bepalen.

Leestekens en andere letters in vette tekst geven andere tekens aan die precies zo moeten worden getypt als ze verschijnen. In dit voorbeeld moet er een komma staan tussen het argument voor het getal en het argument voor de tekenreeks.

Hier volgt een voorbeeld van het gebruik van het **KEY**-commando op basis van dit patroon:

```
KEY 6, "LIST" + CHR$(13)
```

Wanneer u vierkante haakjes om argumenten ziet staan, geeft dit aan dat de argumenten optioneel zijn. U moet de vierkante haakjes niet typen. Bekijk de volgende beschrijving van het **CIRCLE**-commando, dat optionele argumenten aanvaardt:

CIRCLE xc**,** yc**,** straal [**,** vlaggen **,** begin**,** einde]

De volgende voorbeelden van het **CIRCLE**-commando zijn beide geldig. Ze hebben verschillend gedrag op basis van hun verschillende argumenten.

```
CIRCLE 100, 150, 30

CIRCLE 100, 150, 30, 0, 45, 135
```

Deze rangschikking van trefwoorden, symbolen en argumenten wordt *syntaxis* genoemd. Indien u iets weglaat of iets op de verkeerde plek zet, zal de computer het commando niet begrijpen en een *syntaxisfout* melden.

U hoeft zich geen zorgen te maken wanneer u een fout van de MEGA65 krijgt. Op die manier geeft de MEGA65 u te kennen dat er iets niet in orde is, zodat u het probleem eenvoudiger kunt vinden en verhelpen. Indien u bijvoorbeeld de komma in het **KEY**-commando weglaat of vervangt door een punt, zal de MEGA65 reageren met een Syntaxisfout, zoals hieronder te zien is:

```
READY.
KEY 8"VIS"

?SYNTAX ERROR
READY.
KEY 8."VIS"

?SYNTAX ERROR
READY.
```

Expressies kunnen een numerieke waarde zijn zoals `23.7`, een tekenreekswaarde zoals `"HALLO"`, of een complexere berekening die waarden, functies en operatoren combineert om een getal of tekenreeks weer te geven: `"LIST"+CHR$(13)`

Het is belangrijk om het juiste type expressie te gebruiken bij het schrijven van programma's. Indien u per ongeluk het verkeerde type gebruikt, zal de MEGA65 een Type Mismatch-fout weergeven om aan te geven dat het type expressie dat u heeft opgegeven niet overeenkomt met wat werd verwacht. Het volgende commando geeft bijvoorbeeld een Type Mismatch-fout, omdat `"POTATO"` een tekenreeksexpressie is en er een numerieke expressie wordt verwacht:

```
KEY "AARDAPPEL", "SOEP"
```

Commando's zijn instructies die u direct vanaf het `READY.`-prompt kunt gebruiken, of vanuit een programma. Bijvoorbeeld:

```
READY.
PRINT "HALLO DAAR"
HALLO DAAR

READY.
10 PRINT "HALLO DAAR"
RUN
HALLO DAAR
```

U kunt een reeks instructies op één regel plaatsen door ze te scheiden met dubbele punten. Bijvoorbeeld:

```
PRINT "HALLO" : PRINT "HOE GAAT HET?" : PRINT "LEKKER WEERTJE?"
HALLO
HOE GAAT HET?
LEKKER WEERTJE?
```

Commando's in directe modus

Sommige commando's werken alleen in **directe modus** (soms "onmiddellijke modus" genoemd). Dit betekent dat het commando geen onderdeel van een BASIC-programma kan zijn, maar wel direct op het scherm kan worden ingevoerd. Het **RENUMBER**-commando werkt bijvoorbeeld alleen in directe modus, omdat het tot doel heeft de regels van een BASIC-programma te hernummeren.

In de twee **PRINT**-voorbeelden hierboven werd de eerste in directe modus ingevoerd, terwijl de tweede deel uitmaakte van een programma. Het **PRINT**-commando werkt zowel in directe modus als in een programma.

Syntaxisbeschrijvingen van commando's

De volgende tabel beschrijft de overige symbolen die voorkomen in de syntaxisbeschrijvingen van commando's.

Symbool	Betekenis
[ ]	Optioneel
...	De syntaxis tussen vierkante haakjes kan nul of meer keer worden herhaald
< | >	Kies een van beiden
[|]	Neem optioneel één van de keuzen op
{ , }	Ten minste één van de argumenten is verplicht. De komma's links van het laatste opgenomen argument zijn verplicht. Optionele komma's achteraan moeten worden weggelaten. Zie **CURSOR** voor een voorbeeld.
[{ , }]	Vergelijkbaar met { , } maar alle argumenten kunnen worden weggelaten

Lettertypen

Voorbeelden van tekst die op het scherm verschijnt, hetzij door u getypt hetzij door de MEGA65 afgedrukt, worden weergegeven in het schermlettertype: `"LIST"+CHR$(13)`

BASIC 65-constanten

Waarden die direct in een expressie of programma worden getypt, worden *constanten* genoemd. Deze waarden zijn "constant" omdat ze niet veranderen op basis van andere aspecten van de programmastatus.

Hierna volgen de typen constanten die in een BASIC 65-expressie kunnen voorkomen.

Type	Voorbeeld	Voorbeeld
Decimaal geheel getal	32000	-55
Decimaal vastpuntgetal	3.14	-7654.321
Decimaal drijvend-kommagetal	1.5E03	7.7E-02
Hexadecimaal	$D020	$FF
Binair	%11010010	%101
Tekenreeks	"X"	"TEXT"

BASIC 65-variabelen

Een programma bewerkt gegevens door waarden in het geheugen van de computer op te slaan, opgeslagen waarden aan te roepen en deze bij te werken op basis van logica. In BASIC heten geheugenelementen die waarden opslaan *variabelen*. Elke variabele heeft een naam; er bestaan aparte reeksen van variabelenamen voor elk type waarde.

De variabele AA kan bijvoorbeeld een numerieke waarde opslaan. De variabele AA$ kan een tekenreekswaarde opslaan. Commodore BASIC beschouwt deze als twee aparte variabelen, hoewel beide namen beginnen met AA.

Een manier om een waarde aan een variabele toe te wijzen, is met de toewijzingsoperator (=). Voorbeeld:

```
AA = 1.95

AA$ = "HALLO, "
```

Variabelenamen moeten beginnen met een letter en mogen alleen letters en getallen bevatten. De naam mag elke lengte hebben, doch Commodore BASIC herkent alleen de eerste twee letters. SPEED en SP worden als dezelfde variabele beschouwd.

Variabelenamen mogen geen van de BASIC-trefwoorden bevatten. Het gebruik van lange namen is daardoor moeilijk: het is gemakkelijk om per ongeluk een trefwoord te gebruiken. ENFORCEMENT is bijvoorbeeld geen geldige variabelenaam, omdat **FOR** een trefwoord is. Het is gebruikelijk om korte variabelenamen te kiezen om dit soort problemen te voorkomen.

Een variabele kan worden gebruikt in een expressie samen met constanten, variabelen, functies en operatoren. Op het moment van uitvoering wordt de variabele vervangen door de waarde die deze dan bevat.

```
10 INPUT "HOE HEET JE"; NA$
20 MSG$ = "HALLO, " + NA$ + "!"
30 PRINT MSG$
```

In tegenstelling tot sommige programmeertalen hoeven BASIC-variabelen niet te worden gedeclareerd voor gebruik. Een variabele heeft een standaardwaarde van nul voor numerieke variabelen, of de lege tekenreeks ("") voor tekenreeksvariabelen.

Een variabele die een enkele waarde opslaat, wordt ook wel een *scalaire variabele* genoemd. De scalaire variabelentypen en hun waardebereiken zijn als volgt.

Type	Naamsymbool	Bereik	Voorbeeld
Byte	&	0 .. 255	BY& = 23
Integer	%	-32768 .. 32767	I% = 5
Real	geen	-1E37 .. 1E37	XY = 1/3
String	$	lengte = 0 .. 255	AB$ = "TEXT"

Een variabele waarvan de naam bestaat uit één letter gevolgd door het typesymbool (of geen symbool voor reële getallen) is een *snelle variabele*. BASIC 65 slaat de variabele zodanig op dat de waarde sneller toegankelijk is dan bij variabelen met langere namen. Verder gedraagt deze zich zoals elke andere variabele. Dit geldt ook voor functies die met **DEF FN** worden gedefinieerd.

BASIC 65-arrays

Naast scalaire variabelen kent Commodore BASIC ook een type variabele dat meerdere waarden kan opslaan: een *array*.

Het volgende voorbeeld slaat drie tekenreekswaarden op in een array en gebruikt een **FOR**-lus om voor elk element een **PRINT**-melding weer te geven:

```
10 DIM NA$(3)
20 NA$(0) = "DEFT"
30 NA$(1) = "GARDNERS"
40 NA$(2) = "LYDON"
50 FOR I = 0 TO 2
60 PRINT "HALLO, "; NA$(I); "!"
70 NEXT I
```

Elke waarde in een array wordt aangeroepen met de naam van de arrayvariabele en een geheel getal als index. Verwijzingen naar array-elementen gebruiken de indexwaarde, zoals in AA(7) dat verwijst naar het element van de array AA() met index 7. De index begint bij nul: het eerste element heeft index 0. De index kan een numerieke expressie zijn, wat een krachtige manier is om met meerdere gegevenselementen te werken.

Alle waarden in een array moeten van hetzelfde type zijn. Het type wordt aangeduid in de naam van de variabele, net als bij scalaire variabelen. AA() is een array van reële getallen, AA$() is een array van tekenreeksen.

Arrayvariabelenamen worden als apart beschouwd van scalaire variabelenamen. De scalaire variabele AA heeft geen relatie met de arrayvariabele AA().

BASIC moet de maximale grootte van de array kennen voordat deze voor het eerst wordt gebruikt, zodat het geheugen voor de volledige array kan worden toegewezen. Een programma kan de grootte van een array declareren met het **DIM**-trefwoord, waarmee de "dimensies" van de array worden opgegeven. Indien een arrayvariabele wordt gebruikt zonder expliciete declaratie, wijst BASIC een eendimensionale array met 10 elementen toe, en kan de array niet opnieuw gedimensioneerd worden (tenzij u alle variabelen met **CLR** wist).

Een array kan meerdere dimensies hebben, elk met een eigen index gescheiden door een komma. De array moet worden gedeclareerd met de maximale waarde voor elke dimensie. Houd er rekening mee dat BASIC 65 geheugen toewijst voor de gehele array, zodat grote arrays beperkt kunnen worden door het beschikbare geheugen.

```
DIM BO$(3, 3)
BO$(1, 1) = "X"
BO$(0, 0) = "O"
BO$(0, 2) = "X"
BO$(1, 0) = "O"
```

Schermtekst- en kleurenarrays

Een BASIC 65-programma kan tekst op het scherm weergeven op verschillende manieren. Het **PRINT**-commando toont een tekenreeks op de huidige cursorpositie, wat vooral handig is voor terminal-achtige uitvoer. Het **CURSOR**-commando verplaatst de cursor naar een opgegeven positie. Een programma kan deze commando's samen gebruiken om afbeeldingen of gebruikersinterfaces te tekenen.

Een programma kan individuele tekens op het scherm benaderen met behulp van de speciale ingebouwde arrays **T@&()** en **C@&()**. Deze arrays zijn tweedimensionaal met indexen die overeenkomen met de kolom en rij van elk teken op het scherm, beginnend bij (0,0) in de linkerbovenhoek.

T@&(kolom**,** rij**)** is de schermcode van het teken. Schermcodes verschillen van PETSCII-codes. Zie bijlage 3 op bladzijde 305 voor een lijst met schermcodes.

C@&(kolom**,** rij**)** is de kleurcode van het teken. Dit is een itemnummer van het systeempalet. Zie bijlage 6 op bladzijde 327 voor de lijst met kleuren in het standaard systeempalet. De hogere bits stellen ook tekstattributen in, zoals knipperen.

Net als bij gewone arrays kunnen de waarden van scherm- en kleurarrays worden toegewezen of worden gebruikt in expressies om hun huidige waarden op te halen.

```
10 FOR X = 10 TO 30
20 T@&(X, 2) = 1
30 C@&(X, 2) = INT(RND(1) * 16)
40 NEXT X
50 PRINT "KLEUR OP POSITIE 15: "; C@&(15, 2)
```

De afmetingen van deze arrays hangen af van de huidige tekstschermmodus. In 80 × 25 tekstmodus varieert de kolom van 0 – 79 en de rij van 0 – 24. De MEGA65 ondersteunt ook 80 × 50 en 40 × 25 tekstmodi.

BASIC 65-operatoren

Een *operator* is een symbool of trefwoord dat een functie uitvoert in een expressie. De operator werkt op een of twee deelexpressies, die *operanden* worden genoemd. De operator en zijn operanden *evalueren tot* het resultaat van de bewerking.

De * (asterisk) operator voert bijvoorbeeld een vermenigvuldiging uit van twee numerieke operanden. De operator en zijn operanden evalueren tot het resultaat van de vermenigvuldiging.

```
A = 6
PRINT A * 7
```

De +-operator (plus) heeft een verschillende betekenis afhankelijk van het type van de operanden. Bij getaloperanden voegt deze de waarden samen. Bij tekenreeksoperanden voegt deze de twee tekenreeksen samen.

```
A = 64
PRINT A + 1

A$ = "MEGA"
PRINT A$ + "65"
```

De --operator (min) aanvaardt één of twee operanden. Bij één getaloperand keert deze de waarde om. Bij twee getaloperanden trekt deze de tweede waarde af van de eerste.

```
A = 64
PRINT -A

PRINT A - 16
```

Het gelijkheidssymbool (=) wordt gebruikt als zowel een toewijzingsopdracht als een relationele operator. Als toewijzing is het =-symbool een opdracht die de waarde van een variabele bijwerkt. De linkerzijde moet een variabele of array-elementreferentie zijn en het type moet overeenkomen met het type van de expressie aan de rechterzijde. De toewijzing is geen operator: deze maakt geen deel uit van een expressie.

```
AA = 7
NA$ = "DEFT"
```

Als relationele operator gedraagt het =-symbool zich als een expressie. Het evalueert de expressie aan weerszijden van de operator en test vervolgens of de waarden gelijk zijn. Als ze gelijk zijn, evalueert de gelijkheidsoperator tot -1, de weergave van BASIC van "true". Als ze niet gelijk zijn, evalueert de operator tot 0, oftewel "false". De gelijkheidsexpressie kan worden gebruikt met een **IF**-opdracht om de programmastroom te sturen, of als onderdeel van een numerieke expressie. Beide expressies moeten van hetzelfde type zijn.

```
100 IF X = 99 THEN 130
110 X = X + 1
120 GOTO 100
130 PRINT "GEREED."
```

BASIC 65 onderscheidt toewijzing en gelijkheid op basis van de context. Bekijk de volgende twee regels:

```
A = B = 10
```

BASIC 65 verwacht een opdracht en herkent een variabelenaam gevolgd door het symbool =. Het concludeert dat dit een opdracht is die een waarde toekent aan de numerieke variabele A. Vervolgens verwacht het een numerieke expressie rechts van het symbool =.

De operatoren **NOT**, **AND**, **OR** en **XOR** kunnen zowel als logische operatoren als bitgewijze operatoren worden gebruikt. Bij gebruik als logische operatoren werken deze op de booleaanse waarden van hun operanden. Bij gebruik als bitgewijze operatoren werken deze op de afzonderlijke bits van hun operanden.

```
IF X = 99 OR Y < 5 THEN 130

IF Y > 10 AND Y < 20 THEN 150
```

Een booleaanse operator aanvaardt twee getaloperanden en voert een berekening uit op de bits van de operanden. Het resultaat is een getalwaarde.

```
A = 17
PRINT A AND 20
```

In tegenstelling tot andere gevallen waarin operatoren zich verschillend gedragen afhankelijk van hoe ze worden gebruikt, hoeft BASIC 65 niet te bepalen of deze operatoren zich gedragen als logische operatoren of booleaanse operatoren. Omdat "true" en "false" worden weergegeven met zorgvuldig gekozen getallen, gedragen de logische operatoren zich hetzelfde of hun operanden nu voorwaardelijke expressies of getallen zijn. Een "true"-voorwaardelijke expressie is het getal -1, dat intern een binair getal is met alle bits gezet. De logische expressie "true and false" komt overeen

met de binaire booleaanse expressie %....0000 & %....1111. In dit geval evalueert de **AND**-operator tot 0, wat "false" is.

Voorwaardelijke expressies die getallen opleveren kunnen bij sommige slimme programmeertechnieken worden gebruikt. Bijvoorbeeld:

```
A = A - (B > 7)
```

Deze opdracht verhoogt de waarde in A met 1 als de waarde in B groter is dan 7. Anders laat hij deze ongewijzigd. Als de sub-expressie B > 7 waar is, evalueert deze tot -1. A - (-1) is equivalent aan A + 1. Als de sub-expressie onwaar is, evalueert deze tot 0, en A - 0 is equivalent aan A.

Wanneer meerdere operatoren in een enkele expressie worden gebruikt, wordt de volgorde waarin ze worden geëvalueerd gespecificeerd door *precedence*. In de opdracht A * A - B * B worden bijvoorbeeld eerst beide vermenigvuldigingen uitgevoerd en dan de aftrekking. Net als in de algebra kunt u haakjes gebruiken om de volgorde van uitvoering te wijzigen. In de expressie A * (A - B) * B wordt de aftrekking eerst uitgevoerd.

De volledige reeks operatoren en hun volgorde van voorrang worden samengevat in de hiernavolgende secties.

Toegewezen instructie

Symbool	Beschrijving	Voorbeelden
=	Toewijzing	A = 42, A$ = "HALLO", A = B < 42

Unaire wiskundige operatoren

Naam	Symbool	Beschrijving	Voorbeeld
Plus	+	Positief teken	A = +42
Min	-	Negatief teken	B = -42

Binaire wiskundige operatoren

Naam	Symbool	Beschrijving	Voorbeeld
Plus	+	Optellen	A = B + 42
Min	-	Aftrekken	B = A - 42
Ster	*	Vermenigvuldigen	C = A * B
Schuine streep	/	Delen	D = B / 13
Omhoog-pijl	↑	Machtsverheffen	E = 2 ↑ 10
Links verschuiven	<<	Links verschuiven	A = B << 2
Rechts verschuiven	>>	Rechts verschuiven	A = B >> 1

OPMERKING: Het teken ↑ dat voor de verheffing tot een macht wordt gebruikt, wordt ingetoetst met [↑], naast [RESTORE].

Relationele operatoren

Symbool	Beschrijving	Voorbeeld
)	Groter dan	A) 42
)=	Groter dan of gelijk aan	B)= 42
(	Kleiner dan	A (42
(=	Kleiner dan of gelijk aan	B (= 42
=	Gelijk	A = 42
()	Niet gelijk	B () 42

Logische operatoren

Trefwoord	Beschrijving	Voorbeeld
AND	En	A) 42 AND A (84
OR	Of	A) 42 OR A = 0
XOR	Exclusief-of	A) 42 XOR B) 42
NOT	Ontkenning	C = NOT A) B

Booleaanse operatoren

Trefwoord	Beschrijving	Voorbeeld
AND	En	A = B AND $FF
OR	Of	A = B OR $80
XOR	Exclusief-of	A = B XOR 1
NOT	Ontkenning	A = NOT 22

Tekenreekshandeling

Naam	Symbool	Beschrijving	Operandtype	Voorbeeld
Plus	+	Voegt tekenreeksen samen	Tekenreeks	A$ = B$ + ".PRG"

Operatorrangorde

Rangorde	Operatoren
Hoog	↑
	+ - (unair wiskundig)
	* /
	+ - (binair wiskundig)
	(( )) (rekenkundig verschuiven)
	((=))= = ()
	NOT
	AND
Laag	OR XOR

Trefwoorden en tokens deel 1

*	AC	COLLISION	FE17	EXP	BD
+	AA	COLOR	E7	FAST	FE25
-	AB	CONCAT	FE13	FGOSUB	FE48
/	AD	CONT	9A	FGOTO	FE47
<	B3	COPY	F4	FILTER	FE03
<<	FE52	COS	BE	FIND	FE2B
=	B2	CURSOR	FE41	FN	A5
>	B1	CUT	E4	FONT	FE46
>>	FE53	DATA	83	FOR	81
ABS	B6	DCLEAR	FE15	FOREGROUND	FE39
AND	AF	DCLOSE	FE0F	FORMAT	FE37
APPEND	FE0E	DEC	D1	FRE	B8
ASC	C6	DECBIN	CE11	FREAD#	FE1C
ATN	C1	DEF	96	FREEZER	FE4A
AUTO	DC	DELETE	F7	FWRITE#	FE1E
BACKGROUND	FE3B	DIM	86	GCOPY	FE32
BACKUP	F6	DIR	EE	GET	A1
BANK	FE02	DISK	FE40	GO	CB
BEGIN	FE18	DLOAD	F0	GOSUB	8D
BEND	FE19	DMA	FE1F	GOTO	89
BLOAD	FE11	DMODE	FE35	GRAPHIC	DE
BOOT	FE1B	DO	EB	HEADER	F1
BORDER	FE3C	DOPEN	FE0D	HELP	EA
BOX	E1	DOT	FE4C	HEX$	D2
BSAVE	FE10	DPAT	FE36	HIGHLIGHT	FE3D
BUMP	CE03	DSAVE	EF	IF	8B
BVERIFY	FE28	DVERIFY	FE14	IMPORT	DD
CATALOG	FE0C	ECTORY	FE29	INFO	FE4D
CHANGE	FE2C	EDIT	FE45	INPUT	85
CHAR	E0	EDMA	FE21	INPUT#	84
CHDIR	FE4B	ELLIPSE	FE30	INSTR	D4
CHR$	C7	ELSE	D5	INT	B5
CIRCLE	E2	END	80	JOY	CF
CLOSE	A0	ENVELOPE	FE0A	KEY	F9
CLR	9C	ERASE	FE2A	LEFT$	C8
CMD	9D	ERR$	D3	LEN	C3
COLLECT	F3	EXIT	ED	LET	88

Trefwoorden en tokens deel 2

LINE	E5	PRINT	99	SPEED	FE26
LIST	9B	PRINT#	98	SPRCOLOR	FE08
LOAD	93	RCOLOR	CD	SPRITE	FE07
LOADIFF	FE43	RCURSOR	FE42	SPRSAV	FE16
LOCK	FE50	READ	87	SQR	BA
LOG	BC	RECORD	FE12	STEP	A9
LOG10	CE08	REM	8F	STOP	90
LOOP	EC	RENAME	F5	STRBIN$	CE12
LPEN	CE04	RENUMBER	F8	STR$	C4
MEM	FE23	RESTORE	8C	SYS	9E
MERGE	E6	RESUME	D6	TAB(	A3
MID$	CA	RETURN	8E	TAN	C0
MKDIR	FE51	RGRAPHIC	CC	TEMPO	FE05
MOD	CE0B	RIGHT$	C9	THEN	A7
MONITOR	FA	RMOUSE	FE3F	TO	A4
MOUNT	FE49	RND	BB	TRAP	D7
MOUSE	FE3E	RPALETTE	CE0D	TROFF	D9
MOVSPR	FE06	RPEN	D0	TRON	D8
NEW	A2	RPLAY	CE0F	TYPE	FE27
NEXT	82	RREG	FE09	UNLOCK	FE4F
NOT	A8	RSPCOLOR	CE07	UNTIL	FC
OFF	FE24	RSPEED	CE0E	USING	FB
ON	91	RSPPOS	CE05	USR	B7
OPEN	9F	RSPRITE	CE06	VAL	C5
OR	B0	RUN	8A	VERIFY	95
PAINT	DF	RWINDOW	CE09	VIEWPORT	FE31
PALETTE	FE34	SAVE	94	VOL	DB
PASTE	E3	SAVEIFF	FE44	VSYNC	FE54
PEEK	C2	SCNCLR	E8	WAIT	92
PEN	FE33	SCRATCH	F2	WHILE	FD
PIXEL	CE0C	SCREEN	FE2E	WINDOW	FE1A
PLAY	FE04	SET	FE2D	WPEEK	CE10
POINTER	CE0A	SGN	B4	WPOKE	FE1D
POKE	97	SIN	BF	XOR	E9
POLYGON	FE2F	SLEEP	FE0B	^	AE
POS	B9	SOUND	DA		
POT	CE02	SPC(	A6		

Tokens en trefwoorden deel 1

80 END	A5 FN	CA MID$
81 FOR	A6 SPC(	CB GO
82 NEXT	A7 THEN	CC RGRAPHIC
83 DATA	A8 NOT	CD RCOLOR
84 INPUT#	A9 STEP	CE02 POT
85 INPUT	AA +	CE03 BUMP
86 DIM	AB -	CE04 LPEN
87 READ	AC *	CE05 RSPPOS
88 LET	AD /	CE06 RSPRITE
89 GOTO	AE ^	CE07 RSPCOLOR
8A RUN	AF AND	CE08 LOG10
8B IF	B0 OR	CE09 RWINDOW
8C RESTORE	B1 >	CE0A POINTER
8D GOSUB	B2 =	CE0B MOD
8E RETURN	B3 <	CE0C PIXEL
8F REM	B4 SGN	CE0D RPALETTE
90 STOP	B5 INT	CE0E RSPEED
91 ON	B6 ABS	CE0F RPLAY
92 WAIT	B7 USR	CE10 WPEEK
93 LOAD	B8 FRE	CE11 DECBIN
94 SAVE	B9 POS	CE12 STRBIN$
95 VERIFY	BA SQR	CF JOY
96 DEF	BB RND	D0 RPEN
97 POKE	BC LOG	D1 DEC
98 PRINT#	BD EXP	D2 HEX$
99 PRINT	BE COS	D3 ERR$
9A CONT	BF SIN	D4 INSTR
9B LIST	C0 TAN	D5 ELSE
9C CLR	C1 ATN	D6 RESUME
9D CMD	C2 PEEK	D7 TRAP
9E SYS	C3 LEN	D8 TRON
9F OPEN	C4 STR$	D9 TROFF
A0 CLOSE	C5 VAL	DA SOUND
A1 GET	C6 ASC	DB VOL
A2 NEW	C7 CHR$	DC AUTO
A3 TAB(	C8 LEFT$	DD IMPORT
A4 TO	C9 RIGHT$	DE GRAPHIC

Tokens en trefwoorden deel 2

DF PAINT	FE08 SPRCOLOR	FE2F POLYGON
E0 CHAR	FE09 RREG	FE30 ELLIPSE
E1 BOX	FE0A ENVELOPE	FE31 VIEWPORT
E2 CIRCLE	FE0B SLEEP	FE32 GCOPY
E3 PASTE	FE0C CATALOG	FE33 PEN
E4 CUT	FE0D DOPEN	FE34 PALETTE
E5 LINE	FE0E APPEND	FE35 DMODE
E6 MERGE	FE0F DCLOSE	FE36 DPAT
E7 COLOR	FE10 BSAVE	FE37 FORMAT
E8 SCNCLR	FE11 BLOAD	FE39 FOREGROUND
E9 XOR	FE12 RECORD	FE3B BACKGROUND
EA HELP	FE13 CONCAT	FE3C BORDER
EB DO	FE14 DVERIFY	FE3D HIGHLIGHT
EC LOOP	FE15 DCLEAR	FE3E MOUSE
ED EXIT	FE16 SPRSAV	FE3F RMOUSE
EE DIR	FE17 COLLISION	FE40 DISK
EF DSAVE	FE18 BEGIN	FE41 CURSOR
F0 DLOAD	FE19 BEND	FE42 RCURSOR
F1 HEADER	FE1A WINDOW	FE43 LOADIFF
F2 SCRATCH	FE1B BOOT	FE44 SAVEIFF
F3 COLLECT	FE1C FREAD#	FE45 EDIT
F4 COPY	FE1D WPOKE	FE46 FONT
F5 RENAME	FE1E FWRITE#	FE47 FGOTO
F6 BACKUP	FE1F DMA	FE48 FGOSUB
F7 DELETE	FE21 EDMA	FE49 MOUNT
F8 RENUMBER	FE23 MEM	FE4A FREEZER
F9 KEY	FE24 OFF	FE4B CHDIR
FA MONITOR	FE25 FAST	FE4C DOT
FB USING	FE26 SPEED	FE4D INFO
FC UNTIL	FE27 TYPE	FE4F UNLOCK
FD WHILE	FE28 BVERIFY	FE50 LOCK
FE02 BANK	FE29 ECTORY	FE51 MKDIR
FE03 FILTER	FE2A ERASE	FE52 <<
FE04 PLAY	FE2B FIND	FE53 >>
FE05 TEMPO	FE2C CHANGE	FE54 VSYNC
FE06 MOVSPR	FE2D SET	
FE07 SPRITE	FE2E SCREEN	

ALFABETISCH OVERZICHT BASIC-COMMANDO'S

ABS

Token: $B6

Formaat: **ABS**(x)

Retourneert:

De absolute waarde van het numerieke argument **x**.

x numeriek geheel- of reëel getal als expressie-argument.

Opmerkingen:

Het resultaat is van het type reëel.

Voorbeelden:

Gebruik van **ABS**

```
PRINT ABS(-123)
 123

PRINT ABS(4.5)
 4.5

PRINT ABS(-4.5)
 4.5
```

AND

Token: $AF

Formaat: operand **AND** operand

Gebruik: Voert een bitgewijze logische AND-bewerking uit op twee 16-bit-waarden.

Gehele operanden worden ongewijzigd gebruikt. Reële operanden worden omgezet naar een geheel getal van 16 bit in tweecomplementformaat (waarbij de precisie verloren gaat). Logische operanden worden omgezet naar geheel getal van 16 bit met $FFFF (-1) voor TRUE en $0000 (0) voor FALSE.

Expressie	Resultaat
0 AND 0	0
0 AND 1	0
1 AND 0	0
1 AND 1	1

Opmerkingen:

Het resultaat is van het type geheel getal. Indien het resultaat in een logische context wordt gebruikt, wordt de waarde 0 als ONWAAR beschouwd en alle andere niet-nul-waarden als WAAR.

Voorbeelden:

Gebruik van **AND**

```
PRINT 1 AND 3
 1

PRINT 128 AND 64
 0
```

Gebruik van **AND** in een logische context (**IF**)

```
IF (C >= 0 AND C < 256) THEN PRINT "BYTE VALUE"
```

APPEND

Token: $FE $0E

Formaat: **APPEND#** kanaal**,** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Opent een bestaand bestand van het type **SEQ** of **USR** voor schrijven en plaatst de schrijfcursor aan het einde van het bestand.

kanaalnummer waarbij:

- 1 <= kanaal <= 127: regelafsluiting is CR.
- 128 <= kanaal <= 255: regelafsluiting is CR LF.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

APPEND# werkt op dezelfde manier als **DOPEN#... ,W**, met dien verstande dat het bestand al moet bestaan.

De inhoud van het bestand behouden en alle afgedrukte tekst wordt aan het einde toegevoegd.

Het proberen om **APPEND#** op een niet-bestaand bestand uit te voeren, geeft een DOS-foutmelding.

Voorbeelden:

Gebruik van **APPEND#** (bestaande bestanden openen)

```
APPEND#5, "DATA", U9

APPEND#130, (DD$), U(UN%)

APPEND#3, "USER FILE,U"

APPEND#2, "DATA BASE"
```

ASC

Token: $C6

Formaat: **ASC**(tekenreeks)

Retourneert:

De PETSCII-code van het eerste teken van de tekenreeks als argument, als getal.

Opmerkingen:

ASC retourneert nul voor lege tekenreeksen. Dit verschilt van BASIC 2, dat een Illegal Quantity-fout gaf voor lege tekenreeksen.

De inverse functie van **ASC** is **CHR$**. Zie de functie **CHR$** op bladzijde 52 voor meer informatie.

De naam was blijkbaar gekozen als een geheugensteuntje voor “ASCII”, maar de geretourneerde waarde is een PETSCII-code.

Voorbeelden:

Gebruik van **ASC**

```
PRINT ASC("MEGA")
 77

PRINT ASC("")
 0
```

ATN

Token: $C1

Formaat: **ATN**(numerieke expressie)

Retourneert:

De arctangens van het argument.

Het resultaat ligt in het bereik ($-\pi/2$ tot $\pi/2$).

Opmerkingen:

Een vermenigvuldiging van het resultaat met $180/\pi$ zet de waarde om naar de eenheid "graden". **ATN** is de inverse functie van **TAN**.

Voorbeelden:

Gebruik van **ATN**

```
PRINT ATN(0.5)
 0.46364761

PRINT ATN(0.5) * 180 / π
 26.565051
```

AUTO

Token: $DC

Formaat: **AUTO** [stap]

Gebruik: Schakelt automatische regelnummering in of uit tijdens het invoeren van een BASIC-programma. Nadat een nieuwe programmaregel aan de BASIC-editor is aangeboden met RETURN, genereert de functie **AUTO** een nieuw BASIC-regelnummer voor de invoer van de volgende regel. Het nieuwe nummer wordt berekend door **stap** op te tellen bij het huidige regelnummer.

stap regelnummerstijging.

Het typen van **AUTO** zonder argument schakelt de functie uit.

Voorbeelden:

Gebruik van **AUTO**

```
AUTO 10 : REM GEBRUIK AUTO MET STAPPEN VAN 10

AUTO    : REM ZET AUTO AF
```

BACKGROUND

Token: $FE $3B

Formaat: **BACKGROUND** kleur

Gebruik: Stelt de achtergrondkleur van het scherm in.

kleur paletindex, in het bereik 0 - 255

Alle kleuren binnen dit bereik zijn aanpasbaar via het commando **PALETTE**.

Bij opstarten heeft de MEGA65 slechts de eerste 32 kleuren geconfigureerd. Zie bijlage 6 op bladzijde 327 voor de lijst met kleuren in het standaard systeempalet.

Voorbeeld: Gebruik van **BACKGROUND**

```
BACKGROUND 3 : REM KIES CYAAN ALS ACHTERGRONDKLEUR
```

BACKUP

Token: $F6

Formaat: **BACKUP U** bron **TO U** doel
BACKUP D bron **TO D** doel [**,U** eenheid]

Gebruik: Kopieert een schijf naar een andere.

De eerste vorm van **BACKUP**, waarbij eenheden voor bron en doel worden opgegeven, kan alleen worden gebruikt voor de stations die op de interne FDC (Floppy Disk Controller) zijn aangesloten. Eenheden 8 en 9 zijn gereserveerd voor deze controller. Dit kunnen de interne diskette (eenheid 8) en een andere diskette (eenheid 9) zijn die op dezelfde lintkabel is aangesloten, of gemonteerde D81-schijfimages. **BACKUP** kan worden gebruikt om van diskette naar diskette, diskette naar image, image naar diskette en image naar image te kopiëren, afhankelijk van de image-mounts en de aanwezigheid van een tweede fysieke diskette.

De tweede vorm van **BACKUP**, waarbij stations voor bron en doel worden opgegeven, is bedoeld voor dual-drive-eenheden die op de IEC-bus zijn aangesloten. Bijvoorbeeld: CBM 4040, 8050, 8250 via een IEEE-488-naar-IEC-adapter. In dit geval wordt de back-up intern door het schijfstation uitgevoerd.

bron station of schijf# van bronstation.

doel station of schijf# van doelschijf.

Opmerkingen:

De doelschijf wordt geformatteerd en er wordt een identieke kopie van de bronschijf geschreven.

BACKUP kan niet worden gebruikt om van interne apparaten naar IEC-apparaten te kopiëren of omgekeerd.

Voorbeelden:

Gebruik van **BACKUP**

```
BACKUP U8 TO U9      : REM BACKUP STATION OP UNIT 8 NAAR UNIT 9

BACKUP U9 TO U8      : REM BACKUP STATION OP UNIT 9 NAAR UNIT 8

BACKUP D0 TO D1, U10 : REM BACKUP OP EEN DUBBELDRIVE VERBONDEN VIA IEC
```

BANK

Token: $FE $02

Formaat: **BANK** bank getal

Gebruik: Selecteert de geheugenconfiguratie voor BASIC-commando's die 16-bit-adressen gebruiken. Dit zijn **LOAD**, **LOADIFF**, **PEEK**, **POKE**, **SAVE**, **SYS** en **WAIT**. Zie de geheugenindeling van het systeem in *het grote Mega65-boek*, Systeemgeheugenindeling (Bijlage J) voor meer informatie.

Opmerkingen:

Een waarde > 127 selecteert geheugentoegewezen I/O.

De standaardwaarde voor het banknummer bij het opstarten van het systeem is 128.

Deze configuratie heeft RAM van $0000 tot $1FFF, de BASIC- en KERNAL-ROM en I/O van $2000 tot $FFFF.

Voorbeeld: Gebruik van **BANK**

```
BANK 1 : REM KIES GEHEUGENCONFIGURATIE 1
```

BEGIN

Token: $FE $18

Formaat: **BEGIN** ... **BEND**

Gebruik: Het begin van een samengestelde instructie die na **THEN** of **ELSE** moet worden uitgevoerd. Hiermee wordt de beperking van één enkele regel van de standaard **IF** ... **THEN** ... **ELSE**-constructie overwonnen.

Opmerkingen:
Gebruik **GOTO** of **GOSUB** niet om in een samengestelde instructie te springen, omdat dit tot onverwachte resultaten kan leiden.

Voorbeeld: Gebruik van **BEGIN** ... **BEND**

```
10 GET A$
20 IF A$ >= "A" AND A$ <= "Z" THEN BEGIN : REM NEGEER ALLES BEHALVE (A-Z)
30 PW$ = PW$ + A$
40 IF LEN(PW$) > 7 THEN 90
50 BEND
60 IF A$ <> CHR$(13) GOTO 10
90 PRINT "PW="; PW$
```

BEND

Token: $FE $19

Formaat: **BEGIN** ... **BEND**

Gebruik: Het einde van een samengestelde instructie die na **THEN** of **ELSE** moet worden uitgevoerd. Hiermee wordt de beperking van één enkele regel van de standaard **IF** ... **THEN** ... **ELSE**-constructie overwonnen.

Opmerkingen:
Het onderstaande voorbeeld toont een eigenaardigheid in de implementatie van de samengestelde instructie. Indien de voorwaarde als **FALSE** wordt beoordeeld, wordt de uitvoering niet direct na **BEND** zoals het zou moeten, maar aan het begin van de volgende regel.

Voorbeeld: Gebruik van **BEGIN** ... **BEND**

```
5 REM DEMO'T DE BESCHREVEN EIGENAARDIGHEID
10 IF Z > 1 THEN BEGIN : A$ = "EEN"
20 B$ = "TWEE"
30 PRINT A$; " "; B$; : BEND : PRINT " RAAR!"
40 REM PROGRAMMA LOOPT HIER VERDER VOOR Z <= 1
```

BLOAD

Token: $FE $11

Formaat: **BLOAD** bestandsnaam [**,B** bank] [**,P** adres] [**,R**] [**,D** station] [**,U** eenheid]

Gebruik: Laadt een bestand van het type **PRG** in RAM op adres **P**. ("Binair laden.")

BLOAD heeft twee modi: De platte geheugenmodus kan worden gebruikt om een programma te laden op elk adres in het 28-bit (256 MB) adresbereik waar RAM aanwezig is. Dit omvat de standaard RAM-banken 0 tot 5, alsmede de 8 MB "zolder-RAM" op adres $8000000 (800.0000).

Deze modus wordt geactiveerd door een adres bij parameter **P** op te geven dat groter is dan $FFFF. De bankparameter wordt in deze modus genegeerd.

Om compatibiliteit met oudere BASIC-versies te waarborgen, accepteert **BLOAD** ook de syntaxis met een 16-bit-adres op P en een banknummer op B. De zolder-RAM valt buiten het bereik van deze compatibiliteitsmodus.

De optionele parameter **R** ("ruwe modus") interpreteert of gebruikt de eerste twee bytes van het bestand niet als laadadres, hetgeen anders het standaardgedrag is. In de ruwe modus wordt elke byte als gegevens gelezen.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

bank specificeert de te gebruiken RAM-bank. Indien niet opgegeven, wordt de huidige bank, zoals ingesteld met het laatste **BANK**-statement, gebruikt.

adres overschrijft het laadadres dat in de eerste twee bytes van het **PRG**-bestand is opgeslagen.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

BLOAD kan geen bankgrenzen overschrijden.

BLOAD gebruikt het laadadres uit het bestand indien geen **P**-parameter is opgegeven.

Het is mogelijk om **BLOAD** te dwingen een bestand van het type **SEQ** te laden met behulp van de bestandsnaamextensie: `",S"`. Zorg er ook voor

dat de ruwe modus is ingeschakeld (**,R**) om te voorkomen dat de eerste twee bytes als een **PRG**-laadadres worden behandeld.

Voorbeelden:

Gebruik van **BLOAD**

```
BLOAD "ML DATA", B0, U9

BLOAD "SPRITES"

BLOAD "ML ROUTINES", B1, P32768

BLOAD (FI$), B(BA%), P(PA), U(UN%)

BLOAD "CHUNK", P($8000000)

BLOAD "USER DATA,S", P($1600), R
```

BOOT

Token: $FE $1B

Formaat: **BOOT** bestandsnaam [**,B** bank][**,P** adres] [**,D** station] [**,U** eenheid]
BOOT SYS
BOOT

Gebruik: Laadt en voert een programma of bootsector van een schijf uit.

BOOT bestandsnaam laadt een bestand van het type **PRG** in het RAM op adres **P** en bank **B** en begint de code op het laadadres uit te voeren.

BOOT SYS laadt de bootsector (512 bytes in totaal) van sector 0, spoor 1 en apparaat 8 naar adres $0400 in bank 0 en voert daarna een JSR `$0400` uit (Jump To Subroutine).

BOOT zonder parameters probeert een bestand met de naam `AUTOBOOT.C65` van het standaardapparaat te laden en uit te voeren. Dit is een verkorte versie van: `RUN "AUTOBOOT.C65"`

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

bank geeft de te gebruiken RAM-bank op. Indien niet opgegeven, wordt de huidige bank gebruikt zoals die met het laatste **BANK**-commando is ingesteld.

adres overschrijft het laadadres dat in de eerste twee bytes van het **PRG**-bestand is opgeslagen.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Voorbeelden:
Gebruik van **BOOT**

```
BOOT SYS

BOOT (FI$), B(BA%), P(PA), U(UN%)

BOOT
```

BORDER

Token: $FE $3C

Formaat: **BORDER** kleur

Gebruik: Stelt de randkleur van het scherm in.

kleur paletindex in het bereik 0 - 255.

Alle kleuren in dit bereik zijn aanpasbaar met het **PALETTE**-commando. Zie bijlage 6 op bladzijde 327 voor de lijst met kleuren in het standaard systeempalet.

Voorbeeld: Gebruik van **BORDER**

```
BORDER 4 : REM KIES DE RANDKLEUR PAARS
```

BOX

Token: $E1

Formaat: **BOX** x0,y0, x2,y2 [, massief]
BOX x0,y0, x1,y1, x2,y2, x3,y3 [, massief]

Gebruik: Bitmapgrafiek: tekent een kader.

De eerste vorm van **BOX** met twee coördinaatparen en een optionele parameter **massief** tekent een eenvoudige rechthoek, waarbij de coördinaatparen twee diagonaal tegenoverliggende hoeken aangeven.

De tweede vorm met vier coördinaatparen geeft een pad van vier punten op, die met lijnen worden verbonden. Het pad wordt gesloten door de laatste coördinaat met de eerste te verbinden.

De vierhoek wordt getekend met de huidige tekencontext die met **SCREEN**, **PALETTE** en **PEN** is ingesteld. De vierhoek wordt gevuld indien de parameter **massief** niet 0 is.

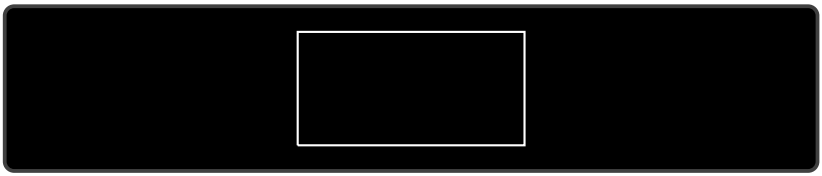
Opmerkingen:

BOX kan met vier coördinaatparen worden gebruikt om elke vorm te tekenen die met vier punten kan worden gedefinieerd, niet alleen rechthoeken. Bijvoorbeeld ruiten, vliegers, trapezoïden en parallellogrammen. Het is ook mogelijk strikachtige vormen te tekenen.

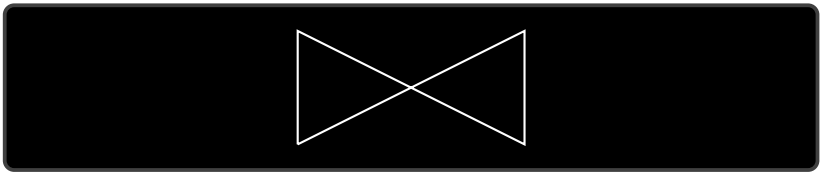
Voorbeelden:

Gebruik van **BOX**

```
BOX 0, 0,  160, 0,  160, 80,  0, 80
```



```
BOX 0, 0,  160, 80,  160, 0,  0, 80
```



```
BOX 20, 0,  140, 0, 160, 80,  0, 80
```



BSAVE

Token: $FE $10

Formaat: **BSAVE** bestandsnaam**, P** begin **TO P** einde [**,B** bank] [**,R**] [**,D** station] [**,U** eenheid]

Gebruik: Slaat een geheugenbereik op als een bestand van het type **PRG**. ("Binair opslaan.")

BSAVE heeft twee modi:

De vlakke geheugenmodus kan worden gebruikt om een geheugenblok op te slaan in het 28-bits (256 MB) adresbereik waarin het RAM is geïnstalleerd. Dit omvat de standaard RAM-banken 0 tot 5, alsmede de 8 MB "zolder-RAM" op adres $8000000 (800.0000).

Deze modus wordt geactiveerd door adressen voor de begin- en eindparameter **P** op te geven die groter zijn dan $FFFF. De bankparameter wordt in deze modus genegeerd. Met deze vlakke geheugenmodus kunnen bereiken groter dan 64 KB worden opgeslagen.

Om compatibiliteitsredenen met oudere BASIC-versies aanvaardt **BSAVE** ook de syntaxis met 16-bits adressen bij **P** en een banknummer bij **B**. De zolder-RAM valt buiten het bereik van deze compatibiliteitsmodus. Deze modus kan geen bankgrenzen overschrijden, dus het begin- en eindadres moeten zich in dezelfde bank bevinden.

De optionele parameter **R** ("ruwe modus") voorkomt dat het 16-bits startadres als de eerste twee bytes wordt geschreven. De adresheader van twee bytes is nodig voor PRG-bestanden die op het opgeslagen adres worden geladen, maar is doorgaans ongewenst voor SEQ-bestanden. **BSAVE** schrijft standaard het startadres als de eerste twee bytes van het bestand.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

Indien het eerste teken van de bestandsnaam het `@`-teken (`@`) is, wordt dit geïnterpreteerd als een "opslaan en vervangen"-operatie. Het wordt afgeraden deze optie op 1541- en 1571-stations te gebruiken, omdat deze een "save and replace bug" in hun DOS bevatten.

begin adres waar het opslaan begint. Dit wordt tevens het laadadres dat in de eerste twee bytes van het **PRG**-bestand wordt opgeslagen.

einde adres waar het opslaan eindigt. **einde-1** is het laatste adres dat voor het opslaan wordt gebruikt.

bank te gebruiken RAM-bank. Indien niet opgegeven, wordt de huidige bank gebruikt zoals die met het laatste **BANK**-commando is ingesteld.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

De lengte van het bestand is **einde - begin + 2**.

Indien het getal na een argumentletter geen decimaal getal is (zoals een hexadecimale constante of een variabelenaam), moet het tussen haakjes worden geplaatst.

Bij het opslaan van een bestand van het type **PRG** vormen de eerste twee bytes doorgaans het 16-bits laadadres. Dit is de standaardinstelling. Indien het startadres zich in een andere bank dan nul bevindt (het startadres is groter dan $FFFF), worden alleen de twee laagste bytes van het adres geschreven.

Om een bestand van het type **SEQ** op te slaan, gebruikt u het bestandsnaamsuffix ",S". Voor bestanden van dit type wordt het 16-bits laadadres doorgaans weggelaten. Geef hiervoor het argument **,R** om "ruwe modus" te activeren.

Voorbeelden:

Gebruik van **BSAVE**

```
BSAVE "ML DATA", P 32768 TO P 33792, B0, U9

BSAVE "SPRITES", P 1536 TO P 2058

BSAVE "ML ROUTINES", B1, P($9000) TO P($A000)

BSAVE (FI$), B(BA%), P(PA) TO P(PE), U(UN%)

BSAVE "USER DATA,S", P($1600) TO P($16FF), R
```

BUMP

Token: $CE $03

Formaat: **BUMP**(type)

Retourneert:

Een bitveld van sprites die op dit moment botsen met andere sprites of schermgegevens.

type het type botsing, waarbij 1:sprite-sprite en 2:sprite-schermgegevens.

Elke bit die in de retourwaarde is gezet, geeft aan dat de sprite die overeenkomt met die bitpositie betrokken is geweest bij een botsing sinds het laatste aanroepen van **BUMP**. Het aanroepen van **BUMP** reset de botsingsmasker, zodat u altijd een overzicht krijgt van de botsingen sinds het laatste aanroepen van **BUMP**.

Sprite	Retour-neert	Masker
0	1	0000 0001
1	2	0000 0010
2	4	0000 0100
3	8	0000 1000
4	16	0001 0000
5	32	0010 0000
6	64	0100 0000
7	128	1000 0000

Opmerkingen:

Het is mogelijk meerdere botsingen te detecteren, maar u moet de sprite-coördinaten evalueren om te bepalen welke sprites met elkaar gebotst hebben.

Voorbeeld: Gebruik van **BUMP**

```
10 S% = BUMP(1) : REM BOTSINGEN SPRITE (-) SPRITE
20 IF (S% AND 6) = 6 THEN PRINT "SPRITE 1 & 2 BOTSEN"
30 REM ---
40 S% = BUMP(2) : REM BOTSINGEN SPRITE (-) SCHERMGEGEVENS
50 IF (S% <> 0) THEN PRINT "EEN SPRITE RAAKT SCHERMGEGEVENS"
```

BVERIFY

Token: $FE $28

Formaat: **BVERIFY** bestandsnaam [**,P** adres] [**,B** bank] [**,D** station] [**,U** eenheid]

Gebruik: Vergelijkt een geheugenbereik met een bestand van het type **PRG**. (“Binair verifiëren.”)

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

bank geeft de te gebruiken RAM-bank op. Indien niet opgegeven, wordt de huidige bank gebruikt zoals die met het laatste **BANK**-commando is ingesteld.

adres adres waar de vergelijking begint. Indien de parameter **P** wordt weggelaten, wordt het laadadres gebruikt dat in de eerste twee bytes van het **PRG**-bestand is opgeslagen.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

BVERIFY kan alleen op gelijkheid testen. In de directe modus geeft **BVERIFY** ofwel het bericht `OK` ofwel `VERIFY ERROR` met het adres van het eerste niet-identieke byte. In de programmamodus stopt een Verify-fout de uitvoer of gaat naar de **TRAP**-foutafhandeling, indien deze actief is.

Voorbeelden:

Gebruik van **BVERIFY**

```
BVERIFY "ML DATA", P 32768, B0, U9

BVERIFY "SPRITES", P 1536

BVERIFY "ML ROUTINES", B1, P(DEC("9000"))

BVERIFY (FI$), B(BA%), P(PA), U(UN%)
```

C@&

Token: N.v.t.

Formaat: **C@&**(kolom**,** rij**)**

Gebruik: Leest of stelt de kleurcode van een teken op de gegeven schermcoördinaten in.

Dit gedraagt zich als een arrayvariabele. Door een waarde toe te wijzen, wordt de kleurcode voor het gegeven teken onmiddellijk gewijzigd.

Opmerkingen:

C@& is een gereserveerde systeemvariabele.

Zie bijlage 6 op bladzijde 327 voor de lijst met kleuren in het standaard systeempalet. Bovenste bits stellen ook tekstattributen in, zoals knipperen.

De coördinaten beginnen bij (0, 0) voor de linkerbovenhoek. Indien de gegeven coördinaten buiten de huidige schermmodus vallen, geeft het openen van de array een Bad Subscript-fout.

Voorbeeld: Zie **T@&**.

CATALOG

Token: $FE $0C

Formaat: **CATALOG** [bestandspatroon] [**,W**] [**,R**] [**,D** station] [**,U** eenheid]
$ [bestandspatroon] [**,W**] [**,R**] [**,D** station] [**,U** eenheid]

Gebruik: Drukt een bestandscatalogus/map van de opgegeven schijf af.

De parameter **W** (Breed) geeft de map in drie kolommen op het scherm weer en stopt nadat het scherm is gevuld met een pagina (63 mapitems). Door op een willekeurige toets te drukken, wordt de volgende pagina weergegeven.

De parameter **R** (Herstelbaar) toont alleen de bestanden die als verwijderd zijn gemarkeerd maar nog steeds herstelbaar zijn, elk gemarkeerd met een '–'. Andere bestanden worden niet getoond.

bestandspatroon is een aangehaalde tekenreeks, bijvoorbeeld: `"DA*"` of een tekenreeksexpressie tussen haakjes, bijv. `(DI$)`

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

CATALOG is een synoniem van **DIRECTORY** en **DIR** en geeft dezelfde lijst.

Het patroon **bestandspatroon** kan worden gebruikt om de lijst te filteren. De tekens * en **?** kunnen als jokerteken worden gebruikt. Door **,T=** aan de patroonreeks toe te voegen, waarbij **T** een bestandstype aangeeft van **P**, **S**, **U**, **R** of **C** (voor **P**RG, **S**EQ, **U**SR, **R**EL, **C**BM), wordt de uitvoer op dat bestandstype gefilterd.

Het afkortingssymbool **$** kan alleen in de directe modus worden gebruikt.

Voorbeelden:

Gebruik van **CATALOG**

```
CATALOG
0 "BLACK SMURF     " BS 2A
508  "STORY PHOBOS"    SEQ
27   "C8096"           PRG
25   "C128"            PRG
104 BLOCKS FREE.
```

```
CATALOG "*,T=S"
0 "BLACK SMURF     " BS 2A
508  "STORY PHOBOS"    SEQ
104 BLOCKS FREE.
```

Gebruik van de brede parameter (**W**)

```
CATALOG W
 0 "BASIC EXAMPLES  "
 1 "BEGIN"          P    1 "FREAD"          P    2 "PAINT.COR"      P
 1 "BEND"           P    1 "FRE"            P    3 "PALETTE.COR"    P
 1 "BUMP"           P    2 "GET#"           P    1 "PEEK"           P
 1 "CHAR"           P    1 "GETKEY"         P    3 "PEN"            P
 1 "CHR$"           P    1 "GET"            P    1 "PLAY"           P
 4 "CIRCLE"         P    2 "GOSUB"          P    2 "POINTER"        P
 1 "CLOSE"          P    2 "GOTO.COR"       P    1 "POKE"           P
 1 "CLR"            P    2 "GRAPHIC"        P    1 "POS"            P
 2 "COLLISION"      P    1 "HELP"           P    1 "POT"            P
 1 "CURSOR"         P    1 "IF"             P    1 "PRINT#"         P
 0 "DATA BASE"      R    2 "INPUT#"         P    1 "PRINT"          P
 1 "DATA"           P    2 "INPUT"          P    1 "RCOLOR.COR"     P
 1 "DEF FN"         P    2 "JOY"            P    1 "READ"           P
 1 "DIM"            P    1 "LINE INPUT#"    P    1 "RECORD"         P
 1 "DO"             P    3 "LINE"           P    1 "REM"            P
 5 "ELLIPSE"        P    1 "LOOP"           P    1 "RESTORE"        P
 1 "ELSE"           P    1 "MID$"           P    1 "RESUME"         P
 1 "EL"             P    1 "MOD"            P    1 "RETURN"         P
 1 "ENVELOPE"       P    1 "MOVSPR"         P    1 "REVERS"         S
 2 "EXIT"           P    1 "NEXT"           P    3 "RGRAPHIC"       P
 1 "FOR"            P    2 "ON"             P    1 "RMOUSE"         P
```

CHANGE

Token: $FE $2C

Formaat: **CHANGE /**zoekstring**/ TO /**vervangstring**/** [**,** regel bereik]
CHANGE "zoekstring**" TO "**vervangstring**"** [**,** regel bereik]

Gebruik: Bewerkt het BASIC-programma dat zich momenteel in het geheugen bevindt om alle voorkomens van één tekenreeks door een andere te vervangen.

Een optioneel **regelbereik** beperkt de zoekactie tot dit bereik, anders wordt het volledige BASIC-programma doorzocht. Bij elk voorkomen van de **zoekstring** wordt de regel weergegeven en wordt de gebruiker om een actie gevraagd:

- **Y** RETURN voert de vervanging uit en zoekt de volgende tekenreeks.
- **N** RETURN voert **geen** vervanging uit en zoekt de volgende tekenreeks.
- ***** RETURN vervangt de huidige en alle volgende overeenkomsten.
- RETURN beëindigt het commando en vervangt de huidige overeenkomst niet.

Opmerkingen:

Vrijwel elk teken dat geen onderdeel van de tekenreeks is, inclusief letters en leestekens, kan in plaats van **/** worden gebruikt.

Het gebruik van het dubbele aanhalingsteken zoekt echter naar tekenreeksen die niet getokeniseerd zijn en dus geen onderdeel van een trefwoord vormen. Het voorbeeld `CHANGE "LOOP" TO "OOPS"` zal het BASIC-trefwoord `LOOP` niet vinden, omdat het trefwoord als token en niet als tekst wordt opgeslagen. `CHANGE /LOOP/ TO /OOPS/` vindt en vervangt het wel (wat tot syntaxisfouten kan leiden).

Vanwege een beperking van de BASIC-parser kan **CHANGE** de trefwoorden **REM** en **DATA** niet herkennen. Zie **FIND**.

Kan alleen in de directe modus worden gebruikt.

Voorbeelden:

Gebruik van **CHANGE**

```
CHANGE "XX$" TO "UU$", 2000-2700

CHANGE /IN/ TO /OUT/

CHANGE &IN& TO &OUT&
```

CHAR

Token: $E0

Formaat: **CHAR** kolom**,** rij**,** hoogte**,** breedte**,** richting**,** tekenreeks [**,** adres van tekenset]

Gebruik: Bitmapgraphics: geeft tekst weer op een grafisch scherm.

kolom (in eenheden van tekenposities) begint de positie van de uitvoer horizontaal. Omdat één kolomeenheid 8 pixels breed is, heeft een schermbreedte van 320 een kolombereik van 0 - 39, terwijl een schermbreedte van 640 een kolombereik van 0 - 79 heeft.

rij (in pixel-eenheden) begint de positie van de uitvoer verticaal. In tegenstelling tot de kolomparameter is de eenheid pixels (geen tekenposities), waarbij de bovenste rij de waarde 0 heeft.

hoogte factor die op de verticale grootte van de tekens wordt toegepast, waarbij 1 de normale grootte (8 pixels) is, 2 de dubbele grootte (16 pixels), enzovoort.

breedte factor die op de horizontale grootte van de tekens wordt toegepast, waarbij 1 de normale grootte (8 pixels) is, 2 de dubbele grootte (16 pixels), enzovoort.

richting bepaalt de afdrukrichting:

- 1: omhoog
- 2: rechts
- 4: omlaag
- 8: links

Het optionele **adres van tekenset** kan worden gebruikt om een tekenset te selecteren. Het standaardadres van de tekenset is $29800, het kleine-letter+hoofdlettergedeelte van lettertype A (PETSCII met enkele ASCII-vervangingen).

U kunt elk adres gebruiken waar een tekenset is opgeslagen, inclusief een eigen tekenset die u in het geheugen heeft geladen. Er zijn drie ingebouwde tekensets (zie ook **FONT**), alsmede de tekensetbuffer van de VIC-IV:

- $29000: lettertype A (ASCII)
- $3D000: lettertype B (vetgedrukt)
- $2D000: lettertype C (CBM)
- $FF7E000: tekensetbuffer van de VIC-IV

Om **CHAR** te gebruiken met eigen tekens die met **CHARDEF** zijn gedefinieerd, gebruikt u het adres van de tekensetbuffer van de VIC-IV, $FF7E000.

Een tekenset met de PETSCII-indeling heeft twee subsets: de set met hoofdletters en grafische tekens en de set met kleine letters en hoofdletters. Met **CHAR** kunt u een subset selecteren op basis van het adres: $xx000 voor hoofdletters en grafische tekens en $xx800 voor kleine letters en hoofdletters. Gebruik bijvoorbeeld tekensetadres $2D800 om de kleine letters van lettertype C te selecteren.

tekenreeks constante of expressie die wordt afgedrukt. Deze tekenreeks mag optioneel een of meer van de volgende besturingstekens bevatten:

Expressie	Sneltoets	Zinnebeeld	Beschrijving
CHR$(2)	CTRL+B	B	Lege cel
CHR$(6)	CTRL+F	F	Teken omdraaien
CHR$(9)	CTRL+I	I	EN met scherm
CHR$(15)	CTRL+O	O	OF met scherm
CHR$(24)	CTRL+X	X	XOR met scherm
CHR$(18)	RVSON	R	Omgekeerd
CHR$(146)	RVSOFF	■	Omgekeerd uit
CHR$(147)	CLR	▢	Venster wissen
CHR$(21)	CTRL+U	U	Onderstrepen
CHR$(25)+"-"	CTRL+Y + "-"	Y–	Links draaien
CHR$(25)+"+"	CTRL+Y + "+"	Y+	Rechts draaien
CHR$(26)	CTRL+Z	◪	Spiegelen
CHR$(157)	Cursor links	▌	Links verplaatsen
CHR$(29)	Cursor rechts	▐	Rechts verplaatsen
CHR$(145)	Cursor op	▢	Omhoog verplaatsen
CHR$(17)	Cursor neer	▢	Omlaag verplaatsen

Opmerkingen:

Merk op dat de beginpositie van de tekenreeks verschillende eenheden heeft in de horizontale en verticale richting. Horizontaal is in kolommen en verticaal in pixels.

Raadpleeg de **CHR$**-functie op bladzijde 52 voor meer informatie.

Voorbeeld: Gebruik van **CHAR**

```
10 SCREEN 640, 400, 2
20 CHAR 28, 180, 4, 4, 2, "MEGA65", $29000
30 GETKEY A$
40 SCREEN CLOSE
```

CHARDEF

Token: $E0 $96

Formaat: **CHARDEF** index, bitmatrix

Gebruik: Wijzigt de weergave van een teken.

index schermcode van het teken dat moet worden gewijzigd (@:0, A:1, B:2, enz.). Zie bijlage 3 op bladzijde 305 voor een lijst van schermcodes. Schermcodes 0 - 255 behoren tot de set van hoofdletters en grafische tekens. Schermcodes 256 - 511 behoren tot de set van kleine letters en hoofdletters.

bitmatrix reeks van 8 byte-waarden die de rasterweergave van het teken definiëren, van de bovenste rij naar de onderste rij. Wanneer meer dan 8 waarden als argumenten worden gebruikt, worden waarden 9 - 16 gebruikt voor het teken index+1, 17 - 24 voor index+2, enzovoort.

Opmerkingen:

De wijzigingen in de bitmap van het teken worden toegepast op de tekensetbuffer van de VIC, die zich in het RAM bevindt op het adres FF7.E000.

Alle wijzigingen zijn tijdelijk. De tekensetbuffer van de VIC wordt overschreven door het commando **FONT** en wordt door een reset hersteld naar PETSCII-lettertype C.

Voorbeelden:

Gebruik van **CHARDEF**

```
CHARDEF 1, $FF, $81, $81, $81, $81, $81, $81, $FF : REM VERANDER 'A' IN RECHTHOEK

CHARDEF 9, $18, $18, $18, $18, $18, $18, $18, $00 : REM MAAK 'I' CURSIEF
```

CHDIR

Token: $FE $4B

Formaat: **CHDIR** mapnaam [**,U** eenheid]

Gebruik: Wijzigt de huidige werkmap.

mapnaam de naam van een map. Ofwel een aangehaalde tekenreeks zoals "EENOFANDEREMAP", of een tekenreeksexpressie in haakjes zoals: (DR$)

Afhankelijk van het type **eenheid** wordt **CHDIR** op verschillende bestandssystemen toegepast.

UNIT 12 is gereserveerd voor de SD-kaart (FAT-bestandssysteem). Dit commando kan worden gebruikt om door submappen te navigeren en schijfimages die daar zijn opgeslagen te koppelen. **CHDIR ".",U12** schakelt over naar de bovenliggende map op **UNIT 12**.

Voor andere eenheden die door CBDOS worden beheerd (doorgaans 8 en 9) wordt **CHDIR** gebruikt om naar submappen op een diskette of schijfimage van het type **D81** te wisselen of deze te verlaten. Bestaande submappen worden weergegeven als bestandstype **CBM** in de bovenliggende map en worden aangemaakt met het commando **MKDIR**. **CHDIR "/",U eenheid** schakelt over naar de hoofdmap.

Voorbeelden:

Gebruik van **CHDIR**

```
CHDIR "AVONTUUR", U12 : REM GA AVONTUUR OP SD-KAART BINNEN
CHDIR "..", U12       : REM GA TERUG NAAR BOVENLIGGENDE DIRECTORY
CHDIR "RACEBAAN", U12 : REM GA SUBDIRECTORY RACEBAAN BINNEN
0 "MEGA65          "   1D
800  "MEGA65-SPELLEN"   CBM
800  "MEGA65-TOOLTJES"  CBM
600  "BASICPROGJES"     CBM
960  BLOCKS FREE.

CHDIR "MEGA65 GAMES", U8 : REM GA SUBDIRECTORY OP DISKETTE BINNEN
CHDIR "/", U8            : REM GA TERUG NAAR DE HOOFDDIRECTORY
```

CHR$

Token: $C7

Formaat: **CHR$(**numerieke expressie**)**

Retourneert:
Een tekenreeks met één teken van de opgegeven PETSCII-waarde.

Opmerkingen:
Het bereik van het argument loopt van 0 - 255, zodat deze functie ook kan worden gebruikt om besturingstekens in te voegen in tekenreeksen. Zelfs het NULL-teken met code 0 is toegestaan. **CHR$** is de omgekeerde functie van **ASC**. De volledige tabel van tekens (en hun PETSCII-codes) staat op bladzijde 309.

Voorbeeld: Gebruik van **CHR$**

```
10 QUOTE$  = CHR$(34)
20 ESCAPE$ = CHR$(27)
30 PRINT QUOTE$; "MEGA65"; QUOTE$ : REM PRINT "MEGA65"
40 PRINT ESCAPE$; "Q";           : REM WIS TOT EINDE REGEL
```

CIRCLE

Token: $E2

Formaat: **CIRCLE** xc, yc, straal [, vlaggen, begin, einde]

Gebruik: Bitmapgraphics: tekent een cirkel. Dit is een speciaal geval van **ELLIPSE**, waarbij dezelfde waarde voor de horizontale en verticale straal wordt gebruikt.

xc de x-coördinaat van het middelpunt in pixels.

yc de y-coördinaat van het middelpunt in pixels.

straal de straal van de cirkel in pixels.

vlaggen bepaalt het vullen, de bogen en de positie van de nulradiaal. De standaardinstelling (nul) is niet vullen, de poten van een boog weergeven en de nulradiaal wijst naar 3 uur.

Bit	Naam	Waarde	Actie indien ingesteld
0	vulling	1	Vul ellips of boog met de huidige penkleur
1	poten	2	Onderdruk het tekenen v/d poten van een boog
2	combs	4	Laat de nulradiaal naar 12 uur wijzen

De eenheden voor de begin- en eindhoek zijn graden in het bereik van 0 tot 360. De nulradiaal begint bij 3 uur en beweegt met de klok mee. Het instellen van bit 2 van vlaggen (waarde 4) verplaatst de nulradiaal naar de 12-uurpositie.

begin de beginhoek van de boog.

einde de eindhoek van de boog.

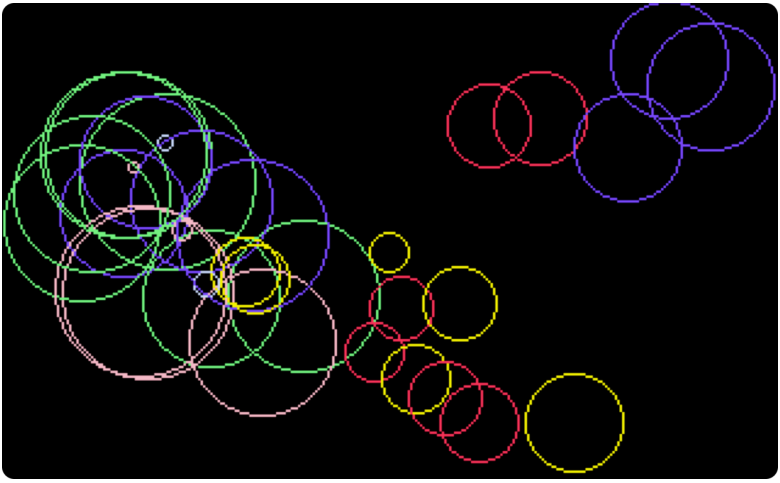
Opmerkingen:

CIRCLE wordt gebruikt om cirkels te tekenen op schermen met een verhouding van 1:1 (bijvoorbeeld: 320 x 200 of 640 x 400). Bij andere resoluties (zoals 640 x 200) wordt de vorm een ellips.

Voorbeeld: Gebruik van **CIRCLE**

```
100 REM CIRKEL (NAAR F.BOWEN)
110 BORDER 0                                          : REM ZWART
120 SCREEN 320, 200, 4                                : REM EENVOUDIGE SCHERMOPZET
130 PALETTE 0, 0,          0,          0,          0 : REM ZWART
140 PALETTE 0, 1, RND(.) * 16, RND(.) * 16,         15 : REM LUKRAKE KLEUREN
150 PALETTE 0, 2, RND(.) * 16,         15, RND(.) * 16
160 PALETTE 0, 3,         15, RND(.) * 16, RND(.) * 16
170 PALETTE 0, 4, RND(.) * 16, RND(.) * 16,         15
180 PALETTE 0, 5, RND(.) * 16,         15, RND(.) * 16
190 PALETTE 0, 6,         15, RND(.) * 16, RND(.) * 16
200 SCNCLR 0                                          : REM WIS SCHERM
210 FOR I = 0 TO 32                                   : REM CIRKELLUS
220 PEN 0, RND(.) * 6 + 1                             : REM LUKRAKE PEN
230 R  = RND(.) * 36 + 1                              : REM STRAAL
240 XC = R + RND(.) * 320 : IF (XC + R) ) 319 THEN 240  : REM X-CENTRUM
250 YC = R + RND(.) * 200 : IF (YC + R) ) 199 THEN 250  : REM Y-CENTRUM
260 CIRCLE XC, YC, R, .                               : REM TEKEN
270 NEXT
280 GETKEY A$                                         : REM WACHT OP TOETS
290 SCREEN CLOSE : BORDER 6
```

Het voorbeeldprogramma gebruikt de willekeurige-getallenfunctie **RND** voor de kleur, grootte en positie van de cirkel. Het toont daarom bij elke uitvoering een ander beeld:



CLOSE

Token: $A0

Formaat: **CLOSE** kanaal

Gebruik: Sluit een invoer- of uitvoerkanaal.

kanaal nummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

Opmerkingen:

Het sluiten van bestanden die voor het einde van het programma zijn geopend, is zeer belangrijk, vooral voor uitvoerbestanden. **CLOSE** maakt uitvoerbuffers leeg en werkt de informatie van de map op schijven bij.

Het niet uitvoeren van **CLOSE** kan bestanden en schijven beschadigen.

BASIC sluit kanalen en bestanden *niet* automatisch wanneer een programma stopt.

Voorbeeld: Gebruik van **CLOSE**

```
10 OPEN 2, 8, 2, "TEST,S,W"
20 PRINT#2, "TEKENREEKSJE TER TEST"
30 CLOSE 2              : REM CLOSE VERZUIMEN BETEKENT EEN ZOMBIEBESTAND
```

CLR

Token: $9C

Formaat: **CLR**
CLR variabele

Gebruik: Wist het variabelengeheugen van BASIC.

Na het uitvoeren van **CLR** worden alle variabelen en matrices niet gedeclareerd. De aanwijzers van de run-time-stack en de tabel met geopende kanalen worden eveneens gereset.

Wanneer een **variabele** wordt opgegeven, wordt deze gewist (op nul gezet). **variabele** kan een numerieke variabele of een tekenreeksvariabele zijn, maar geen matrix.

Opmerkingen:

CLR mag niet worden gebruikt in lussen of subroutines, omdat dit het retouradres vernietigt.

Na **CLR** zijn alle variabelen onbekend en worden ze geïnitialiseerd bij het volgende gebruik.

RUN voert **CLR** automatisch uit.

Voorbeeld: Gebruik van **CLR**

```
10 A = 5 : P$ = "MEGA65"
20 CLR
30 PRINT A; P$

RUN
 0
```

CLRBIT

Token: $9C $FE $4E

Formaat: **CLRBIT** adres**,** bitnummer

Gebruik: Wist (zet terug) een enkele bit op het **adres**.

Wanneer het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank die met **BANK** is ingesteld, gebruikt.

Adressen groter dan of gelijk aan $10000 (65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

bitnummer waarde van 0 - 7.

Opmerkingen:

CLRBIT is een verkorte versie van het gebruik van een bitsgewijze **AND** om een bit te wissen, doch u kunt slechts één bit tegelijk wissen. Zie **SETBIT** om een bit in te stellen.

Voorbeeld: Gebruik van **CLRBIT**

```
10 BANK 128       : REM KIES SYSTEEMMAPPING
20 CLRBIT $D011, 4 : REM SCHAKEL SCHERM UIT
30 CLRBIT $D016, 3 : REM SCHAKEL NAAR 38- OF 76-KOLOMSMODE
```

CMD

Token: $9D

Formaat: **CMD** kanaal [**,** tekenreeks]

Gebruik: Verlegt de standaarduitvoer van het scherm naar een kanaal.

Hiermee kunt u programmalijsten en mappen naar andere uitvoerkanalen afdrukken. Het is ook mogelijk deze uitvoer om te leiden naar een bestand op schijf of een modem.

kanaal nummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

De optionele **tekenreeks** wordt naar het kanaal gestuurd voordat de verlegging begint en kan bijvoorbeeld worden gebruikt voor escapesequenties voor de instelling van een printer of modem.

Opmerkingen:

De **CMD**-modus wordt gestopt met **PRINT#** of door het kanaal te sluiten met **CLOSE**. Het wordt aanbevolen om **PRINT#** te gebruiken voordat u sluit, zodat de uitvoerbuffer gegarandeerd wordt geleegd.

Voorbeeld: Gebruik van **CMD**

```
OPEN 1, 4 : REM LIST OP EEN PRINTER

CMD 1

LIST

PRINT#1

CLOSE 1
```

COLLECT

Token: $F3

Formaat: **COLLECT** [**,D** station] [**,U** eenheid]

Gebruik: Herstelt de Block Availability Map (BAM) van een schijf, waarbij bestanden die niet goed zijn gesloten (splat-bestanden) worden verwijderd en ongebruikte blokken als vrij worden gemarkeerd.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Hoewel dit commando nuttig is om een schijf van splat-bestanden te reinigen, is het gevaarlijk voor schijven met opstartblokken of willekeurig toegankelijke bestanden. Deze blokken zijn niet gekoppeld aan standaard schijfbestanden en worden daarom als vrij gemarkeerd, waardoor ze door verdere schrijfbewerkingen kunnen worden overschreven.

Voorbeelden:

Gebruik van **COLLECT**

```
COLLECT

COLLECT U9

COLLECT D0, U9
```

COLLISION

Token: $FE $17

Formaat: **COLLISION** type [**,** regelnummer]

Gebruik: Schakelt een gebruikergestuurde interrupt-handler voor sprite-collisions in of uit.

Indien een handler is ingeschakeld, onderbreekt een sprite-collision van het opgegeven **type** het BASIC-programma en voert een **GOSUB** uit naar **regelnummer**. Deze handler moet de besturing teruggeven met **RETURN**.

type collisiontype voor deze interrupt-handler:

Type	Beschrijving
1	Sprite - Sprite Collision
2	Sprite - Data - Collision
3	Light Pen

regelnummer regelnummer van een subroutine die de sprite-collision afhandelt en eindigt met **RETURN**.

Een aanroep zonder de parameter regelnummer schakelt de handler uit.

Opmerkingen:

Het is mogelijk de interrupt-handler voor alle typen in te schakelen, maar er kan er slechts één tegelijk worden uitgevoerd. Een interrupt-handler kan niet worden onderbroken door een andere interrupt-handler.

Functies zoals **BUMP**, **LPEN** en **RSPPOS** kunnen worden gebruikt om de betrokken sprites en hun posities te evalueren.

COLLISION was niet voltooid in BASIC 10, maar is wel beschikbaar in BASIC 65.

Voorbeeld: Gebruik van **COLLISION**

```
10 COLLISION 1, 70                                : REM INSCHAKELEN
20 SPRITE 1, 1 : MOVSPR 1, 120,   0 : MOVSPR 1, 0#5
30 SPRITE 2, 1 : MOVSPR 2, 120, 100 : MOVSPR 2, 180#5
40 FOR I = 1 TO 50000 : NEXT
50 COLLISION 1                                    : REM UITSCHAKELEN
60 END
70 REM SPRITE <-> SPRITE INTERRUPTHANDLER
80 PRINT "BUMP RETURNS"; BUMP(1)
90 RETURN                                         : REM KEER TERUG UIT INTERRUPT
```

COLOR

Token: $E7

Formaat: **COLOR** kleur

Gebruik: Stelt de voorgrondtekstkleur in voor opvolgende **PRINT**-commando's.

kleur paletindexnummer in het bereik 0 - 31.

Zie bijlage 6 op bladzijde 327 voor de lijst met kleuren in het standaard systeempalet.

Opmerkingen:

Dit is een andere naam voor **FOREGROUND**.

Voorbeeld: Gebruik van **COLOR**

```
COLOR 2 : PRINT "DIT IS ROOD"

COLOR 3 : PRINT "DIT IS CYAAN"
```

CONCAT

Token: $FE $13

Formaat: **CONCAT** toevoegbestand [**,D** station] **TO** doelbestand [**,D** station] [**,U** eenheid]

Gebruik: Voegt de inhoud van het **toevoegbestand** samen met het **doelbestand**. Na uitvoering bevat het **doelbestand** de inhoud van beide bestanden, terwijl het **toevoegbestand** ongewijzigd blijft.

toevoegbestand is een aangehaalde tekenreeks, bijvoorbeeld: `"DATA"` of een tekenreeksexpressie tussen haakjes, bijvoorbeeld: `(FI$)`

doelbestand is een aangehaalde tekenreeks, bijvoorbeeld: `"SAFE"` of een tekenreeksexpressie tussen haakjes, bijvoorbeeld: `(FS$)`

Als het schijfstation twee stations heeft, is het mogelijk **CONCAT** toe te passen op bestanden die op verschillende schijven zijn opgeslagen. In dat geval moet het station# voor beide bestanden worden opgegeven. Dit is ook noodzakelijk indien beide bestanden op station#1 zijn opgeslagen.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

CONCAT wordt uitgevoerd door het DOS van het schijfstation.

Beide bestanden moeten bestaan en patronen (jokertekens) zijn niet toegestaan.

Alleen bestanden van het type **SEQ** kunnen worden samengevoegd.

Voorbeelden:

Gebruik van **CONCAT**

```
CONCAT "NIEUWE DATA" TO "ARCHIEF", U9

CONCAT "ADRES", D0 TO "ADRESBOEK", D1
```

CONT

Token: $9A

Formaat: **CONT**

Gebruik: Hervat de programma-uitvoering na een onderbreking of stop als gevolg van een **END**- of **STOP**-instructie, of na het indrukken van RUN STOP.

Dit is een nuttig hulpmiddel bij het foutzoeken. Het BASIC-programma kan worden gestopt, waarna variabelen kunnen worden bekeken en zelfs gewijzigd.

De **CONT**-instructie hervat de uitvoering.

Opmerkingen:

CONT kan niet worden gebruikt indien het programma is gestopt als gevolg van een fout. Daarnaast belemmert elke bewerking van het programma de voortzetting. Het stoppen en hervatten kan de schermuitvoer verstoren en kan ook de invoer/uitvoer beïnvloeden.

Voorbeeld: Gebruik van **CONT**

```
10 I = I + 1 : GOTO 10

RUN

BREAK IN 10
READY.
PRINT I
 947

CONT
```

COPY

Token: $F4

Formaat: **COPY** bron [**,D** station] [**,U** eenheid] **TO** [doel] [**,D** station] [**,U** eenheid]

Gebruik: Kopieert een bestand naar een ander bestand, of een of meer bestanden van de ene schijf naar de andere.

bron een aangehaalde tekenreeks, bijv. "DATA" of een tekenreeksexpressie tussen haakjes, bijv. (FI$)

doel een aangehaalde tekenreeks, bijv. "BACKUP" of een tekenreeksexpressie tussen haakjes, bijv. (F$$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Indien geen of één **eenheid**nummer is opgegeven, of indien de eenheidnummers voor en na het **TO**-token gelijk zijn, wordt **COPY** op het schijfstation zelf uitgevoerd en bevinden de bron- en doelbestanden zich op dezelfde schijf.

Indien de bron-eenheid (voor **TO**) verschilt van de doeleenheid (na **TO**), voert **COPY** een door de CPU gestuurde routine uit die de bronbestanden in een RAM-buffer leest en naar de doeleenheid schrijft. In dat geval kan de doelbestandsnaam niet worden gekozen; deze is gelijk aan de bronbestandsnaam. Met de uitgebreide eenheid-naar-eenheid kopieermodus kunnen enkele bestanden, bestanden volgens een patroon of alle bestanden van een schijf worden gekopieerd. Elke combinatie van eenheden is toegestaan: interne diskette, D81-schijfimages, IEC-diskettestations zoals de 1541, 1571, 1581, of CMD-diskettenstations en hardeschijfstations.

Opmerkingen:

De bestandstypen **PRG**, **SEQ** en **USR** kunnen worden gekopieerd. Indien bron en doel zich op dezelfde schijf bevinden, moet de doelbestandsnaam verschillen van de bronbestandsnaam.

COPY kan geen **DEL**-bestanden kopiëren; deze worden doorgaans gebruikt als titels of scheidingstekens in directory's. Deze voldoen niet aan de Commodore DOS-regels en kunnen niet worden benaderd met standaard **OPEN**-routines.

REL-bestanden kunnen niet van eenheid naar eenheid worden gekopieerd.

Voorbeelden:

Gebruik van **COPY**

```
COPY U8 TO U9           : REM KOPIEER ALLE BESTANDEN
COPY "CODES" TO "BACKUP" : REM KOPIEER ENKEL BESTAND
COPY "*.TXT", U8 TO U9   : REM KOPIEER VOLGENS PATROON
COPY "M*", U9 TO U11     : REM KOPIEER VOLGENS PATROON
```

COS

Token: $BE

Formaat: **COS**(expressie in radialen)
COSD(expressie in graden)

Retourneert:

De cosinus van een hoek.

De **COS**-functie verwacht een argument in radialen.

De **COSD**-variant verwacht een argument in graden.

Het resultaat ligt in het bereik (-1.0 tot +1.0).

Voorbeelden:

Gebruik van **COS**

```
PRINT COS(0.7)
 0.76484219

X = 60 : PRINT COS(X * π / 180)
 0.5

PRINT COSD(60)
 0.5
```

CURSOR

Token: $FE $41

Formaat: **CURSOR** <**ON** | **OFF**> [{**,** kolom**,** rij**,** stijl}]
CURSOR kolom**,** rij

Gebruik: Verplaatst de tekstcursor naar de opgegeven positie op het huidige tekstscherm.

ON of **OFF** toont of verbergt de cursor. Wanneer de cursor op **ON** staat, verschijnt deze op de cursorpositie tijdens **GETKEY**.

kolom en **rij** specificeren de nieuwe positie.

stijl stelt een vaste (1) of knipperende (0) cursor in.

Opmerkingen:

Wanneer de cursor "aan" staat, kan het verplaatsen van de cursor met **PRINT** weergaveartefacten veroorzaken waarbij het cursor-aan-teken zichtbaar blijft wanneer een PETSCII-code de cursor verplaatst. In het gebruikelijke geval waarbij **PRINT** wordt gebruikt om door de gebruiker getypte tekens weer te geven, is de aanbevolen techniek om de cursor uit te schakelen voordat er wordt afgedrukt:

```
10 CURSOR ON
20 GETKEY A$
30 CURSOR OFF
40 PRINT A$
50 GOTO 10
```

Voorbeeld: Gebruik van **CURSOR**

```
 10 SCNCLR
 20 CURSOR 1, 2
 30 PRINT "A"; : SLEEP 1
 40 PRINT "B"; : SLEEP 1
 50 PRINT "C"; : SLEEP 1
 60 CURSOR 20, 10
 70 PRINT "D"; : SLEEP 1
 80 CURSOR ,5          : REM ZET DE CURSOR OP RIJ 5 ZONDER VAN KOLOM TE VERANDEREN
 90 PRINT "E"; : SLEEP 1
100 CURSOR 0           : REM ZET DE CURSOR AAN HET BEGIN VAN DE RIJ
110 PRINT "F"; : SLEEP 1
```

CUT

Token: $E4

Formaat: **CUT** x, y, breedte, hoogte

Gebruik: Bitmapgrafiek: kopieert de inhoud van het opgegeven rechthoek met linkerbovenpositie **x, y** en de **breedte** en **hoogte** naar een buffer en vult het gebied vervolgens met de kleur van de geselecteerde pen.

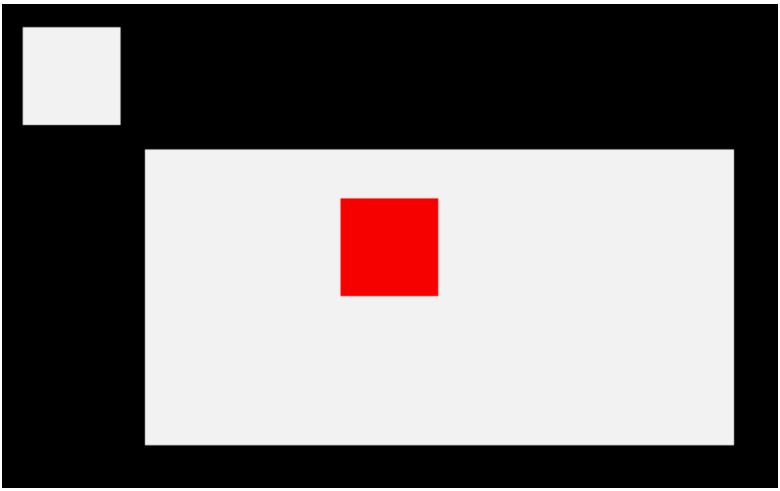
Het uitgesneden gebied kan op elke positie worden ingevoegd met het commando **PASTE**.

Opmerkingen:

De grootte van de rechthoek is beperkt tot de omvang van 1 KB van de buffer. Het geheugengebruik voor een uitgesneden gebied is breedte * hoogte * aantal bitplanes / 8. Dit mag niet 1024 bytes of meer bedragen. Voor een scherm met 4 bitplanes heeft een gebied van bijv. 45 x 45 1012.5 bytes nodig.

Voorbeeld: Gebruik van **CUT**

```
10 SCREEN 320, 200, 2
20 BOX 60, 60, 300, 180, 1 : REM TEKEN EEN WIT VAK
30 PEN 2                   : REM KIES RODE PEN
40 CUT 140, 80, 40, 40     : REM SNIJ EEN OPPERVLAK VAN 40 * 40 UIT
50 PASTE 10, 10            : REM PLAK HET OP NIEUWE POSITIE TERUG
60 GETKEY A$               : REM WACHT OP TOETSDRUK
70 SCREEN CLOSE
```



DATA

Token: $83

Formaat: **DATA** [constante [**,** constante ...]]

Gebruik: Definieert constanten die door **READ**-instructies in een programma kunnen worden gelezen.

Getallen en tekenreeksen zijn toegestaan, maar expressies niet. Items worden gescheiden door komma's. Tekenreeksen die komma's, dubbele punten of spaties bevatten moeten tussen aanhalingstekens worden geplaatst.

RUN initialiseert de gegevenspointer op het eerste item van de eerste **DATA**-instructie en schuift deze voor elk gelezen item op. Het is de verantwoordelijkheid van de programmeur dat het type van de constante en de variabele in de **READ**-instructie overeenkomen.

RESTORE kan worden gebruikt om de gegevenspointer op een specifieke regel te zetten voor volgende leesacties.

Opmerkingen:

Het is een goede programmeerpraktijk om grote hoeveelheden **DATA**-instructies aan het einde van het programma te plaatsen, zodat ze de zoekactie naar regelnummers na **GOTO** en andere instructies met regelnummerdoelen niet vertragen.

Lege items zonder constante tussen komma's zijn toegestaan en worden geretourneerd als nul voor numerieke constanten en als lege tekenreeks voor tekenreeksexpressies.

Voorbeeld: Gebruik van **DATA**

```
10 READ NA$, VE
20 READ N% : FOR I=2 TO N% : READ GL(I) : NEXT I
30 PRINT "PROGRAMMA: "; NA$; "  VERSIE:"; VE
40 PRINT "N-PUNTS GAUSS-LEGENDRE-FACTOREN E1:"
50 FOR I = 2 TO N% : PRINT I; GL(I) : NEXT I
60 END
80 DATA "MEGA65", 1.1
90 DATA 5, 0.5120, 0.3573, 0.2760, 0.2252

RUN
PROGRAMMA: MEGA65  VERSIE: 1.1
N-PUNTS GAUSS-LEGENDRE-FACTOREN E1:
 2  0.512
 3  0.3573
 4  0.276
 5  0.2252
```

DCLEAR

Token: $FE $15

Formaat: **DCLEAR** [**,D** station] [**,U** eenheid]

Gebruik: Verstuurt een initialiseercommando naar het opgegeven eenheid en het station.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Het DOS van het schijfstation sluit alle geopende bestanden, wist alle kanalen, maakt buffers vrij en herschrijft de BAM. Alle geopende kanalen op de computer worden ook gesloten.

Voorbeelden:

Gebruik van **DCLEAR**

```
DCLEAR

DCLEAR U9

DCLEAR D0, U9
```

DCLOSE

Token: $FE $0F

Formaat: **DCLOSE** [**U** eenheid]
DCLOSE# kanaal

Gebruik: Sluit een enkel bestand of alle bestanden voor het opgegeven station.

kanaal kanaalnummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

DCLOSE wordt gebruikt met ofwel een kanaalparameter ofwel een eenheidnummer, maar nooit met beide.

Opmerkingen:

Het is van belang alle geopende bestanden te sluiten voordat het programma eindigt. Anders worden buffers niet vrijgemaakt en kunnen open bestanden die zijn geschreven onvolledig zijn (ook wel "splat-bestanden" genoemd) en niet meer bruikbaar zijn.

Voorbeelden:

Gebruik van **DCLOSE**

```
DCLOSE#2  : REM SLUIT BESTAND GEASSOCIEED AAN KANAAL 2

DCLOSE U9 : REM SLUIT ALLEN OPEN BESTANDEN OP UNIT 9
```

DEC

Token: $D1

Formaat: **DEC**(tekenreeksexpressie)

Retourneert:

De decimale waarde van een hexadecimale tekenreeks.

Het bereik van het argument is “0” tot “FFFFFFFF”.

Opmerkingen:

Toegestane cijfers in de hoofdletter-/graphicsmodus zijn 0 – 9 en A – F (`0123456789ABCDEF`) en in de kleine-letter-/hoofdlettermodus 0 – 9 en a – f (`0123456789abcdef`).

DEC negeert alles na en inclusief het eerste niet-toegestane cijfer of het 8e teken. Een tekenreeks die alleen niet-toegestane cijfers bevat geeft 0 terug.

Om een getal naar een hexadecimale tekenreeks om te zetten, zie **HEX$**.

Om een letterlijke waarde als hexadecimaal getal op te geven, gebruikt u **$** gevolgd door de cijfers: $A12F

Voorbeelden:

Gebruik van **DEC**

```
PRINT DEC("D000")
53248

POKE DEC("600"), 255
```

DECBIN

Token: $CE $11

Formaat: **DECBIN**(tekenreeksexpressie)

Retourneert:

De decimale waarde van een binaire tekenreeks.

Het bereik van het argument is “0” tot “11111111 ... 11111111” (32 enen).

Opmerkingen:

DECBIN negeert alles na het eerste niet-binaire cijfer of het 32e teken. Een tekenreeks die alleen niet-toegestane cijfers bevat geeft 0 terug.

Om een getal naar een binaire tekenreeks om te zetten, zie **STRBIN$**.

Om een letterlijke waarde als binair getal op te geven, gebruikt u **%** gevolgd door de cijfers: %10011101

Voorbeeld: Gebruik van **DECBIN**

```
PRINT DECBIN("1101000000000000")
53248
```

DEF FN

Token: $96 (DEF) $A5 (FN)

Formaat: **DEF FN** naam(reëel-getal-variabele) = [expressie]

Gebruik: Definieert een gebruikersfunctie van één instructie met een argument van het type reëel getal, welke een reëel-getal-waarde retourneert wanneer deze wordt geëvalueerd.

De definitie moet worden uitgevoerd voordat de functie in expressies kan worden gebruikt. Het argument is een dummy-variabele, welke wordt vervangen door het argument wanneer de functie wordt aangeroepen.

Opmerkingen:

Het functie-argument is geen reëel-getal-variabele en overschrijft geen variabele met die naam. Het vertegenwoordigt uitsluitend de argumentwaarde binnen de functiedefinitie.

Merk op dat **DEF FN** in dit geval eigenlijk bestaat uit twee aparte tokens: **DEF** en **FN**.

Voorbeeld: Gebruik van **DEF FN**

```
10 PD = π / 180
20 DEF FN CD(X) = COS(X * PD) : REM COS VOOR GRADEN
30 DEF FN SD(X) = SIN(X * PD) : REM SIN VOOR GRADEN
40 FOR D = 0 TO 360 STEP 90
50 PRINT USING "####"; D;
60 PRINT USING " ##.##"; FNCD(D);
70 PRINT USING " ##.##"; FNSD(D)
80 NEXT D

RUN
  0  1.00  0.00
 90  0.00  1.00
180 -1.00  0.00
270  0.00 -1.00
360  1.00  0.00
```

DELETE

Token: $F7

Formaat: **DELETE** [regelbereik]
DELETE bestandsnaam [**,D** station] [**,U** eenheid] [**,R**]

Gebruik: De eerste vorm verwijdert een bereik van regels uit het BASIC-programma. De tweede vorm verwijdert een of meer bestanden van een schijf.

regel bereik bestaat uit de eerste en laatste te verwijderen regel, of een enkel regelnummer. Indien het eerste nummer wordt weggelaten, wordt de eerste BASIC-regel aangenomen. Het tweede nummer in de bereikaanduiding standaard op de laatste BASIC-regel.

bestandsnaam is een tekenreeks tussen aanhalingstekens, bijvoorbeeld: "SAFE" of een tekenreeksexpressie tussen haakjes, bijvoorbeeld: (F$$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

De **R**-vlag wordt gebruikt om een eerder verwijderd bestand te herstellen. Dit werkt alleen indien er geen schrijfbewerkingen tussen verwijdering en herstel hebben plaatsgevonden, welke de inhoud van het bestand hadden kunnen wijzigen.

Opmerkingen:
DELETE is een synoniem van **SCRATCH** en **ERASE**.

Voorbeelden:
Gebruik van **DELETE**

```
DELETE 100       : REM VERWIJDER REGEL 100

DELETE 240-350   : REM VERWIJDER REGELS VAN 240 TOT 350

DELETE 500-      : REM VERWIJDER VAN 500 TOT EIND

DELETE -70       : REM VERWIJDER VAN START TOT 70

DELETE "DRM", U9 : REM VERWIJDER BESTAND DRM OP UNIT 9

DELETE "*=SEQ"   : REM VERWIJDER ALLE SEQUENTIELE BESTANDEN

DELETE "R*=PRG"  : REM VERWIJDER PROGRAMMABESTANDEN BEGINNEND MET 'R'
```

DIM

Token: $86

Formaat: **DIM** naam(limieten) [**,** naam(limieten) ...]

Gebruik: Geeft de vorm, grenzen en het type van een BASIC-array aan.

Als declaratie-instructie moet deze slechts eenmaal worden uitgevoerd en vóór elk gebruik van de gedeclareerde arrays. Een array kan één of meer dimensies hebben. Eendimensionale arrays worden vaak vectoren genoemd, terwijl twee of meer dimensies een matrix definiëren. De onderste grens van een dimensie is altijd nul, terwijl de bovengrens bepaald is zoals gedefinieerd. De regels voor variabelenamen gelden ook voor arraynamen. U kunt byte-arrays, integer-arrays, reëel-getal-arrays en string-arrays aanmaken. Het is toegestaan om dezelfde identifier te gebruiken voor scalarvariabelen en arrayvariabelen. Het linker haakje na de naam onderscheidt arraynamen.

Opmerkingen:

Byte-arrays verbruiken één byte per element, integer-arrays twee bytes, reëel-getal-arrays vijf bytes en string-arrays drie bytes voor de stringdescriptor plus de lengte van de string zelf.

Indien een array-identifier wordt gebruikt zonder voorafgaande declaratie, wordt impliciet een eendimensionale array met een limiet van 10 gedeclareerd.

Voorbeeld: Gebruik van **DIM**

```
10 DIM A%(8)     : REM ARRAY VAN 9 ELEMENTEN
20 DIM XX(2, 3)  : REM ARRAY VAN 3X4 = 12 ELEMENTEN
30 FOR I = 0 TO 8 : A%(I) = PEEK(256 + I) : PRINT A%(I); : NEXT : PRINT
40 FOR I = 0 TO 2 : FOR J = 0 TO 3 : READ XX(I, J) : PRINT XX(I, J); : NEXT J, I
50 END
60 DATA 1, -2, 3, -4, 5, -6, 7, -8, 9, -10, 11, -12

RUN
 45  52  50  0  0  0  0  0  0
 1 -2  3 -4  5 -6  7 -8  9 -10  11 -12
```

DIR

Token: $EE (DIR) $FE $29 (ECTORY)

Formaat: **DIR** [bestandspatroon] [**,W**] [**,P**] [**,R**] [**,D** station] [**,U** eenheid]
DIRECTORY [bestandspatroon] [**,W**] [**,P**] [**,R**] [**,D** station] [**,U** eenheid]
$ [bestandspatroon] [**,W**] [**,R**] [**,D** station] [**,U** eenheid]
DIR [bestandspatroon] **,U12** [**,P**]

Gebruik: Geeft de bestandsmap/catalogus van de opgegeven schijf weer.

De parameter **W** (Breed) geeft de directory in drie kolommen op het scherm weer en stopt nadat het scherm is gevuld met een pagina (63 directory-items). Door op een willekeurige toets te drukken, wordt de volgende pagina weergegeven.

De parameter **P** (Paginering) geeft de map in één kolom weer en onderbreekt na elke volle schermpagina. Druk op de Q-toets om de lijstopgave op de huidige pagina te onderbreken. Druk op een andere toets om de volgende pagina weer te geven.

De parameter **R** (Herstelbaar) toont alleen de bestanden die als verwijderd zijn gemarkeerd maar nog steeds herstelbaar zijn, elk gemarkeerd met een '-'. Andere bestanden worden niet getoond.

bestandspatroon is een tekenreeks tussen aanhalingstekens ("D*") of een tekenreeksexpressie tussen haakjes: (D$)

Het argument **U12** geeft de inhoud van de SD-kaart weer. Dit kan worden gecombineerd met het argument **P** voor een gepagineerde weergave. Het ondersteunt een bestandspatroon. Andere argumenten worden niet ondersteund.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

DIR is een synoniem van **CATALOG** en **DIRECTORY**, en produceert dezelfde lijstopgave.

De **bestandspatroon** kan worden gebruikt om de lijstopgave te filteren. Binnen het patroon:

- * komt overeen met een willekeurig aantal tekens van elk type.
- # komt overeen met één cijfer.
- $ komt overeen met één letter.

- ? komt overeen met één teken van elk type.

Door **,T=** toe te voegen aan de patroontekenreeks, waarbij **T** een bestandstype opgeeft van **P**, **S**, **U**, **R** of **C** (voor **P**RG, **S**EQ, **U**SR, **R**EL, **C**BM), wordt de uitvoer gefilterd op dat bestandstype.

Het afkortingssymbool **$** kan alleen in de directe modus worden gebruikt.

Voorbeeld: Gebruik van **DIR**

```
DIR

DIR "MEG*",U12
DIR "ME????.D81",U12
DIR "ME$$65.D81",U12
DIR "MEGA##.D81",U12
DIR "M*.D81",U12
0 "BLACK SMURF     " BS 2A
508  "STORY PHOBOS"    SEQ
27   "C8096"           PRG
25   "C128"            PRG
104 BLOCKS FREE.
```

DISK

Token: $FE $40

Formaat: **DISK** commando [**,U** eenheid]
@ commando [**,U** eenheid]

Gebruik: Verstuurt een commandotekenreeks naar het opgegeven schijfstation.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

commando een tekenreeksexpressie.

Opmerkingen:

De commandotekenreeks wordt door het schijfstation geïnterpreteerd en moet compatibel zijn met de gebruikte DOS-versie. Raadpleeg de handleiding van het schijfstation voor mogelijke commando's.

Het gebruik van **DISK** zonder commandotekenreeks geeft de schijfstatus weer.

De afkorting **@** kan alleen in de directe modus worden gebruikt.

Voorbeelden:

Gebruik van **DISK**

```
DISK "I0"   : REM INITIALISEER DISK IN DRIVE 0

DISK "U0>9" : REM ZET DEVICENUMMER OP 9
```

DLOAD

Token: $F0

Formaat: **DLOAD** bestandsnaam [**,D** station] [**,U** eenheid]
DLOAD "$[patroon=type]" [**,D** station] [**,U** eenheid]
DLOAD "$$[patroon=type]" [**,D** station] [**,U** eenheid]

Gebruik: De eerste vorm laadt een bestand van het type PRG in het geheugen dat gereserveerd is voor BASIC-programma's.

De tweede vorm laadt een map in het geheugen, welke vervolgens met **LIST** kan worden bekeken. De structuur lijkt op een BASIC-programma, doch in plaats van regelnummers worden bestandsgroottes weergegeven.

De derde vorm lijkt op de tweede, maar de bestanden krijgen opeenvolgende regelnummers. Deze lijstopgave kan als een BASIC-programma worden gescrold met de toetsen **F9** of **F11**, bewerkt, weergegeven, opgeslagen of afgedrukt.

Een filter kan worden toegepast door een patroon of een patroon met een type op te geven. Het asterisk * komt overeen met de rest van de naam, terwijl het ? met elk willekeurig teken overeenkomt. De typespecificatie kan een van de tekens (P, S, U, R, C) zijn, respectievelijk programma-, sequentieel, gebruiker-, relatief- of CBM-bestand.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:
Het laadadres dat in de eerste twee bytes van het PRG-bestand is opgeslagen, wordt genegeerd. Het programma wordt altijd in BASIC-geheugen geladen. Hierdoor kunnen BASIC-programma's worden geladen die op andere computers met andere geheugenconfiguraties zijn opgeslagen. Na het laden wordt het programma opnieuw gekoppeld en is het klaar om te worden uitgevoerd (**RUN**) of bewerkt.

Het is mogelijk om **DLOAD** in een lopend programma te gebruiken. Dit wordt "overlayen" of "ketenen" genoemd. Indien u dit doet, vervangt het nieuw geladen programma het huidige en start de uitvoering automatisch op de eerste regel van het nieuwe programma. Variabelen, arrays

en strings van de huidige uitvoering worden behouden en kunnen ook door het nieuw geladen programma worden gebruikt.

Elke **DLOAD**, van een programma of een directorylijst, vervangt een programma dat zich momenteel in het geheugen bevindt.

Voorbeelden:

Gebruik van **DLOAD**

```
DLOAD "APOCALYPSE"

DLOAD "MEGA TOOLS", U9

DLOAD (FI$), U(UN%)

DLOAD "$"      : REM LAAD HELE DIRECTORY - MET BESTANDSGROOTTES

DLOAD "$$"     : REM LAAD HELE DIRECTORY - SCROLLBAAR

DLOAD "$$X*=P" : REM DIRECTORY MET PRG-BESTANDEN BEGINNEND MET 'X'
```

DMA

Token: $FE $1F

Formaat: **DMA** commando [**,** lengte**,** bronadres**,** bronbank**,** doeladres**,** doelbank[**,** sub]]

Gebruik: **DMA** ("Direct Memory Access") is verouderd en is vervangen door **EDMA**.

Het **commando** bestuurt de functie met behulp van de twee onderste bits: 0: kopiëren, 1: mengen, 2: verwisselen, 3: vullen.

lengte aantal bytes (in het bereik 0 tot 65535). OPMERKING: Het opgeven van lengte 0 wordt geïnterpreteerd als een lengte van 65536 (exact 64KB).

bronadres (16-bits) van het leesgebied of het vulbyte.

bronbank nummer van de bron (genegeerd in de vulmodus).

doeladres van het schrijfgebied (16-bits).

doelbank nummer van het doel.

sub het te gebruiken subcommando.

Opmerkingen:

DMA heeft toegang tot het onderste 1MB-adresbereik, georganiseerd in 16 banken van 64KB. Om deze beperking te omzeilen, gebruikt u **EDMA**, welke toegang heeft tot het volledige 256MB-adresbereik.

OPMERKING: Alleen kopiëren en vullen zijn geïmplementeerd in de MEGA65-DMAcontroller ten tijde van het schrijven. Andere DMAgic-commandobits kunnen ook worden gezet, bijvoorbeeld om het kopiëren van gegevens in de omgekeerde richting toe te laten of het bron- of doeladres vast te houden.

Voorbeelden:

Gebruik van **DMA**

```
DMA 0, 80*25, 2048, 0, 0, 4  : REM BEWAAR SCHERM OP $0000 IN BANK 4

DMA 3, 80*25, 32, 0, 2048, 0 : REM MAAK SCHERM LEEG

DMA 0, 80*25, 0, 4, 2048, 0  : REM HERSTEL SCHERM VANAF $0000 IN BANK 4
```

DMODE

Token: $FE $35

Formaat: **DMODE** jam**,** complement**,** sjabloon**,** stijl**,** dikte

Gebruik: Bitmapgraphics: stelt de parameters van de "weergavemodus" van de grafische context in, welke door tekencommando's worden gebruikt.

Indien **jam** 1 is, worden kleurwaarden gecombineerd wanneer een pixel op dezelfde locatie als een andere pixel wordt getekend. De standaardwaarde is 0: de pixel overschrijven.

Indien **complement** 1 is, wordt een nieuwe kleur geproduceerd gebaseerd op de XOR van de bitplaten wanneer een pixel op dezelfde locatie als een andere pixel wordt getekend. De standaardwaarde is 0: de pixel overschrijven.

Indien **stijl** 1 is, worden bij het tekenen het sjabloon van **DPAT** gebruikt voor lijnen en vulling. De standaardwaarde is 0: ononderbroken lijnen en vulling.

Opmerkingen:

De parameters **sjabloon** en **dikte** zijn niet geïmplementeerd en hebben geen effect. Deze argumenten zijn behouden uit het ontwerpdocument van de C65 voor mogelijke toekomstige implementatie. Eveneens suggereert de C65-documentatie een derde **stijl**-waarde welke eveneens niet is geïmplementeerd.

De C65-documentatie suggereert dat **jam**-modus een aparte penkleur gebruikt bij het tekenen over de achtergrond, ingesteld met **PEN 1,C**. Dit lijkt niet te zijn geïmplementeerd.

Zie ook **DPAT**.

Voorbeeld: Gebruik van **DMODE**

```
 10 SCREEN 320, 200, 5
 20 PEN 2
 30 BOX 50, 50, 100, 100, 1
 40 PEN 6
 50 DMODE 1, 0, 0, 0, 0
 60 BOX 40, 80, 170, 84, 1
 70 DMODE 0, 1, 0, 0, 0
 80 BOX 40, 90, 170, 94, 1
 90 GETKEY A$
100 SCREEN CLOSE
```

DO

Token: $EB

Formaat: **DO** ... **LOOP**
DO [<**UNTIL** | **WHILE**> logische expressie] ... opdrachten [**EXIT**]
LOOP [<**UNTIL** | **WHILE**> logische expressie]

Gebruik: **DO** definieert het begin van een BASIC-lus.

Het gebruik van **DO** ... **LOOP** zonder modificatoren creëert een oneindige lus, welke alleen met de instructie **EXIT** kan worden afgesloten. De lus kan worden beheerd door **UNTIL** of **WHILE** toe te voegen na het **DO** of het **LOOP**.

Opmerkingen:
DO-lussen mogen genest worden. Een instructie **EXIT** sluit alleen de huidige lus af.

Voorbeelden:
Gebruik van **DO**

```
10 PW$ = "" : DO
20 GET A$ : PW$ = PW$ + A$
30 LOOP UNTIL LEN(PW$) > 7 OR A$ = CHR$(13)

10 DO                          : REM WACHT OP GEBRUIKERSBESLISSING
20 GET A$
30 LOOP UNTIL A$ = "Y" OR A$ = "N"

10 DO WHILE ABS(EPS) > 0.001
20 GOSUB 2000                  : REM ITERATIESUBROUTINE
30 LOOP

10 I% = 0                      : REM INTEGERLUS 1-100
20 DO : I% = I %+ 1
30 LOOP WHILE I% < 101
```

DOPEN

Token: $FE $0D

Formaat: **DOPEN#** kanaal**,** bestandsnaam>**,L** [rec-lengte]] [**,W**] [**,D** station] [**,U** eenheid]

Gebruik: Opent een bestand voor lezen of schrijven.

kanaal getal, waarbij:

- 1 <= kanaal <= 127: regelafsluiting is CR.
- 128 <= kanaal <= 255: regelafsluiting is CR LF.

L geeft aan dat het bestand een relatief bestand is, welke wordt geopend voor lezen/schrijven en willekeurige toegang.

rec-lengte is de recordlengte en is verplicht voor het aanmaken van relatieve bestanden. Bij bestaande relatieve bestanden wordt **rec-lengte** als veiligheidscontrole gebruikt indien opgegeven.

W opent een bestand voor schrijftoegang. Het bestand mag niet bestaan.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

DOPEN# kan worden gebruikt om alle bestandstypen te openen. Het sequentiële bestandstype **SEQ** is standaard. Het relatieve bestandstype **REL** wordt gekozen met de parameter **L**. Andere bestandstypen moeten in de bestandsnaam worden opgegeven, bijv. door ",P" toe te voegen aan de bestandsnaam voor **PRG**-bestanden of ",U" voor **USR**-bestanden.

Indien het eerste teken van de bestandsnaam een "@" (apestaartje) is, wordt dit geïnterpreteerd als een "opslaan en vervangen"-operatie. Het gebruik van deze optie op 1541- en 1571-stations wordt niet aanbevolen, omdat deze een "opslaan en vervangen-fout" in hun DOS bevatten.

Voorbeelden:

Gebruik van **DOPEN**

```
DOPEN#5, "DATA", U9

DOPEN#130, (DD$), U(UN%)

DOPEN#3, "USER FILE,U"

DOPEN#2, "DATA BASE", L240

DOPEN#4, "MYPROG,P"
```

DOT

Token: $FE $4C

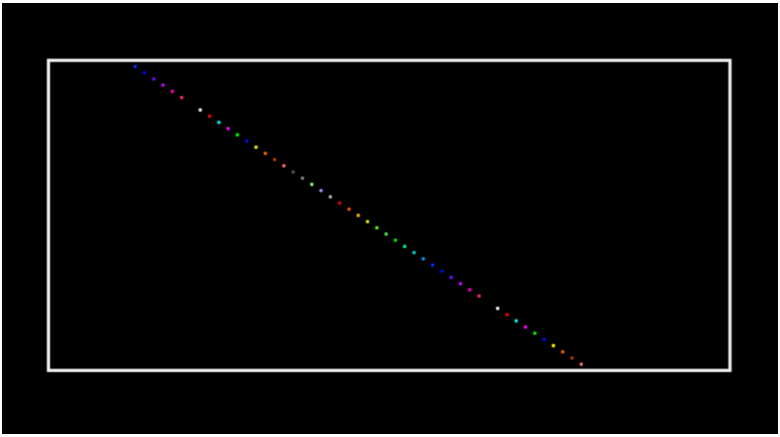
Formaat: **DOT** x, y [,kleur]

Gebruik: Bitmapgraphics: tekent een pixel op schermcoördinaten (x, y).

De optionele derde parameter bepaalt de te gebruiken kleur. Indien niet opgegeven, wordt de huidige penkleur gebruikt.

Voorbeeld: Gebruik van **DOT**

```
10 SCREEN 320, 200, 5
20 BOX 50, 50, 270, 150
30 VIEWPORT 50, 50, 220, 100
40 FOR I = 0 TO 127
50 DOT I + I + I, I + I, I
60 NEXT
70 GETKEY A
80 SCREEN CLOSE
```



DPAT

Token: $FE $36

Formaat: **DPAT** type [**,** getal**,** patroon ...]

Gebruik: Bitmapgraphics: stelt het tekenpatroon van de grafische context in voor tekencommando's.

Er zijn vier voorgedefinieerde patronen die geselecteerd kunnen worden door het typenummer (1, 2, 3 of 4) als enkele parameter op te geven.

- 1: Dunne stip.
- 2: dikke stip.
- 3: onderbroken.
- 4: stip-streep.

Een typenummer van nul geeft een door de gebruiker gedefinieerd patroon aan. Dit patroon kan ingesteld worden met een bitreeks van 8, 16, 24 of 32 bits. Het aantal gebruikte patroonbytes wordt als tweede parameter opgegeven. Dit bepaalt hoeveel patroonbytes (1, 2, 3 of 4) volgen.

- Type: 0 - 4.
- Aantal: aantal volgende patroonbytes (1 - 4).
- Patroon: patroonbytes.

Opmerkingen:

Om het tekenpatroon te gebruiken, stelt u met **DMODE** de **stijl**-modus in op 1.

Het effect van een tekenpatroon is afhankelijk van het commando dat tekent. **LINE** en de omtrekken van **BOX** en **POLYGON** tekenen het patroon in de richting van de lijn. Een gevulde rechthoek getekend met **BOX** geeft het patroon horizontaal weer in de vulling. Andere gevulde figuren, zoals **CIRCLE**, **ELLIPSE** en niet-rechthoekige **BOX**, kunnen het patroon op onvoorspelbare wijze toepassen op basis van hun vullingsalgoritme.

Zie ook **DMODE**.

Voorbeeld: Gebruik van **DPAT**

```
10 SCREEN 320, 200, 5
20 DMODE 0, 0, 0, 1, 0
30 PEN 2
40 DPAT 1
50 BOX 100, 100, 149, 149, 1
60 PEN 3
70 DPAT 0, 4, %00110011, %00111100, %00001111, %11110000
80 BOX 150, 150, 199, 199, 1
90 GETKEY A$
100 SCREEN CLOSE
```

DS

Token: N.v.t.

Formaat: **DS**

Gebruik: De status van de laatste schijfbewerking.

Dit is een vluchtige variabele. Elk gebruik activeert het uitlezen van de schijfstatus van het momenteel actieve schijfstation.

DS is gekoppeld aan de tekenreeksvariabele **DS$**, welke tegelijkertijd wordt bijgewerkt.

Het uitlezen van de schijfstatus van een schijfstation wist automatisch elke foutstatus op dat station. De variabelen **DS** en **DS$** behouden hun waarden tot de volgende schijfbewerking.

Opmerkingen:

DS is een gereserveerde systeemvariabele.

Voorbeeld: Gebruik van **DS**

```
100 DOPEN#1, "DATA"
110 IF DS <> 0 THEN PRINT "COULD NOT OPEN FILE DATA" : STOP
```

DS$

Token: N.v.t.

Formaat: **DS$**

Gebruik: De status van de laatste schijfbewerking in de tekstvorm "<code>,<bericht>,<spoor>,<sector>".

DS$ is gekoppeld aan de numerieke variabele **DS**. DS$ wordt ingesteld op `00,OK,00,00` indien er geen fout is opgetreden; anders wordt deze ingesteld op een DOS-foutmelding (vermeld in de handleidingen van de schijfstations).

Opmerkingen:

DS$ is een gereserveerde systeemvariabele.

De spoorwaarde wordt ook gebruikt voor het rapporteren van hoeveelheden, zoals het aantal verwijderde bestanden na een **DELETE**.

Voorbeeld: Gebruik van **DS$**

```
100 DOPEN#1, "DATA"
110 IF DS <> 0 THEN PRINT DS$ : STOP
```

DSAVE

Token: $EF

Formaat: **DSAVE** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Slaat het BASIC-programma in het geheugen op als een bestand van het type **PRG**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

De maximale lengte van de bestandsnaam is 16 tekens.

DSAVE geeft een fout indien er al een bestand met de opgegeven naam bestaat. Om een bestaand bestand te vervangen, plaatst u een apestaartje (@) voor de bestandsnaam, tussen de aanhalingstekens.

OPMERKING: Opslaan met vervanging wordt niet aanbevolen op 1541- en 1571-stations, vanwege een bug in oudere versies van het besturingssysteem van het schijfstation welke gegevensverlies kan veroorzaken.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

DVERIFY kan na **DSAVE** gebruikt worden om te controleren of het opgeslagen programma op de schijf identiek is aan het programma in het geheugen.

Voorbeelden:

Gebruik van **DSAVE**

```
DSAVE "AVONTUUR"

DSAVE "CMYPROG"

DSAVE "ZORK-I", U9

DSAVE "KERKERGEWELF", D1, U10
```

DT$

Token: N.v.t.

Formaat: **DT$**

Gebruik: De huidige datum, als tekenreeks.

De datumwaarde wordt bijgewerkt vanuit de RTC (Real-Time Clock). De tekenreeks **DT$** heeft het formaat `DD-MMM-JJJJ`, bijvoorbeeld `04-APR-2021`.

Opmerkingen:

DT$ is een gereserveerde systeemvariabele. Voor meer informatie over het instellen van de Real-Time Clock, raadpleeg *het grote Mega65-boek*, Configuratiehulpprogramma (paragraaf 5).

Voorbeeld: Gebruik van **DT$**

```
100 PRINT "VANDAAG IS HET: "; DT$
```

DVERIFY

Token: $FE $14

Formaat: **DVERIFY** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Controleert of het BASIC-programma in het geheugen gelijk is aan een bestand van het type **PRG**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

DVERIFY kan alleen op gelijkheid testen. Het geeft geen informatie over het aantal of de positie van bytes met verschillende waarden. **DVERIFY** sluit af met het bericht OK of met een Verify-fout. Indien een Verify-fout optreedt, is het weergegeven adres altijd $2001.

Voorbeelden:

Gebruik van **DVERIFY**

```
DVERIFY "AVONTUUR"

DVERIFY "ZORK-I", U9

DVERIFY "KERKERGEWELF", D1, U10
```

EDIT

Token: $FE $45 (EDIT) $91 (ON) $FE $24 (OFF)

Formaat: **EDIT** <**ON** | **OFF**>

Gebruik: Schakelt de tekstbewerkingsmodus van de schermeditor in of uit.

EDIT ON schakelt de tekstbewerkingsmodus in. In deze modus kunt u bestanden van het type SEQ als tekstbestanden aanmaken, bewerken, opslaan en laden met dezelfde regeleditor die u gebruikt om BASIC-programma's te schrijven. In deze modus geldt het volgende:

- De prompt wordt weergegeven als OK in plaats van READY.
- De editor voert geen tokenisering/parsing uit. Alle tekst die na een regelnummer wordt ingevoerd, blijft zuivere tekst; BASIC-trefwoorden zoals **FOR** en **GOTO** worden niet omgezet in BASIC-tokens, zoals in de programmode het geval is.
- De regelnummers worden uitsluitend gebruikt voor tekstorganisatie, sortering, verwijdering, weergave etc.
- Wanneer de tekst met **DSAVE** naar een bestand wordt opgeslagen, wordt een sequentieel bestand (type **SEQ**) geschreven, geen programma- (**PRG**-) bestand. Regelnummers worden *niet* naar het bestand geschreven.
- **DLOAD** in de tekstmodus kan uitsluitend sequentiële bestanden laden. Regelnummers worden automatisch gegenereerd voor bewerkingsdoeleinden.
- De tekstmodus is alleen van toepassing op regels met een regelnummer. Regels zonder regelnummer worden, zoals gebruikelijk, als BASIC-commando's uitgevoerd.

EDIT OFF schakelt de tekstbewerkingsmodus uit en keert terug naar de BASIC-programmabewerkingsmodus. De MEGA65 start in de BASIC-programmabewerkingsmodus.

Sequentiële bestanden die met de tekseditor zijn aangemaakt, kunnen op het scherm weergegeven worden (zonder ze te laden) met behulp van **TYPE**.

Voorbeelden:

Gebruik van **EDIT**

```
ready.
edit on

ok.
100 Dit is een eenvoudige teksteditor.
dsave "voorbeeld"

ok.
new

ok.
catalog

0 "demoleeg        " 00 3d
1 "voorbeeld         seq
3159 blocks free

ok.
type "voorbeeld"
Dit is een eenvoudige teksteditor.

ok.
dload "voorbeeld"

loading

ok.
list

100 Dit is een eenvoudige teksteditor.

ok.
edit off

ready.
```

EDMA

Token: $FE $21

Formaat: **EDMA** commando**,** lengte**,** bron**,** doel

Gebruik: Kopieert of werkt grote hoeveelheden geheugen snel.

EDMA ("Extended Direct Memory Access") is de snelste methode om geheugengebieden te manipuleren met behulp van de DMA-controller. Raadpleeg *het grote Mega65-boek*, F018-compatibele controller voor Direct Memory Access (DMA) (Bijlage P) voor meer informatie over EDMA.

commando 0: kopiëren, of 3: vullen.

Omdat de commandobits hetzelfde register delen met andere bits, kunt u bijvoorbeeld bit 5 gebruiken om de lusomkering te activeren. Dit werkt ook bij overlappende geheugenregio's voor bron en doel. Zie het onderstaande voorbeeld.

lengte aantal bytes (in het bereik 0 tot 65535). OPMERKING: Het opgeven van lengte 0 wordt geïnterpreteerd als een lengte van 65536 (exact 64KB).

bron 28-bits adres van het leesgebied of het vulbyte.

doel 28-bits adres van het schrijfgebied.

Opmerkingen:

EDMA heeft toegang tot het volledige 256MB-adresbereik, met maximaal 28 bits voor de adressen van bron en doel.

Voorbeelden:

Gebruik van **EDMA**

```
EDMA 0, $800, $F700, $8000000 : REM KOPIEER SCALAIRE VARIABELEN NAAR ZOLDER-RAM

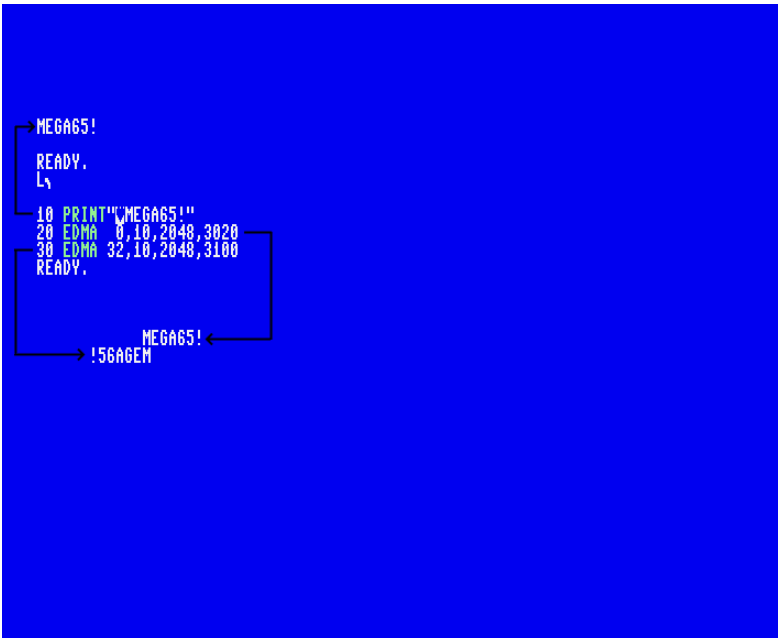
EDMA 3, 80*25, 32, 2048       : REM MAAK HET SCHERM LEEG

EDMA 0, 80*25, 2048, $8000800 : REM KOPIEER SCHERM NAAR ZOLDER-RAM
```

Door 32 (bit 5) aan de commandoparameter toe te voegen, kan de DMA-bewerking in omgekeerde volgorde worden uitgevoerd:

```
10 PRINT "■MEGA65!"
20 EDMA  0, 10, 2048, 3020 : REM 2048 IS BEGIN SCHERMGEHEUGEN
30 EDMA 32, 10, 2048, 3100 : REM 3020 EN 3100 ZIJN ONDERSTE DEEL SCHERM
```

Lijst en uitvoer van het omgekeerde voorbeeld:



EL

Token: N.v.t.

Formaat: **EL**

Gebruik: Het regelnummer waar de meest recente BASIC-fout is opgetreden, of de waarde -1 indien er geen fout is opgetreden.

Opmerkingen:

EL is een gereserveerde systeemvariabele.

Deze variabele wordt doorgaans gebruikt in een **TRAP**-routine, waarbij het foutregelnummer uit **EL** wordt gelezen.

Voorbeeld: Gebruik van **EL**

```
 10 TRAP 100
 20 PRINT SQR(-1)                        : REM OPZETTELIJKE FOUT
 30 PRINT "BIJ REGEL 30"                 : REM PROGRAMMALOOP GAAT HIER VERDER
 40 END
100 IF ER > 0 THEN PRINT ERR$(ER); " ERROR"
110 PRINT " OP REGEL"; EL
120 RESUME NEXT                          : REM GA VERDER NA FOUT
```

ELLIPSE

Token: $FE $30

Formaat: **ELLIPSE** xc**,** yc**,** xr**,** yr[**,** vlaggen **,** begin**,** einde]

Gebruik: Bitmapgraphics: tekent een ellips.

xc de x-coördinaat van het midden in pixels.

yc de y-coördinaat van het midden in pixels.

xr de x-straal van de ellips in pixels.

yr de y-straal van de ellips in pixels.

vlaggen bepaalt de vulling, bogen en de oriëntatie van de nulradiaal (combs-vlag, genoemd naar **retroCombs**). Standaardinstelling (nul) is: niet vullen, poten tekenen, starten met tekenen om 3 uur.

Bit	Naam	Waarde	Actie bij ingesteld
0	vulling	1	Vul ellips of boog met de huidige penkleur
1	poten	2	Onderdruk het tekenen v/d poten van een boog
2	combs	4	Laat de nulradiaal naar 12 uur wijzen

De eenheid van de begin- en eindhoek is graden in het bereik van 0 tot 360. De nulradiaal begint om 3 uur en loopt met de klok mee. De combs-vlag verschuift de nulradiaal en de startpositie naar de 12-uurpositie.

begin beginhoek om een elliptische boog te tekenen.

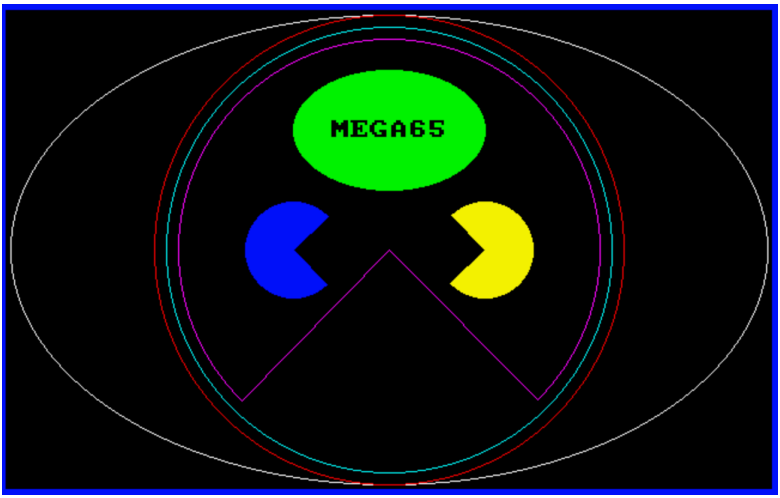
einde eindhoek om een elliptische boog te tekenen.

Opmerkingen:

ELLIPSE wordt gebruikt om ellipsen op schermen met verschillende resoluties te tekenen. Indien een volledige ellips getekend moet worden, moeten begin en einde of weggelaten worden of beide op nul worden gezet (niet 0 en 360). Het tekenen en vullen van volledige ellipsen is aanzienlijk sneller dan het gebruik van elliptische bogen.

Voorbeeld: Gebruik van **ELLIPSE**

```
100 S% = 2 : D% = 3 : W% = 320 * S% : H% = 200 * S%        : REM SCHERMINSTELLINGEN
110 CX% = W% / 2 : CY% = H% / 2                            : REM MIDDELPUNT EN STRAAL
120 RX% = W% / 2 : RY% = H% / 2
130 SCREEN W%, H%, D%                                      : REM OPEN SCHERM
140 ELLIPSE CX%, CY%, CX% - 4, CY% - 4
150 PEN 2 : CIRCLE CX%, CY%, RY% - 4, 2
160 PEN 3 : CIRCLE CX%, CY%, RY% - 14, 2
170 PEN 4 : CIRCLE CX%, CY%, RY% - 24, 0, 135, 45
180 PEN 5 : ELLIPSE CX%, CY% / 2, RX% / 4, RY% / 4, 1
190 PEN 6 : CIRCLE 120 * S%, CY%, 40, 1, 45, 315
200 PEN 7 : CIRCLE 200 * S%, CY%, 40, 1, 225, 135
210 PEN 0 : CHAR 34, CY% / 2 - 8, 2, 2, 2, "MEGA65", $3D000
220 GETKEY A$                                              : REM WACHT OP TOETSDRUK
230 SCREEN CLOSE                                           : REM SLUIT GRAFISCH SCHERM
```



ELSE

Token: $D5

Formaat: **IF** expressie **THEN** waar-clausule [**ELSE** onwaar-clausule]

Gebruik: **ELSE** is een optioneel onderdeel van een **IF**-statement.

expressie logische of numerieke expressie. Een numerieke expressie wordt als **FALSE** beschouwd indien de waarde nul is en als **TRUE** voor elke niet-nul waarde.

waar-clausule een of meer instructies die direct na **THEN** op dezelfde regel beginnen. Een regelnummer na **THEN** voert in plaats daarvan een **GOTO** uit naar die regel.

onwaar-clausule een of meer instructies die direct na **ELSE** op dezelfde regel beginnen. Een regelnummer na **ELSE** voert in plaats daarvan een **GOTO** uit naar die regel.

Opmerkingen:

Voor **ELSE** moet een dubbele punt staan. Na **ELSE** mag geen dubbele punt of einde-van-regel staan.

De standaard **IF** ... **THEN** ... **ELSE**-structuur is beperkt tot één regel, maar de **waar-clausule** en **onwaar-clausule** kunnen worden uitgesmeerd over meerdere regels met behulp van een samengestelde instructie tussen **BEGIN** ... **BEND**.

Wanneer de **waar-clausule** geen **BEGIN** ... **BEND** gebruikt, moet **ELSE** op dezelfde regel staan als **IF**.

Voorbeelden:

Gebruik van **ELSE**

```
100 REM ELSE
110 ROOD$ = CHR$(28) : ZWART$ = CHR$(144) : WIT$ = CHR$(5)
120 INPUT "GEEF EEN GETAL"; V
130 IF V < 0 THEN PRINT ROOD$; : ELSE PRINT ZWART$;
140 PRINT V                                    : REM PRINT NEGATIEVE GETALLEN IN ROOD
150 PRINT WIT$
160 INPUT "PROGRAMMA STOPPEN (J/N)"; A$
170 IF A$ = "J" THEN END
180 IF A$ = "N" THEN 120 : ELSE 160
```

Gebruik van **ELSE** met **BEGIN** ... **BEND**

```
100 A=0 : GOSUB 200
110 A=1 : GOSUB 200
120 END
200 IF A=0 THEN BEGIN
210 PRINT "HALLO"
220 BEND : ELSE BEGIN
230 PRINT "TOT ZIENS"
240 BEND
250 RETURN
```

END

Token: $80

Formaat: **END**

Gebruik: Beëindigt de uitvoering van het BASIC-programma.

De READY-prompt verschijnt en de computer schakelt over naar de directe modus om op toetsenbordinvoer te wachten.

Opmerkingen:

END wist *geen* kanalen en sluit *geen* bestanden.

Variabele-definities blijven na **END** geldig.

Het programma kan worden voortgezet met het **CONT**-statement. Na het uitvoeren van de laatste regel van een programma wordt **END** automatisch uitgevoerd.

Voorbeeld: Gebruik van **END**

```
10 IF V < 0 THEN END : REM NEGATIEF GETAL STOPT PROGRAMMA
20 PRINT V
```

ENVELOPE

Token: $FE $0A

Formaat: **ENVELOPE** n [{, aanzet, terugval, aanhouden, wegsterven, golfvorm, pw}]

Gebruik: Stelt de parameters in voor de synthese van een muziekinstrument voor gebruik met **PLAY**.

n envelopeslot (0 - 9).

aanzet aanzetsnelheid (0 - 15).

terugval terugvalsnelheid (0 - 15).

aanhouden aanhoudniveau (0 - 15).

wegsterven wegsterfsnelheid (0 - 15).

golfvorm 0: driehoek, 1: zaagtand, 2: vierkant/puls, 3: ruis of 4: ringmodulatie.

pw pulsbreedte (0 - 4095) voor de golfvorm.

Er zijn 10 slots voor het opslaan van instrumentparameters, vooraf ingesteld met de volgende standaardwaarden:

n	A	D	S	R	WF	PW	Instrument
0	0	9	0	0	2	1536	piano
1	12	0	12	0	1		accordeon
2	0	0	15	0	0		calliope
3	0	5	5	0	3		drum
4	9	4	4	0	0		fluit
5	0	9	2	1	1		gitaar
6	0	9	0	0	2	512	klavecimbel
7	0	9	9	0	2	2048	orgel
8	8	9	4	1	2	512	trompet
9	0	9	0	0	0		xylofoon

Voorbeeld: Gebruik van **ENVELOPE**

```
10 ENVELOPE 9, 10, 5, 10, 5, 2, 4000
20 VOL 9, 9
30 TEMPO 30
40 PLAY "T9O4Q CDEFGAB U3T8 CDEFGAB L", "T5O3Q H CGEQG T7 HCGEQG L"
```

ER

Token: N.v.t.

Formaat: **ER**

Gebruik: Het nummer van de meest recente BASIC-fout die heeft plaatsgevonden, of -1 wanneer er geen fout was opgetreden.

Opmerkingen:

ER is een gereserveerde systeemvariabele.

Deze variabele wordt doorgaans in een **TRAP**-routine gebruikt, waarbij het foutnummer aan **ER** wordt ontleend.

De mogelijke waarden van **ER** zijn als volgt:

ER	Betekenis	Vertaling
0	OK	Oké
1	TOO MANY FILES	Teveel bestanden
2	FILE OPEN	Bestand open
3	FILE NOT OPEN	Bestand niet open
4	FILE NOT FOUND	Bestand niet gevonden
5	DEVICE NOT PRESENT	Apparaat niet aanwezig
6	NOT INPUT FILE	Geen invoerbestand
7	NOT OUTPUT FILE	Geen uitvoerbestand
8	MISSING FILE NAME	Ontbrekende bestandsnaam
9	ILLEGAL DEVICE NUMBER	Ongeldig apparaatnummer
10	NEXT WITHOUT FOR	NEXT zonder FOR
11	SYNTAX	Syntaxis(fout)
12	RETURN WITHOUT GOSUB	RETURN zonder GOSUB
13	OUT OF DATA	data is op
14	ILLEGAL QUANTITY	ongeldige hoeveelheid
15	OVERFLOW	overloop(fout)
16	OUT OF MEMORY	geheugen is op
17	UNDEFINED STATEMENT	onbekende opdracht
18	BAD SUBSCRIPT	ongeldige index
19	REDIMENSIONED ARRAY	array al gedimensioneerd
20	DIVISION BY ZERO	deling door nul
21	ILLEGAL DIRECT	ongeldig direct
22	TYPE MISMATCH	strijdig type
23	STRING TOO LONG	tekenreeks te lang
24	FILE DATA	bestandsdata
25	FORMULA TOO COMPLEX	formule te complex
26	CAN'T CONTINUE	kan niet doorgaan
27	UNDEFINED FUNCTION	ongedefinieerde functie
28	VERIFY	verificatie(fout)
29	LOAD	laad(fout)
30	BREAK	afbreek(fout)

voortgezet ...

ER	Betekenis	Vertaling
31	CAN'T RESUME	kan niet doorgaan
32	LOOP NOT FOUND	lus niet gevonden
33	LOOP WITHOUT DO	LOOP zonder DO
34	DIRECT MODE ONLY	uitsluitend directe modus
35	SCREEN NOT OPEN	scherm niet open
36	BAD DISK	slechte disk
37	BEND NOT FOUND	BEND niet gevonden
38	LINE NUMBER TOO LARGE	regelnummer te hoog
39	UNRESOLVED REFERENCE	onopgeloste verwijzing
40	UNIMPLEMENTED COMMAND	ongeïmplementeerd commando
41	FILE READ	bestandlees(fout)
42	EDIT MODE	edit-modus
43	RESERVED NAME	gereserveerde naam
44	CBDOS DRIVES ONLY	uitsluitend CBDOS-drives
45	OK	Oké

Voorbeeld: Gebruik van **ER**

```
10 TRAP 100
20 PRINT SQR(-1)        : REM OPZETTELIJKE FOUT
30 PRINT "BIJ REGEL 30"  : REM PROGRAMMALOOP GAAT HIER VERDER
40 END
100 IF ER > 0 THEN PRINT ERR$(ER); " ERROR";
110 PRINT " OP REGEL"; EL
120 RESUME NEXT         : REM GA VERDER NA FOUT
```

ERASE

Token: $FE $2A

Formaat: **ERASE** bestandsnaam [**,D** station] [**,U** eenheid] [**,R**]

Gebruik: Verwijdert een schijfbestand.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

R herstelt een eerder verwijderd bestand. Dit werkt alleen wanneer er geen schrijfbewerkingen hebben plaatsgevonden tussen het verwijderen en het herstellen, de inhoud van de schijf mogelijk hebben gewijzigd.

Opmerkingen:

ERASE is een synoniem van **SCRATCH** en **DELETE**.

In de directe modus wordt het succes en het aantal verwijderde bestanden afgedrukt. De voorlaatste waarde uit het bericht bevat het aantal succesvol verwijderde bestanden.

Voorbeelden:

Gebruik van **ERASE**

```
ERASE "DRM", U9          : REM WIS HET BESTAND "DRM" OP UNIT 9
01, FILES SCRATCHED,01,00

ERASE "OUD*"             : REM WIS ALLE BESTANDEN BEGINEND MET "OUD"
01, FILES SCRATCHED,04,00

ERASE "R*=PRG"           : REM WIS ALLE PROGRAMMABESTANDEN BEGINNEND MET "R"
01, FILES SCRATCHED,09,00
```

ERR$

Token: $D3

Formaat: **ERR$**(getal)

Retourneert:

De tekenreeks met de beschrijving van een gegeven BASIC-foutnummer.

nummer BASIC-foutnummer (bereik 1 - 44).

Deze functie wordt doorgaans gebruikt in een **TRAP**-routine, waarbij het foutnummer wordt gelezen uit de gereserveerde variabele **ER**.

Opmerkingen:

Een BASIC-foutnummer buiten het bereik (1 - 44) geeft een `OK`.

Voorbeelden:

Gebruik van **ERR$**

```
10 TRAP 100
20 PRINT SQR(-1)        : REM OPZETTELIJKE FOUT
30 PRINT "BIJ REGEL 30"  : REM PROGRAMMALOOP GAAT HIER VERDER
40 END
100 IF ER > 0 THEN PRINT ERR$(ER); " ERROR";
110 PRINT " OP REGEL"; EL
120 RESUME NEXT         : REM GA VERDER NA FOUT
```

Alle foutnummers en hun beschrijvingen weergeven

```
100 SCNCLR
110 FOR N = 0 TO 255
120 PRINT USING "###"; N; ": "; ERR$(N)
130 IF MOD(N, 16) <> 15 THEN 160
140 PRINT "[ MEER ]"
150 GETKEY A$
160 NEXT N
170 PRINT "DONE"
```

EXIT

Token: $ED

Formaat: **EXIT**

Gebruik: Verlaat de huidige **DO** ... **LOOP** en vervolgt de uitvoering bij het eerste instructie na **LOOP**.

Opmerkingen:
In geneste lussen verlaat **EXIT** alleen de huidige lus en vervolgt de uitvoering in de buitenste lus (indien aanwezig).

Voorbeeld: Gebruik van **EXIT**

```
1 REM EXIT
10 OPEN 2, 8, 0, "$"          : REM OPEN CATALOGUS
15 IF DS THEN PRINT DS$ : STOP : REM KAN NIET LEZEN
20 GET#2, D$, D$              : REM LAADADRES WEGGOOIEN
25 DO                         : REM REGELLUS
30   GET#2, D$, D$            : REM REGELPOINTER WEGGOOIEN
35   IF ST THEN EXIT          : REM END-OF-FILE
40   GET#2, LO, HI            : REM BESTANDSGROOTTEBYTES
45   S = LO + 256 * HI        : REM BESTANDSGROOTTE
50   LINE INPUT#2, F$         : REM BESTANDSNAAM
55   PRINT S; F$              : REM PRINT BESTANDSINGANG
60 LOOP
65 CLOSE 2
```

EXP

Token: $BD

Formaat: **EXP**(numerieke expressie)

Retourneert:
De waarde van de wiskundige constante het getal van Euler (2.71828183) verheven tot de macht van het argument.

Opmerkingen:
Een argument groter dan 88 veroorzaakt een Overflow-fout.

Voorbeelden:
Gebruik van **EXP**

```
PRINT EXP(1)
 2.7182818

PRINT EXP(0)
 1

PRINT EXP(LOG(2))
 2
```

FAST

Token: $FE $25

Formaat: **FAST** [snelheid]

Gebruik: Stelt de kloksnelheid van de CPU in op 1 MHz, 3.5 MHz of 40 MHz.

snelheid kloksnelheid van de CPU, waarbij:

- 1: Stelt de CPU in op 1 MHz.
- 3: Stelt de CPU in op 3.5 MHz.
- Iets anders dan **1** of **3** stelt de CPU in op 40 MHz.

Opmerkingen:

Hoewel het mogelijk is **FAST** aan te roepen met elk reëel getal, wordt het fractionele deel (de decimale punt en alle cijfers daarna) genegeerd.

FAST is een synoniem van **SPEED**.

FAST heeft geen effect indien eerder `POKE 0, 65` is gebruikt om de CPU op 40 MHz in te stellen.

Voorbeelden:

Gebruik van **FAST**

```
FAST     : REM ZET SNELHEID MAXIMAAL (40 MHZ)

FAST 1   : REM ZET SNELHEID OP 1 MHZ

FAST 3   : REM ZET SNELHEID OP 3.5 MHZ

FAST 3.5 : REM ZET SNELHEID OP 3.5 MHZ
```

FGOSUB

Token: $FE $48

Formaat: **FGOSUB** numerieke expressie

Gebruik: Evalueert de gegeven numerieke expressie en voert vervolgens **GOSUB** uit op het subroutine bij het resulterende regelnummer.

Opmerkingen:

Wees voorzichtig bij het gebruik van **RENUMBER** om de regelnummers van uw programma te wijzigen; zorg dat eventuele **FGOSUB**-instructies nog steeds de beoogde nummers gebruiken. Indien het regelnummer niet bestaat, geeft FGOSUB een Undefined Statement-fout.

Voorbeeld: Gebruik van **FGOSUB**

```
 10 INPUT "WELKE SUBROUTINE UITVOEREN 100, 200, 300"; LI
 20 FGOSUB LI                  : REM HOPELIJK BESTAAT DIT REGEL#
 30 GOTO 10                    : REM HERHAAL
100 PRINT "OP REGEL 100" : RETURN
200 PRINT "OP REGEL 200" : RETURN
300 PRINT "OP REGEL 300" : RETURN
```

FGOTO

Token: $FE $47

Formaat: **FGOTO** numerieke expressie

Gebruik: Evalueert de gegeven numerieke expressie en gaat vervolgens met **GOTO** naar het resulterende regelnummer.

Opmerkingen:

Wees voorzichtig bij het gebruik van **RENUMBER** om de regelnummers van uw programma te wijzigen; zorg dat eventuele **FGOTO**-instructies nog steeds de beoogde nummers gebruiken. Indien het regelnummer niet bestaat, treedt een Undefined Statement-fout op.

Voorbeeld: Gebruik van **FGOTO**

```
 10 INPUT "WELK REGEL# UITVOEREN 100, 200, 300"; LI
 20 FGOTO LI           : REM HOPELIJK BESTAAT DIT REGEL#
 30 END
100 PRINT "OP REGEL 100" : END
200 PRINT "OP REGEL 200" : END
300 PRINT "OP REGEL 300" : END
```

FILTER

Token: $FE $03

Formaat: **FILTER** sid [{**,** frequentie**,** lp**,** bp**,** hp**,** res}]

Gebruik: Stelt de parameters van een SID-geluidsfilter in.

sid 1: rechter SID, 2: linker SID.

frequentie grensfrequentie van het filter (0 - 2047).

lp laagdoorlaatfilter (0: uit, 1: aan).

bp banddoorlaatfilter (0: uit, 1: aan).

hp hoogdoorlaatfilter (0: uit, 1: aan).

res resonantie (0 - 15).

Opmerkingen:

Ontbrekende parameters behouden hun huidige waarde. Het effectieve filter is de som van alle filterinstellingen. Hiermee worden bandstop- en notch-effecten mogelijk gemaakt.

Voorbeeld: Gebruik van **FILTER**

```
 10 PLAY "T7X103P9C"
 15 SLEEP 0.02
 20 PRINT "ZWAAI MET LAAGDOORLAATFILTER" : L = 1 : B = 0 : H = 0 : GOSUB 100
 30 PRINT "ZWAAI MET BANDDOORLAATFILTER" : L = 0 : B = 1 : H = 0 : GOSUB 100
 40 PRINT "ZWAAI MET HOOGDOORLAATFILTER" : L = 0 : B = 0 : H = 1 : GOSUB 100
 50 GOTO 20
100 REM *** ZWAAI ***
110 FOR F = 50 TO 1950 STEP 50
120 IF F >= 1000 THEN FF = 2000 - F : ELSE FF = F
130 FILTER 1, FF, L, B, H, 15
140 PLAY "X1"
150 SLEEP 0.02
160 NEXT F
170 RETURN
```

FIND

Token: $FE $2B

Formaat: **FIND /**tekenreeks**/** [**,** regelbereik]
FIND "tekenreeks**"** [**,** regelbereik]

Gebruik: Doorzoekt het BASIC-programma dat zich momenteel in het geheugen bevindt op alle voorkomens van een tekenreeks.

Het zoekt in een opgegeven regelbereik (indien gespecificeerd); anders wordt het volledige BASIC-programma doorzocht.

Bij elk voorkomen van de tekenreeks wordt de regel weergegeven met de tekenreeks gemarkeerd.

NO SCROLL kan worden gebruikt om de uitvoer te pauzeren.

Opmerkingen:

Bijna elk teken dat geen deel uitmaakt van de tekenreeks, waaronder letters en leestekens, kan in plaats van de "/" (schuine streep) worden gebruikt.

Het gebruik van dubbele aanhalingstekens (**"**) als scheidingsteken heeft een speciaal effect: de zoektekst wordt niet getokeniseerd. **FIND "FOR"** zoekt naar de drie letters "F," "O," en "R," niet naar het BASIC-sleutelwoord **FOR**. Daardoor kan het het woord **FOR** vinden in stringconstanten of **REM**-instructies, maar niet in programmacode.

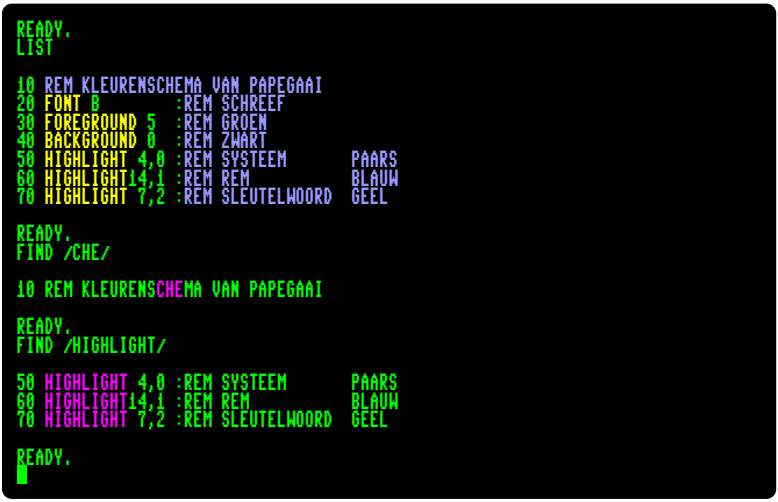
Anderzijds vindt **FIND /FOR/** alle voorkomens van het BASIC-sleutelwoord, maar niet de tekst "FOR" in strings.

Gedeeltelijke sleutelwoorden kunnen niet worden gezocht. **FIND /LOO/** vindt bijvoorbeeld het sleutelwoord **LOOP** niet.

Vanwege de wijze waarop BASIC wordt verwerkt, is het zoeken van de sleutelwoorden **REM** en **DATA** alleen mogelijk met de dubbelepunt als scheidingsteken: **FIND :REM TODO:** Dit werkt niet met het **CHANGE**-commando.

FIND is een editorcommando dat alleen in de directe modus kan worden gebruikt.

Voorbeeld: Gebruik van **FIND**


READY.
LIST
10 REM KLEURENSCHEMA VAN PAPEGAAI
20 FONT B :REM SCHREEF
30 FOREGROUND 5 :REM GROEN
40 BACKGROUND 0 :REM ZWART
50 HIGHLIGHT 4,0 :REM SYSTEEM PAARS
60 HIGHLIGHT 4,1 :REM REM BLAUW
70 HIGHLIGHT 7,2 :REM SLEUTELWOORD GEEL
READY.
FIND /CHE/
10 REM KLEURENSCHEMA VAN PAPEGAAI
READY.
FIND /HIGHLIGHT/
50 HIGHLIGHT 4,0 :REM SYSTEEM PAARS
60 HIGHLIGHT 4,1 :REM REM BLAUW
70 HIGHLIGHT 7,2 :REM SLEUTELWOORD GEEL
READY.

FN

Token: $A5

Formaat: **FN** naam(numerieke expressie)

Gebruik: **FN**-functies zijn door de gebruiker gedefinieerde functies die een numerieke expressie als argument aanvaarden en een reële waarde retourneren. Ze moeten eerst met **DEF FN** worden gedefinieerd voordat ze kunnen worden gebruikt.

naam naam van de functie zoals eerder gedefinieerd met **DEF FN**.

Voorbeeld: Gebruik van **FN**

```
10 PD = π / 180
20 DEF FN CD(X) = COS(X * PD)     : REM COS VOOR GRADEN
30 DEF FN SD(X) = SIN(X * PD)     : REM SIN VOOR GRADEN
40 FOR D = 0 TO 360 STEP 90
50 PRINT USING "####"; D;
60 PRINT USING " ##.##"; FNCD(D);
70 PRINT USING " ##.##"; FNSD(D)
80 NEXT D

RUN
  0  1.00  0.00
 90  0.00  1.00
180 -1.00  0.00
270  0.00 -1.00
360  1.00  0.00
```

FONT

Token: $FE $46

Formaat: **FONT** <**A** | **B** | **C**>

Gebruik: Wisselt alle tekens naar het opgegeven ingebouwde lettertype.

FONT A is het PETSCII-lettertype waarbij verschillende kleine letters zijn vervangen door ASCII-leestekens.

FONT B is een alternatieve weergave van **FONT A**.

FONT C is het PETSCII-lettertype. Dit is de standaardinstelling wanneer de MEGA65 voor het eerst wordt ingeschakeld.

Hiermee worden alle wijzigingen die met het **CHARDEF**-commando zijn aangebracht, ongedaan gemaakt.

De ASCII-symbolen van lettertypen **A** en **B** worden ingevoerd door de toetsen in de onderstaande tabel in te drukken, waarvan er enkele ook het ingedrukt houden van de  -toets vereisen. De codes voor hoofd- letters en kleine letters zijn omgedraaid ten opzichte van ASCII.

Code	Toets	PETSCII	ASCII
$5C	Pond	£	\ (schuine streep)
$5E	Pijl omhoog (naast RESTORE)	↑	^ (accentcircumflex)
$5F	Pijl links (naast 1)	←	_ (laagstreep)
$7B	MEGA + dubbele punt	┼	{ (openende accolade)
$7C	MEGA + punt	┬	| (verticale streep)
$7D	MEGA + puntkomma	┤	} (sluitende accolade)
$7E	MEGA + komma	├	~ (tilde)

Opmerkingen:

De extra ASCII-symbolen die door **FONT A** en **B** worden aangeboden, zijn alleen beschikbaar bij gebruik van de kleine-letterset.

FONT D, **FONT E** en **FONT F** komen overeen met respectievelijk **A**, **B** en **C**, waarbij het lettertype automatisch naar de kleine-lettermodus wordt geschakeld.

Voorbeelden:

Gebruik van **FONT**

```
FONT A : REM ASCII - ACTIVEER (!)←↑
FONT B : REM VERGELIJKBAAR MET A, MET SCHREEF
FONT C : REM COMMODORE-FONT (STANDAARD)
```

FOR

Token: $81

Formaat: **FOR** index = begin **TO** einde [**STEP** stap] ... **NEXT** [index]

Gebruik: **FOR**-instructies starten een BASIC-lus met een indexvariabele.

index kan bij elke iteratie met een constante waarde worden verhoogd of verlaagd. Standaard wordt de variabele met 1 verhoogd. De index-variabele moet een reële variabele zijn.

begin wordt gebruikt om de index te initialiseren.

einde wordt aan het einde van een iteratie gecontroleerd en bepaalt of nog een iteratie wordt uitgevoerd of dat de lus wordt afgesloten.

stap definieert de wijziging die aan het einde van een iteratie op de indexvariabele wordt toegepast. Positieve stapwaarden verhogen deze, terwijl negatieve waarden deze verlagen. Standaard is de waarde 1.0 indien niet opgegeven.

Opmerkingen:

Bij positieve verhogingen moet **einde** groter of gelijk zijn aan **begin**, terwijl bij negatieve verhogingen **einde** kleiner of gelijk moet zijn aan **begin**.

Het is een slechte programmeerpraktijk om de waarde van de **index**-variabele binnen de lus te wijzigen of met **GOTO** in of uit een lus te springen.

Voorbeelden:

Gebruik van **FOR**

```
10 FOR D = 0 TO 360 STEP 30
20 R = D * π / 180
30 PRINT D; R; SIN(R); COS(R); TAN(R)
40 NEXT D

10 DIM M(20, 20)
20 FOR I = 0 TO 20
30 FOR J = I TO 20
40 M(I, J) = I + 100 * J
50 NEXT J, I
```

FOREGROUND

Token: $FE $39

Formaat: **FOREGROUND** kleur

Gebruik: Stelt de voorgrondtekstkleur in voor de volgende **PRINT**-commando's.

kleur paletindex in het bereik 0 - 31.

Zie bijlage 6 op bladzijde 327 voor de lijst met kleuren in het standaard systeempalet.

Opmerkingen:

Dit is een andere naam voor **COLOR**.

Voorbeeld: Gebruik van **FOREGROUND**



FORMAT

Token: $FE $37

Formaat: **FORMAT** schijfnaam [**,I** id] [**,D** station] [**,U** eenheid]

Gebruik: Formatteert een schijf. OPMERKING: *Hierbij worden alle gegevens op de schijf gewist.*

I schijf-ID. De maximale lengte is 2 tekens.

schijfnaam is een tekenreeks tussen aanhalingstekens, bijv. `"DATA"` of een tekenreeksexpressie tussen haakjes, bijv. `(DN$)`. De maximale lengte is 16 tekens.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

FORMAT is een andere naam voor het commando **HEADER**.

Voor nieuwe diskettes die nog niet in MEGA65 (1581)-formaat zijn geformatteerd, moet het schijf-ID worden opgegeven met de parameter **I**. Hiermee wordt het formaatcommando overgeschakeld naar een laagformattering, waarbij sector-ID's worden geschreven en alle inhoud wordt gewist. Dit kost enige tijd, omdat elk blok op de diskette wordt geschreven.

Indien de parameter **I** wordt weggelaten, wordt een snelle formattering uitgevoerd. Dit is alleen mogelijk als de schijf al is geformatteerd als MEGA65- of 1581-diskette. Een snelle formattering schrijft de nieuwe schijfnaam, wist de eerste directorysector en de bloktoewijzingstabel, en markeert alle blokken als vrij. Het schijf-ID wordt niet gewijzigd. Omdat de directory wordt gewist, kunnen bestanden na een snelle formattering niet meer worden hersteld met **ERASE R**. U kunt meer lezen over **ERASE** op bladzijde 110.

Voorbeelden:

Gebruik van **FORMAT**

```
FORMAT "AVONTUUR", ID,K   : REM SCHIJF VOLLEDIG FORMATTEREN MET NAAM "ADVENTURE" EN ID "DK"

FORMAT "ZORK-I", U9       : REM SCHIJF SNEL FORMATTEREN IN UNIT 9 MET NAAM "ZORK-I"

FORMAT "GEWELF", D1, U10  : REM SCHIJF SNEL FORMATTEREN IN DRIVE 1 UNIT 10 MET NAAM "GEWELF"
```

FRE

Token: $B8

Formaat: **FRE**(bank)

Retourneert:

Informatie over het geheugengebruik of andere systeemkenmerken.

FRE(0) retourneert het aantal vrije bytes in bank 0, die wordt gebruikt voor de BASIC-programmabron.

FRE(1) retourneert het aantal vrije bytes in bank 1, de bank voor BASIC-variabelen, arrays en strings. **FRE(1)** activeert tevens "garbage collection", een proces dat strings in gebruik bovenaan de bank verzamelt en daarmee het stringgeheugen defragmenteert.

FRE(4) en **FRE(5)** retourneren respectievelijk het geheugengebruik van banken 4 en 5 als een 8-bits bitveld. Elke bit vertegenwoordigt 8KB geheugen in de bank. Bit 0 van **FRE(4)** bijvoorbeeld vertegenwoordigt het geheugengebied 4.0000 - 4.1FFF. Indien een bit is ingesteld, is het gereserveerd, hetzij door **MEM** hetzij door het systeem. Zie **MEM**.

FRE(-1) retourneert de ROM-versie, een zescijferig getal in het formaat 92XXXX.

Voorbeeld: Gebruik van **FRE**

```
10 PM = FRE(0)
20 VM = FRE(1)
30 RV = FRE(-1)
40 PRINT PM; " BYTES BESCHIKBAAR VOOR PROGRAMMA"
50 PRINT VM; " BYTES BESCHIKBAAR VOOR VARIABELEN"
60 PRINT RV; " ROM-VERSIE"
```

FREAD#

Token: $FE $1C

Formaat: **FREAD#** kanaal**,** pointer**,** grootte

Gebruik: Leest **grootte** bytes van **kanaal** in het geheugen, beginnend op het 32-bits adres **pointer**.

kanaal nummer, dat werd opgegeven bij een eerdere aanroep van commando's zoals **DOPEN** of **OPEN**.

FREAD# kan worden gebruikt om gegevens direct van schijf in een variabele te lezen. Het wordt aanbevolen de **POINTER**-instructie te gebruiken voor het pointer-argument en de grootteparameter te berekenen door het aantal elementen te vermenigvuldigen met de itemgrootte.

Type	Itemgrootte
Byte-array	1
Integer-array	2
Real-array	5

Opmerkingen:

Houd er rekening mee dat de **POINTER**-functie met een stringargument *niet* het stringadres retourneert, maar de stringdescriptor. Het wordt afgeraden **FREAD#** te gebruiken voor strings of stringarrays, tenzij u volledig op de hoogte bent van de interne werking van de stringopslag.

Om in een array te lezen, moet u altijd een array-index opgeven zodat **POINTER** het adres van een element retourneert. Het startadres van array `XY()` is `POINTER(XY(0))`. `POINTER(XY)` retourneert het adres van de scalaire variabele `XY`.

Voorbeeld: Gebruik van **FREAD#**

```
100 N = 23
110 DIM B&(N), C&(N)
120 DOPEN#2, "TEKST"
130 FREAD#2, POINTER(B&(0)), N
140 DCLOSE#2
150 FOR I = 0 TO N - 1 : PRINT CHR$(B&(I)); : NEXT
160 FOR I = 0 TO N - 1 : C&(I) = B&(N - 1 - I) : NEXT
170 DOPEN#2, "REVERS,W"
180 FWRITE#2, POINTER(C&(0)), N
190 DCLOSE#2
```

FREEZER

Token: $FE $4A

Formaat: **FREEZER**

Gebruik: Opent het Freezer-menu.

Opmerkingen:
Het invoeren van het commando **FREEZER** is een alternatief voor het indrukken en loslaten van de RESTORE-toets.

Voorbeeld: Gebruik van **FREEZER**

```
FREEZER : REM FREEZERMENU AANROEPEN
```

FWRITE#

Token: $FE $1E

Formaat: **FWRITE#** kanaal**,** pointer**,** grootte

Gebruik: Schrijft **grootte** bytes naar **kanaal** vanuit het geheugen, beginnend op het 32-bits adres **pointer**.

kanaal kanaalnummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

FWRITE# kan worden gebruikt om de waarde van een variabele naar een bestand te schrijven.

Het wordt aanbevolen de **POINTER**-instructie te gebruiken voor het pointer-argument en de grootteparameter te berekenen door het aantal elementen te vermenigvuldigen met de itemgrootte.

Zie de tabel met itemgroottes voor **FREAD#** op bladzijde 126.

Opmerkingen:

Houd er rekening mee dat de **POINTER**-functie met een stringargument *niet* het stringadres retourneert, maar de stringdescriptor. Het wordt afgeraden **FWRITE#** te gebruiken voor strings of stringarrays, tenzij u volledig op de hoogte bent van de interne werking van de stringopslag.

Om een array te schrijven, moet u altijd een array-index opgeven zodat **POINTER** het adres van een element retourneert. Het startadres van array `XY()` is `POINTER(XY(0))`. `POINTER(XY)` retourneert het adres van de scalaire variabele `XY`.

Voorbeeld: Gebruik van **FWRITE#**

```
100 N = 23
110 DIM B&(N), C&(N)
120 DOPEN#2, "TEKST"
130 FREAD#2, POINTER(B&(0)), N
140 DCLOSE#2
150 FOR I = 0 TO N - 1 : PRINT CHR$(B&(I)); : NEXT
160 FOR I = 0 TO N - 1 : C&(I) = B&(N - 1 - I) : NEXT
170 DOPEN#2, "REVERS,W"
180 FWRITE#2, POINTER(C&(0)), N
190 DCLOSE#2
```

GCOPY

Token: $FE $32

Formaat: **GCOPY** x, y, breedte, hoogte

Gebruik: Bitmapgrafieken: kopieert de inhoud van de opgegeven rechthoek met de bovenliggende positie **x, y** en de **breedte** en **hoogte** naar een buffer.

Het gekopieerde gebied kan op elke positie worden geplakt met het commando **PASTE**.

Opmerkingen:

De grootte van de rechthoek is begrensd door de omvang van 1KB van de buffer. De geheugenvereiste voor een gebied is breedte * hoogte * aantal bitplanes / 8. Dit mag niet groter zijn dan of gelijk aan 1024 bytes. Voor een scherm met 4 bitplanes bijvoorbeeld heeft een gebied van 45 x 45 1012,5 bytes nodig.

Voorbeeld: Gebruik van **GCOPY**

```
10 SCREEN 320, 200, 2
20 BOX 60, 60, 300, 180, 1 : REM TEKEN EEN WIT VAK
30 GCOPY 140, 80, 40, 40   : REM KOPIEER EEN GEBIED VAN 40 * 40
40 PASTE 10, 10            : REM PLAK HET OP EEN NIEUWE POSITIE
50 GETKEY A$               : REM WACHT OP TOETSDRUK
60 SCREEN CLOSE
```

GET

Token: $A1

Formaat: **GET** variabele

Gebruik: Haalt het volgende teken, of de byte-waarde van het volgende teken, uit de toetsenbordwachtrij.

Indien de **variabele** die wordt ingesteld een stringvariabele is en de wachtrij leeg is, wordt een lege string toegewezen; anders wordt een string van één teken aangemaakt en toegewezen.

Indien de **variabele** een numerieke variabele is, wordt de byte-waarde van de toets toegewezen; anders wordt nul toegewezen indien de wachtrij leeg is. **GET** wacht niet op toetsinvoer en is daarom bruikbaar om op regelmatige intervallen of in lussen op toetsaanslagen te controleren.

Opmerkingen:

GETKEY werkt vergelijkbaar, maar wacht tot er een toets is ingedrukt.

Voorbeeld: Gebruik van **GET**

```
10 DO : GET A$ : LOOP UNTIL A$ <> ""
20 IF A$ = "W" THEN PRINT "NOORD"  : REM GA NOORD
30 IF A$ = "A" THEN PRINT "WEST"   : REM GA WEST
40 IF A$ = "S" THEN PRINT "OOST"   : REM GA OOST
50 IF A$ = "Z" THEN PRINT "ZUID"   : REM GA ZUID
60 IF A$ = CHR$(13) THEN END       : REM ERUIT
70 GOTO 10
```

GET#

Token: $A1 '#'

Formaat: **GET#** kanaal**,** variabele [**,** variabele ...]

Gebruik: Leest een enkele byte van het kanaalargument en wijst tekenreeksen van één teken toe aan stringvariabelen, of een 8-bits binair getal aan numerieke variabelen.

Dit is bruikbaar om tekens (of bytes) een voor een uit een invoerstroom te lezen.

kanaal nummer, dat werd opgegeven bij een eerdere aanroep van commando's zoals **DOPEN** of **OPEN**.

Opmerkingen:

Alle waarden van 0 tot en met 255 zijn geldig, zodat **GET#** ook kan worden gebruikt om binaire gegevens te lezen.

Voorbeeld: Gebruik van **GET#**

```
10 OPEN 2, 8, 0, "$"          : REM OPEN CATALOGUS
15 IF DS THEN PRINT DS$ : STOP : REM KAN NIET LEZEN
20 GET#2, D$, D$              : REM LAADADRES WEGGOOIEN
25 DO                         : REM REGELLUS
30   GET#2, D$, D$            : REM REGELPOINTER WEGGOOIEN
35   IF ST THEN EXIT          : REM END-OF-FILE
40   GET#2, LO, HI            : REM BESTANDSGROOTTEBYTES
45   S=LO+256*HI              : REM BESTANDSGROOTTE
50   LINE INPUT#2, F$         : REM BESTANDSNAAM
55   PRINT S;F$               : REM PRINT BESTANDSINGANG
60 LOOP
65 CLOSE 2
```

GETKEY

Token: $A1 (GET) $F9 (KEY)

Formaat: **GETKEY** variabele

Gebruik: Haalt het volgende teken, of de byte-waarde van het volgende teken, uit de toetsenbordwachtrij. Indien de wachtrij leeg is, wacht het programma tot er een toets is ingedrukt.

Nadat er een toets is ingedrukt, wordt de **variabele** ingesteld en gaat de programma-uitvoering verder. Bij gebruik met een stringvariabele wordt een string van één teken aangemaakt en toegewezen. Indien de variabele een numerieke variabele is, wordt de byte-waarde toegewezen.

Voorbeeld: Gebruik van **GETKEY**

```
10 GETKEY A$                    : REM WACHTEN LEES TEKEN
20 IF A$ = "W" THEN PRINT "NOORD" : REM GA NOORD
30 IF A$ = "A" THEN PRINT "WEST"  : REM GA WEST
40 IF A$ = "S" THEN PRINT "OOST"  : REM GA OOST
50 IF A$ = "Z" THEN PRINT "ZUID"  : REM GA ZUID
60 IF A$ = CHR$(13) THEN END      : REM EINDE
70 GOTO 10
```

GO64

Token: $CB (GO) $36 $34 ("64")

Formaat: **GO64**

Gebruik: Schakelt de MEGA65 over naar de C64-compatibele modus.

Indien u zich in de directe modus bevindt, wordt een beveiligingsprompt `ARE YOU SURE?` weergegeven, waarop u met Y moet antwoorden om door te gaan.

U kunt terugkeren naar de MEGA65-modus met het commando `SYS 58552`.

Opmerkingen:

Indien u met `SYS 58552` terugkeert naar de MEGA65, vindt de overschakeling onmiddellijk plaats. Er wordt geen bevestiging gevraagd.

Voorbeeld: Gebruik van **GO64**

```
READY.
GO64
ARE YOU SURE? Y
```

Terugkeren naar MEGA65

```
READY.
SYS 58552
```

GOSUB

Token: $8D

Formaat: **GOSUB** regelnummer

Gebruik: **GOSUB** (GOto SUBroutine) zet de programma-uitvoering voort op de opgegeven BASIC-regel, waarbij de huidige BASIC-programmateller en regelnummer op de run-tijdstack worden bewaard.

Hiermee kan de uitvoering worden hervat na het **GOSUB**-statement, zodra een **RETURN**-statement in de aangeroepen subroutine is uitgevoerd.

Aanroepen van subroutines via **GOSUB** mogen genest worden, maar de subroutines moeten altijd eindigen met **RETURN**, anders kan een stack overflow optreden.

Opmerkingen:

In tegenstelling tot andere programmeertalen ondersteunt BASIC geen argumenten of lokale variabelen voor subroutines. Programma's kunnen worden geoptimaliseerd door subroutines te groeperen aan het begin van de programmabron. De **GOSUB**-aanroepen hebben dan lage regelnummers met minder cijfers om te verwerken. De subroutines worden ook sneller gevonden, omdat het zoeken naar subroutines vaak begint aan het begin van het programma.

Voorbeeld: Gebruik van **GOSUB**

```
 10 GOTO 100                                  : REM NAAR HOOFDPROGRAMMA
 20 REM *** SUBROUTINE DISKSTATUSCONTROLE ***
 30 DD = DS : IF DD THEN PRINT "DISKFOUT"; DS$
 40 RETURN
 50 REM *** SUBROUTINEPROMPT J/N ***
 60 DO : INPUT "DOORGAAN (J/N)"; A$
 70 LOOP UNTIL A$ = "J" OR A$ = "N"
 80 RETURN
 90 REM *** HOOFDPROGRAMMA ***
100 DOPEN#2, "BIG DATA"
110 GOSUB 30 : IF DD THEN DCLOSE#2 : GOSUB 60   : REM VRAAG
120 IF A$ = "N" THEN END
130 GOTO 100                                  : REM PROBEER NOGMAALS
```

GOTO

Token: $89 (GOTO) of $CB $A4 (GO TO)

Formaat: **GOTO** regel
GO TO regel

Gebruik: Zet de programma-uitvoering voort op de opgegeven BASIC-regel.

Opmerkingen:

Als het doel**regel**nummer groter is dan het huidige regelnummer, begint de zoekactie vanaf de huidige regel en gaat door naar hogere regelnummers. Als het doel**regel**nummer lager is, begint de zoekactie bij het eerste **regel**nummer van het programma. Het is mogelijk de uitvoeringssnelheid van het programma te optimaliseren door veelgebruikte doelen aan het begin te groeperen (met lagere regelnummers).

GOTO (als één woord geschreven) wordt sneller uitgevoerd dan **GO TO**.

Voorbeeld: Gebruik van **GOTO**

```
 10 GOTO 100                                  : REM NAAR HOOFDPROGRAMMA
 20 REM *** SUBROUTINE DISKSTATUSCONTROLE ***
 30 DD = DS : IF DD THEN PRINT "DISKFOUT"; DS$
 40 RETURN
 50 REM *** SUBROUTINEPROMPT J/N ***
 60 DO : INPUT "DOORGAAN (J/N)"; A$
 70 LOOP UNTIL A$ = "J" OR A$ = "N"
 80 RETURN
 90 REM *** HOOFDPROGRAMMA ***
100 DOPEN#2, "BIG DATA"
110 GOSUB 30 : IF DD THEN DCLOSE#2 : GOSUB 60   : REM VRAAG
120 IF A$ = "N" THEN END
130 GOTO 100                                  : REM PROBEER NOGMAALS
```

GRAPHIC

Token: $DE

Formaat: **GRAPHIC CLR**

Gebruik: Bitmapgrafieken: initialiseert het BASIC-bitmapgrafiekensysteem. Het wist het grafisch geheugen en scherm, en stelt alle parameters van de grafische context in op hun standaardwaarden.

Nadat het grafisch systeem is gewist, kunnen commando's zoals **LINE**, **PALETTE**, **PEN**, **SCNCLR** en **SCREEN** worden gebruikt om parameters van het grafisch systeem in te stellen.

Voorbeeld: Gebruik van **GRAPHIC**

```
110 GRAPHIC CLR           : REM INITIALISEER
120 SCREEN DEF 1, 1, 1, 2 : REM 640 X 400 X 2
130 SCREEN OPEN 1         : REM OPEN HET
140 SCREEN SET 1, 1       : REM TOON HET
150 PALETTE 1, 0, 0,  0, 0 : REM ZWART
160 PALETTE 1, 1, 0, 15, 0 : REM GROEN
170 SCNCLR 0              : REM VUL SCHERM MET ZWART
180 PEN 0, 1              : REM KIES PEN
190 LINE 50, 50, 590, 350 : REM TEKEN LIJN
200 GETKEY A$             : REM WACHT OP TOETSDRUK
210 SCREEN CLOSE 1        : REM SLUIT SCHERM EN HERSTEL PALET
```

HASBIT

Token: $CE $13

Formaat: **HASBIT(**adres**,** bitnummer**)**

Retourneert:

Waar (-1) indien de waarde op het adres de opgegeven bit heeft ingesteld, anders onwaar (0).

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

bitnummer waarde in het bereik van 0 - 7.

Een bankwaarde > 127 wordt gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enz.

Voorbeelden:

Gebruik van **HASBIT**

```
BANK 128

SETBIT $D031,6

PRINT HASBIT($D031,6)
-1

CLRBIT $D031,6

PRINT HASBIT($D031,6)
0
```

HEADER

Token: $F1

Formaat: **HEADER** schijfnaam [**,I** id] [**,D** station] [**,U** eenheid]

Gebruik: Formatteert een schijf. OPMERKING: *Hierbij worden alle gegevens op de schijf gewist.*

I schijf-ID. Maximale lengte is 2 tekens.

schijfnaam is een tekenreeks tussen aanhalingstekens, bijv. "DATA" of een tekenreeksexpressie tussen haakjes, bijv. (DN$). De maximale lengte is 16 tekens.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

HEADER is een andere naam voor het commando **FORMAT**.

Voor nieuwe diskettes die nog niet in MEGA65 (1581)-formaat zijn geformatteerd, moet het schijf-ID worden opgegeven met de parameter **I**. Hiermee wordt het formaatcommando overgeschakeld naar een laagformattering, waarbij sector-ID's worden geschreven en alle inhoud wordt gewist. Dit kost enige tijd, omdat elk blok op de diskette wordt geschreven.

Indien de parameter **I** wordt weggelaten, wordt een snelle formattering uitgevoerd. Dit is alleen mogelijk als de schijf al is geformatteerd als MEGA65- of 1581-diskette. Een snelle formattering schrijft de nieuwe schijfnaam, wist de eerste directorysector en de bloktoewijzingstabel, en markeert alle blokken als vrij. Het schijf-ID wordt niet gewijzigd. Omdat de directory wordt gewist, kunnen bestanden na een snelle formattering niet meer worden hersteld met **ERASE R**. U kunt meer lezen over **ERASE** op bladzijde 110.

Voorbeelden:

Gebruik van **HEADER**

```
HEADER "AVONTUUR", IDK      : REM EEN DISKETTE VOLLEDIG FORMATTEREN MET NAAM "AVON-
TUUR" EN ID "DK"

HEADER "ZORK-I", U9         : REM EEN DISKETTE IN UNIT 9 SNEL FORMATTEREN MET NAAM "ZORK-I"

HEADER "KERKERGEWELF", D1, U10 : REM EEN DISKETTE SNEL FORMATTE-
REN IN DRIVE 1 VAN UNIT 10 MET NAAM "KERKERGEWELF"
```

HELP

Token: $EA

Formaat: **HELP**

Gebruik: Geeft informatie over de plaats waar een fout is opgetreden in een BASIC-programma.

Wanneer het BASIC-programma door een fout wordt onderbroken, kan **HELP** gebruikt worden om meer informatie te verkrijgen. De vertaalde regel wordt weergegeven, waarbij de foutieve instructie gemarkeerd of onderstreept wordt getoond.

Opmerkingen:

Geeft BASIC-fouten weer. Voor fouten met betrekking tot schijf-I/O moet de schijfstatusvariabele **DS** of de schijfstatustekenreeks **DS$** worden gebruikt.

Voorbeeld: Gebruik van **HELP**

```
10 A = 1.E20
20 B = A + A : C = EXP(A) : PRINT A, B, C
RUN

?OVERFLOW ERROR IN 20
READY.
HELP

20 B = A + A : C=EXP(A) : PRINT A, B, C
```

HEX$

Token: $D2

Formaat: **HEX$**(numerieke expressie)

Retourneert:

Een vier- of achtteken hexadecimale weergave van het argument.

Het argument moet in het bereik van 0 - 4294967295 liggen, wat overeenkomt met de hex-getallen $00000000-$FFFFFFFF. Indien het argument tussen 0 - 65535 ligt, wordt een tekenreeks van vier tekens geretourneerd. Voor alle waarden boven 65535 wordt in plaats daarvan een tekenreeks van acht tekens geretourneerd.

Opmerkingen:

Indien reële getallen als argumenten worden gebruikt, wordt het decimale deel genegeerd. Met andere woorden, reële getallen worden niet afgerond.

Om een hexadecimale tekenreeks om te zetten in een getal, zie **DEC**.

Voorbeelden:

Gebruik van **HEX$**

```
PRINT HEX$(10), HEX$(100), HEX$(1000.9)
000A     0064     03E8

PRINT HEX$(4294967295)
FFFFFFFF
```

HIGHLIGHT

Token: $FE $3D

Formaat: **HIGHLIGHT** kleur [**,** modus]

Gebruik: Stelt de kleuren in die voor de code-markering worden gebruikt.

Er kunnen verschillende kleuren worden ingesteld voor systeemmeldingen, **REM**-instructies en BASIC-trefwoorden.

kleur een van de eerste 16 kleuren in het huidige palet. Zie bijlage 6 op bladzijde 327 voor de lijst met kleuren in het standaard systeempalet.

modus geeft aan waarvoor de kleur wordt gebruikt.

- 0: Systeemmeldingen (de standaardmodus).
- 1: **REM**-instructies.
- 2: BASIC-trefwoorden.

Opmerkingen:

De kleur voor systeemmeldingen wordt gebruikt bij het weergeven van foutmeldingen en in de uitvoer van **CHANGE**, **FIND** en **HELP**. De kleuren voor **REM**-instructies en BASIC-trefwoorden worden door **LIST** gebruikt.

Voorbeeld: Gebruik van **HIGHLIGHT**

```
LIST

10 REM *** DIT IS HALLO WERELD ***
20 PRINT "HALLO WERELD"

READY.
HIGHLIGHT 0,2

READY.
LIST

10 REM *** DIT IS HALLO WERELD ***
20 PRINT "HALLO WERELD"

READY.
```

IF

Token: $8B

Formaat: **IF** expressie **THEN** waar-clausule [**ELSE** onwaar-clausule]

Gebruik: Voert instructies voorwaardelijk uit.

expressie logische of numerieke expressie. Een numerieke expressie wordt als **FALSE** beschouwd indien de waarde nul is en als **TRUE** voor elke waarde ongelijk aan nul.

waar-clausule een of meer instructies die direct na **THEN** op dezelfde regel beginnen. Een regelnummer na **THEN** voert in plaats daarvan een **GOTO** uit naar die regel.

onwaar-clausule een of meer instructies die direct na **ELSE** op dezelfde regel beginnen. Een regelnummer na **ELSE** voert in plaats daarvan een **GOTO** uit naar die regel.

Opmerkingen:

De standaardstructuur **IF** ... **THEN** ... **ELSE** is beperkt tot één regel. De **waar-clausule** en **onwaar-clausule** kunnen echter worden uitgesmeerd over meerdere regels met behulp van een samengestelde instructie tussen **BEGIN** ... **BEND**.

Een **ELSE**-clause kan een andere **IF**-instructie bevatten. Indien de interne **IF**-instructie meerdere regels beslaat met **BEGIN** ... **BEND**, moet de gehele interne **IF**-instructie ook tussen **BEGIN** ... **BEND** worden geplaatst.

Vanwege een ongebruikelijke beperking bij **BEGIN** ... **BEND**: indien een **THEN**-clause **BEGIN** ... **BEND** gebruikt en er een **ELSE**-clause aanwezig is, moet het trefwoord **ELSE** op dezelfde regel staan als de vorige **BEND**-instructie. Zie het onderstaande voorbeeld.

Voorbeeld: Gebruik van **IF**

```
10 ROOD$ = CHR$(28) : ZWART$ = CHR$(144) : WIT$ = CHR$(5)
20 INPUT "GEEF EEN GETAL"; V
30 IF V < 0 THEN PRINT ROOD$; : ELSE PRINT ZWART$;
40 PRINT V                                  : REM PRINT NEGATIEVE GETALLEN IN HET ROOD
50 PRINT WIT$
60 INPUT "PROGRAMMA STOPPEN (J/N)"; A$
70 IF A$ = "J" THEN END
80 IF A$ = "N" THEN 20 : ELSE 60
```

```
 10 X=1
 20 IF X=0 THEN BEGIN
 30   PRINT "NUL"
 40 BEND:ELSE BEGIN
 50   IF X=1 THEN BEGIN
 60     PRINT "EEN"
 70   BEND:ELSE BEGIN
 80     IF X=2 THEN BEGIN
 90       PRINT "TWEE"
100     BEND:ELSE PRINT "ANDERS"
110   BEND
120 BEND
```

IMPORT

Token: $DD

Formaat: **IMPORT** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Laadt BASIC-code in tekstformaat van een bestand van het type **SEQ** in het geheugen dat voor BASIC-programma's is gereserveerd.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Het programma wordt in het BASIC-geheugen geladen en van tekst omgezet naar de getokeniseerde vorm van **PRG**-bestanden. Hiermee kunnen BASIC-programma's worden geladen die als platte tekst als programmalijst zijn opgeslagen.

Na het laden wordt het programma opnieuw gelinkt en is het gereed om te worden **RUN** of bewerkt. Het is mogelijk **IMPORT** te gebruiken om een programmatekstbestand van de schijf samen te voegen met een programma dat zich al in het geheugen bevindt. Elke regel die uit het bestand wordt gelezen, wordt op dezelfde manier verwerkt alsof de gebruiker deze via de schermeditor zou hebben ingevoerd.

Er is geen **EXPORT**-tegenhanger, omdat deze functie al beschikbaar is. De volgorde `DOPEN#1,"LISTING",W:CMD 1:LIST:DCLOSE#1` converteert het programma in het geheugen naar tekst en schrijft het naar het bestand dat in de **DOPEN**-instructie is opgegeven.

IMPORT kan alleen in de directe modus worden gebruikt.

Voorbeelden:

Gebruik van **IMPORT**

```
IMPORT "APOCALYPS"

IMPORT "MEGA-TOOLTJES", U9

IMPORT (FI$), U(UN%)
```

INFO

Token: $FE $4D

Formaat: **INFO**

Gebruik: Geeft informatie over de uitvoeromgeving.

Opmerkingen:

Het commando **INFO** geeft informatie over de BASIC-uitvoeromgeving, waaronder:

- De videomodus (PAL, NTSC).
- De versie van de ROM.
- De CPU-snelheid.
- De huidige **MEM**-instelling.
- Het gebruikte geheugen en het beschikbare geheugen voor programmatekst en variabelen.

Voorbeeld: Gebruik van **INFO**

```
READY.
INFO

INFO: PAL     DATA BYTES  USED /  FREE
---------------------------------------
ROM-V   920408   PROGRAM:    7 / 55030
SPEED    40 MHZ  SCALARS:    0 /  1472
BANK4 --------   STRINGS:    0 / 54980
BANK5 --------   ARRAYS :    0 / 54980
```

INPUT

Token: $85

Formaat: **INPUT** [prompt <**,** | **;**>] variabele [**,** variabele ...]

Gebruik: Vraagt de gebruiker om toetsenbordinvoer door een optionele prompttekst en vraagteken op het scherm weer te geven.

prompt optionele tekenreeksexpressie die als prompt wordt afgedrukt.

Indien het scheidingsteken tussen **prompt** en **variabelenlijst** een komma (**,**) is, wordt de cursor direct na de prompt geplaatst. Indien het scheidingsteken een puntkomma (**;**) is, worden in plaats daarvan een vraagteken en een spatie aan de prompt toegevoegd.

variabelenlijst een of meer variabelen die de invoer ontvangen.

De invoer wordt verwerkt nadat de gebruiker RETURN indrukt.

Opmerkingen:

De gebruiker moet ervoor zorgen dat het juiste type invoer wordt ingevoerd, zodat deze overeenkomt met de typen van de **variabelenlijst**. Het aantal invoerelementen moet eveneens overeenkomen met het aantal variabelen. Een overschot aan invoerelementen wordt genegeerd, terwijl te weinig invoerelementen een nieuwe vraag om invoer uitlokt met de prompt ??.

Het invoeren van niet-numerieke tekens voor variabelen van het type geheel getal of reëel getal levert een Type Mismatch-fout op.

Strings voor stringvariabelen moeten tussen dubbele aanhalingstekens (**"**) staan indien ze spaties of komma's bevatten.

Veel programma's die een veilige invoerroutine nodig hebben, gebruiken **LINE INPUT** en een aangepaste parser om programmafouten door verkeerde invoer van de gebruiker te voorkomen.

Voorbeeld: Gebruik van **INPUT**

```
 10 DIM N$(100), A%(100), S$(100)
 20 DO
 30 INPUT "NAAM, LEEFTIJD, GESLACHT";NA$, AG%, SE$
 40 IF NA$ = "" THEN 30
 50 IF NA$ = "EIND" THEN EXIT
 60 IF AG% < 18 OR AG% > 100 THEN PRINT "LEEFTIJD?" : GOTO 30
 70 IF SE$ <> "M" AND SE$ <> "V" THEN PRINT "GESLACHT?" : GOTO 30
 80 REM CHECK OK, OPSLAAN IN ARRAY
 90 N$(N) = NA$ : A%(N) = AG% : S$(N) = SE$ : N = N + 1
100 LOOP UNTIL N = 100
110 PRINT "N;" NAMEN ONTVANGEN"
```

INPUT#

Token: $84

Formaat: **INPUT#** kanaal**,** variabele [**,** variabele ...]

Gebruik: Leest een record van een invoerapparaat, bijv. een schijfbestand, en wijst de gegevens toe aan de variabelen in de lijst.

kanaal nummer, dat werd opgegeven bij een eerdere aanroep van commando's zoals **DOPEN** of **OPEN**.

variabelenlijst een of meer variabelen die de invoer ontvangen.

Het invoerrecord moet worden afgesloten met een RETURN-teken en mag niet langer zijn dan de invoerbuffer (160 tekens).

Opmerkingen:

Het type en het aantal gegevens in een record moeten overeenkomen met de variabelenlijst. Het lezen van niet-numerieke tekens voor variabelen van het type geheel getal of reëel getal levert een File Data-fout op. Strings voor stringvariabelen moeten tussen dubbele aanhalingstekens staan indien ze spaties of komma's bevatten.

LINE INPUT# kan worden gebruikt om een volledig record in één stringvariabele te lezen.

Volgordelijke bestanden die met **INPUT#** kunnen worden gelezen, kunnen door programma's met **PRINT#** of met de editor worden aangemaakt.

Voorbeeld: Gebruik van **INPUT#**

```
READY.
EDIT ON                                    : REM BESTAND AANMAKEN

OK.
10 "CHUCK PEDDLE", 1937, "ONTWERPER VAN DE 6502"
20 "JACK TRAMIEL", 1928, "OPRICHTER VAN CBM"
30 "BILL MENSCH", 1945, "HARDWARE"

DSAVE "CBM-PERSONEN"

OK.
EDIT OFF

READY.
```

Lees bestand

```
 10 DIM N$(100), B%(100), S$(100)
 20 DOPEN#2, "CBM-PERSONEN"       : REM OPEN SEQ-BESTAND
 25 IF DS THEN PRINT DS$ : STOP   : REM OPENFOUT
 30 FOR I = 0 TO 100
 40 INPUT#2, N$(I), B%(I), S$(I)
 50 IF ST AND 64 THEN 80          : REM END OF FILE
 60 IF DS THEN PRINT DS$ : GOTO 80 : REM DISKFOUT
 70 NEXT I
 80 DCLOSE#2
110 PRINT "I + 1; "RECORDS GELEZEN"
120 FOR J = 0 TO I : PRINT N$(J) : NEXT J

RUN
3 RECORDS GELEZEN
CHUCK PEDDLE
JACK TRAMIEL
BILL MENSCH

TYPE "CBM-PERSONEN"
"CHUCK PEDDLE", 1937, "ONTWERPER VAN DE 6502"
"JACK TRAMIEL", 1928, "OPRICHTER VAN CBM"
"BILL MENSCH", 1945, "HARDWARE"
```

INSTR

Token: $D4

Formaat: **INSTR**(hooiberg**,** naald [**,** begin])

Gebruik: Zoekt de positie van de tekenreeksexpressie **naald** in de tekenreeksexpressie **hooiberg** en retourneert de index van het eerste voorkomen, of nul indien er geen overeenkomst is.

De tekenreeksexpressie **hooiberg** wordt doorzocht op het voorkomen van de tekenreeksexpressie **naald**.

Een verbeterde versie van tekenreekszoeken met patroonherkenning wordt gebruikt indien het eerste teken van de zoekreeks een '£' (pondteken) is. Het pondteken maakt geen deel uit van de zoekopdracht, maar maakt het mogelijk het '.' (punt) als jokerteken te gebruiken, dat overeenkomt met elk willekeurig teken. Het tweede speciale patroonteken is het '*' (asterisk) teken. De asterisk in de zoekreeks geeft aan dat het voorgaande teken nooit, eenmaal of herhaaldelijk mag voorkomen om als overeenkomst te worden beschouwd.

Het optionele argument **begin** is een integerexpressie die de startpositie voor de zoekopdracht in **hooiberg** bepaalt. Indien het ontbreekt, wordt standaard één gebruikt.

Opmerkingen:

Indien een van de tekenreeksen leeg is of er geen overeenkomst is, retourneert de functie nul.

Voorbeelden:

Gebruik van **INSTR**

```
I = INSTR("ABCDEF", "CD")    : REM I = 3
I = INSTR("ABCDEF", "XY")    : REM I = 0
I = INSTR("RAIIIN", "£A*IN") : REM I = 5
I = INSTR("ABCDEF", "£C.E")  : REM I = 3
I = INSTR(A$ + B$, C$)
```

INT

Token: $B5

Formaat: **INT(**numerieke expressie**)**

Retourneert:

Het gehele getal van een getal.

Dit wordt berekend als de "floor" van het getal, waarbij het decimale deel naar negatieve oneindigheid wordt afgerond. `INT(3.1) = 3`; `INT(-3.1) = -4`.

Deze functie is *niet* beperkt tot het gebruikelijke 16-bit integerbereik (-32768 tot 32767), omdat reële rekenkunde wordt gebruikt. Het toegestane bereik wordt daarom bepaald door de grootte van de reële mantissa, die 32 bits breed is (-2147483648 tot 2147483647).

Opmerkingen:

Het is niet nodig de functie **INT** te gebruiken om reële waarden toe te wijzen aan integer- of bytevariabelen, omdat deze conversie impliciet wordt uitgevoerd.

Voorbeelden:

Gebruik van **INT**

```
X = INT(1.9)      : REM X = 1
X = INT(-3.1)     : REM X = -4
X = INT(100000.5) : REM X = 100000
N% = INT(100000.5) : REM ?ILLEGAL QUANTITY ERROR
```

JOY

Token: $CF

Formaat: **JOY**(poort)

Retourneert:

De stand van de joystick voor de geselecteerde controller**poort** (1 of 2).

Een **poort**-waarde van 3 retourneert de samengevoegde stand van beide joystickpoorten, zodat een enkele joystick op willekeurige poort gebruikt kan worden.

De stok kan in acht richtingen bewogen worden, welke met de klok mee genummerd worden vanaf de bovenste stand.

	Links	**Midden**	**Rechts**
Omhoog	8	1	2
Midden	7	0	3
Omlaag	6	5	4

JOY() retourneert de stokrichting, plus een getal voor elke ingedrukte knop:

Knop	**Waarde**	**Functie**
1	128	A (trekker)
2	256	B
3	512	C of Start
4	1024	Start
5	2048	Select

Opmerkingen:

De retourwaarde van **poort** 3, wanneer twee joysticks zijn aangesloten en tegelijkertijd bewogen worden, is niet gedefinieerd.

Het overgrote deel van de Commodore- en Atari-joysticks heeft één vuurknop. De C64GS-controller en Atari CX78-gamepad hebben elk twee actieknoppen. Sommige moderne Commodore-compatibele gamecontrollers en adapters ondersteunen drie knoppen en kunnen de derde knop toewijzen aan een actieknop of een knop met het label "Start".

Het `5plusbuttonsJoystick`-protocol van `crystalct` specificeert een manier voor Commodore-programmatuur om tot vijf knoppen uit te lezen. Knoppen 4 en 5 hergebruiken de signalen van de richtingstok, waardoor de richtingstok mogelijk niet werkt terwijl deze knoppen worden ingedrukt. Een typische vijfknopscontroller wijst deze knoppen toe aan "Start" en "Select".

Het drieknoppenprotocol gebruikt de paddle-signalen van de joystickpoort voor knoppen 2 en 3. **JOY()** kan onverwachte waarden retourneren

wanneer paddles zijn aangesloten. Gebruik **POT()** om paddle-controllers uit te lezen.

De waarden voor elk van de vijf knoppen zijn afzonderlijke bitwaarden. Een programma kan de **AND**-operator gebruiken om elk bit afzonderlijk te testen, los van de richtingstok en de andere knoppen. De expressie `JOY(2) AND 128` is bijvoorbeeld waar wanneer knop 1 van de joystick in poort 2 wordt ingedrukt, ongeacht de stand van de andere knoppen.

Voorbeeld: Gebruik van **JOY**

```
 10 N = JOY(1)
 20 IF N AND 128 THEN PRINT "VUUR!"
 30 REM              N    NO   O    ZO   Z    ZW   W    NW
 40 ON N AND 15 GOSUB 100, 200, 300, 400, 500, 600, 700, 800
 50 GOTO 10
100 PRINT "GA NOORD"     : RETURN
200 PRINT "GA NOORDOOST" : RETURN
300 PRINT "GA OOST"      : RETURN
400 PRINT "GA ZUIDOOST"  : RETURN
500 PRINT "GA ZUID"      : RETURN
600 PRINT "GA ZUIDWEST"  : RETURN
700 PRINT "GA WEST"      : RETURN
800 PRINT "GA NOORDWEST" : RETURN
```

KEY

Token: $F9

Formaat: **KEY**
KEY <**ON** | **OFF**>
KEY <**LOAD** | **SAVE**> bestandsnaam
KEY getal**,** tekenreeks
KEY RESTORE

Gebruik: Beheert de functietoetsmacro's in de BASIC-editor.

Elke functietoets kan een tekenreeks toegewezen krijgen die wordt ingetypt wanneer de toets wordt ingedrukt. De functietoetsen hebben standaardtoewijzingen bij het opstarten, welke met het **KEY**-commando gewijzigd kunnen worden.

KEY : toont de huidige toewijzingen.

KEY ON : schakelt de functietoetsteksten in. De toetsen verzenden de toegewezen tekenreeksen wanneer ze worden ingedrukt.

KEY OFF : schakelt de functietoetsteksten uit. De toetsen verzenden hun tekencode wanneer ze worden ingedrukt.

KEY LOAD bestandsnaam : laadt toetstoewijzingen uit een bestand.

KEY SAVE bestandsnaam : slaat toetstoewijzingen op in een bestand.

KEY getal**,** tekenreeks : wijst de tekenreeks toe aan de toets met het opgegeven nummer.

nummer kan elke waarde binnen dit bereik zijn:

- 1 - 14: Komt overeen met toetsen van F1 tot en met F14
- 15: Komt overeen met HELP
- 16: Komt overeen met SHIFT RUN STOP

KEY RESTORE : herstelt alle functietoetsmacro's naar hun standaardtoewijzingen.

Standaardtoewijzingen:

```
KEY
KEY 1,CHR$(27)+"X"
KEY 2,CHR$(27)+"C"
KEY 3,"DIR"+CHR$(13)
KEY 4,"DIR "+CHR$(34)+"*=PRG"+CHR$(34)+CHR$(13)
KEY 5,"LI"
KEY 6,"KEY6"+CHR$(141)
KEY 7,"LL"
KEY 8,"MONITOR"+CHR$(13)
KEY 9,"LP"
KEY 10,"KEY10"+CHR$(141)
KEY 11,"LU"
KEY 12,"KEY12"+CHR$(141)
KEY 13,CHR$(27)+"O"
KEY 14,"LU"+CHR$(27)+"O"
KEY 15,"HELP"+CHR$(13)
KEY 16,"RUN "+CHR$(34)+"*"+CHR$(34)+CHR$(13)
```

Opmerkingen:

De som van de lengtes van alle toegewezen tekenreeksen mag maximaal 240 tekens bedragen. Speciale tekens zoals RETURN of QUOTE worden ingevoerd met hun codes via de **CHR$**-functie. Zie **CHR$** op bladzijde 52 voor meer informatie.

Voorbeelden:

Gebruik van **KEY**

```
KEY ON                    : REM FUNCTIETOETSEN AAN
KEY OFF                   : REM FUNCTIOTOETSEN UIT
KEY                       : REM TOON TOEWIJZING
KEY 2, "PRINT π" + CHR$(14) : REM WIJS PRINT PI AAN F2 TOE
KEY SAVE "MIJN TOETSSET"  : REM BEWAAR HUIDIGE DEFINITIES IN BESTAND
KEY LOAD "SET ELF"        : REM LAAD DEFINITIES UIT BESTAND
KEY RESTORE
```

LEFT$

Token: $C8

Formaat: **LEFT$**(tekenreeks**,** n)

Retourneert:

Een tekenreeks met de eerste **n** tekens van het argument **tekenreeks**.

Indien de lengte van **tekenreeks** gelijk aan of kleiner dan **n** is, zal de resulterende tekenreeks identiek zijn aan het argument.

tekenreeks de tekenreeksexpressie.

n numerieke expressie (0 - 255).

Opmerkingen:

Lege tekenreeksen en lengte nul zijn toegestane waarden.

Voorbeeld: Gebruik van **LEFT$**

```
PRINT LEFT$("MEGA65", 4)
MEGA
```

LEN

Token: $C3

Formaat: **LEN**(tekenreeks)

Retourneert:
De lengte van een tekenreeks.

tekenreeks de tekenreeksexpressie.

Opmerkingen:
BASIC-tekenreeksen kunnen elk teken bevatten, inclusief het nulteken. De lengte van een tekenreeks wordt intern opgeslagen in een tekenreeks-descriptor.

Voorbeeld: Gebruik van **LEN**

```
PRINT LEN("MEGA65" + CHR$(13))
 7
```

LET

Token: $88

Formaat: [**LET**] variabele = expressie

Gebruik: Wijst waarden (of resultaten van expressies) toe aan variabelen.

Opmerkingen:

Het **LET**-statement is overbodig en niet vereist. Toewijzing aan variabelen kan zonder **LET** gedaan worden, maar het is in BASIC 65 behouden gebleven voor achterwaartse compatibiliteit.

Voorbeelden:

Gebruik van **LET**

```
LET A = 5 : REM LANGER EN TRAGER

A = 5     : REM KORTER EN SNELLER
```

LINE

Token: $E5

Formaat: **LINE** x-beg**,** y-beg[**,** x-volgend1**,** y-volgend1 ...]

Gebruik: Bitmapgrafiek: tekent een lijn of een reeks lijnen.

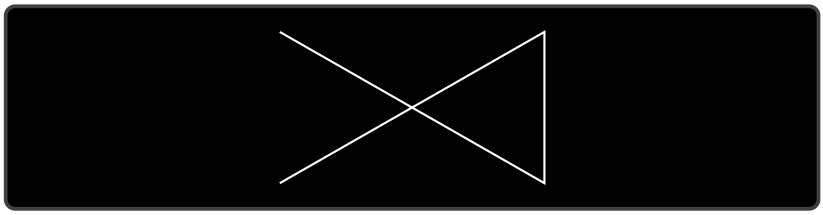
Indien slechts één coördinatenpaar wordt opgegeven, tekent **LINE** een punt.

Indien meer dan één paar wordt gedefinieerd, wordt een lijn getekend op het huidige grafische scherm vanaf de coördinaat (**x-beg**, **y-beg**) naar de volgende coördinatenpaar(en).

Alle momenteel gedefinieerde modes en waarden van de grafische context worden gebruikt.

Voorbeeld: Gebruik van **LINE**

```
10 SCREEN 320, 200, 2                    : REM SCHERM 320 X 200 X 2
20 PEN 1                                 : REM TEKENPEN KLEUR 1 (WIT)
30 LINE 25, 25, 295, 175, 295, 25, 25, 175 : REM TEKEN LIJNEN
40 GETKEY A$                             : REM WACHT OP TOETSDRUK
50 SCREEN CLOSE                          : REM SLUIT SCHERM EN HERSTEL PALET
```



LINE INPUT

Token: $E5 $85

Formaat: **LINE INPUT** [prompt <**,** | **;**>] tekenreeksvariabele [**,** tekenreeksvariabele ...]

Gebruik: Vraagt de gebruiker om invoer vanaf het toetsenbord, waarbij een optionele prompttekenreeks en vraagteken op het scherm worden afgedrukt.

prompt optionele tekenreeksexpressie die als prompt wordt afgedrukt.

Indien het scheidingsteken tussen **prompt** en de eerste **tekenreeksvariabele** een ',' (komma) is, wordt de cursor direct achter de prompt geplaatst. Indien het scheidingsteken een ';' (puntkomma) is, worden een vraagteken en een spatie aan de prompt toegevoegd.

tekenreeksvariabele een of meer tekenreeksvariabelen die elk één regel invoer accepteren.

Opmerkingen:

Dit verschilt van **INPUT** in de wijze waarop de invoer wordt verwerkt. **LINE INPUT** accepteert elk teken dat op een regel wordt ingetypt als een enkele tekenreekswaarde. Alleen de RETURN-toets produceert geen teken.

Indien de variabelenlijst meer dan één variabele bevat, zal **LINE INPUT** de gehele eerste regel gebruiken voor de eerste variabele en het ??-prompt tonen voor elke volgende variabele.

LINE INPUT werkt alleen met tekenreeksvariabelen. Indien een niet-tekenreeksvariabele wordt gebruikt, resulteert **LINE INPUT** in een Type Mismatch-fout nadat de gegevens zijn ingevoerd.

Voorbeeld: Gebruik van **LINE INPUT**

```
10 LINE INPUT "GEEF EEN ZIN IN: ", PH$
20 PRINT "DE ZIN DIE U INVOERDE:"; CHR$(13); "  "; PH$

RUN
VOER EEN ZIN IN: JIJ ZEGT "AARDAPPEL," IK ZEG "AARDAPPEL."
DE ZIN DIE U INVOERDE:
  JIJ ZEI "AARDAPPEL," IK ZEI "AARDAPPEL."
```

LINE INPUT#

Token: $E5 $84

Formaat: **LINE INPUT#** kanaal**,** variabele [**,** variabele ...]

Gebruik: Leest één record per variabele van een invoerapparaat (zoals een schijfstation) en wijst de gelezen gegevens toe aan de variabele. De records moeten afgesloten worden door een **RETURN**-teken, welke niet naar de tekenreeksvariabele gekopieerd wordt. Een lege regel die alleen uit het **RETURN**-teken bestaat, resulteert derhalve in een lege tekenreeks.

kanaal nummer, dat werd opgegeven bij een eerdere aanroep van commando's zoals **DOPEN** of **OPEN**.

variabelenlijst een of meer variabelen die de invoer ontvangen.

Opmerkingen:

In de variabelenlijst kunnen alleen tekenreeksvariabelen of tekenreeksarray-elementen worden gebruikt. In tegenstelling tot andere INPUT-commando's interpreteert of verwijdert **LINE INPUT#** geen aanhalingstekens in de invoer. Deze worden net als alle andere tekens als gegevens geaccepteerd. Records mogen niet langer zijn dan de invoerbuffer, welke 160 tekens bedraagt.

Voorbeeld: Gebruik van **LINE INPUT#**

```
 10 DIM N$(100)
 20 DOPEN#2, "DATA"
 30 FOR I = 0 TO 100
 40 LINE INPUT#2, N$(I)
 50 IF ST = 64 THEN 80           : REM EINDE BESTAND
 60 IF DS THEN PRINT DS$ : GOTO 80 : REM DISKFOUT
 70 NEXT I
 80 DCLOSE#2
110 PRINT "I; " RECORDS GELEZEN"
```

LIST

Token: $9B

Formaat: **LIST** [**P**] [regelbereik]
LIST [**P**] bestandsnaam [**,U** eenheid]

Gebruik: Toont een bereik van regels uit het BASIC-programma in het geheugen.

Bij een enkel regelnummer toont **LIST** die regel.

Bij een bereik van regelnummers toont **LIST** alle regels in dat bereik. Een bereik kan twee getallen zijn gescheiden door een '–' (koppelteken), of het begin of einde van het bereik kan weggelaten worden om het begin of einde van het programma impliciet te maken. Zie de voorbeelden hieronder.

Indien een bestandsnaam wordt opgegeven, wordt het bereik van regels direct uit een BASIC-programmabestand getoond.

Opmerkingen:

De optionele parameter **P** schakelt paginamodus in. Na het tonen van een schermvol regels stopt de lijst en wordt het [MORE]-prompt onderaan het scherm weergegeven. Door **Q** in te drukken wordt de paginamodus afgesloten; elke andere toets gaat verder naar de volgende pagina.

De uitvoer van **LIST** kan naar andere apparaten worden omgeleid via **CMD**.

Een andere manier om een programmalijst uit het geheugen op het scherm weer te geven, is het gebruik van de toetsen F9 en F11, of CTRL P en CTRL V, om een BASIC-lijst op het scherm omhoog of omlaag te scrollen.

Voorbeelden:

Gebruik van **LIST**

```
LIST 100      : REM LIST REGEL 100

LIST 240-350  : REM LIST ALLE REGELS VAN 240 TOT 350

LIST 500-     : REM LIST VAN 500 TOT EIND

LIST -70      : REM LIST VANAF BEGIN TOT 70

LIST "DEMO"   : REM LIST BESTAND "DEMO"

LIST P        : REM LIST PROGRAMMA GEPAGINEERD

LIST P "MURX" : REM LIST BESTAND "MURX" GEPAGINEERD
```

LOAD

Token: $93

Formaat: **LOAD** bestandsnaam [**,** eenheid [**,** vlag]]
LOAD "$[patroon=type]" [**, eenheid**]
LOAD "$$[patroon=type]" [**, eenheid**]
/ bestandsnaam [**,** eenheid [**,** vlag]]

Gebruik: De eerste vorm laadt een bestand van het type **PRG** in het geheugen dat gereserveerd is voor BASIC-programma's.

De tweede vorm laadt een map in het geheugen, welke vervolgens met **LIST** kan worden bekeken. De structuur lijkt op een BASIC-programma, doch in plaats van regelnummers worden bestandsgroottes weergegeven.

De derde vorm lijkt op de tweede, maar de bestanden krijgen opeenvolgende regelnummers. Deze lijstopgave kan als een BASIC-programma worden gescrold met de toetsen **F9** of **F11**, bewerkt, weergegeven, opgeslagen of afgedrukt.

Er kan een filter worden toegepast door een patroon of een patroon en een type op te geven. Het asterisk-teken (*) komt overeen met de rest van de bestandsnaam, terwijl het (?) overeenkomt met willekeurig één teken. De typespecificatie kan één van de tekens (P, S, U, R) zijn, respectievelijk programma-, sequentieel, gebruiker- of relatief bestand.

Een veelgebruikte toepassing van het afkortingsteken **/** is het snel laden van **PRG**-bestanden. Hiervoor:

1. Druk een schijfmap af met **DIR** of **CATALOG**.
2. Verplaats de cursor naar de gewenste regel.
3. Typ '/' in de eerste kolom van de regel en druk op RETURN.

Na het indrukken van RETURN wordt het bestand op de regel met het voorafgaande '/' geladen. Tekens voor en achter de bestandsnaam '"' (dubbele aanhalingstekens) worden genegeerd. Dit geldt uitsluitend voor **PRG**-bestanden.

bestandsnaam is een tekenreeks tussen aanhalingstekens, bijv. `"PROG"`, of een tekenreeksexpressie `("N$)`

Het nummer van de **eenheid** is optioneel. Indien dit niet wordt opgegeven, wordt de standaard schijfeenheid gebruikt.

Indien de **vlag** een ongelijkwaardige waarde heeft, wordt het bestand geladen op het adres dat uit de eerste twee bytes van het bestand

wordt gelezen. Anders wordt het geladen naar het begin van het BASIC-geheugen en wordt het laadadres in het bestand genegeerd.

Opmerkingen:

LOAD laadt bestanden van het type **PRG** in bank 0 van het RAM-geheugen, dat ook voor de broncode van BASIC-programma's wordt gebruikt.

LOAD "*" kan worden gebruikt om het eerste **PRG**-bestand van de opgegeven **eenheid** te laden.

LOAD "$" kan worden gebruikt om de bestandslijst van de opgegeven **eenheid** te laden. Bij gebruik van **LOAD "$"** kan **LIST** worden gebruikt om de lijst op het scherm weer te geven. Hierbij wordt eventueel reeds in het geheugen aanwezig programma overschreven.

LOAD is opgenomen in BASIC 65 voor achterwaartse compatibiliteit met BASIC V2.

Het afkortingsteken **/** kan uitsluitend in de directe modus worden gebruikt.

Standaard gebruikt de C64 **eenheid** 1, welke is toegewezen aan datasette-bandrecorders op de cassettepoort. De MEGA65 gebruikt standaard **eenheid** 8, welke is toegewezen aan het interne schijfstation. Hierdoor is het niet nodig om ,8 toe te voegen aan **LOAD**-commando's die hiervan gebruikmaken.

Voorbeelden:

Gebruik van **LOAD**

```
LOAD "APOCALYPS"    : REM LAAD EEN BESTAND GEHETEN APOCALYPS IN BASICGEHEUGEN

LOAD "MEGA TOOLS",9 : REM LAAD EEN BESTAND "MEGA TOOLS" VANAF UNIT 9 IN BASICGEHEUGEN

LOAD "*",8,1        : LAAD EERSTE BESTAND OP UNIT 8 IN RAM ZOALS IN BESTAND AANGEGEVEN

LOAD "$"            : REM LAAD HELE DIRECTORY - MET BESTANDSGROOTTES

LOAD "$$"           : REM LAAD HELE DIRECTORY - SCROLLBAAR

LOAD "$$X*=P"       : REM DIRECTORY, MET PRG-BESTANDEN BEGINNEND OP 'X'
```

LOADIFF

Token: $FE $43

Formaat: **LOADIFF** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Bitmapgraphics: laadt een IFF-bestand in het grafisch geheugen.

IFF (Interchange File Format) wordt ondersteund door veel verschillende toepassingen en besturingssystemen. **LOADIFF** gaat ervan uit dat de bestanden bitplane-graphics bevatten die overeenkomen met het actieve grafische scherm qua resolutie en kleurdiepte.

Ondersteunde resoluties zijn:

Breedte	Hoogte	Bitplanes	Kleuren	Geheugen
320	200	max. 8	max. 256	max. 64KB
640	200	max. 8	max. 256	max. 128KB
320	400	max. 8	max. 256	max. 128KB
640	400	max. 4	max. 16	max. 128KB

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Er zijn tools beschikbaar om populaire afbeeldingsformaten naar IFF te converteren. Deze tools zijn beschikbaar op diverse besturingssystemen, zoals AMIGA OS, macOS, Linux en Windows. Bijvoorbeeld **ImageMagick** is een gratis grafisch pakket dat een tool genaamd **convert** bevat, welke in combinatie met de **ppmtoilbm**-tool uit het **Netbpm**-pakket kan worden gebruikt om IFF-bestanden aan te maken.

Om `convert` en `ppmtoilbm` te gebruiken voor het converteren van een JPG-bestand naar een IFF-bestand op Linux:

```
convert <myImage.jpg> <myImage.ppm>

ppmtoilbm -aga <myImage.ppm> > <myImage.iff>
```

Voorbeeld: Gebruik van **LOADIFF**

```
100 BANK 128 : SCNCLR
110 REM TOON PLAATJES IN 320 X 200 X 7 RESOLUTIE
120 GRAPHIC CLR : SCREEN DEF 0, 0, 0, 7 : SCREEN OPEN 0 : SCREEN SET 0, 0
130 FOR I = 1 TO 7 : READF$
140 LOADIFF(F$+".IFF") : SLEEP 4 : NEXT
150 DATA ALIEN, BEAKER, JOKER, PICARD, PULP, TROOPER, RIPLEY
160 SCREEN CLOSE 0
170 PALETTE RESTORE
```

LOCK

Token: $FE $50

Formaat: **LOCK** <bestandsnaam|patroon> [**,D** station] [**,U** eenheid]

Gebruik: Vergrendelt een bestand op de schijf, waardoor het niet kan worden bijgewerkt of verwijderd.

Het opgegeven bestand of de set bestanden die aan het patroon voldoen, wordt vergrendeld en kan niet worden verwijderd met de commando's **DELETE**, **ERASE** of **SCRATCH**.

Het commando **UNLOCK** verwijdert de vergrendeling.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

In de directe modus wordt het aantal vergrendelde bestanden weergegeven. Het op-een-na laatste getal in het bericht bevat het aantal vergrendelde bestanden.

Voorbeelden:

Gebruik van **LOCK**

```
LOCK "DRM", U9        : REM VERGRENDEL BESTAND DRM OP UNIT 9
03,FILES LOCKED,01,00

LOCK "BS*"            : REM VERGENDEL BESTANDEN BEGINNEND OP "BS"
03,FILES LOCKED,04,00
```

LOG

Token: $BC

Formaat: **LOG(**numerieke expressie**)**

Retourneert:

De natuurlijke logaritme van een getal.

De natuurlijke logaritme gebruikt het getal van Euler (2.71828183) als grondtal. Op een wetenschappelijke rekenmachine wordt deze functie doorgaans aangeduid als "LN".

Opmerkingen:

Voor het gemak bevat BASIC 65 afzonderlijke functies voor logaritmen met grondtal 10 en grondtal 2: respectievelijk **LOG10()** en **LOG2()**.

Om logaritmen met andere grondtallen te berekenen, gebruik dan de volgende logaritmische identiteit:

$\log_b(n) =$ `LOG(n) / LOG(b)`

Voorbeelden:

Gebruik van **LOG**

```
PRINT LOG(1)
 0

PRINT LOG(0)
?ILLEGAL QUANTITY ERROR

PRINT LOG(4)
 1.3862944

PRINT LOG(100) / LOG(10)
 2
```

LOG10

Token: $CE $08

Formaat: **LOG10**(numerieke expressie)

Retourneert:

De decimale logaritme van de numerieke expressie.

De decimale logaritme gebruikt 10 als grondtal.

Voorbeelden:

Gebruik van **LOG10**

```
PRINT LOG10(1)
 0

PRINT LOG10(0)
?ILLEGAL QUANTITY ERROR

PRINT LOG10(5)
 0.69897001

PRINT LOG10(100); LOG10(10); LOG10(1); LOG10(0.1); LOG10(0.01)
 2  1  0 -1 -2
```

LOG2

Token: $CE $16

Formaat: **LOG2**(numerieke expressie)

Retourneert:

De binaire logaritme van de numerieke expressie.

De binaire logaritme gebruikt 2 als grondtal.

Voorbeelden:

Gebruik van **LOG2**

```
PRINT LOG2(1)
 0

PRINT LOG2(0)
?ILLEGAL QUANTITY ERROR

PRINT LOG2(5)
 2.3219281

PRINT LOG2(4); LOG2(2); LOG2(1); LOG2(0.5); LOG2(0.25)
 2  1  0 -1 -2
```

LOOP

Token: $EC

Formaat: **DO** ... **LOOP**
DO [<**UNTIL** | **WHILE**> logische expressie] ... opdrachten [**EXIT**]
LOOP [<**UNTIL** | **WHILE**> logische expressie]

Gebruik: **DO** ... **LOOP** definieert het bereik van een BASIC-lus. Het gebruik van **DO** ... **LOOP** zonder modifiers creëert een oneindige lus, welke alleen kan worden afgebroken met de **EXIT**-instructie. De lus kan worden beheerd door **UNTIL** of **WHILE** toe te voegen na **DO** of **LOOP**.

Opmerkingen:
DO-lussen mogen genest worden. Een instructie **EXIT** sluit alleen de huidige lus af.

Voorbeelden:
Gebruik van **DO** ... **LOOP**

```
10 PW$ = "" : DO
20 GET A$ : PW$ = PW$ + A$
30 LOOP UNTIL LEN(PW$) > 7 OR A$ = CHR$(13)

10 DO : REM WACHT OP GEBRUIKERSBESLISSING
20 GET A$
30 LOOP UNTIL A$ = "J" OR A$ = "N"

10 DO WHILE ABS(EPS) > 0.001
20 GOSUB 2000               : REM ITERATIESUBROUTINE
30 LOOP

10 I% = 0                   : REM INTEGERLUS 1-100
20 DO : I% = I% + 1
30 LOOP WHILE I% < 101
```

LPEN

Token: $CE $04

Formaat: **LPEN**(coördinaat)

Retourneert:

De status van een lightpen-periferie.

Deze functie vereist een CRT-beeldbuis (of televisie) en een lightpen. Ze werkt niet met een LCD- of LED-scherm. De lightpen moet op poort 1 worden aangesloten.

LPEN(0) retourneert de X-positie van de lightpen, in het bereik 60 - 320.

LPEN(1) retourneert de Y-positie van de lightpen, in het bereik 50 - 250.

Opmerkingen:

De X-resolutie bedraagt twee pixels, waardoor **LPEN(0)** uitsluitend even getallen retourneert.

Er is een felle achtergrondkleur nodig om de lightpen te activeren. De **COLLISION**-instructie kan worden gebruikt om een interrupt-handler in te schakelen.

Voorbeeld: Gebruik van **LPEN**

```
PRINT LPEN(0), LPEN(1) : REM PRINT LICHTPENCOORDINATEN
 100       200
```

MEM

Token: $FE $23

Formaat: **MEM** masker4**,** masker5

Gebruik: Reserveert geheugen in bank 4 of 5, zodat het bitmapgrafieksysteem dit niet zal gebruiken.

masker4 en **masker5** zijn byte-waarden, geïnterpreteerd als een bitveld. Elke bit die op 1 staat, vertegenwoordigt een segment van 8 KB geheugen dat is gereserveerd voor programma-gebruik. Het eerste argument beschrijft bank 4 en het tweede argument beschrijft bank 5.

bit	geheugensegment
0	$0000 - $1FFF
1	$2000 - $3FFF
2	$4000 - $5FFF
3	$6000 - $7FFF
4	$8000 - $9FFF
5	$A000 - $BFFF
6	$C000 - $DFFF
7	$E000 - $FFFF

Opmerkingen:

Na het reserveren van geheugen met **MEM** zal de grafische bibliotheek de gereserveerde gebieden niet gebruiken, zodat deze voor andere doeleinden kunnen worden aangewend. Toegang tot bank 4 en 5 is mogelijk met de commando's **PEEK, WPEEK, POKE, WPOKE** en **EDMA**.

Indien een grafisch scherm niet kan worden geopend omdat het resterende geheugen ontoereikend is, geeft het **SCREEN**-commando een "Out of Memory"-fout.

Enkele commando's in de directe modus, zoals **RENUMBER**, maken gebruik van geheugen in bank 4 en 5 en respecteren de **MEM**-reserveringen niet. **MEM**-reserveringen worden uitsluitend gegarandeerd tijdens de uitvoering van programma's.

Wanneer de 80 × 50-tekstmodus is ingeschakeld, wordt segment 0 van bank 4 automatisch gereserveerd en gebruikt voor schermgegevens. Het gebruikt altijd segment 0, ook indien dit voorheen met **MEM** of een grafisch scherm was gereserveerd. Indien uw programma gebruikmaakt van de 80 × 50-tekstmodus, gebruik segment 0 van bank 4 dan niet voor andere doeleinden.

Om geheugen dat door **MEM** is gereserveerd weer vrij te geven, roep **MEM** opnieuw aan met alle bits op 0. Om geheugen dat door **SCREEN** is gereserveerd vrij te geven, sluit het scherm. Zie **SCREEN CLOSE**. Om

geheugen dat door de 80 × 50-tekstmodus is gereserveerd vrij te geven, schakel over naar een van de andere tekstmodi.

FRE(*4*) en **FRE(*5*)** retourneren bitvelden die het geheugen weergeven dat respectievelijk in bank 4 en 5 is gereserveerd door **MEM**, **SCREEN** en de 80 × 50-tekstmodus tezamen. Zie **FRE()**.

Voorbeeld: Gebruik van **MEM**

```
10 MEM 1, 3                : REM RESERVEER $40000 - $41FFF EN $50000 - $53FFF
20 SCREEN 320, 200         : REM SCREEN ZAL GEEN GERESERVEERDE SEGMENTEN GEBRUIKEN
40 EDMA 3, $2000, 0, $4000 : REM VUL SEGMENT MET NULLEN
```

MERGE

Token: $E6

Formaat: **MERGE** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Laadt een BASIC-programmabestand vanaf de schijf en voegt dit toe aan het programma in het geheugen.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Het laadadres dat in de eerste twee bytes van het bestand is opgeslagen, wordt genegeerd. Het geladen programma vervangt niet het programma in het geheugen (wat **DLOAD** wel doet), maar wordt eraan toegevoegd. Na het laden wordt het programma opnieuw gelinkt en is het gereed om te worden uitgevoerd of bewerkt.

Het is de verantwoordelijkheid van de gebruiker om te waarborgen dat er geen conflicten met regelnummers bestaan tussen het programma in het geheugen en het samengevoegde programma. Het eerste regelnummer van het samengevoegde programma moet groter zijn dan het laatste regelnummer van het programma in het geheugen.

MERGE kan uitsluitend in de directe modus worden gebruikt.

Voorbeeld: Gebruik van **MERGE**

```
DLOAD "HOOFDPROGRAMMA"
MERGE "BIBLIOTHEEK"
```

MID$

Token: $CA

Formaat: **MID$**(tekenreeks**,** index**,** n)
MID$(tekenreeksvariabele**,** index**,** n**)** = tekenreeksexpressie

Gebruik: Als functie: geeft een deeltekenreeks van een string terug. Als instructie: vervangt een deeltekenreeks van een tekenreeksvariabele door een andere string.

tekenreeks de tekenreeksexpressie.

index de startindex (1 - 255).

n de lengte van de deeltekenreeks (0 - 255).

Opmerkingen:
Lege tekenreeksen en lengtes van nul zijn toegestane waarden.

Voorbeeld: Gebruik van **MID$**

```
10 A$ = "MEGA65"
20 PRINT MID$(A$, 3, 4)
30 MID$(A$, 4, 1) = "0"
40 PRINT A$

RUN
GA65
MEG065
```

MKDIR

Token: $FE $51

Formaat: **MKDIR** mapnaam **,L** grootte [**,U** eenheid]

Gebruik: Maakt een submap aan op een diskette of D81-schijfimage.

mapnaam de naam van een map. Ofwel een aangehaalde tekenreeks zoals "EENOFANDEREMAP", of een tekenreeksexpressie in haakjes zoals: (DR$)

MKDIR kan alleen worden gebruikt op stations die door CBDOS worden beheerd. Dit betreft het interne diskettestation en SD-kaartafbeeldingen van het type D81. Het commando kan niet worden gebruikt op externe stations die op de seriële IEC-bus zijn aangesloten.

De parameter **grootte** geeft het aantal sporen op dat voor de submap worden gereserveerd, waarbij één spoor bestaat uit 40 sectoren van 256 bytes. Het eerste spoor van het gereserveerde bereik wordt als mappenspoor voor de submap gebruikt.

De minimale grootte is 3 sporen, de maximale 38 sporen. Er moet een aaneengesloten gebied van lege sporen op de diskette (of D81-afbeelding) aanwezig zijn dat groot genoeg is voor het aanmaken van de submap. Indien een dergelijk gebied niet beschikbaar is, wordt een Disk Full-fout gemeld.

Er kunnen meerdere submappen worden aangemaakt, mits er voldoende lege sporen beschikbaar zijn.

Na het succesvol aanmaken van de submap vindt er automatisch een **CHDIR** naar deze submap plaats.

CHDIR "/" wisselt terug naar de hoofdmap.

Voorbeeld: Gebruik van **MKDIR**

```
MKDIR "SUBDIR", L5 : REM MAAK SUBDIRECTORY MET 5 TRACKS
DIR
0 "SUBDIR          "   1D
160 BLOCKS FREE.
```

MOD

Token: $CE $0B

Formaat: **MOD**(teller, deler)

Retourneert:
De rest van een delingsbewerking.

Opmerkingen:
In andere programmeertalen zoals C is deze functie geïmplementeerd als operator (%). In BASIC is zij als functie geïmplementeerd.

Voorbeeld: Gebruik van **MOD**

```
FOR I = 0 TO 8 : PRINT MOD(I, 4); : NEXT I
 0  1  2  3  0  1  2  3  0
```

MONITOR

Token: $FA

Formaat: **MONITOR**

Gebruik: Roept de KERNAL-machinetaalmonitor aan.

Opmerkingen:

Het gebruik van de **MONITOR** vereist kennis van de CSG4510 / 6502 / 6510-processors, de assembleertaal die zij gebruiken, en hun architecturen. Meer informatie over de **MONITOR** is beschikbaar in *het grote Mega65-boek*, Machinetaalmonitor (Bijlage O).

Om de monitor te verlaten, druk op **X**.

Hulptekst kan worden weergegeven met **?** of **H**.

Voorbeeld: Gebruik van **MONITOR**

```
MONITOR
```

```
BS MONITOR COMMANDS:ABCDEFGHJMR!UXG.,);?$+&%'LSV
    PC   SR AC XR YR ZR BP  SP  NVEBDIZC
; 00FFA2 00 00 00 00 00 00 01F8 --------
ASSEMBLE     - A ADDRESS MNEMONIC OPERAND
BITMAPS      - B [FROM]
COMPARE      - C FROM TO WITH
DISASSEMBLE  - D [FROM [TO]]
FILL         - F FROM TO FILLBYTE
GO           - G [ADDRESS]
HUNT         - H FROM TO (STRING OR BYTES)
JSR          - J ADDRESS
LOAD         - L FILENAME [UNIT [ADDRESS]]
MEMORY       - M [FROM [TO]]
REGISTERS    - R
SAVE         - S FILENAME UNIT FROM TO
TRANSFER     - T FROM TO TARGET
VERIFY       - V FILENAME [UNIT [ADDRESS]]
EXIT         - X
.<DOT>       - . ADDRESS MNEMONIC OPERAND
)<GREATER>   - ) ADDRESS BYTE SEQUENCE
;<SEMICOLON> - ; REGISTER CONTENTS
@DOS         - @ [DOS COMMAND]
?HELP        - ?
```

MOUNT

Token: $FE $49

Formaat: **MOUNT** bestandsnaam [**,U** eenheid]
MOUNT [**U** eenheid]
MOUNT OFF [U eenheid]

Gebruik: Mount een schijfimagebestand van het type D81 of D64 vanaf de SD-kaart op station 8 (standaard) of station 9 (**U9**).

MOUNT zonder argumenten wijst het interne fysieke diskettestation toe aan station 9 en heft de toewijzing van station 9 op. Indien **eenheid** wordt opgegeven, wordt alleen dat station gewijzigd.

MOUNT OFF heft de toewijzing van zowel station 8 als station 9 op. Dit is equivalent aan de "no disk"-instelling in het Freezer-menu. Indien **eenheid** wordt opgegeven, wordt alleen dat station gewijzigd.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

eenheid is ofwel 8 voor het eerste virtuele station, of 9 voor het tweede virtuele station.

Opmerkingen:

MOUNT kan zowel in de directe modus als in een programma worden gebruikt.

Station 8 en station 9 verwijzen respectievelijk naar de virtuele stations 0 en 1. U kunt de stationnummers voor deze stations wijzigen met het **SET DISK**-commando. Het **MOUNT**-commando verwacht altijd **U8** of **U9** als argument om naar deze stations te verwijzen, zelfs indien de werkelijke stationnummers door **SET DISK** of in het Freezer-menu zijn gewijzigd.

MOUNT zonder argumenten (of met het argument **U9**) wijst station 9 eigenlijk toe aan een hypothetische Commodore 1565 externe diskettestation. Een standaard MEGA65 heeft geen 1565-poort en dergelijke stations zijn er nog niet, waardoor station 9 effectief niet wordt gemount.

OPMERKING: Het **MOUNT**-commando onthoudt de meest recente schijfimage-bestandsnaam voor elk station en probeert de schijfimages opnieuw te mounten na het indrukken van de resetknop. **MOUNT** zonder bestandsnaam, of **MOUNT OFF**, doet de meest recente schijfimage-bestandsnamen vergeten. Schijfimages die via de Freezer zijn gemount, worden niet op deze manier onthouden. Deze functionaliteit wordt herzien en kan zich in een latere versie anders gedragen.

Voorbeelden:

Gebruik van **MOUNT**

```
MOUNT "APOCALYPS.D81"  : REM MOUNT IMAGE OP UNIT 8

MOUNT "BASIC.D81", U9  : REM MOUNT IMAGE OP UNIT 9

MOUNT (FI$), U(UN%)    : REM MOUNT MET VARIABELE ARGUMENTEN

MOUNT                  : REM INTERNE DRIVE OP UNIT 8, U9 NIET TOEGEWEZEN
MOUNT U8               : REM INTERNE DRIVE OP UNIT 8, U9 ONVERANDERD

MOUNT OFF              : REM BEIDE SCHIJVEN OP "GEEN DISK"
MOUNT OFF U9           : REM ENKEL UNIT 9 OP "GEEN DISK"
```

MOUSE

Token: $FE $3E

Formaat: **MOUSE ON** [{**,** poort**,** sprite**,** activeringspunt**,** pos}]
MOUSE OFF

Gebruik: Schakelt de muisdriver in en verbindt de muis op de opgegeven poort met de muispijltjessprite.

poort muispoort 1 of 2 (standaard 2), of "3" zodat een muis in beide poorten het sprite kan verplaatsen.

sprite nummer van de muispijltjessprite (standaard 0).

activeringspunt locatie van het "activeringspunt" dat de positie en het klikdoelwit bepaalt (x, y) (standaard 0, 0).

positie beginspositie van de muis (x, y). Indien niet opgegeven, wordt de laatst bekende positie van het sprite gebruikt.

MOUSE OFF schakelt de muisdriver uit en verbergt het bijbehorende sprite.

Opmerkingen:

Het "activeringspunt" van de muis bepaalt welk deel van de muisafbeelding als klikdoelwit wordt beschouwd, zoals de bovenkant van een pijl of het midden van een richtkruis. Het activeringspunt blijft altijd binnen de schermrand. Het standaard **activeringspunt** is (0, 0), wat de linkerbovenhoek van het sprite vertegenwoordigt.

Bij het opstarten van het systeem wordt sprite 0 geïnitialiseerd als een afbeelding van een muispijltje, met het activeringspunt op (0, 0).

Gebruik **RMOUSE** om de locatie en de knopstatus van de muis te testen. Dit retourneert de coördinaten van de linkerbovenhoek van het sprite, niet de coördinaten van het activeringspunt. Om de coördinaten van het activeringspunt te verkrijgen, telt u de locatie van het activeringspunt op bij de sprite-coördinaten.

positie kan een absolute coördinaat zijn, of een relatieve coördinaat ten opzichte van de huidige muispositie, vergelijkbaar met **MOVSPR**.

U kunt een van de hoge-resolutie-sprite-modi van de MEGA65 gebruiken voor de muispijltjessprite. De modi moeten worden ingesteld voordat de muis met **MOUSE** wordt geactiveerd.

Zie ook **SET FAST MOUSE**.

Voorbeelden:

Gebruik van **MOUSE**

```
REM LAAD DATA IN SPRITE #0 ALVORENS HET TE GEBRUIKEN
MOUSE ON, 1          : REM MUIS AAN MET SPRITE #0
MOUSE OFF            : REM MUIS UIT

MOUSE ON, 1, 0, 2, 4  : REM ZET DE HOTSPOT TOP (2,4)
RMOUSE X, Y, B        : REM HAAL MUISSPRITE-COORDINATEN OP
X = X + 2 : Y = Y + 4 : REM BEREKEN COORDINATEN VAN DE HOTSPOT

REM ZET BEGINPOSITIE OP (300,75)
MOUSE ON, 1, 0, 0, 0, 300,75

MOUSE ON, 3    : REM MUIS VERBONDEN MET EENDER WELKE POORT BEWEEGT PIJLTJE

SETBIT $D076,0 : REM VERTICALE HIRES VOOR SPRITE 0 AAN
SETBIT $D054,4 : REM HORIZONTAL HIRES VOOR ALLE SPRITES AAN
MOUSE ON, 1    : REM MUISPIJLTJE IS HIRES OP BEIDE ASSEN
```

MOVSPR

Token: $FE $06

Formaat: **MOVSPR** nummer**,** positie
MOVSPR nummer**,** begin-positie **TO** eind-positie**,** snelheid

Gebruik: Verplaatst een sprite naar een positie op het scherm.

Elke **positie**-parameter bestaat uit twee 16-bits waarden, welke een absolute coördinaat, een relatieve coördinaat, een hoek of een snelheid specificeren. Het waardetype wordt bepaald door een prefix:

- **+waarde** relatieve coördinaat: positieve offset.
- **-waarde** relatieve coördinaat: negatieve offset.
- **#waarde** snelheid.

Indien geen prefix wordt opgegeven, wordt de absolute coördinaat of hoek gebruikt.

De positie-parameter kan derhalve worden gebruikt om:

- het sprite op een absolute positie op het scherm te plaatsen.
- een verplaatsing ten opzichte van de huidige positie op te geven.
- een relatieve beweging vanaf een opgegeven positie te activeren.
- beweging met een hoek en snelheid vanaf de huidige positie te beschrijven.

MOVSPR nummer, positie wordt gebruikt om de sprite onmiddellijk op de positie te plaatsen of, in het geval van een hoek#snelheid-argument, de verdere beweging te beschrijven.

In de tweede vorm plaatst het sprite op de beginpositie, definieert de bestemmingspositie en de snelheid van de beweging.

Het sprite wordt op de beginpositie geplaatst en beweegt in een rechte lijn naar de bestemming op de opgegeven snelheid. De coördinaten moeten absoluut of relatief zijn. De beweging wordt beheerd door de BASIC-interrupt handler en vindt parallel aan de programma-uitvoering plaats.

nummer sprite-nummer (0 - 7).

positie coördinaten: x,y | xrel,y | x,yrel | xrel,yrel | hoek#snelheid.

x absolute schermcoördinaat in pixels.

y absolute schermcoördinaat in pixels.

xrel relatieve schermcoördinaat in pixels.

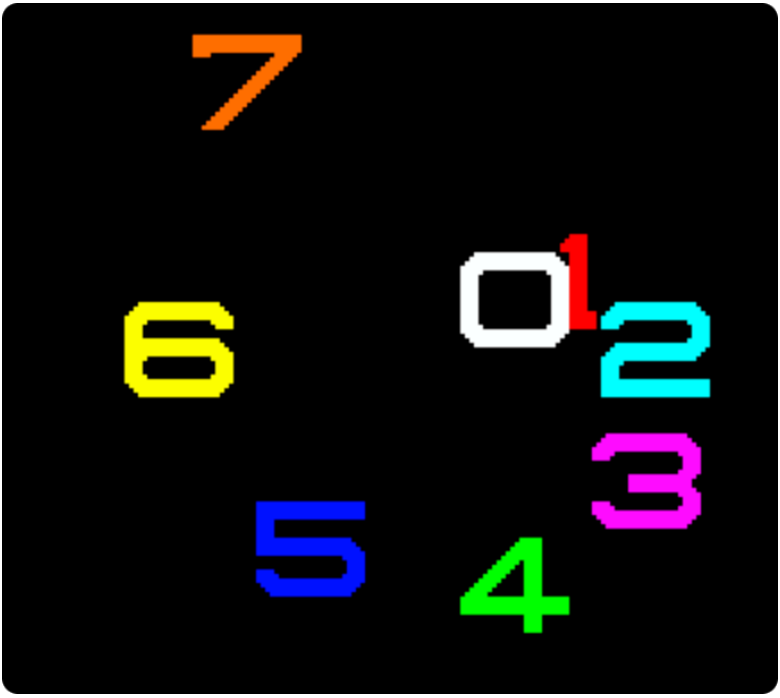
yrel relatieve schermcoördinaat in pixels.

hoek kompasrichting voor de spritebeweging [graden]. 0: omhoog, 90: rechts, 180: omlaag, 270: links, 45: rechtsboven, enzovoort.

snelheid snelheid van de beweging, geconfigureerd als een zwevend getal in het bereik 0.0 – 127.0, in pixels per frame. PAL heeft 50 frames per seconde, NTSC heeft 60 frames per seconde. Een snelheidswaarde van 1.0 verplaatst het sprite met 50 pixels per seconde in de PAL-modus.

Voorbeeld: Gebruik van **MOVSPR**

```
100 CLR : SCNCLR : SPRITECLR
110 BLOAD "DEMOSPRITES1", B0, P1536
130 FOR I = 0 TO 7 : C = I + 1 : SP = 0.07 * (I + 1)
140 MOV SPR I, 160, 120
145 MOV SPR I, 45*I#SP
150 SPRITE I, 1, C,, 0, 0
160 NEXT
170 SLEEP 3
180 FOR I = 0 TO 7 : MOVSPR I, 0#0 : NEXT
```



NEW

Token: $A2

Formaat: **NEW**
NEW RESTORE

Gebruik: Wist het BASIC-programma in het geheugen en reset alle BASIC-parameters naar hun standaardwaarden.

Omdat **NEW** parameters en pointers reset, maar het adresbereik van een BASIC-programma dat in het geheugen aanwezig was niet overschrijft, is het mogelijk het programma te herstellen. Indien er na **NEW** geen **LOAD**-operaties of bewerkingen hebben plaatsgevonden, kan het programma worden hersteld met **NEW RESTORE**.

Voorbeelden:

Gebruik van **NEW**

```
NEW         : REM RESET BASIC EN WIS PROGRAMMA

NEW RESTORE : REM PROBEERD GE'NEW'D PROGRAMMA TE HERSTELLEN
```

NEXT

Token: $82

Formaat: **FOR** index = begin **TO** einde [**STEP** stap] ... **NEXT** [index]

Gebruik: Markeert het einde van de lus die bij de opgegeven indexvariabele hoort. Een lus die met **FOR** wordt gedefinieerd, moet eindigen met **NEXT**.

De **index**-variabele kan bij elke iteratie met een constante waarde **stap** worden verhoogd of verlaagd. Standaard wordt de variabele met 1 verhoogd. De indexvariabele moet een reële variabele zijn.

begin de waarde waarmee de index wordt geïnitialiseerd.

einde wordt aan het einde van een iteratie gecontroleerd en bepaalt of nog een iteratie wordt uitgevoerd of dat de lus wordt afgesloten.

stap definieert de verandering die aan het einde van elke iteratie op de indexvariabele wordt toegepast. Positieve stapwaarden verhogen de variabele, negatieve waarden verlagen deze. Standaard is 1,0 indien niet opgegeven.

Opmerkingen:

De **index**-variabele na **NEXT** is optioneel. Indien deze wordt weggelaten, wordt de variabele van de huidige lus aangenomen. Meerdere opeenvolgende **NEXT**-instructies kunnen worden gecombineerd door de indexen als een door komma's gescheiden lijst op te geven. De instructies "NEXT I : NEXT J : NEXT K" en "NEXT I, J, K" zijn equivalent.

Voorbeelden:

Gebruik van **NEXT**

```
10 FOR D = 0 TO 360 STEP 30
20 R = D * π / 180
30 PRINT D;R;SIN(R);COS(R);TAN(R)
40 NEXT D

10 DIM M(20, 20)
20 FOR I = 0 TO 20
30 FOR J = I TO 20
40 M(I, J) = I + 100 * J
50 NEXT J, I
```

NOT

Token: $A8

Formaat: **NOT** operand

Gebruik: Voert een bitgewijze logische NOT-bewerking uit op een 16-bits waarde.

Gehele operanden worden als zodanig gebruikt, terwijl reële operanden worden omgezet naar een geheel getal van 16 bit in het tweecomplementformaat (waarbij precisie verloren gaat).

Logische operanden worden omgezet naar een geheel getal van 16 bit, waarbij $FFFF (decimaal -1) voor TRUE en $0000 (decimaal 0) voor FALSE wordt gebruikt.

Expressie	Resultaat
NOT 0	-1
NOT -1	0

Opmerkingen:

Het resultaat is van het type geheel getal.

Voorbeelden:

Gebruik van **NOT**

```
PRINT NOT 3
-4

PRINT NOT -65
64

C = 257
OK = C ( 256 AND C )= 0
IF (NOT OK) THEN PRINT "GEEN BYTEWAARDE"
```

OFF

Token: $FE $24

Formaat: trefwoord **OFF**

Gebruik: **OFF** is een secundair trefwoord dat in combinatie met primaire trefwoorden wordt gebruikt, zoals **KEY** en **MOUSE**.

Opmerkingen:
OFF kan niet op zichzelf worden gebruikt.

Voorbeelden:
Gebruik van **OFF**

```
KEY OFF   : REM FUNCTIESNELTOETSEN UIT

MOUSE OFF : REM MUISPIJLTJE UIT
```

ON

Token: $91

Formaat: **ON** expressie **GOSUB** regelnummer [**,** regelnummer ...]
ON expressie **GOTO** regelnummer [**,** regelnummer ...]
trefwoord **ON**

Gebruik: Voert een **GOSUB** of **GOTO** uit naar een regelnummer dat door een getalsexpressie wordt geselecteerd.

Afhankelijk van het resultaat van de expressie wordt het doel voor **GOSUB** en **GOTO** gekozen uit de tabel met regelnummers aan het einde van de instructie.

Wanneer **ON** als secundair trefwoord wordt gebruikt, wordt het gecombineerd met primaire trefwoorden zoals **KEY** en **MOUSE**.

expressie positieve numerieke waarde. Reële getallen worden omgezet naar een geheel getal (met verlies van nauwkeurigheid) in tweecomplementformaat. Logische operanden worden omgezet naar 16-bits gehele getallen, waarbij $FFFF (decimaal -1) voor WAAR en $0000 (decimaal 0) voor ONWAAR wordt gebruikt.

Opmerkingen:

Negatieve waarden voor **expressie** onderbreken het programma met een Illegal Quantity-fout.

De **regelnummerlijst** geeft de doelen op voor waarden van 1, 2, 3 enzovoort.

Een expressieresultaat van nul, of een resultaat dat groter is dan het aantal doelregels, heeft geen effect en het programma gaat verder met de volgende instructie.

Voorbeeld: Gebruik van **ON**

```
 20 KEY   ON : REM FUNCTIESNELTOETSEN AAN
 30 MOUSE ON : REM MUISDRIVER AAN
 40 N = JOY(1) : IF N AND 128 THEN PRINT "VUUR! ";
 60 REM              N    NO   O    ZO   Z    ZW   W    NW
 70 ON N AND 15 GOSUB 100, 200, 300, 400, 500, 600, 700, 800
 80 GOTO 40
100 PRINT "GA NOORD"     : RETURN
200 PRINT "GA NOORDOOST" : RETURN
300 PRINT "GA OOST"      : RETURN
400 PRINT "GA ZUIDOOST"  : RETURN
500 PRINT "GA ZUID"      : RETURN
600 PRINT "GA ZUIDWEST"  : RETURN
700 PRINT "GA WEST"      : RETURN
800 PRINT "GA NOORDWEST" : RETURN
```

OPEN

Token: $9F

Formaat: **OPEN** kanaal**,** eerste adres [**,** secundair adres [**,** bestandsnaam]]

Gebruik: Opent een invoer/uitvoerkanaal voor een apparaat.

kanaal getal, waarbij:

- 1 <= kanaal <= 127: regelafsluiting is CR.
- 128 <= kanaal <= 255: regelafsluiting is CR LF.

eerste adres apparaatnummer. Voor IEC-apparaten is het eenheidsnummer het primaire adres. De volgende primaire adreswaarden zijn mogelijk:

Eenheid	Apparaat
0	Toetsenbord
1	Standaard van het systeem
2	RS232-seriële verbinding
3	Scherm
4 - 7	IEC-printer en plotter
8 - 31	IEC-schijfstations

Het **secundaire adres** heeft enkele gereserveerde waarden voor IEC-schijfeenheden: 0: laden, 1: opslaan, 15: commandokanaal. De waarden 2 - 14 kunnen voor schijfbestanden worden gebruikt.

bestandsnaam is een aangehaalde tekenreeks, bijv. `"DATA"` of een tekenreeksexpressie. De syntaxis verschilt van **DOPEN#**, omdat de **bestandsnaam** voor **OPEN** alle bestandsattributen bevat, bijvoorbeeld: `"0:DATA,S,W"`

Opmerkingen:

Voor IEC-schijfeenheden wordt het gebruik van **DOPEN#** aanbevolen.

Indien het eerste teken van de bestandsnaam een "@" (apestaartje) is, wordt dit geïnterpreteerd als een "opslaan en vervangen"-operatie. Het gebruik van deze optie op 1541- en 1571-stations wordt niet aanbevolen, omdat deze een "opslaan en vervangen-fout" in hun DOS bevatten.

Voorbeeld: Gebruik van **OPEN**

```
OPEN 4, 4                    : REM OPEN PRINTER
CMD 4                        : REM LEID STANDAARDUITVOER NAAR 4 OM
LIST                         : REM PRINT LISTING OP PRINTERDEVICE 4
OPEN 3, 8, 3, "0:USRBESTAND,U"
OPEN 2, 9, 2, "0:DATA,S,W"
```

OR

Token: $B0

Formaat: operand **OR** operand

Gebruik: Voert een bitgewijze logische OR-bewerking uit op twee 16-bits waarden.

Gehele operanden worden ongewijzigd gebruikt. Reële operanden worden omgezet naar een geheel getal van 16 bit (met verlies van nauwkeurigheid) in tweecomplementformaat. Logische operanden worden omgezet naar 16-bits gehele getallen, waarbij $FFFF (decimaal -1) voor WAAR en $0000 (decimaal 0) voor ONWAAR wordt gebruikt.

Expressie	Resultaat
0 OR 0	0
0 OR 1	1
1 OR 0	1
1 OR 1	1

Opmerkingen:

Het resultaat is van het type geheel getal. Indien het resultaat in een logische context wordt gebruikt, wordt de waarde 0 als ONWAAR beschouwd en alle andere niet-nul-waarden als WAAR.

Voorbeelden:

Gebruik van **OR**

```
PRINT 1 OR 3
 3

PRINT 128 OR 64
 192

IF (C<0 OR C>255) THEN PRINT "GEEN BYTEWAARDE"
```

PAINT

Token: $DF

Formaat: **PAINT** x, y, modus [, gebiedsrandkleur]

Gebruik: Bitmapgrafiek: voert een vulbewerking uit op een afgesloten grafisch gebied met de huidige tekenkleur.

x, y coördinaatpaar, dat zich binnen het te vullen gebied moet bevinden.

modus specificeert de tekenmodus:

- 0: De kleur van pixel (x,y) bepaalt de kleur die door de tekenkleur wordt vervangen.
- 1: De **gebiedsrandkleur** bepaalt het gebied dat met de tekenkleur wordt gevuld.
- 2: Vult het gebied dat verbonden is met pixel (x, y).

randkleur van het gebied bepaalt de kleurindex voor modus 1.

Voorbeeld: Gebruik van **PAINT**

```
10 SCREEN 320, 200, 2      : REM OPEN SCHERM
20 PALETTE 0, 1, 10, 15, 10 : REM KLEUR 1 OP LICHTGROEN
30 PEN 1                   : REM ZET TEKENPEN (PEN 0) OP LICHTGROEN (1)
40 LINE 160, 0, 240, 100   : REM 1E REGEL
50 LINE 240, 100, 80, 100  : REM 2E REGEL
60 LINE 80, 100, 160, 0    : REM 3E REGEL
70 PAINT 160, 10           : REM VUL DRIEHOEK MET PENKLEUR
80 GETKEY A$               : REM WACHT OP TOETSDRUK
90 SCREEN CLOSE            : REM GRAFISCH SCHERM UIT
```

PALETTE

Token: $FE $34

Formaat: **PALETTE** scherm, kleur, rood, groen, blauw
PALETTE COLOR kleur, rood, groen, blauw
PALETTE RESTORE

Gebruik: **PALETTE** kan worden gebruikt om een item van het systeemkleurenpalet of het palet van een scherm te wijzigen.

PALETTE RESTORE herstelt het systeempalet naar de standaardwaarden.

scherm schermnummer (0 - 3).

COLOR trefwoord voor het wijzigen van het systeempalet.

kleur index naar paletitem (0 - 255). PALETTE kan kleuren definiëren buiten de standaard systeempaletitems 0 - 31.

rood intensiteit (0 - 15).

groen intensiteit (0 - 15).

blauw intensiteit (0 - 15).

Voorbeelden:

Gebruik van **PALETTE**

```
10 REM VERANDER SYSTEEMKLEURINDEX
20 REM --- INDEX 9 (BRUIN) TO (DONKERBLAUW)
30 PALETTE COLOR 9, 0, 0, 7
```

```
 10 GRAPHIC CLR           : REM INITIALISEER
 20 SCREEN DEF 1, 0, 0, 2 : REM 320 X 200
 30 SCREEN OPEN 1         : REM OPEN
 40 SCREEN SET 1, 1       : REM MAAK SCHERM ACTIEF
 50 SCREEN CLR 0          : REM MAAK SCHERM ZWART
 60 PALETTE 1, 0,  0, 0, 0 : REM 0 = ZWART
 70 PALETTE 1, 1, 15, 0, 0 : REM 1 = ROOD
 80 PALETTE 1, 2,  0, 0,15 : REM 2 = BLAUW
 90 PALETTE 1, 3,  0,15, 0 : REM 3 = GROEN
100 PEN 2                 : REM ZET TEKENPEN (PEN 0) OP BLAUW (2)
110 LINE 160, 0, 240, 100  : REM 1E REGEL
120 LINE 240, 100, 80, 100 : REM 2E REGEL
130 LINE 80, 100, 160, 0   : REM 3E REGEL
140 PAINT 160, 10          : REM VUL DRIEHOEK MET HUIDIGE PENKLEUR
150 GETKEY K$              : REM WACHT OP TOETSDRUL
160 SCREEN CLOSE 1         : REM GRAFISCH SCHERM UIT
```

PASTE

Token: $E3

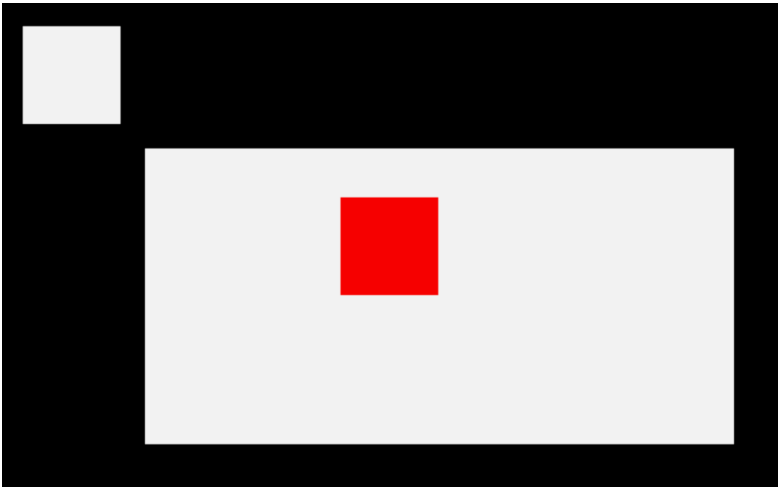
Formaat: **PASTE** x, y

Gebruik: Bitmapgrafiek: plakt de inhoud van de **CUT**- of **GCOPY**-buffer op het scherm. De argumenten **x, y** geven de linkerbovenhoek van de plakpositie op het scherm aan, als pixelcoördinaten.

Opmerkingen:
De grootte van de rechthoekje wordt beperkt door de 1KB-grootte van de buffer. Het geheugenvereiste voor een gebied is breedte * hoogte * aantal bitplanes / 8. Dit mag niet 1024 bytes of meer bedragen. Bij een scherm met 4 bitplanes heeft een gebied van 50 x 50 bijvoorbeeld 1.250 bytes nodig.

Voorbeeld: Gebruik van **PASTE**

```
10 SCREEN 320, 200, 2
20 BOX 60, 60, 300, 180, 1 : REM TEKEN EEN WIT VAK
30 PEN 2                   : REM KIES RODE PEN
40 CUT 140, 80, 40, 40     : REM SNIJ EEN GEBIED VAN 40 * 40 UIT
50 PASTE 10,10             : REM PLAK HET OP NIEUWE POSITIE
60 GETKEY A$               : REM WACHT OP TOETSDRUK
70 SCREEN CLOSE
```



PEEK

Token: $C2

Formaat: **PEEK**(adres)

Retourneert:

De byte-waarde die in het geheugen op het opgegeven **adres** is opgeslagen, als een natuurlijk getal van 8 bit.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

Opmerkingen:

Banken 0 - 127 geven toegang tot RAM- of ROM-banken. Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enzovoort.

Voorbeeld: Gebruik van **PEEK**

```
10 BANK 128                          : REM KIES SYSTEEMBANK
20 L = PEEK($02F8)                   : REM DOELADRES USR-JUMP LAAG
30 H = PEEK($02F9)                   : REM DOELADRES USR-JUMP HOOG
40 T = L + 256 * H                   : REM 16-BIT JUMPADRES
50 PRINT "ADRES USR-FUNCTIEAANROEP";T
```

PEN

Token: $FE $33

Formaat: **PEN** kleur

Gebruik: Bitmapgrafiek: stelt de kleur (paletitem) van de grafische pen in voor het huidige scherm.

kleur paletindex, uit het palet van het huidige scherm.

Zie bijlage 6 op bladzijde 327 voor de lijst met kleuren in het standaard systeempalet.

Opmerkingen:

De ontwerpdocumentatie van de C65 beschrijft een versie van **PEN** met twee argumenten, waarmee het paletitem voor een van de twee pennen wordt ingesteld: de tekenpenkleur (pen 0) en een secundaire kleur die door de **jam**-tekenmodus wordt gebruikt (pen 1). Deze functie van **jam** lijkt niet te zijn geïmplementeerd. Zie **DMODE**.

Voorbeeld: Gebruik van **PEN**

```
 10 GRAPHIC CLR             : REM INITIALISEER
 20 SCREEN DEF 1, 0, 0, 2   : REM 320 X 200
 30 SCREEN OPEN 1           : REM OPEN
 40 SCREEN SET 1, 1         : REM MAAK SCHERM ACTIEF
 50 PALETTE 1, 0,  0,  0,  0 : REM 0 = ZWART
 60 PALETTE 1, 1, 15,  0,  0 : REM 1 = ROOD
 70 PALETTE 1, 2,  0,  0, 15 : REM 2 = BLAUW
 80 PALETTE 1, 3,  0, 15,  0 : REM 3 = GROEN
 90 PEN 1                   : REM ZET TEKENPEN (PEN 0) OP ROOD (1)
100 LINE 160, 0, 240, 100   : REM TEKEN RODE LIJN
110 PEN 2                   : REM ZET TEKENPEN (PEN 0) OP BLAUW (2)
120 LINE 240, 100, 80, 100  : REM TEKEN BLAUWE LIJN
130 PEN 3                   : REM ZET TEKENPEN (PEN 0) OP GROEN (3)
140 LINE 80, 100, 160, 0    : REM TEKEN GROENE LIJN
150 GETKEY K$               : REM WACHT OP TOETSDRUK
160 SCREEN CLOSE 1          : REM GRAFISCH SCHERM UIT
```

PIXEL

Token: $CE $0C

Formaat: **PIXEL**(x, y)

Gebruik: Bitmapgrafiek: de kleur van een pixel op de opgegeven positie.

x absoluut schermcoördinaat.

y absoluut schermcoördinaat.

Retourneert:
De kleur op positie (x, y). **PIXEL** geeft een Scherm niet geopend-fout terug indien er geen scherm is geopend.

Voorbeeld: Gebruik van **PIXEL**

```
10 SCREEN 320, 200, 4          : REM BOUW SCHERM OP MET 10.000 WILLEKEURIGE PIXELS
20 FOR I = 1 TO 10000 : DOT RND(1) * 320, RND(1) * 200, RND(1) * 16 : NEXT
30 MOUSE ON, 1, 0
40 RMOUSE X, Y, B              : REM LEES MUIS X, Y
50 MOVSPR 0, X, Y              : REM BEWEEG PIJLTJE
60 BORDER PIXEL(X, Y)          : REM ZET RANDKLEUR OP KLEUR OP POSITIE
70 GET A$ : IF A$ = "" THEN 40 : REM EEN TOETSDRUK BEEINDIGT PROGRAMMA
80 BORDER 6 : MOUSE OFF
90 SCREEN CLOSE
```

PLAY

Token: $FE $04

Formaat: **PLAY** [{tekenreeks1, tekenreeks2, tekenreeks3, tekenreeks4, tekenreeks5, tekenreeks6}]

Gebruik: Start het afspelen van een reeks muzieknoten of stopt een momenteel spelende reeks.

PLAY zonder argumenten doet alle stems zwijgen en reset alle variabelen van het muzieksysteem (zoals **TEMPO**).

PLAY aanvaardt maximaal zes kommagescheiden tekenreeksen, waarbij elke tekenreeks de reeks noten en directieven beschrijft die op een specifieke stem van de twee beschikbare SID-chips worden afgespeeld, waardoor tot 6-kanaals polyfonie mogelijk is.

PLAY gebruikt SID1 (voor stems 1 tot 3) en SID3 (voor stems 4 tot 6) van de 4 SID-chips van het systeem. Standaard zijn SID1 en SID2 enigszins rechtsgeoriënteerd en SID3 en SID4 licht linksgeoriënteerd in de stereomix.

```
PLAY "CEG"
PLAY "C", "E", "G"
```

Binnen een **PLAY**-tekenreeks is een muzieknoot een teken (A, B, C, D, E, F of G) dat eventueel door een wijzigingsteken kan worden voorafgegaan.

Mogelijke wijzigingstekens zijn:

Teken	Effect
#	Kruis
$	Mol
.	Punt
W	Hele noot
H	Halve noot
Q	Kwartnoot
I	Achtste noot
S	Zestiende noot
R	Rust

Let op: het wijzigingsteken (**.**) staat vóór de nootnaam, niet erna zoals in traditionele bladmuziek.

Directieven bestaan uit een letter gevolgd door een cijfer. Directieven blijven van kracht voor alle volgende noten, totdat de parameter door een andere directief wordt gewijzigd.

Teken	Directief	Bereik
O	Octaaf	0 - 6
T	Instrumentomhulling	0 - 9
V	Volume	0 - 9
X	Filter	0 - 1
M	Modulatie	0 - 9
P	Portamento	0 - 9
L	Herhaling	N.v.t.

Een octaaf is een bereik noten van C tot B. Het standaard octaaf is 4, het "middelste" octaaf.

Instrumentomhullingen beschrijven de aard van het geluid. Zie **ENVELOPE** voor een lijst met standaard omhullingsstijlen en informatie over het aanpassen van de tien omhullingen.

De modulatiedirectief voegt een toonhoogtegebaseerde vibrato aan uw noot toe, met de sterkte die u opgeeft (1 - 9). Een waarde van 0 schakelt het uit.

De portamentodirectief schuift op de snelheid die u opgeeft (1 - 9) tussen opeenvolgende noten. Een waarde van 0 schakelt het uit. Noten worden aan het eind van elke noot nog steeds uitgeschakeld (gate-off) zolang portamento is ingeschakeld.

Indien een tekenreeks eindigt met de directief **L**, wordt het patroon na voltooiing teruggelusd naar het begin van de tekenreeks.

U kunt een tekenreeks voor een bepaalde stem overslaan, zodat een al spelend patroon op die stem doorgaat, door lege argumenten te gebruiken:

```
PLAY "O4EDCDEEERL",,, "O2CGEGCGEGL"
```

Een voorbeeld met stem 2 en stem 5

```
PLAY , "O5T2IGAGFEDCEGO6.QCL",,, "O3T2.QG.B O4ICO3GE.QCL"
```

RPLAY(stem) test of er muziek wordt afgespeeld op de opgegeven stem en retourneert 1 indien dit het geval is, of 0 indien niet.

Een beperking waar u rekening mee moet houden, is dat BASIC-strings een maximale lengte van 255 bytes hebben. Indien uw melodie deze lengte overschrijdt, kunt u de melodie opsplitsen in meerdere tekenreeksen en vervolgens **RPLAY(stem)** gebruiken om te bepalen wanneer de eerste tekenreeks is geëindigd, waarna de volgende tekenreeks kan worden afgespeeld.

Instrumentomhullingen kunnen worden aangepast met het **ENVELOPE**-statement. De standaardinstellingen voor de omhullingen staan op bladzijde 107.

Retourneert:

Het **PLAY**-statement maakt gebruik van een interruptgestuurde routine die de tekenreeks begint te verwerken en de melodie af te spelen. De programma-uitvoering gaat door met het volgende statement en blokkeert niet totdat de melodie is geëindigd. Dit verschilt van de Commodore 128, die de programma-uitvoering tijdens het afspelen onderbreekt.

De 6 stemkanalen die door het **PLAY**-commando worden gebruikt (op SID1 en SID3) zijn onafhankelijk van de 6 kanalen die door het **SOUND**-commando worden gebruikt (op SID2 en SID4). Geluidseffecten onderbreken de muziek niet en omgekeerd.

Voorbeelden:

Gebruik van **PLAY**

```
 5 REM *** VOORBEELD VAN EEN EENVOUDIGE ZWAAI ***
10 ENVELOPE 9, 10, 5, 10, 5, 0, 300
20 VOL 8, 8
30 TEMPO 30
40 PLAY "O5T9HCIDCDEHCG IGAGFEFDEWCL", "O2T0QCGEGCGEG DBGB CGEGL"
```

```
 5 REM *** VOORBEELD VAN MODULATIE + PORTAMENTO ***
10 TEMPO 20
20 M$ = "M5 T2O5P0QD P5FP0RP5QG .AI#AQA HGQE.C IDQE HFQD .DI#CQD HEQ#CQO4HA"
30 M$ = M$ + "O5QDHFQG.AI#AQA HGQE.C IDQEFED#CO4BO5HC DO4AFD P0R L"
40 B$ = "T0QRO2H.D.F.CO1.A.#A.G.A QAIO2AGFE H.D.F.CO1.A.#A.AO2 .D DL"
50 PLAY M$, B$
```

POINTER

Token: $CE $0A

Formaat: **POINTER**(variabele)

Retourneert:

Het huidige adres van een variabele of een array-element als een 32-bits pointer.

Voor stringvariabelen is dit het adres van de stringdescriptor, niet van de string zelf. De stringdescriptor bestaat uit drie bytes: lengte, lage byte van het stringadres, hoge byte van het stringadres. Het stringadres is een offset in bank 1.

Voor scalaire variabelen van een numeriek type is dit het adres van de waarde. Het formaat hangt af van het type. Een bytevariabele (`A&`) beslaat één byte in het tweecomplement-formaat voor gehele getallen. Een integervariabele (`A%`) beslaat twee bytes, waarbij de meest significante byte als eerste komt. Een reële variabele (`A`) beslaat vijf bytes in het compacte zwevende-kommagetalformaat.

Om het adres van een array te verkrijgen, gebruikt u **POINTER** met het eerste element van de array (index `0` in elke dimensie). Array-elementen worden opeenvolgend opgeslagen in het formaat van het scalaire record, waarbij de meest linkse index de kortste stap gebruikt. Een array die is gedefinieerd als `DIM A%(3, 2)` begint op adres `POINTER(A%(0, 0))`, heeft tweebiterecords en is geordend als:

(0, 0) (1, 0) (2, 0) (3, 0) (0, 1) (1, 1) (2, 1) (3, 1) ...

Opmerkingen:

De adreswaarden vanarraysen hun elementen zijn constant tijdens de uitvoering van het programma. De adressen van strings (niet hun descriptors) kunnen echter te allen tijde wijzigen door “garbage collection”.

Voorbeeld: Gebruik van **POINTER**

```
10 BANK 0                          : REM SCALAIRE VARS ZITTEN IN BANK 0
20 H$ = "HALLO"                    : REM WIJS TEKENREEKS TOE AAN H$
30 P = POINTER(H$)                 : REM VRAAG DESCRIPTORADRES OP
40 PRINT "DESCRIPTOR OP: $"; HEX$(P)
50 L = PEEK(P) : SP = WPEEK(P+1)   : REM LENGTE & STRINGPOINTER
60 PRINT "LENGTE = "; L            : REM PRINT LENGTE
70 BANK 1                          : REM STRINGS ZITTEN IN BANK 1
80 FOR I% = 0 TO L - 1 : PRINT PEEK(SP + I%); : NEXT : PRINT
90 FOR I% = 0 TO L - 1 : PRINT CHR$(PEEK(SP + I%)); : NEXT : PRINT

RUN
DESCRIPTOR AT: $FD75
LENGTH =  5
 72  69  76  76  79
HELLO
```

POKE

Token: $97

Formaat: **POKE** adres**,** waarde [**,** waarde ...]

Retourneert:

Schrijft één of meer bytes in het geheugen of geheugengestuurde I/O, beginnend bij **adres**.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

Indien **waarde** in het bereik van 0 - 255 valt, wordt deze in het geheugen gepoked; anders wordt de lage byte van de waarde gebruikt. Een commando als `POKE AD,V AND 255` kan daarom worden geschreven als `POKE AD,V`, omdat **POKE** sowieso de lage byte gebruikt.

Opmerkingen:

Het adres wordt voor elke databyte verhoogd, waardoor een geheugenbereik met één enkele **POKE** kan worden geschreven.

Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enz.

Voorbeeld: Gebruik van **POKE**

```
10 BANK 128          : REM KIES SYSTEEMBANK
20 POKE $02F8, 0, 24 : REM ZET USR-VECTOR OP $1800
```

POLYGON

Token: $FE $2F

Formaat: **POLYGON** x, y, xrad, yrad, zijden [{, tekenzijden, overspannen, hoek, massief}]

Gebruik: Bitmapgrafiek: tekent een regelmatige n-hoek. De veelhoek wordt getekend met de huidige tekensituatie die is ingesteld met **SCREEN**, **PALETTE** en **PEN**.

x,y centrumcoördinaten.

xrad,yrad straal in x- en y-richting.

zijden aantal zijden van de veelhoek.

tekenzijden aantal te tekenen zijden.

overspannen tekent een lijn van het centrum naar het beginpunt indien ingesteld (1).

hoek beginhoek.

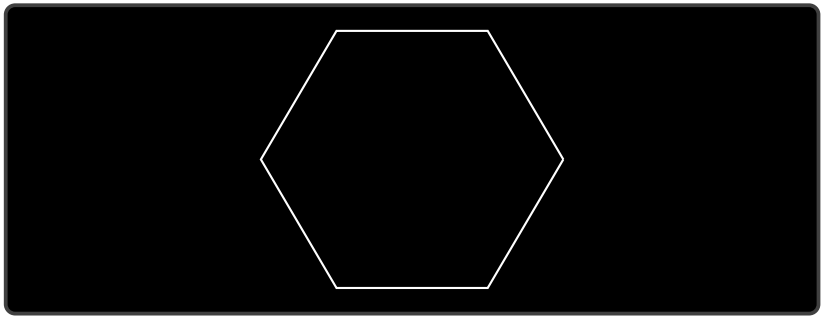
massief vult (1) of omlijnt (0) de veelhoek.

Opmerkingen:

Een regelmatige veelhoek is zowel isogoon als isotoxaal, wat betekent dat alle zijden en hoeken gelijk zijn.

Voorbeeld: Gebruik van **POLYGON**

```
100 SCREEN 320, 200, 1          : REM OPEN SCHERM VAN 320 x 200
110 POLYGON 160, 100, 40, 40, 6 : REM TEKEN HONINGGRAAT
120 GETKEY A$                   : REM WACHT OP TOETSDRUK
130 SCREEN CLOSE                : REM GRAFISCH SCHERM UIT
```



POS

Token: $B9

Formaat: **POS**(dummy)

Retourneert:
De huidige cursorkolom ten opzichte van het momenteel gebruikte venster.

dummy numerieke waarde, die wordt genegeerd.

Opmerkingen:
POS geeft de kolompositie van de schermcursor. Het werkt niet voor omgeleide uitvoer.

Voorbeeld: Gebruik van **POS**

```
10 IF POS(0) > 72 THEN PRINT : REM VOEG RETURN IN
```

POT

Token: $CE $02

Formaat: **POT**(paddle)

Retourneert:

De positie van een paddle-invoerapparaat.

paddle paddle-nummer (1 - 4).

De lage byte van de retourwaarde is de paddle-waarde, waarbij 0 de met de klok mee is en 255 de antimet de klok mee.

Een waarde groter dan 255 geeft aan dat de vuurknop ook wordt ingedrukt.

Opmerkingen:

Analoge paddles zijn onnauwkeurig. Het bereik kan kleiner zijn dan 0 - 255 en de waarden die door **POT** worden geretourneerd, kunnen enigszins fluctueren.

Paddles die zijn gemaakt voor Atari-spelconsoles geven andere waarden terug dan paddles voor Commodore-computers. Commodore-paddles leveren de meest nauwkeurige waarden in het bereik 0 - 255 bij een volledige draai van de paddle. Atari-paddles gaan doorgaans slechts voor een deel van een draai van 0 naar 255, wat resulteert in een snellere beweging en een "dode zone" voor de rest van de draai.

Voorbeeld: Gebruik van **POT**

```
10 X = POT(1)    : REM LEES PADDLE #1
20 B = X ) 255   : REM WAAR (-1) INDIEN VUURKNOP INGEDRUKT
30 V = X AND 255 : REM WAARDE PADDLE #1
```

PRINT

Token: $99

Formaat: **PRINT** argumenten

Gebruik: Drukt een reeks waarden af die zijn opgemaakt naar de huidige uitvoerstroom, doorgaans het scherm.

Waarden worden opgemaakt op basis van hun type. Voor meer controle over de opmaak, zie **PRINT USING**.

De volgende expressies en tekens kunnen in de argumentenlijst voorkomen:

- **numeriek**: De afdruk begint met een spatie voor positieve en nulwaarden, of een minteken voor negatieve waarden. Integere waarden worden afgedrukt met het benodigde aantal cijfers. Reële waarden worden afgedrukt in vastekommavorm (doorgaans 9 cijfers) of wetenschappelijke notatie indien de waarde buiten het bereik van 0.01 tot 999999999 valt.
- **tekenreeks**: De tekenreeks kan bestaan uit afdrukbare tekens en controlcodes. Afdrukbare tekens worden op de cursorpositie afgedrukt. Controlcodes worden uitgevoerd.
- Puntkomma (**;**) scheidt argumenten van de lijst. Er worden geen tekens afgedrukt. Een puntkomma aan het einde van de argumentenlijst onderdrukt het automatische terugloopteken (carriage return).
- Komma (**,**) verplaatst de cursor naar de volgende tabpositie.

Opmerkingen:

De functies **SPC** en **TAB** kunnen in de argumentenlijst worden gebruikt voor positionering.

CMD kan worden gebruikt om afgedrukte tekens om te leiden naar een ander apparaat dan het scherm.

Voorbeeld: Gebruik van **PRINT**

```
10 FOR I = 1 TO 10
20 PRINT I, I * I, SQR(I)
30 NEXT
```

PRINT#

Token: $98

Formaat: **PRINT#** kanaal**,** argumenten

Gebruik: Drukt een reeks waarden af die zijn opgemaakt naar het apparaat dat is toegewezen aan **kanaal**.

Waarden worden opgemaakt op basis van hun type. Voor meer controle over de opmaak, zie **PRINT USING**.

kanaal kanaalnummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

De volgende argumententypen worden verwerkt:

- **numeriek** De afdruk begint met een spatie voor positieve en nulwaarden, of een minteken voor negatieve waarden. Integere waarden worden afgedrukt met het benodigde aantal cijfers. Reële waarden worden afgedrukt in vastekommavorm (doorgaans 9 cijfers) of wetenschappelijke notatie indien de waarde buiten het bereik van 0.01 tot 999999999 valt.
- **tekenreeks** Kan bestaan uit afdrukbare tekens en controlcodes. Afdrukbare tekens worden op de cursorpositie afgedrukt, terwijl controlcodes worden uitgevoerd.
- Puntkomma (**;**) scheidt argumenten van de lijst. Er worden geen tekens afgedrukt. Een puntkomma aan het einde van de argumentenlijst onderdrukt het automatische terugloopteken (carriage return).
- Komma (**,**) verplaatst de cursor naar de volgende tabpositie.

Opmerkingen:

De functies **SPC** en **TAB** zijn niet geschikt voor apparaten anders dan het scherm.

Voorbeeld: Gebruik van **PRINT#**

```
10 DOPEN#2, "TABLE", W, U8
20 FOR I = 1 TO 10
30 PRINT#2, I, I * I, SQR(I)
40 NEXT
50 DCLOSE#2
```

U kunt controleren of het bestand **'TABLE'** is geschreven door `DIR "TA*"` in te typen, en vervolgens de inhoud van het bestand bekijken door `TYPE "TABLE"` in te typen.

PRINT USING

Token: $98 $FB of $99 $FB

Formaat: **PRINT**[**#** kanaal**,**] **USING** indeling**;** argument

Gebruik: Geeft een reeks waarden weer die met behulp van een patroon zijn opgemaakt, naar de huidige uitvoerstroom (doorgaans het scherm) of een uitvoerkanaal.

Het argument kan een string of een numerieke waarde zijn. De opmaak van de uitvoer wordt bepaald door de **opmaak**-tekenreeks.

Het **kanaal**nummer, dat eerder is opgegeven bij een aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN**. Indien geen kanaal wordt opgegeven, gaat de uitvoer naar het scherm.

indeling is een stringvariabele of een stringconstante die de regels voor het opmaken definieert. Bij het gebruik van een getal als **argument** kan de opmaak zowel in CBM-stijl plaatsvinden (met een patroon als "###.##") als in C-stijl, met een specificatie voor **breedte**.**precisie** zoals "%3D %7.2F %4X".

Het te formatteren **argument**-nummer. Indien het argument niet in het format past, bijvoorbeeld bij het afdrukken van een variabele met vier cijfers in een reeks van drie hashes "###", wordt in plaats daarvan een asterisk (*) gebruikt.

Opmerkingen:

De formatstring wordt slechts op één argument toegepast, doch het is mogelijk om er meer toe te voegen met opeenvolgende **USING indeling;argument**-reeksen.

Het **argument** kan afdrukbare tekens en controlcodes bevatten. Afdrukbare tekens worden op de cursorpositie weergegeven, terwijl controlcodes worden uitgevoerd.

Het aantal #-tekens bepaalt de breedte van de uitvoer. Indien het eerste teken van de formatstring een gelijkteken (**=**) is, wordt het argument gecentreerd. Indien het eerste teken een groter-dan-teken (**>**) is, wordt het argument rechts uitgelijnd.

Bij een **F**-patroon in C-stijl telt de **breedte** de cijfers mee, maar niet het decimaalteken, zodat `%6.4F` zeven tekens afdrukt en `%7.2F` acht tekens. Dit verschilt van C, waar de breedte het decimaalteken wel meetelt.

Bij een patroon in C-stijl kan met een voorloop-nul worden gewerkt in plaats van spaties, door de **breedte** van een '0' te laten voorafgaan, zoals: `%07.2F` Voorloopnullen worden verwijderd indien de waarde negatief is. Velden na het decimaalteken en hexadecimale patronen worden standaard links met nullen aangevuld.

Voorbeelden:

Gebruik van **PRINT USING**

```
PRINT USING "##.##"; π, USING " [%6.4F] "; SQR(2)
 3.14 [ 1.4142]

PRINT USING " ( # # # ) ";12*31
 ( 3 7 2 )

PRINT USING "####"; "ABCDE"
ABC

PRINT USING ")####"; "ABCDE"
CDE

PRINT USING "ADDRESS:$%4X";65000
ADDRESS:$FDE8

A$ = "####,####,####.#" : PRINT USING A$; 1E7 / 3
  3,333,333.3
```

RCOLOR

Token: $CD

Formaat: **RCOLOR**(kleurbron)

Retourneert:

De huidige kleurindex voor de geselecteerde kleurbron.

De kleurbronnen zijn:

- 0: Achtergrondkleur (VIC $D021).
- 1: Tekstkleur ($F1).
- 2: Markeerkleur ($02D8).
- 3: Randkleur (VIC $D020).

Voorbeeld: Gebruik van **RCOLOR**

```
10 C = RCOLOR(3) : REM KLEURINDEX VAN RANDKLEUR
```

RCURSOR

Token: $FE $42

Formaat: **RCURSOR** {kolvar**,** rijvar}

Gebruik: Leest de huidige cursorkolom en –rij in variabelen in.

Opmerkingen:
De rij- en kolomwaarden beginnen bij nul, waarbij de meest linker kolom nul is en de bovenste rij nul.

Voorbeeld: Gebruik van **RCURSOR**

```
100 CURSOR ON, 20, 10
110 PRINT "[HERE]";
120 RCURSOR X, Y
130 PRINT " COL:"; X; " ROW:"; Y

RUN


                [HERE] COL: 26  ROW: 10
```

RDISK

Token: $CE $15

Formaat: **RDISK**(eigenschapsnummer)

Retourneert:

De waarde van een eigenschap van de meest recente schijfoperatie.

De eigenschappen van schijfoperaties zijn:

- 0: Het aantal bytes dat is gelezen of geschreven.
- 1: Het apparaatnummer.
- 2: Het stationnummer van een multi-station IEC-apparaat, indien van toepassing.

Voorbeeld: Gebruik van **RDISK**

```
DLOAD "MYPROGRAM",U9

10 DU=RDISK(1)
15 REM : LAAD BRONNENBESTAND VANAF ZELFDE STATION ALS PROGRAMMA
20 BLOAD "MIJNBRONNEN",U(DU)
```

READ

Token: $87

Formaat: **READ** variabele [**,** variabele ...]

Gebruik: Leest waarden uit **DATA**-regels in variabelen in.

De **variabelenlijst** bevat willekeurige geldige variabelen.

Alle soorten constanten (gehele getallen, reële getallen en strings) kunnen worden gelezen, maar geen expressies. Elementen worden door komma's gescheiden. Strings die komma's, dubbele punten of spaties bevatten, moeten tussen aanhalingstekens worden geplaatst.

RUN initialiseert de gegevenspointer op het eerste element van de eerste **DATA**-regel en schuift deze bij elk gelezen element op. Het is de verantwoordelijkheid van de programmeur dat het type van de constante en de variabele in de **READ**-regel overeenkomen. Lege elementen zonder constante tussen komma's zijn toegestaan en worden als nul geïnterpreteerd voor numerieke variabelen en als een lege string voor stringvariabelen.

RESTORE kan worden gebruikt om de gegevenspointer op een specifieke regel te zetten voor volgende leesacties.

Opmerkingen:

Het is goede programmeringspraktijk om grote hoeveelheden **DATA**-regels aan het einde van het programma te plaatsen, zodat ze de zoekactie naar regelnummers na **GOTO** en andere instructies met regelnummerdoelen niet vertragen.

Voorbeeld: Gebruik van **READ**

```
10 READ NA$, VE
20 READ N% : FOR I = 2 TO N% : READ GL(I) : NEXT I
30 PRINT "PROGRAM:"; NA$; "   VERSION:"; VE
40 PRINT "N-POINT GAUSS-LEGENDRE FACTORS E:"
50 FOR I = 2 TO N% : PRINT I; GL(I) : NEXT I
30 STOP
80 DATA "MEGA65", 1.1
90 DATA 5, 0.5120, 0.3573, 0.2760, 0.2252
```

RECORD

Token: $FE $12

Formaat: **RECORD#** kanaal**,** record [**,** byte]

Gebruik: Positioneert de lees-/schrijfpointer van een relatief bestand.

Het **kanaal**nummer, dat eerder is opgegeven bij een aanroep van commando's zoals **DOPEN** of **OPEN**.

Het doel**record** (1 - 65535).

De **byte**-positie in het record.

RECORD kan alleen worden gebruikt voor bestanden van het type **REL**, welke relatieve bestanden zijn die directe toegang mogelijk maken.

RECORD positioneert de bestandspointer op het opgegeven recordnummer. Indien dit recordnummer niet bestaat en er voldoende ruimte op de schijf is waar **RECORD** naartoe schrijft, wordt het bestand uitgebreid tot het gevraagde aantal records door lege records toe te voegen. Wanneer dit gebeurt, geeft de schijfstatus het bericht "RECORD NOT PRESENT" weer, maar dit is geen fout!

Na een aanroep van **INPUT#** of **PRINT#** schuift de bestandspointer door naar de volgende recordpositie.

Opmerkingen:

De Commodore-schijfstations hebben een bug in hun DOS, waardoor gegevens verloren kunnen gaan bij het gebruik van relatieve bestanden. Een aanbevolen tijdelijke oplossing is om het commando **RECORD** tweemaal aan te roepen, voor en na de I/O-bewerking.

Voorbeeld: Gebruik van **RECORD**

```
100 DOPEN#2, "DATA BASE", L240        : REM OPENEN OF AANMAKEN
110 FOR I% = 1 TO 20                  : REM SCHRIJFLUS
120 PRINT#2, "RECORD #"; I%           : REM SCHRIJF RECORD
130 NEXT I%                           : REM HERHAAL
140 DCLOSE#2                          : REM SLUIT BESTAND
150                                   : REM NU TESTEN
160 DOPEN#2, "DATA BASE", L240        : REM HEROPEN
170 FOR I% = 20 TO 2 STEP -2          : REM LEES BESTAND ACHTERWAARDS
180 RECORD#2, I%                      : REM POSITIONEER OP RECORD
190 INPUT#2, A$                       : REM LEES RECORD
200 PRINT A$; : IF I% AND 2 THEN PRINT
210 NEXT I%                           : REM HERHAAL
220 DCLOSE#2                          : REM SLUIT BESTAND

RUN
RECORD # 20 RECORD # 18
RECORD # 16 RECORD # 14
RECORD # 12 RECORD # 10
RECORD # 8 RECORD # 6
RECORD # 4 RECORD # 2
```

REM

Token: $8F

Formaat: **REM**

Gebruik: **REM** (afkorting van remark) negeert alle volgende tekens op een regel BASIC-code als een code-opmerking.

Voorbeeld: Gebruik van **REM**

```
10 REM *** PROGRAMMATITEL ***
20 N = 1000     : REM AANTAL ELEMENTEN
30 DIM NA$(N)
```

RENAME

Token: $F5

Formaat: **RENAME** oud **TO** nieuw [**,D** station] [**,U** eenheid]

Gebruik: Hernoemt een schijfbestand.

Het **oude** bestand is ofwel een string tussen aanhalingstekens, bijv. `"DATA"` of een stringexpressie tussen haakjes, bijv. `(FI$)`.

Het **nieuwe** bestand is ofwel een string tussen aanhalingstekens, bijv. `"BACKUP"` of een stringexpressie tussen haakjes, bijv. `(FS$)`.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

RENAME wordt uitgevoerd in de DOS van het schijfstation. Het kan alle reguliere bestandstypen hernoemen: **PRG**, **REL**, **SEQ**, **USR** en **CBM**. Het oude bestand moet bestaan en het nieuwe bestand mag niet bestaan. Er kunnen slechts enkele bestanden worden hernoemd; jokertekens zoals '*' en '?' zijn niet toegestaan. Het bestandstype kan niet worden gewijzigd.

Voorbeeld: Gebruik van **RENAME**

```
RENAME "CODES" TO "BACKUP" : REM HERNOEM ENKEL BESTAND OP STANDAARD UNIT
```

RENUMBER

Token: $F8

Formaat: **RENUMBER** [**C**] [{nieuw**,** stapgrootte**,** bereik}]

Gebruik: Herenummereert regels van een BASIC-programma.

Het **nieuwe** beginregelnummer van het regelbereik dat hernummerd moet worden. De standaardwaarde is 10.

De **stapgrootte** die moet worden gebruikt. De standaardwaarde is 10.

Het **bereik** van regels dat hernummerd moet worden. De standaardwaarden lopen van de eerste tot de laatste regel.

Indien de vlag **C** is opgegeven, verwijdert **RENUMBER** spaties die vóór regelnummers staan. Dit vergroot de kans dat de gewijzigde regelnummers de regel niet te lang maken voor weergave op het scherm, in zeldzame gevallen waarin de regel al dicht bij de maximale logische regellengte ligt. Dit heeft geen invloed op de werking van de code.

RENUMBER wijzigt alle regelnummers in het gekozen bereik en past ook alle verwijzingen aan in instructies die **GOSUB**, **GOTO**, **RESTORE**, **RUN**, **TRAP**, **COLLISION** enz. gebruiken.

RENUMBER kan alleen in de directe modus worden uitgevoerd. Indien een probleem wordt gedetecteerd, zoals geheugenoverloop, onopgeloste verwijzingen of regelnummeroverstroming (meer dan 64000 regels), wordt de uitvoering gestopt met een foutmelding en blijft het programma ongewijzigd.

RENUMBER kan met 0 - 3 parameters worden aangeroepen. Niet-opgegeven parameters gebruiken hun standaardwaarden.

Opmerkingen:

RENUMBER kan voor grote programma's enkele minuten nodig hebben om uit te voeren.

RENUMBER kan alleen in de directe modus worden gebruikt.

Dit commando maakt tijdelijk geheugen in banken 4 en 5 aan en kan eventueel opgeslagen gegevens overschrijven.

Voorbeelden:

Gebruik van **RENUMBER**

```
RENUMBER                : REM NUMMERS WORDEN 10, 20, 30, ...
RENUMBER 100, 5         : REM NUMMERS WORDEN 100, 105, 110, 115, ...
RENUMBER C 601, 1, 500  : REM OPTIMALISATIE, HERNUMMER BEGINNEND BIJ 500 TOT 601, 602, ...
RENUMBER 100, 5, 120-180 : REM HERNUMMER REGELS 120-180 NAAR 100, 105, ...

10 GOTO  20
20 GOTO 10

RENUMBER 100, 10        : REM BEHOUD SPATIES
100 GOTO  110
110 GOTO 100

RENUMBER C 100,10       : REM OPTIMALISATIE
100 GOTO110
110 GOTO100
```

RESTORE

Token: $8C

Formaat: **RESTORE** [regel]

Gebruik: Zet de interne pointer voor **READ** uit **DATA**-regels.

De nieuwe **regel** voor **RESTORE**. De standaardwaarde is de eerste programmaregel.

Opmerkingen:

De nieuwe pointerdoel**regel** hoeft geen **DATA**-regels te bevatten. Elke **READ** schuift de pointer automatisch door naar de volgende **DATA**-regel.

Voorbeeld: Gebruik van **RESTORE**

```
10 DATA 3, 1, 4, 1, 5, 9, 2, 6
20 DATA "MEGA65"
30 DATA 2, 7, 1, 8, 2, 8, 9, 5
40 FOR I = 1 TO 8 : READ P : PRINT P : NEXT
50 RESTORE 30
60 FOR I = 1 TO 8 : READ P : PRINT P : NEXT
70 RESTORE 20
80 READ A$ : PRINT A$
```

RESUME

Token: $D6

Formaat: **RESUME** [regel | **NEXT**]

Gebruik: Hervat de normale programma-uitvoering in een **TRAP**-routine na het afhandelen van een fout.

RESUME zonder parameters probeert de instructie die de fout veroorzaakte opnieuw uit te voeren. De **TRAP**-routine heeft de plaats waar de fout optrad inmiddels moeten onderzoeken en herstellen.

Het **regel**nummer waarop de programma-uitvoering wordt hervat.

NEXT hervat de uitvoering na de instructie die de fout veroorzaakte. Dit kan de volgende instructie op dezelfde regel zijn, gescheiden door een dubbele punt (:), of de instructie op de volgende regel.

Opmerkingen:

RESUME kan niet in de directe modus worden gebruikt.

Voorbeeld: Gebruik van **RESUME**

```
 10 TRAP 100
 20 FOR I = 1 TO 100
 30 PRINT EXP(I)
 40 NEXT
 50 PRINT "GESTOPT BIJ I ="; I
 60 END
100 PRINT ERR$(ER) : RESUME 50
```

RETURN

Token: $8E

Formaat: **RETURN**

Gebruik: Geeft de besturing terug vanuit een subroutine die is aangeroepen met **GOSUB** of een gebeurtenisafhandelaar die is gedeclareerd met **COLLISION**.

De uitvoering gaat verder bij de instructie na de **GOSUB**-aanroep.

In het geval van de **COLLISION**-afhandelaar gaat de uitvoering verder bij de instructie waarvandaan de afhandelaar werd aangeroepen.

Voorbeeld: Gebruik van **RETURN**

```
 10 SCNCLR            : REM MAAK SCHERM LEEG
 20 FOR I = 1 TO 20   : REM BEGIN EEN LUS
 30 GOSUB 100         : REM ROEP SUBROUTINE AAN
 40 NEXT I            : REM HERHAAL
 50 END               : REM EINDE PROGRAMMA
100 CURSOR ON, I, I, 0 : REM ACTIVEER EN POSITIONEER CURSOR
110 PRINT "X";        : REM PRINT X
120 SLEEP 0.5         : REM WACHT 0.5 SECONDEN
130 CURSOR OFF        : REM KNIPPERENDE CURSOR AFZETTEN
140 RETURN            : REM TERUG NAAR DE PLAATS VAN AANROEP
```

RGRAPHIC

Token: $CC

Formaat: **RGRAPHIC**(scherm, parameter)

Retourneert:
Bitmap-graphics: de status van een gegeven grafische scherminstelling.

Parameter	Beschrijving
0	Open (1), Gesloten (0), of Ongeldig (>1)
1	Breedte (0=320, 1=640)
2	Hoogte (0=200, 1=400)
3	Diepte (1 - 8 bitplanes)
4	Gebruikte bitplanes (bitmasker)
5	Gebruikte blokken bank 4 (bitmasker)
6	Gebruikte blokken bank 5 (bitmasker)
7	Tekenvenster # (0 - 3)
8	Weergavevenster # (0 - 3)
9	Tekentstanden (bitmasker)
10	Patroontype (bitmasker)

Voorbeeld: Gebruik van **RGRAPHIC**

```
 10 GRAPHIC CLR                              : REM INITIALISEER
 20 SCREEN DEF 0, 1, 0, 4                    : REM SCHERM 0: 640 X 200 X 4
 30 SCREEN OPEN 0                            : REM OPEN
 40 SCREEN SET 0, 0                          : REM TEKEN OP 0 EN TOON 0
 50 SCNCLR 0                                 : REM SCHERM LEEGMAKEN
 60 PEN 0, 1                                 : REM KIES KLEUR
 70 LINE 0, 0, 639, 199                      : REM TEKEN LIJN
 80 FOR I = 0 TO 10 : A(I) = RGRAPHIC(0, I) : NEXT
 90 SCREEN CLOSE 0
100 FOR I = 0 TO 6 : PRINT I; A(I) : NEXT    : REM PRINT INFO

RUN
 0  1
 1  1
 2  0
 3  4
 4  15
 5  15
 6  15
```

RIGHT$

Token: $C9

Formaat: **RIGHT$**(tekenreeks**,** n**)**

Retourneert:

Een tekenreeks met de laatste **n** tekens van **tekenreeks**.

Indien de lengte van **tekenreeks** kleiner of gelijk is aan **n**, is de resulterende tekenreeks identiek aan het argument.

tekenreeks tekenreeksexpressie.

n numerieke expressie (0 - 255).

Opmerkingen:

Lege tekenreeksen en lengtes van nul zijn toegestane waarden.

Voorbeeld: Gebruik van **RIGHT$**

```
PRINT RIGHT$("MEGA65", 2)
65
```

RMOUSE

Token: $FE $3F

Formaat: **RMOUSE** x-variabele**,** y-variabele**,** knopvariabele

Gebruik: Leest de muispositie en de knopstatus.

x-variabele numerieke variabele waarin de x-positie wordt opgeslagen.

y-variabele numerieke variabele waarin de y-positie wordt opgeslagen.

knopvariabele numerieke variabele die de knopstatus ontvangt. De linker knop zet bit 7, de rechter knop zet bit 0.

De coördinaten zijn compatibel met sprite-coördinaten. Dit is beperkt tot het zichtbare scherm binnen de rand, met de coördinaten (24, 50) linksboven.

Waarde	Status
0	Geen knop
1	Rechter knop
128	Linker knop
129	Beide knoppen

Opmerkingen:

RMOUSE plaatst -1 in alle variabelen indien de muis niet is aangesloten of is uitgeschakeld.

Voorbeeld: Gebruik van **RMOUSE**

```
10 MOUSE ON, 1, 1                              : REM MUIS IN POORT 1 MET SPRITE 1
20 RMOUSE XP, YP, BU                           : REM LEES MUISSTATUS
30 IF XP < 0 THEN PRINT "GEEN MUIS IN POORT 1" : STOP
40 PRINT "MUIS:"; XP; YP; BU
50 MOUSE OFF                                   : REM MUIS UIT
```

RND

Token: $BB

Formaat: **RND**(type)

Retourneert:

Een pseudowillekeurig getal.

Dit wordt een “pseudowillekeurig” getal genoemd, omdat computers geen werkelijk willekeurige getallen kunnen genereren. Pseudowillekeurige getallen worden rekenkundig afgeleid van een ander getal, een “seed” genoemd, dat reproduceerbare reeksen genereert. **type** bepaalt welke seedmethode wordt gebruikt:

- type = 0: Gebruik de systeemklok.
- type < 0: Gebruik de waarde van **type** als seed.
- type > 0: Leid een nieuw willekeurig getal af van het vorige.

Opmerkingen:

Willekeurige getallenreeksen met een seed produceren dezelfde reeks voor identieke seeds.

Het algoritme wordt tijdens het opstarten geïnitialiseerd met een seed vanuit de Real-Time Clock en andere factoren, waardoor **RND**(1) onwaarschijnlijk dezelfde reeks twee keer retourneert. Dit verschilt van de Commodore 64, die altijd dezelfde initiële seed gebruikte. Indien **RND** ooit met een negatieve waarde wordt aangeroepen, wordt die waarde als nieuwe seed gebruikt en worden de reeksen die door **RND**(1) worden gegenereerd voorspelbaar. Dit is met name nuttig bij het uitvoeren van bepaalde tests. Gebruik **RND**(0) om opnieuw te seeden met een onvoorspelbare waarde.

Elke aanroep van **RND**(0) genereert een nieuwe seed op basis van de systeemklok en andere factoren. Het herhaald aanroepen van **RND**(0) produceert doorgaans een betere verdeling van waarden dan op een Commodore 64, vanwege de precisie van de seedbronnen.

Voorbeeld: Gebruik van **RND**

```
10 DEF FN DI(X) = INT(RND(0) * 6) + 1 : REM DUBBELSTEENFUNCTIE
20 FOR I = 1 TO 10                    : REM 10 KEER TONEN
30 PRINT I; FN DI(0)                  : REM PRINT GEDOBBELDE WAARDE
40 NEXT
```

RPALETTE

Token: $CE $0D

Formaat: **RPALETTE**(scherm**,** index**,** rgb)

Retourneert:

De rode, groene of blauwe waarde van een kleurenpaletindex.

scherm schermnummer (0 - 3), of een negatieve waarde om een van de vier systeempaletten te selecteren: -1 voor systeempalet 0 (het standaard systeempalet), -2 voor systeempalet 1, -3 voor palet 2, of -4 voor palet 3.

index kleurenpaletindex.

rgb (0: rood, 1: groen, 2: blauw).

Voorbeeld: Gebruik van **RPALETTE**

```
10 SCREEN 320, 200, 4                : REM DEFINIEER EN OPEN SCHERM
20 R = RPALETTE(0, 3, 0)             : REM LEES ROOD
30 G = RPALETTE(0, 3, 1)             : REM LEES GROEN
40 B = RPALETTE(0, 3, 2)             : REM LEES BLAUW
50 SCREEN CLOSE                      : REM SLUIT SCHERM
60 PRINT "PALETINDEX 3 RGB ="; R; G; B

RUN
PALETINDEX 3 RGB = 0  15  15
```

RPEN

Token: $D0

Formaat: **RPEN**(n)

Retourneert:

De kleurindex van pen **n**.

n pennummer (0 – 2), waarbij:

- 0: Tekenpen.
- 1: Wispen.
- 2: Omtrekpen.

Voorbeeld: Gebruik van **RPEN**

```
 10 GRAPHIC CLR          : REM INITIALISEER
 20 SCREEN DEF 0, 1, 0, 4 : REM SCHERM 0: 640 X 200 X 4
 30 SCREEN OPEN 0         : REM OPEN
 40 SCREEN SET 0, 0       : REM TEKEN OP 0 EN TOON 0
 50 SCNCLR 0              : REM LEEGMAKEN
 60 PEN 0, 1              : REM KIES KLEUR
 70 X = RPEN(0)
 80 Y = RPEN(1)
 90 C = RPEN(2)
100 SCREEN CLOSE 0
110 PRINT "KLEUR VAN TEKENPEN = "; X

RUN
KLEUR VAN TEKENPEN =  1
```

RPLAY

Token: $CE $0F

Formaat: **RPLAY**(stem)

Retourneert:

Test of er muziek wordt afgespeeld op het opgegeven stemkanaal.

stem het te beoordelen stemkanaal, van 1 tot en met 6.

Retourneert 1 indien er muziek op het kanaal wordt afgespeeld, anders 0.

Voorbeeld: Gebruik van **RPLAY**

```
10 PLAY "O4ICDEFGABO5CR", "O2QCGEGCO1GCR"
20 IF RPLAY(1) OR RPLAY(2) THEN GOTO 20   : REM WACHT OP EINDE LIEDJE
```

RPT$

Token: $CE $14

Formaat: **RPT$**(tekenreeks, aantal)

Retourneert:
Een nieuwe tekenreeks die bestaat uit **tekenreeks** herhaald **aantal** keer.

Opmerkingen:
Indien de nieuwe tekenreeks langer zou zijn dan 255 tekens, wordt een String Too Long-fout gerapporteerd.

Indien **tekenreeks** een lege tekenreeks is of **aantal** nul bedraagt, retourneert **RPT$()** de lege tekenreeks.

Voorbeeld: Gebruik van **RPT$**

```
PRINT RPT$("=", 30)
==============================

PRINT RPT$("HALLO ", 3),"DAAR"
HALLO HALLO HALLO DAAR
```

RREG

Token: $FE $09

Formaat: **RREG** [{areg, xreg, yreg, zreg, sreg}]

Gebruik: Leest de waarden die zich in de CPU-registers bevonden na een **SYS**-aanroep, in de opgegeven variabelen.

areg accumulatoreind waarde.

xreg X-registerwaarde.

yreg Y-registerwaarde.

zreg Z-registerwaarde.

sreg statusregisterwaarde.

Opmerkingen:
De registerwaarden na een **SYS**-aanroep worden in het systeemgeheugen opgeslagen. Hierdoor kan **RREG** ze ophalen.

Voorbeeld: Gebruik van **RREG**

```
10 POKE $1800, $18, $8A, $65, $06, $60
20 REM        CLC  TXA  ADC   06  RTS
30 SYS $1800, 77, 11                : REM A = 77 EN X = 11
40 RREG AC, X, Y, Z, S
50 PRINT "REGISTER:"; AC; X; Y; Z; S
```

RSPCOLOR

Token: $CE $07

Formaat: **RSPCOLOR**(n)

Retourneert:
De kleurinstelling van een multicolor-spritekleur.

n multicolor-spritenummer:

- 1: Haal multicolor #1 op.
- 2: Haal multicolor #2 op.

Opmerkingen:
Raadpleeg **SPRITE** en **SPRCOLOR** voor meer informatie.

Voorbeeld: Gebruik van **RSPCOLOR**

```
10 SPRITE 1, 1      : REM ZET SPRITE 1 AAN
20 C1% = RSPCOLOR(1) : REM LEES KLEUR #1
30 C2% = RSPCOLOR(2) : REM LEES KLEUR #2
```

RSPEED

Token: $CE $0E

Formaat: **RSPEED**(n)

Retourneert:

De huidige CPU-kloksnelheid in MHz.

n numeriek dummyargument dat wordt genegeerd.

De retourwaarde is een geheel getal, dat overeenkomt met de werkelijke CPU-snelheid als volgt:

- 1: 1 MHz
- 3: 3.5 MHz
- 40: 40.5 MHz

Opmerkingen:

RSPEED(n) retourneert niet de juiste waarde indien eerder `POKE 0, 65` is gebruikt om de hoogste snelheid (40 MHz) in te schakelen.

De MEGA65 ondersteunt ook het draaien van de CPU op 2 MHz, vergelijkbaar met de Commodore 128. Dit is niet beschikbaar als retourwaarde voor **RSPEED**. In dat geval retourneert **RSPEED** 1.

Raadpleeg het **SPEED**-commando voor meer informatie.

Voorbeeld: Gebruik van **RSPEED**

```
10 X = RSPEED(0)                        : REM VRAAG KLOKSNELHEID OP
20 IF X = 1  THEN PRINT "1 MHZ"   : GOTO 50
30 IF X = 3  THEN PRINT "3.5 MHZ" : GOTO 50
40 IF X = 40 THEN PRINT "40 MHZ"
50 END
```

RSPPOS

Token: $CE $05

Formaat: **RSPPOS**(sprite, n)

Retourneert:
De positie of snelheid van een sprite.

sprite het spritenummer.

n spriteparameter om op te halen:

- 0: X-positie.
- 1: Y-positie.
- 2: Snelheid.

Opmerkingen:
Raadpleeg de commando's **MOVSPR** en **SPRITE** voor meer informatie.

Voorbeeld: Gebruik van **RSPPOS**

```
10 SPRITE 1, 1      : REM ZET SPRITE 1 AAN
20 XP = RSPPOS(1, 0) : REM LEES X VAN SPRITE 1
30 YP = RSPPOS(1, 1) : REM LEES Y VAN SPRITE 1
30 SP = RSPPOS(1, 2) : REM LEES SNELHEID VAN SPRITE 1
```

RSPRITE

Token: $CE $06

Formaat: **RSPRITE**(sprite, n)

Retourneert:

Een spriteparameter.

sprite sprite-nummer (0 - 7)

n spriteparameter om terug te geven (0 - 5):

- 0: Ingeschakeld (0 of 1). 0 betekent dat de sprite uitgeschakeld is.
- 1: Voorgrondkleur (0 - 15).
- 2: Achtergrondprioriteit (0 of 1).
- 3: X-uitgebreid (0 of 1). 0 betekent dat de sprite niet X-uitgebreid is.
- 4: Y-uitgebreid (0 of 1). 0 betekent dat de sprite niet Y-uitgebreid is.
- 5: Multikleur (0 of 1). 0 betekent dat de sprite geen multikleur heeft.

Opmerkingen:

Raadpleeg de commando's **MOVSPR** en **SPRITE** voor meer informatie.

Voorbeeld: Gebruik van **RSPRITE**

```
10 SPRITE 1, 1       : ZET SPRITE 1 AAN
20 EN = RSPRITE(1, 0) : REM SPRITE 1 INGESCHAKELD ?
30 FG = RSPRITE(1, 1) : REM SPRITE 1 VOORGRONDKLEURINDEX
40 BP = RSPRITE(1, 2) : REM SPRITE 1 ACHTERGRONDPRIORITEIT
50 XE = RSPRITE(1, 3) : REM SPRITE 1 X VERDUBBELD ?
60 YE = RSPRITE(1, 4) : REM SPRITE 1 Y VERDUBBELD ?
70 MC = RSPRITE(1, 5) : REM SPRITE 1 MEERKLEUREN ?
```

RSPRSYS

Token: $CE $17

Formaat: **RSPRSYS**(n)

Retourneert:
Een spritesysteemparameter.

n parameter om terug te geven (0):

- 0: Hoge-resolutiemodus van sprite. 1=ingeschakeld, 0=uitgeschakeld.

Opmerkingen:
Raadpleeg het commando **SPRITE SYS** voor meer informatie.

Voorbeeld: Gebruik van **RSPRSYS**

```
SPRITE SYS R1

PRINT RSPRSYS(0)
1
```

RUN

Token: $8A

Formaat: **RUN** [regelnummer]
RUN bestandsnaam [**,D** station] [**,U** eenheid]
[↑] bestandsnaam

Gebruik: Voert het BASIC-programma in het geheugen uit of laadt en voert een programma vanaf schijf uit.

Indien een bestandsnaam is opgegeven, wordt het programmabestand in het geheugen geladen en uitgevoerd; anders wordt het programma dat zich op dat moment in het geheugen bevindt, gebruikt.

De ↑ kan als sneltoets worden gebruikt wanneer deze in de directe modus in de linkerste kolom staat. U kunt hiermee een programma vanuit een directorylijst laden en uitvoeren door de cursor naar de regel met de bestandsnaam te verplaatsen, de ↑ aan het begin van de regel te typen en op return te drukken. Tekens vóór en achter de aangehaalde bestandsnaam worden genegeerd (vergelijkbaar met PRG bijvoorbeeld).

regelnummer een bestaand regelnummer van het programma in het geheugen vanaf welke het programma moet worden uitgevoerd.

bestandsnaam is een aangehaalde tekenreeks, bijv. "PROG" of een tekenreeksexpressie in haakjes, bijv. (PR$). Het bestandstype moet **PRG** zijn.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

RUN herstelt eerst alle interne pointers naar hun standaardwaarden. Er zijn derhalve geen variabelen, arrays of strings gedefinieerd. De runtijdstack wordt eveneens hersteld en de tabel met geopende bestanden wordt gewist.

Om de programma-uitvoering te starten of voort te zetten zonder alles te herstel, gebruikt u **GOTO**.

Voorbeelden:

Gebruik van **RUN**

```
RUN "VLUCHTSIM" : REM LAAD EN PROGRAMMA "FLIGHTSIM" EN VOER HET UIT

RUN 1000        : REM VOER PROGRAMMA IN HET GEHEUGEN UIT, BEGIN OP REGEL 1000

RUN             : REM VOER PROGRAMMA IN HET GEHEUGEN UIT
```

RWINDOW

Token: $CE $09

Formaat: **RWINDOW**(n)

Retourneert:

Een parameter van het huidige tekstvenster.

n de schermparameter om op te halen:

- 0: Breedte van het huidige tekstvenster.
- 1: Hoogte van het huidige tekstvenster.
- 2: Aantal kolommen op het scherm (40 of 80).
- 3: Aantal regels op het scherm (25 of 50).

Opmerkingen:

Raadpleeg het commando **WINDOW** voor meer informatie.

Voorbeeld: Gebruik van **RWINDOW**

```
10 W = RWINDOW(2)          : REM LEES SCHERMBREEDTE
20 IF W = 80 THEN BEGIN    : REM IS 80 KOLOMSMODE ACTIEF?
30   PRINT CHR$(27) + "X"; : REM JA, SCHAKEL NAAR 40 KOLOMMEN
40 BEND
```

SAVE

Token: $94

Formaat: **SAVE** bestandsnaam [**,** eenheid]
[←] bestandsnaam [**,** eenheid]

Gebruik: Slaat een BASIC-programma op in een bestand van het type **PRG**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

SAVE gebruikt de verouderde syntaxis voor besturingstekens in de bestandsnaam. Zo vervangt "@0:BESTANDSNAAM" een bestaand bestand op station 0 van een multi-stationeenheid met de naam BESTANDSNAAM. Deze syntaxis verschilt van die van nieuwere commando's zoals **DSAVE**.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

SAVE is verouderd en alleen ter compatibiliteit met oudere programma's geïmplementeerd. Gebruik in plaats daarvan **DSAVE**.

De sneltoets [←] bevindt zich naast **1**. Kan alleen in de directe modus worden gebruikt.

OPMERKING: Opslaan met vervanging wordt niet aanbevolen op 1541- en 1571-stations, vanwege een bug in oudere versies van het besturingssysteem van het schijfstation welke gegevensverlies kan veroorzaken.

Voorbeelden:

Gebruik van **SAVE**

```
SAVE "AVONTUUR"

SAVE "ZORK-I", 8

SAVE "1:KERKERGEWELF", 9
```

SAVEIFF

Token: $FE $44

Formaat: **SAVEIFF** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Bitmapgraphics: slaat het huidige grafische scherm op in een schijfbestand in **IFF**-formaat.

IFF (Interchange File Format) wordt ondersteund door veel verschillende toepassingen en besturingssystemen. **SAVEIFF** slaat de afbeelding, het kleurengeheugen en de resolutieparameters op.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$) De maximale lengte van de bestandsnaam is 16 tekens. Indien het eerste teken van de bestandsnaam een '@' (apestaartje) is, wordt dit geïnterpreteerd als een "opslaan en vervangen"-bewerking. Het wordt niet aanbevolen deze optie te gebruiken op 1541- en 1571-stations, omdat deze een "opslaan en vervangen-fout" in hun DOS bevatten.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Bestanden die met **SAVEIFF** zijn opgeslagen, kunnen worden geladen met **LOADIFF**. Er zijn programma's beschikbaar om veelgebruikte afbeeldingsformaten naar IFF te converteren. Deze programma's zijn beschikbaar voor diverse besturingssystemen, zoals Amiga OS, macOS, Linux en Windows. **ImageMagick** is bijvoorbeeld een gratis grafisch hulpprogramma dat een tool genaamd **convert** bevat, waarmee IFF-bestanden kunnen worden gemaakt in combinatie met de tool **ppmtoilbm** uit het **Netbpm**-pakket.

Voorbeeld: Gebruik van **SAVEIFF**

```
10 SCREEN 320, 200, 2        : REM SCHER, #0 320 X 200 X 2
20 PEN 1                     : REM TEKENPENKLEUR 1 (WIT)
30 LINE 25, 25, 295, 175     : REM TEKEN LIJN
40 SAVEIFF "LINE-EXAMPLE", U8 : REM BEWAAR HUIDIG BEELD IN BESTAND
50 SCREEN CLOSE              : REM GRAFISCH SCHERM UIT EN HERSTEL PALET
```

SCNCLR

Token: $E8

Formaat: **SCNCLR** [kleur]

Gebruik: Wist een tekstvenster of een bitmapgrafisch scherm.

SCNCLR (zonder argumenten) wist het huidige tekstvenster. Het standaardvenster beslaat het volledige scherm.

SCNCLR kleur wist het grafische scherm door het te vullen met de opgegeven **kleur**.

Voorbeeld: Gebruik van **SCNCLR**

```
 10 GRAPHIC CLR             : REM INITIALISEER
 20 SCREEN DEF 1, 0, 0, 2   : REM SCHERM #1 320 X 200 X 2
 30 SCREEN OPEN 1           : REM OPEN SCHERM 1
 40 SCREEN SET 1, 1         : REM OP SCHERM 1 TEKENEN, SCHERM 1 TONEN
 50 SCNCLR 0                : REM SCHERM SCHOONMAKEN
 60 PALETTE 1, 1, 15, 15, 15 : REM DEFIEER KLEUR 1 ALS WIT
 70 PEN 0, 1                : REM TEKENPEN
 80 LINE 25, 25, 295, 175   : REM TEKEN LIJN
 90 SLEEP 10                : REM WACHT 10 SECONDEN
100 SCREEN CLOSE 1          : REM GRAFISCH SCHERM UIT EN HERSTEL PALET
```

SCRATCH

Token: $F2

Formaat: **SCRATCH** bestandsnaam [**,D** station] [**,U** eenheid] [**,R**]

Gebruik: Verwijdert ("krast") een schijfbestand.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

R herstelt een eerder verwijderd bestand. Dit werkt alleen indien er geen schrijfbewerkingen hebben plaatsgevonden tussen verwijdering en herstel, waardoor de inhoud van de schijf mogelijk is gewijzigd.

Opmerkingen:

SCRATCH is een synoniem van **ERASE** en **DELETE**.

In de directe modus wordt het succes en het aantal verwijderde bestanden afgedrukt. De voorlaatste waarde uit het bericht bevat het aantal succesvol verwijderde bestanden.

Voorbeelden:

Gebruik van **SCRATCH**

```
SCRATCH "DRM", U9          : REM VERWIJDER BESTAND "DRM" OP UNIT 9
01, FILES SCRATCHED,01,00

SCRATCH "OUD*"             : REM VERWIJDER ALLE BESTANDEN BEGINNEND OP "OUD"
01, FILES SCRATCHED,04,00

SCRATCH "R*=PRG"           : REM VRWIJDER ALLE PROGRAMMABESTANDEN BEGINNEND OP 'R'
01, FILES SCRATCHED,09,00
```

SCREEN

Token: $FE $2E

Formaat: **SCREEN** [scherm,] breedte**,** hoogte**,** diepte
SCREEN CLR kleur
SCREEN DEF scherm**,** breedte-vlag**,** hoogte-vlag**,** diepte
SCREEN SET teken-scherm**,** weergave-scherm
SCREEN OPEN [scherm]
SCREEN CLOSE [scherm]

Gebruik: Bitmapgraphics: beheert een grafisch scherm.

Bij het voorbereiden van het scherm voor het tekenen van graphics zijn er twee benaderingen beschikbaar: een vereenvoudigde benadering en een gedetailleerde benadering.

Vereenvoudigde benadering:

De eerste versie van **SCREEN** (met pixelunits voor breedte en hoogte) is de eenvoudigste manier om een grafisch scherm te starten en is de aanbevolen methode indien slechts één scherm nodig is (d.w.z. er is geen tweede scherm nodig voor double buffering). Deze versie voert alle voorbereidende werkzaamheden voor u uit en aanvaardt commando's zoals **GRAPHIC CLR**, **SCREEN CLR**, **SCREEN DEF**, **SCREEN OPEN** en **SCREEN SET** namens u uit. De volgende parameters worden ondersteund:

SCREEN [scherm,] breedte, hoogte, diepte

- screen: Het schermnummer (0 - 3) is optioneel. Indien geen schermnummer is opgegeven, wordt scherm 0 gebruikt. Om deze benadering zo eenvoudig mogelijk te houden, wordt aangeraden het standaardscherm 0 te gebruiken.
- width: 320 of 640 (standaard 320).
- height: 200 of 400 (standaard 200).
- depth: 1..8 (standaard = 8), kleuren = 2 ^depth.

De argumentenparsing is foutbestendig en gebruikt standaardwaarden voor breedte (320) en hoogte (200) indien het geparsede argument ongeldig is.

Deze versie van **SCREEN** start met een voorgedefinieerd kleurengeheugen, stelt de achtergrond in op zwart en de pen in op wit, zodat er direct met de standaardwaarden kan worden getekend.

De gedetailleerde benadering vereist daarentegen het instellen van het kleurengeheugen en de penkleur voordat er kan worden getekend.

De **kleur**-waarde moet in het bereik 0 - 15 liggen. Zie bijlage 6 op bladzijde 327 voor de lijst met kleuren in het standaard kleurenpalet.

Wanneer u klaar bent met uw grafische scherm, roept u eenvoudig **SCREEN CLOSE** [scherm] aan om terug te keren naar het tekstscherm.

Gedetailleerde benadering:

De overige versies van **SCREEN** voeren specifieke acties uit en zijn bedoeld voor geavanceerde grafische programma's die meerdere schermen openen of "double buffering" vereisen. Indien u de vereenvoudigde benadering heeft gekozen, heeft u geen van deze versies nodig, met uitzondering van **SCREEN CLOSE**.

SCREEN CLR kleur (of **SCNCLR** kleur).
Wist het actieve grafische scherm door het te vullen met **kleur**.

SCREEN DEF scherm, breedtevlag, hoogtevlag, dieptevlag
Definieert de resolutieparameters voor het geselecteerde scherm. De breedte- en hoogtevlag geven aan of hoge resolutie (1) of lage resolutie (0) is geselecteerd.

- screen: Schermnummer 0 - 3.
- breedtevlag: 0 - 1 (0:320, 1:640 pixel).
- hoogtevlag: 0 - 1 (0:200, 1:400 pixel).
- depth: 1 - 8 (2 - 256 kleuren).

Merk op dat de waarden voor breedte en hoogte hier **vlaggen** zijn en **geen pixeleenheden**.

SCREEN SET tekenscherm, toonscherm
Stelt schermnummers (0 - 3) in voor het teken- en het weergavescherm; terwijl het ene scherm wordt weergegeven, kunt u op een apart scherm tekenen en later tussen de schermen wisselen. Dit staat bekend als dubbele buffering.

SCREEN OPEN scherm
Kent bronnen toe en initialiseert de grafische context voor het geselecteerde **screen** (0 - 3). Een optionele variabelenaam als extra argument ontvangt het resultaat van het commando, waarna het op succes kan worden getest.

SCREEN CLOSE [scherm]
Sluit **scherm** (0 - 3) en geeft bronnen vrij. Indien geen waarde is opgegeven, wordt standaard 0 gebruikt. Merk op dat het sluiten van een scherm automatisch **PALETTE RESTORE** uitvoert.

Opmerkingen:

Grafische schermen worden toegewezen in bank 4 en 5 van het geheugen. Indien een BASIC-programma ook geheugen in bank 4 of 5

gebruikt, kunnen geheugengebieden worden gereserveerd om te voorkomen dat **SCREEN** deze toewijst. Dit gebeurt met het commando **MEM**. Zie **MEM**.

Voorbeelden:

Gebruik van **SCREEN**

```
 5 REM *** VEREENVOUDIGDE AANPAK ***
10 SCREEN 320, 200, 2    : REM SCHERM #0: 320 X 200 X 2
20 PEN 1                 : REM TEKENPENKLEUR = 1 (WIT)
30 LINE 25, 25, 295, 175 : REM TEKEN LIJN
40 GETKEY A$             : REM WACHT OP TOETSDRUK
50 SCREEN CLOSE          : REM SLUIT SCHERM 0 (HERSTEL PALET)
```

```
 5 REM *** UITGEBREIDE AANPAK ***
10 GRAPHIC CLR             : REM INITIALISEER
20 SCREEN DEF 1, 0, 0, 2   : REM SCHERM #1: 320 X 200 X 2
30 SCREEN OPEN 1           : REM OPEN SCHERM 1
40 SCREEN SET 1, 1         : REM TEKEN OP SCHERM 1 EN TOON SCHERM 1
50 SCREEN CLR 0            : REM SCHERM SCHOONMAKEN
60 PALETTE 1, 1, 15, 15, 15 : REM DEFINIEER KLEUR 1 ALS WIT
70 PEN 0, 1                : REM TEKENPEN
80 LINE 25, 25, 295, 175   : REM TEKEN EEN LIJN
90 SLEEP 10                : REM WACHT 10 SECONDEN
100 SCREEN CLOSE 1         : REM SLUIT SCHERM 1 (HERSTEL PALET)
```

SET

Token: $FE $2D

Formaat: **SET DEF** eenheid
SET DISK oud **TO** nieuw
SET FAST MOUSE <**ON** | **OFF**>
SET VERIFY <**ON** | **OFF**>

Gebruik: **SET DEF** herbepaalt de standaardeenheid voor schijftoegang, welke door de DOS op 8 is geïnitialiseerd. Commando's die geen eenheid expliciet specificeren, gebruiken deze standaardeenheid.

SET DISK wordt gebruikt om het eenheidsnummer van een schijfstation tijdelijk te wijzigen.

SET FAST MOUSE schakelt het verdubbelen van de snelheid van de BASIC **MOUSE** in of uit.

SET VERIFY schakelt de DOS verify-after-write-modus voor 3.5"-stations in of uit.

Opmerkingen:
Deze instellingen zijn geldig totdat een reset of afsluiting plaatsvindt.

Voorbeelden:
Gebruik van **SET**

```
DIR             : REM TOON DIRECTORY VAN UNIT 8

SET DEF 11      : REM UNIT 11 WORDT STANDAARD

DIR             : REM TOON DIRECTORY OP UNIT 11

DLOAD "*"       : REM LAAD EERSTE BESTAND VAN UNIT 11

SET DISK 8 TO 9 : REM VERANDER UNIT# VAN STATION 8 NAAR 9

DIR U9          : REM TOON DIRECTORY VAN UNIT 9 (VOORHEEN 8)

SET FAST MOUSE ON  : REM MUIS DUBBEL ZO SNEL

SET VERIFY ON   : REM ACTIVEER VERIFICATIE NA SCHRIJVEN
```

SETBIT

Token: $FE $2D $FE $4E

Formaat: **SETBIT** adres**,** bitnummer

Gebruik: Zet een enkel bit op het opgegeven **adres**.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

bitnummer waarde in het bereik van 0 - 7.

Een bankwaarde > 127 wordt gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enz.

Voorbeeld: Gebruik van **SETBIT**

```
10 BANK 128       : REM KIES SYSTEEMBANK
20 SETBIT $D011, 6 : REM SCHAKEL UITGEBREIDE ACHTERGRONDKLEURMODUS IN
30 SETBIT $D01B, 0 : REM ZET ACHTERGRONDPRIORITEIT VOOR SPRITE 0
```

SGN

Token: $B4

Formaat: **SGN**(numerieke expressie)

Retourneert:
Het teken van een numerieke expressie, als getal.

- -1: Negatief argument.
- 0: Nul.
- 1: Positief, niet-nul argument.

Voorbeeld: Gebruik van **SGN**

```
10 ON SGN(X) + 2 GOTO 100, 200, 300 : REM DOELEN VOOR MINUS, NUL, PLUS
20 Z = SGN(X) * ABS(Y)              : REM COMBINEER TEKEN VAN X MET WAARDE VAN Y
```

SIN

Token: $BF

Formaat: **SIN**(expressie in radialen)
SIND(expressie in graden)

Retourneert:

De sinus van een hoek.

De **SIN**-functie verwacht een argument in radialen.

De **SIND**-variant verwacht een argument in graden.

Het resultaat ligt in het bereik (-1.0 tot +1.0).

Voorbeelden:

Gebruik van **SIN**

```
PRINT SIN(0.7)
 0.64421769

X = 30 : PRINT SIN(X * π / 180)
 0.5

PRINT SIND(30)
 0.5
```

SLEEP

Token: $FE $0B

Formaat: **SLEEP** seconden

Gebruik: Pauzeert de uitvoering voor de opgegeven duur.

Het argument is een positief zwevend-kommagetal in seconden. De precisie bedraagt 1 microseconde.

Opmerkingen:

Het indrukken van RUN STOP onderbreekt de slaap.

Voorbeelden:

Gebruik van **SLEEP**

```
SLEEP 10     : REM WACHT 10 SECONDEN

SLEEP 0.0005 : REM SLAAP 500 MICROSECONDEN

SLEEP 0.01   : REM SLAAP  10 MILLISECONDEN

SLEEP DD     : REM HAAL DE SLAAPTIJD UIT VARIABLE DD

SLEEP 600    : REM SLAAP 10 MINUTEN
```

SOUND

Token: $DA

Formaat: **SOUND** stem**,** frequentie**,** duur [{**,** dir**,** minimum**,** zwaai**,** golf**,** puls}]
SOUND CLR

Gebruik: **SOUND** speelt een geluidseffect af.

voice stemnummer (1 - 6).

frequentie frequentie (0 - 65535).

duur duur in jiffy's (0 - 32767). Een jiffy is afhankelijk van de beeldstandaard. Bij PAL zijn er 50 jiffy's per seconde en bij NTSC 60.

dir richting (0: omhoog, 1: omlaag, 2: oscillerend).

min minimumfrequentie (0 - 65535).

sweep zwaaistap (0 - 65535): de hoeveelheid die elke jiffy bij de frequentie wordt opgeteld (richting 0) of daarvan wordt afgetrokken (richting 1). De frequentie zwaait tussen **min** en **freq**, slaat om naar het andere uiteinde zodra hij er voorbij komt, of keert om wanneer **dir** gelijk is aan 2 (oscilleren).

wave golfvorm (0: driehoek, 1: zaagtand, 2: vierkant, 3: ruis).

pulse pulsbreedte (0 - 4095).

SOUND CLR dempt alle klank van **SOUND** en **PLAY** en herstelt het klanksysteem en alle parameters naar hun standaardwaarde.

Opmerkingen:

SOUND begint het geluidseffect af te spelen en gaat onmiddellijk verder met de uitvoering van het volgende BASIC-statement terwijl het geluidseffect wordt afgespeeld. Hierdoor kunnen afbeeldingen of tekst worden weergegeven en tegelijkertijd geluiden worden afgespeeld.

SOUND gebruikt SID2 (voor stemmen 1 tot 3) en SID4 (voor stemmen 4 tot 6) van de vier SID-chips van het systeem. Standaard zijn SID1 en SID2 enigszins naar rechts gedraaid en SID3 en SID4 enigszins naar links in de stereomix.

De zes stemkanalen die door het **SOUND**-commando worden gebruikt (op SID2 en SID4) zijn onderscheiden van de zes kanalen die door het **PLAY**-commando worden gebruikt (op SID1 en SID3). Geluidseffecten onderbreken geen muziek en omgekeerd.

Voorbeeld: Gebruik van **SOUND**

```
10 IF PEEK($D06F) AND $80 THEN J = 60 : ELSE J = 50 : REM J IS JIFFIES PER SECONDE
20 SOUND 1, 7382, J                        : REM SPEEL BLOKGOLF OP STEM 1, 1 SECONDE LANG
30 SOUND 2,  800, J*60                     : REM SPEEL BLOKGOLF OP STEM 2, 1 MINUUT LANG
40 SOUND 3, 4000, 120, 2, 2000, 400, 1        : REM SPEEL ZWAAIENDE ZAAGTAND OP STEM 3
50 SLEEP 3                                    : REM LAAT DE GELUIDEN 3 SECONDEN SPELEN
60 SOUND CLR                                    : REM GELUID STIL, HERSTEL PARAMETERS
```

SPC

Token: $A6

Formaat: **SPC**(kolommen)

Retourneert:

Als argument voor **PRINT** een tekenreeks van PETSCII-codes voor cursor naar rechts, geschikt om de cursor het opgegeven aantal kolommen verder te plaatsen.

Het afdrukken hiervan komt overeen met het indrukken van → **<kolom>** keer.

Dit is geen echte functie en genereert geen tekenreeks. Het kan alleen als argument voor **PRINT** worden gebruikt.

Opmerkingen:

De naam van deze functie is afgeleid van “spaces”, wat misleidend is. De functie drukt **cursor-rechts tekens** af, geen spaties. De inhoud van de tekenposities die worden overgeslagen, wordt niet gewijzigd.

Voorbeeld: Gebruik van **SPC**

```
10 FOR I = 8 TO 12
20 PRINT SPC(-(I < 10)); I : REM TRUE = -1, FALSE = 0
30 NEXT I
RUN
 8
 9
10
11
12
```

SPEED

Token: $FE $26

Formaat: **SPEED** [snelheid]

Gebruik: Stelt de processorkloksnelheid in op 1 MHz, 3.5 MHz of 40.5 MHz.

snelheid kloksnelheid van de CPU, waarbij:

- 1: Stelt de CPU in op 1 MHz.
- 3: Stelt de CPU in op 3.5 MHz.
- Elke andere waarde dan **1** of **3** stelt de CPU in op 40.5 MHz.

Opmerkingen:

Hoewel **SPEED** met elk zwevend-kommagetal kan worden aangeroepen, wordt het deel na de decimale punt (de punt en alle cijfers daarna) genegeerd.

SPEED is een synoniem van **FAST**.

SPEED heeft geen effect indien `POKE 0, 65` eerder is gebruikt om de CPU op 40 MHz in te stellen.

De MEGA65 ondersteunt ook het draaien van de CPU op 2 MHz, vergelijkbaar met de Commodore 128. Dit is niet beschikbaar als optie voor **SPEED**.

Voorbeelden:

Gebruik van **SPEED**

```
SPEED     : REM SET SNELHEID OP MAXIMUM (40 MHZ)

SPEED 1   : REM SET SNELHEID OP 1 MHZ

SPEED 3   : REM SET SNELHEID OP 3.5 MHZ

SPEED 3.5 : REM SET SNELHEID OP 3.5 MHZ
```

SPRCOLOR

Token: $FE $08

Formaat: **SPRCOLOR** [{mc1**,** mc2}]

Gebruik: Stelt de meerkleuren van sprites in.

SPRITE, dat de attributen van een sprite instelt, stelt alleen de voorgrondkleur in. Gebruik **SPRCOLOR** voor het instellen van de twee extra kleuren van multikleur-sprites.

Opmerkingen:

De kleuren die met **SPRCOLOR** worden ingesteld, hebben invloed op alle sprites. Raadpleeg het **SPRITE**-commando voor meer informatie.

Het laatste argument van **SPRITE** schakelt de multikleur-modus voor de sprite in.

Voorbeeld: Gebruik van **SPRCOLOR**

```
10 SPRITE 1, 1, 2,,,, 1 : REM ZET SPRITE 1 AAN (FG = 2)
20 SPRCOLOR 4, 5        : REM MC1 = 4, MC2 = 5
```

SPRITE

Token: $FE $07

Formaat: **SPRITE CLR**
SPRITE LOAD bestandsnaam [**,D** station] [**,U** eenheid]
SPRITE SAVE bestandsnaam [**,D** station] [**,U** eenheid]
SPRITE num [{**,** toon**,** kleur**,** prio**,** expx**,** expy**,** modus}]
SPRITE SYS R hires-mode

Gebruik: **SPRITE CLR** wist alle spritegegevens en herstelt alle pointers en attributen naar hun standaardwaarden.

SPRITE LOAD laadt spritegegevens vanuit **bestandsnaam** in het spritegeheugen.

SPRITE SAVE slaat spritegegevens uit het spritegeheugen op in **bestandsnaam**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

De **SPRITE**-vorm schakelt een sprite in of uit en stelt de attributen in:

num sprite-nummer.

toon sprite tonen of verbergen (1: toon, 0: verberg).

kleur sprite-voorgrondkleur.

prio de prioriteit (0: sprite voor tekst, 1: sprite achter tekst).

expx X-vergroting in- of uitschakelen (1: aan, 0: uit).

expy Y-vergroting in- of uitschakelen (1: aan, 0: uit).

modus multikleur-modus van sprite in- of uitschakelen (1: aan, 0: uit).

SPRITE SYS stelt eigenschappen van het spritesysteem als geheel in. Het aanvaardt één benoemd argument: **R hires-mode** schakelt de hoge-resolutiemodus van sprites in (1) of uit (0). Indien ingeschakeld, gebruiken alle sprites hoge-resolutiepixels in zowel verticale als horizontale dimensie, waardoor ze half zo groot lijken als in de lage-resolutiemodus. De hoge-resolutiemodus gebruikt ook een ander coördinatenvlak voor sprites.

Opmerkingen:
SPRCOLOR moet worden gebruikt om extra kleuren voor multikleur-sprites in te stellen (mode = 1).

RSPRITE() retourneert eigenschappen van individuele sprites die met het **SPRITE**-commando zijn ingesteld.

RSPRSYS() retourneert eigenschappen van het spritesysteem die met het **SPRITE SYS**-commando zijn ingesteld.

Voorbeeld: Gebruik van **SPRITE**

```
10 CLR : SCNCLR : SPRITE CLR
20 SPRITE LOAD "DEMOSPRITES1"
30 FOR I = 0 TO 7 : C = I : IF C = 6 THEN C = 8
40 MOVSPR I, 60 + 30 * I, 0 TO 60 + 30 * I, 65 + 20 * I, 3 : SPRITE I, 1, C,, 1, 1 : NEXT : SLEEP 3
50 FOR I = 0 TO 7 : SPRITE I,,,, 0, 0 : NEXT : SLEEP 3 : SPRITE CLR
60 FOR I = 0 TO 7 : MOVSPR I, 45 * I#5 : NEXT : FOR I = 0 TO 7 : SPRITE I, 1 : NEXT
70 FOR I = 0 TO 7 : X = 60 + 30 * I : Y = 65 + 20 * I : DO
80 LOOP UNTIL(X = RSPPOS(I, .)) AND (Y = RSPPOS(I, 1)) : MOVSPR I, .#. : NEXT
```

SPRSAV

Token: $FE $16

Formaat: **SPRSAV** bron**,** bestemming

Gebruik: Kopieert spritegegevens tussen twee sprites, of tussen een sprite en een stringvariabele.

bron sprite-nummer of stringvariabele.

bestemming sprite-nummer of stringvariabele.

Opmerkingen:

Bron en bestemming kunnen een sprite-nummer of een stringvariabele zijn,

SPRSAV kan alleen met de basisvorm van sprites (C64-compatibel) worden gebruikt. Deze sprites beslaan 64 bytes geheugen en produceren tekenreeksen van lengte 64 indien de bestemmingparameter een stringvariabele is.

Uitgebreide sprites en sprites met variabele hoogte kunnen niet met **SPRSAV** worden gebruikt.

Een tekenreeksarray van spritegegevens kan worden gebruikt om veel vormen op te slaan en snel naar het spritegeheugen te kopiëren met het commando **SPRSAV**.

Het is ook een handige methode om vormen van individuele sprites van of naar een schijfbestand te lezen of te schrijven.

Voorbeeld: Gebruik van **SPRSAV**

```
10 SPRITE LOAD "SPRITEDATA" : REM LAAD DATA VOOR 8 SPRITES
20 SPRITE 1, 1              : REM ZET SPRITE 1 AAN
30 SPRSAV 1, 2              : REM KOPIEER DATA VAN SPRITE 1 NAAR SPRITE 2
40 SPRITE 2, 1              : REM ZET SPRITE 2 AAN
50 SPRSAV 1, A$             : REM BEWAAR DATA VAN SPRITE 1 IN STRING A$
```

SQR

Token: $BA

Formaat: **SQR**(numerieke expressie)

Retourneert:
De vierkantswortel van een numerieke expressie.

Opmerkingen:
Het argument mag niet negatief zijn. Indien **SQR()** een negatieve waarde krijgt, wordt een Illegal Quantity-fout gegenereerd.

Voorbeeld: Gebruik van **SQR**

```
PRINT SQR(2)
 1.4142136
```

ST

Token: N.v.t.

Formaat: **ST**

Gebruik: De status van de laatste I/O-bewerking.

Indien **ST** nul is, trad er geen fout op; anders is deze ingesteld op een apparaat-afhankelijke foutcode.

Opmerkingen:

ST is een gereserveerde systeemvariabele.

Voorbeeld: Gebruik van **ST**

```
100 MX = 100 : DIM T$(MX)                          : REM DATA-ARRAY
110 DOPEN#1, "DATA"                                : REM OPEN BESTAND
120 IF DS THEN PRINT"KON NIET OPENEN" : STOP
130 LINE INPUT#1, T$(N) : N = N + 1                : REM LEES EEN RECORD
140 IF N > MX THEN PRINT "TE VEEL DATA" : GOTO 160
150 IF ST = 0 THEN 130                             : REM ST = 64 FOR END-OF-FILE
160 DCLOSE#1
170 PRINT N; " RECORDS GELEZEN"
```

STEP

Token: $A9

Formaat: **FOR** index = begin **TO** einde [**STEP** stap] ... **NEXT** [index]

Gebruik: **STEP** is een optioneel onderdeel van een **FOR**-lus.

De **index**-variabele kan na elke iteratie met een constante waarde worden verhoogd of verlaagd. Standaard wordt de variabele met 1 verhoogd. De indexvariabele moet een reële variabele zijn.

begin beginwaarde van de index.

einde wordt aan het einde van een iteratie gecontroleerd en bepaalt of er nog een iteratie wordt uitgevoerd of dat de lus wordt afgebroken.

stap bepaalt de wijziging die op het einde van een lusiteratie op de **index** wordt toegepast. Positieve stapwaarden verhogen de index, negatieve waarden verlagen deze. Standaard is 1.0 indien niet opgegeven.

Opmerkingen:

Bij positieve stappen moet **einde** groter zijn dan of gelijk aan **begin**. Bij negatieve stappen moet **einde** kleiner zijn dan of gelijk aan **begin**.

Het is een slechte programmeerpraktijk om de waarde van de **index**-variabele binnen de lus te wijzigen of met **GOTO** in of uit een lus te springen.

Voorbeeld: Gebruik van **STEP**

```
10 FOR D = 0 TO 360 STEP 30
20 R = D * π / 180
30 PRINT D; R; SIN(R); COS(R); TAN(R)
40 NEXT D
```

STOP

Token: $90

Formaat: **STOP**

Gebruik: Onderbreekt de uitvoering van het BASIC-programma.

Er wordt een bericht weergegeven met het regelnummer waarop het programma is gestopt. De READY-prompt verschijnt en de computer schakelt over naar de directe modus, in afwachting van toetsenbordinvoer.

Opmerkingen:

Alle variabeledefinities blijven na **STOP** geldig. Deze kunnen worden bekeken of gewijzigd, en het programma kan worden hervat met **CONT**. Elke bewerking van de programmaregels maakt verdere hervatting evenwel onmogelijk.

De programma-uitvoering kan worden hervat met **CONT**.

Voorbeeld: Gebruik van **STOP**

```
10 IF V < 0 THEN STOP : REM PROGRAMMA STOPT INDIEN NEGATIEF
20 PRINT SQR(V)       : REM PRINT VIERKANTSWORTEL
```

STR$

Token: $C4

Formaat: **STR$**(numerieke expressie)

Retourneert:
Een tekenreeks met de opgemaakte waarde van het argument, alsof deze met **PRINT** naar de tekenreeks is afgedrukt.

Opmerkingen:
Om een getal naar een decimale tekenreekswaarde om te zetten, zie **VAL**.

Voorbeeld: Gebruik van **STR$**

```
10 A$ = "DE WAARDE VAN PI IS " + STR$(π)
20 PRINT A$

RUN
DE WAARDE VAN PI IS 3.1415927
```

STRBIN$

Token: $CE $12

Formaat: **STRBIN$**(numerieke expressie)

Retourneert:
De getalwaarde als tekenreeks van de binaire representatie.

Opmerkingen:
Breukdelen worden genegeerd.

Om een binaire tekenreeks naar een getal om te zetten, zie **DECBIN**.

Voorbeeld: Gebruik van **STRBIN$**

```
PRINT STRBIN$(245)
11110101
```

SYS

Token: **SYS** : $9E
SYS TO : $9E $A4

Formaat: **SYS** adres [{**,** areg**,** xreg**,** yreg**,** zreg**,** sreg}]
SYS TO address [{**,A** areg**,X** xreg**,Y** yreg**,Z** zreg**,S** sreg **,I** BASIC-interruptmode **,O** MAPL-offset }]

Gebruik: Roept een machinetaal-subroutine aan.

adres startadres van de subroutine. Dit kan een in het ROM aanwezige KERNAL-routine zijn of elke andere routine die eerder is geladen of met **POKE** in het RAM is geplaatst.

areg waarde van de CPU-accumulator.

xreg waarde van het CPU X-register.

yreg waarde van het CPU Y-register.

zreg waarde van het CPU Z-register.

sreg waarde van het CPU-statusregister.

SYS laadt de argumenten (indien aanwezig) in registers, en aanroep daarna de subroutine. De aangeroepen routine moet met een **RTS**-instructie worden afgesloten. Nadat de subroutine is teruggekeerd, slaat hij de nieuwe registerinhoud op en geeft hij de besturing terug aan het BASIC-programma.

Indien de adreswaarde 16-bits is ($0000 - $FFFF), wordt de **BANK**-waarde gebruikt om het feitelijke adres te bepalen. Indien het adres hoger is dan $FFFF, wordt het geïnterpreteerd als een lineair 24-bits adres en wordt de waarde van **BANK** genegeerd.

SYS-vorm: Het **SYS**-commando aanvaardt het adres en optionele registerinhoud als positieargumenten, gescheiden door komma's. Als de waarde van een register niet is opgegeven, is die waarde ongedefinieerd. Zorg er dus voor dat u elke registerparameter instelt die de subroutine gebruikt.

In tegenstelling tot andere BASIC-commando's die geheugen benaderen, gelden er beperkingen op welke adressen **SYS** kan benaderen:

- **SYS** kan alleen banken 0 - 5 benaderen en heeft geen toegang tot zolder-RAM of het bovenste geheugen, zelfs niet met lange adressen.
- Alleen offsets $2000 - $7FFF binnen een gegeven bank verwijzen daadwerkelijk naar het geheugen van die bank.
- **SYS** kan alleen offsets $0000 - $1FFF in bank 0 benaderen.

- Het benaderen van offsets $8000 - $FFFF benadert altijd geheugen alsof **BANK** op 128 staat (inclusief ROM- en I/O-registermappingen), zelfs wanneer **BANK** op een andere bank staat of wanneer lange adressen worden gebruikt.

SYS TO-vorm:

Het **SYS TO**-commando aanvaardt het adres als eerste argument en de overige argumenten als benoemde argumenten. Deze vorm ondersteunt bovendien extra mogelijkheden ten opzichte van **SYS**, waaronder de mogelijkheid om code overal in de eerste 384KB van het geheugen uit te voeren.

Om een register in te stellen voor het aanroepen van de subroutine, geef het benoemde argument dat bij het register hoort op. Om bijvoorbeeld het X-register in te stellen en vervolgens een subroutine op adres $1600 aan te roepen: `SYS TO $1600,X($FF)`. De registerparameters heten A, X, Y, Z en S.

MAPL-offset Overschrijft het MAP-offsetregister voor 16-bit-adressen $2000 to $7FFF. De benoemde O-parameter accepteert een drie-nybble-waarde die met 256 wordt vermenigvuldigd om de offset te bepalen. Dit commando koppelt bijvoorbeeld $2000 - $7FFF TO $54000 - $59FFF, en aanroep vervolgens $3100 (+ $52000 = $55100): `SYS TO $3100,O($520)`

BASIC-interruptmode Een aanroepvorm waarbij BASIC tijdens de aanroep ingeschakeld blijft. Om deze modus in te schakelen, geef het argument I met waarde 1 op. In deze modus blijft BASIC ROM in het geheugen aanwezig en blijven interrupt-gestuurde functies ingeschakeld. Dit is nuttig voor het aanroepen van korte machinetaalroutines vanuit BASIC terwijl **PLAY** muziek afspeelt of **MOVSPR** sprites bewegen.

Een genoemd register kan worden gevolgd door een decimale waarde of door een numerieke expressie tussen haakjes. Om een hexadecimale waarde te gebruiken, plaats deze tussen haakjes.

Opmerkingen:

De registerwaarden na een **SYS**-aanroep worden in het systeemgeheugen opgeslagen. **RREG** kan worden gebruikt om deze waarden op te halen.

Ondanks de ongebruikelijke beperkingen op adressen is **SYS** een krachtige manier om BASIC- en machinetaal te combineren. Voor korte routines is geheugen in bank 0 op offsets $1800 - $1EFF beschikbaar voor programmeergebruik. Indien er zorgvuldig voor wordt gezorgd dat het einde van het BASIC-programma niet wordt overschreven, kunnen machinetaalroutines elders in bank 0 worden geladen, tot offset $BFFF.

Het correcte gebruik van **SYS** (d.w.z. zonder het systeem te storen) vereist enige technische vaardigheid. Zie voor meer informatie en voorbeel-

den *het grote Mega65-boek*, Programmeren met het geheugen (hoofdstuk 15).

Voorbeeld: Gebruik van **SYS**

```
10 REM VERANDER DE RANDKLEUR MET SYS
20 BANK 0
30 POKE $4000, $EE, $20, $D0, $60   : REM MACHINETAAL VOOR INC $D020 : RTS
40 SYS $4000                        : REM ROEP SUBROUTINE AAN OP $4000 / BANK $00
50 GETKEY A$ : IF A$ <> "Q" THEN 40 : REM GA DOOR TOTDAT OP 'Q' IS GEDRUKT

REM : ROEP $1600 AAN MET HET Y-REGISTER GEZET OP $CC
SYS $1600,,,$CC

REM : ROEP $1600 AAN MET HET Y-REGISTER GEZET OP $CC
SYS TO $1600,Y($CC)

REM : ROEP $1600 AAN MET BASIC ZICHTBAAR
SYS TO $1600,I1

REM : $2000-$7FFF BEELDT HET GEHEUGEN AF VANAF $54000-$59FFF, DAN WORDT $3100 AANGEROEPEN
SYS TO $3100,0($520)
```

T@&

Token: N.v.t.

Formaat: **T@&**(kolom**,** rij**)**

Gebruik: Leest of stelt de schermcode van een teken op de gegeven schermcoördinaten in.

Dit gedraagt zich als een arrayvariabele. Het toewijzen van een waarde wijzigt de schermcode van het gegeven teken onmiddellijk.

Opmerkingen:

T@& is een gereserveerde systeemvariabele.

Schermcodes zijn niet hetzelfde als PETSCII-codes. Zie bijlage 3 op bladzijde 305 voor een lijst met schermcodes.

De coördinaten beginnen bij (0, 0) voor de linkerbovenhoek. Indien de gegeven coördinaten buiten het huidige schermformaat vallen, geeft het benaderen van de array een Bad Subscript-fout.

Voorbeeld: Gebruik van **T@&** en **C@&**

```
10 PRINT CHR$(147)
20 FOR R = 0 TO 3
30 FOR C = 0 TO 63
40 T@&(C, R) = R * 64 + C
50 C@&(C, R)= MOD(C, 16)
60 NEXT C
70 NEXT R
80 CURSOR 0, 4
```

TAB

Token: $A3

Formaat: **TAB**(kolom)

Retourneert:

Plaatst de cursor op **kolom**.

Dit wordt alleen gedaan als de doelkolom zich *rechts* van de huidige cursorpositie bevindt, anders zal de cursor niet bewegen. De kolomtelling begint met 0 als de meest linkse kolom.

Opmerkingen:

Deze functie moet niet worden verward met TAB, die de cursor naar het volgende tabulatiepunt verplaatst.

Voorbeeld: Gebruik van **TAB**

```
10 FOR I = 1 TO 5
20 READ A$
30 PRINT "* " A$ TAB(10) " *"
40 NEXT I
50 END
60 DATA EEN, TWEE, DRIE, VIER, VIJF

RUN
* EEN      *
* TWEE     *
* DRIE     *
* VIER     *
* VIJF     *
```

TAN

Token: $C0

Formaat: **TAN**(expressie in radialen)
TAND(expressie in graden)

Retourneert:

De tangens van een hoek.

De **TAN**-functie verwacht een argument in radialen.

De **TAND**-variant verwacht een argument in graden.

Het resultaat ligt in het bereik (-1.0 tot +1.0).

Voorbeeld: Gebruik van **TAN**

```
PRINT TAN(0.7)
 0.84228838

X = 45 : PRINT TAN(X * π / 180)
 1

PRINT TAND(45)
 1
```

TEMPO

Token: $FE $05

Formaat: **TEMPO** snelheid

Gebruik: Stelt de weergavesnelheid in voor **PLAY**.

snelheid bereik van 1 - 255.

De duur (in seconden) van een hele noot wordt berekend met $duration = 24/speed$.

Voorbeeld: Gebruik van **TEMPO**

```
10 VOL 8, 8
20 FOR T = 24 TO 18 STEP -2
30 TEMPO T
40 PLAY "T0M3O4QGAGFED", "T2O4M5P0H.DP5GB", "T5O3IGAGAGAABABAB"
50 IF RPLAY(1) THEN GOTO 50
60 NEXT T
70 PLAY "T0O5QCO4GEH.C", "T2O5IEFEDEDCEGO6P8CP0R", "T5O3ICDCDEFEDCO4C"
```

THEN

Token: $A7

Formaat: **IF** expressie **THEN** waar-clausule [**ELSE** onwaar-clausule]

Gebruik: **THEN** is onderdeel van een **IF**-structuur.

expressie is een logische of numerieke expressie. Een numerieke expressie wordt als **FALSE** geëvalueerd indien de waarde nul is en als **TRUE** bij elke niet-nul waarde.

waar-clausule is één of meer instructies die direct na **THEN** op dezelfde regel beginnen. Een regelnummer na **THEN** voert in plaats daarvan een **GOTO** naar die regel uit.

onwaar-clausule is één of meer instructies die direct na **ELSE** op dezelfde regel beginnen. Een regelnummer na **ELSE** voert in plaats daarvan een **GOTO** naar die regel uit.

Opmerkingen:

De standaard **IF** ... **THEN** ... **ELSE**-structuur is beperkt tot één regel. De **waar-clausule** en **onwaar-clausule** kunnen evenwel worden uitgebreid naar meerdere regels met behulp van een samengestelde instructie tussen **BEGIN** ... **BEND**.

Voorbeeld: Gebruik van **THEN**

```
10 ROOD$ = CHR$(28) : ZWART$ = CHR$(144) : WIT$ = CHR$(5)
20 INPUT "GEEF EEN GETAL"; V
30 IF V < 0 THEN PRINT ROOD$; : ELSE PRINT ZWART$;
40 PRINT V                                  : REM PRINT NEGATIEVE GETALLEN IN HET ROOD
50 PRINT WIT$
60 INPUT "PROGRAMMA BEEINDIGEN (J/N)"; A$
70 IF A$ = "J" THEN END
80 IF A$ = "N" THEN 20 : ELSE 60
```

TI

Token: N.v.t.

Formaat: **TI**

Gebruik: Een hogesnelheidstimer-variabele met een resolutie van 1 microseconde.

Deze wordt gestart of gereset met **CLR TI** en kan op dezelfde manier als elke andere variabele in expressies worden benaderd.

Opmerkingen:

TI is een gereserveerde systeemvariabele.

De waarde in **TI** is het aantal seconden (met 7 decimalen) sinds de laatste reset of start.

Voorbeeld: Gebruik van **TI**

```
100 CLR TI                              : REM START TIMER
110 FOR I% = 1 TO 10000 : NEXT          : REM DOE IETS
120 ET = TI                             : REM SLA VERLOPEN TIJD OP IN ET
130 PRINT "LOOPTIJD:"; ET; " SECONDEN"
```

TI$

Token: N.v.t.

Formaat: **TI$**

Gebruik: De huidige tijd van de dag, als een tekenreeks.

De tijdwaarde wordt bijgewerkt door de RTC (Real-Time Clock). De tekenreeks **TI$** heeft het formaat HH:MM:SS.

TI$ is een read-only variabele die de registers van de RTC uitleest en de waarden formatteert tot een tekenreeks. Dit verschilt van andere Commodore-computers die geen RTC hebben.

Opmerkingen:

TI$ is een gereserveerde systeemvariabele.

Het is mogelijk om de registers van de RTC direct te benaderen via **PEEK**s. Het beginadres van de registers is $FFD7110 (FFD.7110).

Raadpleeg voor meer informatie over het instellen van de Real-Time Clock de sectie Configuratiehulpprogramma op bladzijde *het grote Mega65-boek*, Configuratiehulpprogramma (paragraaf 5).

Voorbeelden:

Gebruik van **TI$**

```
100 REM ****** LEES RTC: ALLE WAARDEN ZIJN IN BCD ******
110 RT = $FFD7110                : REM ADRES VAN RTC
120 FOR I = 0 TO 5               : REM SS, MM, UU, DD, MA, JJ
130 T(I) = PEEK(RT + I)          : REM LEES REGISTERS
140 NEXT I                       : REM GEBRUIK ALLEEN LAATSTE TWEE CIJFERS
150 T(2) = T(2) AND 127          : REM VLAG VOOR 24-UURS MODE WEGHALEN
160 T(5) = T(5) + $2000          : REM 2000 BIJ JAAR OPTELLEN
170 FOR I = 2 TO 0 STEP -1       : REM TIJDINFORMATIE
180 PRINT USING ">##"; HEX$(T(I));
190 NEXT I
RUN
12 52 36
```

```
PRINT DT$;TI$
05-APR-2021 15:10:00
```

TO

Token: $A4

Formaat: trefwoord **TO**

Gebruik: **TO** is een secundair trefwoord dat in combinatie met primaire trefwoorden wordt gebruikt, zoals **BACKUP**, **BSAVE**, **CHANGE**, **CONCAT**, **COPY**, **FOR**, **GO**, **RENAME** en **SET DISK**.

Opmerkingen:
TO kan niet op zichzelf worden gebruikt.

Voorbeeld: Gebruik van **TO**

```
10 GO TO 1000           : REM ZOALS GOTO 1000

10 GOTO 1000            : REM KORTER EN SNELLER

10 FOR I = 1 TO 10      : REM TO IS DEEL VAN DE LUS
20 PRINT I : NEXT       : REM LUSEINDE

COPY "CODES" TO "BACKUP" : REM KOPIEER ENKEL BESTAND
```

TRAP

Token: $D7

Formaat: **TRAP** [regelnummer]

Gebruik: Registreert (of wist) een foutafhandelings-subroutine voor BASIC.

Wanneer een foutafhandelaar is geregistreerd, wordt er bij een fout in een BASIC-programma de subroutine aangeroepen in plaats van het programma te beëindigen. Tijdens de subroutine bevat de systeemvariabele **ER** het foutnummer. De **TRAP**-foutafhandelaar kan dan beslissen of de uitvoering moet worden gestopt (**STOP**) of hervat (**RESUME**).

TRAP zonder argument schakelt de foutafhandelaar uit; fouten worden dan afgehandeld door de normale systeemroutines.

Voorbeeld: Gebruik van **TRAP**

```
 10 TRAP 100
 20 FOR I = 1 TO 100
 30 PRINT EXP(I)
 40 NEXT
 50 PRINT "STOPPED FOR I ="; I
 60 END
100 PRINT ERR$(ER) : RESUME 50
```

TROFF

Token: $D9

Formaat: **TROFF**

Gebruik: Schakelt de traceermodus uit (ingeschakeld door **TRON**).

Wanneer de traceermodus actief is, wordt elk regelnummer afgedrukt voordat het wordt uitgevoerd. **TROFF** schakelt de traceermodus uit.

Voorbeeld: Gebruik van **TROFF**

```
10 TRON             : REM ACTIVEER TRACEMODUS
20 FOR I = 85 TO 100
30 PRINT I; EXP(I)
40 NEXT
50 TROFF            : REM DEACTIVEER TRACEMODUS

RUN
[10][20][30] 85  8.2230125E36
[40][30] 86  2.2352466E37
[40][30] 87  6.0760302E37
[40][30] 88  1.6516362E38
[40][30] 89
?OVERFLOW ERROR IN 30
READY.
```

TRON

Token: $D8

Formaat: **TRON**

Gebruik: Schakelt de traceermodus in.

Wanneer de traceermodus actief is, wordt elk regelnummer afgedrukt voordat het wordt uitgevoerd. **TRON** schakelt de traceermodus in.

Dit is nuttig voor het debuggen van de besturingsstroom van een BASIC-programma. Voeg **TRON**- en **TROFF**-opdrachten toe aan het programma rond de regels die gedebugd moeten worden.

Voorbeeld: Gebruik van **TRON**

```
10 TRON             : REM ACTIVEER TRACEMODUS
20 FOR I = 85 TO 100
30 PRINT I; EXP(I)
40 NEXT
50 TROFF            : REM DEACTIVEER TRACEMODUS

RUN
[10][20][30] 85  8.2230125E36
[40][30] 86  2.2352466E37
[40][30] 87  6.0760302E37
[40][30] 88  1.6516362E38
[40][30] 89
?OVERFLOW ERROR IN 30
READY.
```

TYPE

Token: $FE $27

Formaat: **TYPE** [**P**] bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Drukt de inhoud van een bestand af dat tekst bevat gecodeerd als PETSCII.

Indien de **P**-vlag is opgegeven, pauzeert de weergave bij elke schermvol tekst. Druk op **Q** om de paginamodus te beëindigen; elke andere toets gaat verder naar de volgende pagina.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

TYPE kan niet worden gebruikt om BASIC-programma's af te drukken. Gebruik **LIST** voor programma's. **TYPE** kan alleen **SEQ**-bestanden verwerken die PETSCII-tekst bevatten, gescheiden door het CR-teken (carriage return). Het CR-teken kan naar een bestand worden geschreven met **CHR$(13)**. **TYPE** opent het bestand als een **SEQ**-bestand. Gebruik van **TYPE** met een bestand van een ander type geeft een schijffout `64,FILE TYPE MISMATCH`.

Zie het **EDIT**-commando voor een manier om tekstbestanden interactief te maken en aan te passen.

Voorbeelden:

Gebruik van **TYPE**

```
TYPE "README"

TYPE "README.LST", U9

TYPE P "MOBYDICK"
```

UNLOCK

Token: $FE $4F

Formaat: **UNLOCK** bestandsnaam/patroon [**,D** station] [**,U** eenheid]

Gebruik: Ontgrendelt een bestand op de schijf dat voorheen is vergrendeld met **LOCK**.

Het opgegeven bestand of de set bestanden die aan het patroon voldoen, wordt ontgrendeld en is niet meer beschermd. Het kan daarna worden verwijderd met de commando's **DELETE**, **ERASE** of **SCRATCH**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Het ontgrendelen van een bestand dat al ontgrendeld is, heeft geen effect.

In de directe modus wordt het aantal ontgrendelde bestanden afgedrukt. De voorlaatste waarde uit het bericht bevat het aantal ontgrendelde bestanden.

Voorbeelden:

Gebruik van **UNLOCK**

```
UNLOCK "SNOOPY", U9     : REM ONTGRENDEL BESTAND SNOOPY OP UNIT 9
03,FILES UNLOCKED,01,00

UNLOCK "BS*"            : REM ONTGRENDEL ALLE BESTANDEN BEGINNEND OP "BS"
03,FILES UNLOCKED,04,00
```

UNTIL

Token: $FC

Formaat: **DO** ... **LOOP**
DO [<**UNTIL** | **WHILE**> logische expressie] ... opdrachten [**EXIT**]
LOOP [<**UNTIL** | **WHILE**> logische expressie]

Gebruik: **DO** ... **LOOP** definieert de reikwijdte van een BASIC-lus. Wanneer **DO** en **LOOP** zonder modifiers worden gebruikt, ontstaat er een oneindige lus die alleen kan worden afgebroken met **EXIT**. De lus kan worden beheerd door **UNTIL** of **WHILE** toe te voegen na **DO** of **LOOP**.

Opmerkingen:
DO-lussen kunnen genest worden. Een **EXIT**-opdracht beëindigt alleen de huidige lus.

Voorbeelden:
Gebruik van **UNTIL**

```
10 PW$ = "" : DO
20 GET A$ : PW$ = PW$ + A$
30 LOOP UNTIL LEN(PW$) > 7 OR A$ = CHR$(13)

10 DO
20 GET A$
30 LOOP UNTIL A$ = "Y" OR A$ = "N"
```

USING

Token: $FB

Formaat: **PRINT**[**#** kanaal**,**] **USING** indeling**;** argument

Gebruik: Ontleedt de **formaat**-tekenreeks en evalueert het argument. Het argument kan een tekenreeks of een numerieke waarde zijn. Het formaat van de resulterende uitvoer wordt bepaald door de **formaat**-tekenreeks.

Het **kanaal**nummer, dat eerder is opgegeven bij een aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN**. Indien geen kanaal wordt opgegeven, gaat de uitvoer naar het scherm.

Een **indeling**-tekenreeksvariabele of een tekenreeksconstante die de opmaakregels vastlegt. Bij het gebruik van een getal als **argument** kan het patroon worden opgegeven als een sjabloon zoals `###.##` of met een %[0]<width>.<precision>-specificatie, zoals `%3D %7.2F %4X`. Als een breedtespecificatie een voorlopende nul heeft, wordt een positief getal links met nullen aangevuld tot aan de decimale punt: `%07.2F`

Het **argument**-nummer dat moet worden opgemaakt. Als het argument niet in het formaat past, bijvoorbeeld bij het afdrukken van een variabele met vier cijfers in een reeks van drie haken (###), worden in plaats daarvan sterretjes gebruikt.

Opmerkingen:

De indelingstekenreeks wordt slechts op één argument toegepast, maar het is mogelijk om meer dan één **USING indeling;argument**-volgorde aan elkaar te koppelen.

Het **argument** kan afdrukbare tekens en controlcodes bevatten. Afdrukbare tekens worden op de cursorpositie weergegeven, terwijl controlcodes worden uitgevoerd.

Het aantal #-tekens bepaalt de breedte van de uitvoer. Indien het eerste teken van de indelingstekenreeks een '=' (gelijkteken) is, wordt het argument gecentreerd. Indien het eerste teken een '>' (groter-dan-teken) is, wordt het argument rechts uitgelijnd.

Hoewel het breedte-precisieformaatpatroon lijkt op de `printf()`-functie van de programmeertaal C, gedraagt het zich niet helemaal hetzelfde. Het breedteargument is het totale aantal cijfers in het patroon. De punt en het optionele precisieargument geven aan dat een decimale punt moet worden gebruikt, met het opgegeven aantal precisiecijfers rechts van de punt. `%7.2F` is equivalent aan: `#####.##`

Het opvullen van links met nullen is alleen beschikbaar met een breedte-precisiepatroon.

Voorbeelden:

USING met een bijbehorende **PRINT#**

```
PRINT USING "##.##"; 4, USING " [%7.4F] "; SQR(2)
 3.14 [  1.4142]

PRINT USING "##.##"; 4, USING " [%07.4F] "; SQR(2)
 3.14 [001.4142]

PRINT USING " ( # # # ) "; 12 * 31
 ( 3 7 2 )

PRINT USING "###"; "ABCDE"
ABC

PRINT USING ">###"; "ABCDE"
CDE

PRINT USING "ADDRESS: $%4X"; 65000
ADDRESS: $FDE8

A$ = "####,####,####.#":PRINT USING A$; 1E7 / 3
  3,333,333.3
```

USR

Token: $B7

Formaat: **USR**(numerieke expressie)

Gebruik: Roept een assembleertaal-routine aan waarvan het geheugenadres is opgeslagen op $02F8 - $02F9.

Het resultaat van de **numerieke expressie** wordt geschreven naar floating-point-accumulator 1.

Na het uitvoeren van de assembleroutine geeft BASIC de inhoud van floating-point-accumulator 1 terug.

Opmerkingen:

Banken 0 - 127 geven toegang tot RAM- of ROM-banken. Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enzovoort.

De floating-point-accumulator is een voorziening van de KERNAL.

Voorbeeld: Gebruik van **USR**

```
10 WPOKE $2F8, $7F33 : REM NEGATIEROUTINE
20 PRINT USR(π)
30 PRINT USR(-5)

RUN
-3.1415927
 5
```

VAL

Token: $C5

Formaat: **VAL**(tekenreeksexpressie)

Retourneert:
Het decimale floating point getal dat door een tekenreeks wordt weergegeven.

Opmerkingen:
VAL verwerkt tekens vanaf het begin van de tekenreeks die lijken op een BASIC-decimaal getal, inclusief een optioneel minteken, cijfers, een decimaal scheidingsteken en een exponent. Indien een ongeldig teken wordt aangetroffen, stopt de verwerking en wordt het resultaat tot aan dat punt in de tekenreeks geretourneerd. Indien het eerste teken ongeldig is, wordt 0 geretourneerd.

Om een getal naar een tekenreeks om te zetten, zie **STR$**. Indien u alleen een getal wilt afdrukken, onthoud dan dat **PRINT** numerieke expressies accepteert zonder dat een conversie naar een tekenreeks nodig is.

Voorbeeld: Gebruik van **VAL**

```
PRINT VAL("78E2")
 7800

PRINT VAL("7+5")
 7

PRINT VAL("1.256")
 1.256

PRINT VAL("$FFFF")
 0
```

VERIFY

Token: $95

Formaat: **VERIFY** bestandsnaam [**,** eenheid [**,** binvlag]]

Gebruik: **VERIFY** zonder **binvlag** vergelijkt een BASIC-programma in het geheugen met een schijfbestand van het type **PRG**. Het doet hetzelfde als **DVERIFY**, maar de syntaxis verschilt.

VERIFY met **binvlag** vergelijkt een binair bestand in het geheugen met een schijfbestand van het type **PRG**. Het doet hetzelfde als **BVERIFY**, maar de syntaxis verschilt.

bestandsnaam is een tekenreeks tussen aanhalingstekens, bijv. "PROG" of een tekenreeksexpressie.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

VERIFY kan alleen op gelijkheid testen. Het geeft geen informatie over het aantal of de positie van verschillende bytes. **VERIFY** eindigt met het bericht OK of met een Verify-fout.

VERIFY is verouderd in BASIC 65. Het is alleen geïmplementeerd voor achterwaartse compatibiliteit. Het wordt aanbevolen om **DVERIFY** en **BVERIFY** te gebruiken in plaats hiervan.

Voorbeelden:

Gebruik van **VERIFY**

```
VERIFY "ADVENTURE"

VERIFY "ZORK-I", 9

VERIFY "1:DUNGEON", 10
```

VIEWPORT

Token: $FE $31

Formaat: **VIEWPORT CLR**
VIEWPORT DEF x, y, breedte, hoogte

Gebruik: Bitmapgrafiek: beheert de viewport van een scherm.

VIEWPORT DEF definieert een knipgebied met de oorsprong (bovenste linker positie) ingesteld op **x, y** en de **breedte** en **hoogte**. Alle volgende grafiekcommando's zijn beperkt tot het **VIEWPORT**-gebied.

VIEWPORT CLR vult het knipgebied met de kleur van de tekenpen.

Opmerkingen:
Het knipgebied kan worden gereset naar het volledige scherm met het commando `VIEWPORT DEF 0, 0, WIDTH, HEIGHT` met dezelfde waarden voor breedte en hoogte als in het **SCREEN**-commando.

Voorbeeld: Gebruik van **VIEWPORT**

```
10 SCREEN 320, 200, 2
20 VIEWPORT DEF 20, 30, 100, 120 : REM GEBIED 20->119, 30->149
30 PEN 1                         : REM KIES KLEUR 1
40 VIEWPORT CLR                  : REM VUL GEBIED MET PENKLEUR
50 GETKEY A$                     : REM WACHT OP TOETSDRUK
60 SCREEN CLOSE
```

VOL

Token: $DB

Formaat: **VOL** right, left

Gebruik: Stelt het volume in voor de geluidsuitvoer met **SOUND** of **PLAY**. De waarde varieert van 0 (uit) tot 15 (hardst).

rechts volume voor SID's 1 en 2.

links volume voor SID's 3 en 4.

Opmerkingen:

De termen "rechts" en "links" verwijzen naar de standaard pan-instellingen voor de SID-chips in de audiomixer. Het werkelijke volume en de panpositie voor elk paar SID's hangt af van de huidige instellingen van de audiomixer. U kunt de audio-instellingen aanpassen in het Freezer-menu.

Voorbeeld: Gebruik van **VOL**

```
10 TEMPO 22
20 FOR V = 2 TO 12 STEP 2
30 VOL V, 16-V
40 PLAY "T0M3O4QGAGFED", "T2O4M5P0H.DP5GB", "T5O3IGAGAGAABABAB", "G"
50 IF RPLAY(1) THEN GOTO 50
60 NEXT V
70 PLAY "T0O5QCO4GEH.C", "T2O5IEFEDEDCEGO6P9CP0R", "T5O3ICDCDEFEDCO4C", "C"
```

VSYNC

Token: $FE $54

Formaat: **VSYNC** rasterlijn

Gebruik: Wacht tot de geselecteerde rasterlijn actief is.

rasterlijn PAL (0 - 311), NTSC (0 - 261).

Dit onderbreekt de uitvoering van het BASIC-programma totdat de schermupdate de opgegeven verticale pixelcoördinaat bereikt. Het betreft een zeer korte onderbreking: het scherm wordt 50 keer per seconde bijgewerkt bij PAL en 60 keer per seconde bij NTSC. Dit is nuttig om grafische parameters op specifieke momenten in de schermupdate aan te passen en de programmalogica van BASIC te synchroniseren met de verversingssnelheid van het scherm.

Voorbeeld: Gebruik van **VSYNC**

```
10 BORDER 3  : REM VERANDER RANDKLEUR IN CYAAN
20 VSYNC 100 : REM WACHT OP RASTERLIJN 100
30 BORDER 7  : REM VERANDER RANDKLEUR IN GEEL
40 VSYNC 260 : REM WACHT OP RASTERLIJN 260
50 GOTO 10   : REM LOOP
```

WAIT

Token: $92

Formaat: **WAIT** adres**,** enmasker [**,** xormasker]

Gebruik: Onderbreekt het BASIC-programma totdat het gevraagde bitpatroon van het opgegeven adres is gelezen.

adres adres in de huidige geheugenbank dat wordt gelezen.

enmasker masker dat via **AND** wordt toegepast.

xormasker masker dat via **XOR** wordt toegepast.

WAIT leest de byte-waarde van **adres** en past de maskers toe:
result = PEEK(adres) AND enmasker XOR xormasker.

De onderbreking eindigt als het resultaat ongelijk is aan nul; anders wordt de leesactie herhaald. Indien de voorwaarde niet wordt bereikt, kan de computer voor onbepaalde tijd vastlopen.

Opmerkingen:

WAIT wordt doorgaans gebruikt om hardware-registers of systeemvariabelen te onderzoeken en te wachten op een gebeurtenis, bijv. een joystick-gebeurtenis, muisevenement, toetsaanslag of wanneer een bepaalde rasterlijn op het scherm zal worden getekend.

Voorbeeld: Gebruik van **WAIT**

```
10 BANK 128
20 WAIT 54801, 1 : REM WACHT TOTDAT OP DE LINKER-SHIFT-TOETS WORDT GEDRUKT
```

WHILE

Token: $FD

Formaat: **DO** ... **LOOP**
DO [<**UNTIL** | **WHILE**> logische expressie] ... opdrachten [**EXIT**]
LOOP [<**UNTIL** | **WHILE**> logische expressie]

Gebruik: **DO** ... **LOOP** definieert de reikwijdte van een BASIC-lus. Wanneer **DO** en **LOOP** zonder modifiers worden gebruikt, ontstaat er een oneindige lus die alleen kan worden afgebroken met **EXIT**. De lus kan worden beheerd door **UNTIL** of **WHILE** toe te voegen na **DO** of **LOOP**.

Opmerkingen:
DO-lussen kunnen genest worden. Een **EXIT**-opdracht beëindigt alleen de huidige lus.

Voorbeelden:
Gebruik van **WHILE**

```
10 DO WHILE ABS(EPS) > 0.001
20 GOSUB 2000              : REM ITERATIESUBROUTINE
30 LOOP

10 I% = 0                  : REM INTEGERLUS 1-100
20 DO I% = I% + 1
30 LOOP WHILE I% < 101
```

WINDOW

Token: $FE $1A

Formaat: **WINDOW** links**,** boven**,** rechts**,** onder[**,** wissen]

Gebruik: Bepaalt de positie en grootte van het venster op het tekstscherm.

links linkerkolom.

boven bovenste rij.

rechts rechterkolom.

onder onderste rij.

wissen vlag voor het wissen van het tekstvenster.

Standaard vinden tekstupdates plaats op het volledige beschikbare tekstscherm. **WINDOW** beperkt het updategebied tot een rechthoek van de beschikbare schermruimte.

Opmerkingen:

De rijwaarden lopen van 0 tot 24. De kolomwaarden lopen van 0 tot respectievelijk 39 of 79, afhankelijk van de schermmodus.

Er kan slechts één venster op het scherm aanwezig zijn. Tweemaal op CLR HOME drukken of **PRINT**en van **CHR$(19)CHR$(19)** herstelt het venster naar de standaardinstelling (volledig scherm).

Voorbeelden:

Gebruik van **WINDOW**

```
WINDOW  0,  1, 79, 24   : REM SCHERM ZONDER TOPREGEL

WINDOW  0,  0, 79, 24, 1 : REM VENSTER OP VOLLEDIG SCHERM LEEGGEMAAKT

WINDOW  0, 12, 79, 24   : REM ONDERSTE HELFT VAN SCHERM

WINDOW 20,  5, 59, 15   : REM KLEIN GECENTREERD VENSTER
```

WPEEK

Token: $CE $10

Formaat: **WPEEK**(adres)

Retourneert:

De 16-bits woordwaarde die in het geheugen is opgeslagen op **adres** (lage byte) en **adres**+1 (hoge byte), als een natuurlijk getal van 16 bit.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

Opmerkingen:

Banken 0 - 127 geven toegang tot RAM- of ROM-banken. Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enzovoort.

Voorbeeld: Gebruik van **WPEEK**

```
10 UA = WPEEK($02F8)                    : REM DOELADRES VAN USR-JUMP
20 PRINT "DOELADRES USR-FUNCTIE"; UA
```

WPOKE

Token: $FE $1D

Formaat: **WPOKE** adres**,** woord [**,** woord ...]

Retourneert:

Schrijft een of meer 16-bits woorden in het geheugen of geheugengekoppelde I/O, beginnend bij **adres**.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

woord waarde van 0 - 65535. Het eerste woord wordt opgeslagen op adres (lage byte) en adres+1 (hoge byte). Het tweede woord wordt opgeslagen op adres+2 (lage byte) en adres+3 (hoge byte), enzovoort. Indien een waarde groter is dan 65535, worden alleen de twee laagste bytes gebruikt.

Opmerkingen:

Het adres wordt met twee verhoogd voor elk datawoord, zodat een geheugenbereik met één enkele **WPOKE**-opdracht kan worden geschreven.

Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enz.

Voorbeeld: Gebruik van **WPOKE**

```
10 BANK 128          : REM KIES SYSTEEMBANK
20 WPOKE $02F8, $1800 : REM ZET USR-VECTOR OP $1800
```

XOR

Token: $E9

Formaat: operand **XOR** operand

Gebruik: Voert een bitsgewijze logische Exclusive OR-bewerking uit op twee 16-bits waarden.

Operanden van het type geheel getal worden ongewijzigd gebruikt. Reële operanden worden omgezet naar een geheel getal van 16 bit (waarbij precisie verloren gaat). Logische operanden worden omgezet naar een geheel getal van 16 bit, waarbij $FFFF (decimaal -1) voor TRUE en $0000 (decimaal 0) voor FALSE wordt gebruikt.

Expressie	Resultaat
0 XOR 0	0
0 XOR 1	1
1 XOR 0	1
1 XOR 1	0

Opmerkingen:

Het resultaat is van het type geheel getal. Indien het resultaat in een logische context wordt gebruikt, wordt de waarde 0 als ONWAAR beschouwd en alle andere niet-nul-waarden als WAAR.

Voorbeeld: Gebruik van **XOR**

```
FOR I = 0 TO 8 : PRINT I XOR 5; : NEXT I
 5  4  7  6  1  0  3  2  13
```

HOOFDSTUK 3

Schermcodes

- Schermcodes

SCHERMCODES

Een tekstteken wordt in het schermgeheugen weergegeven door een schermcode. Er bestaan 256 mogelijke schermcodes, die elk verwijzen naar een afbeelding in de huidige tekenset.

Een volledige tekenset bevat twee groepen van 256 afbeeldingen: één voor de hoofdlettermodus en één voor de kleinelettermodus, derhalve 512 afbeeldingen in totaal. Slechts één modus kan tegelijk worden weergegeven. De ingebouwde tekensets gebruiken de eerste 128 tekens van elke groep voor normale tekens en de volgende 128 voor de geïnverteerde versies van dezelfde tekens.

In BASIC biedt de speciale array **T@&()** toegang tot de tekens op het scherm via kolom- en rij-indexen. De waarden in deze speciale array zijn schermcodes. Het **FONT**-commando schakelt tussen de ingebouwde tekensets. Het **CHARDEF**-commando wijzigt de afbeelding die bij een schermcode hoort.

OPMERKING: Schermcodes verschillen van PETSCII-codes. PETSCII-codes worden gebruikt om tekstgegevens op te slaan, te verzenden en te ontvangen en om te bepalen hoe strings op het scherm worden weergegeven. Wanneer een PETSCII-teken op het scherm wordt weergegeven, wordt de bijbehorende schermcode in het schermgeheugen geschreven. Voor een overzicht van PETSCII-codes, zie bijlage 4 op bladzijde 309.

De volgende tabel geeft een overzicht van de schermcodes. Indien een code afhankelijk van de modus een ander teken oplevert, wordt het teken vermeld als "hoofdletters / kleine letters".

0 @	**13** M / m	**26** Z / z	**39** '
1 A / a	**14** N / n	**27** [	**40** (
2 B / b	**15** O / o	**28** £	**41**)
3 C / c	**16** P / p	**29**]	**42** *
4 D / d	**17** Q / q	**30** ↑	**43** +
5 E / e	**18** R / r	**31** ←	**44** ,
6 F / f	**19** S / s	**32** spatie	**45** -
7 G / g	**20** T / t	**33** !	**46** .
8 H / h	**21** U / u	**34** "	**47** /
9 I / i	**22** V / v	**35** #	**48** 0
10 J / j	**23** W / w	**36** $	**49** 1
11 K / k	**24** X / x	**37** %	**50** 2
12 L / l	**25** Y / y	**38** &	**51** 3

52 4
53 5
54 6
55 7
56 8
57 9
58 :
59 ;
60 <
61 =
62 >
63 ?
64
65 / A
66 / B
67 / C
68 / D
69 / E
70 / F
71 / G
72 / H
73 / I
74 / J
75 / K
76 / L
77 / M
78 / N
79 / O
80 / P
81 / Q
82 / R
83 / S
84 / T
85 / U
86 / V
87 / W
88 / X
89 / Y
90 / Z
91
92
93
94 π /

OPMERKING: In de ingebouwde tekensets zijn codes 128-255 geïnverteerde versies van codes 0-127.

HOOFDSTUK 4

PETSCII-codes

- PETSCII-codes

PETSCII-CODES

PETSCII is de tekencodering die Commodore-8-bitcomputers gebruiken voor gegevens. De codering omvat afdrukbare tekens zoals letters, cijfers, leestekens en grafische symbolen; Controlcodes om de weergaveparameters te wijzigen en de cursor van de schermeditor te bedienen; en voorstellingen van toetsaanslagen die via het toetsenbord kunnen worden ingevoerd. Een programma kan PETSCII-codes gebruiken voor uitvoer naar het scherm, voor invoer via het toetsenbord of voor het lezen en schrijven van tekengegevens in bestanden of via seriële communicatie.

In BASIC aanvaardt de functie **CHR$()** een PETSCII-code en geeft die code als string terug. Zo drukt `PRINT CHR$(65)` de letter A af. De functie **ASC()** aanvaardt een string van één teken en geeft het getal van de PETSCII-code terug. De commando's **GET** en **GETKEY** lezen een toetsaanslag van het toetsenbord en geven de bijbehorende PETSCII-code terug.

Machinetaalprogramma's kunnen de GETIN-routine van de KERNAL of het PETSCIIKEY-hardwareregister gebruiken om toetsenbordinvoer als PETSCII-codes te lezen.

0

1 Alternatief palet

2 Onderstrepen aan

3

4 Standaardpalet

5 Wit

6

7 Bel

8

9 TAB

10 Nieuwe regel

11 Schakelt SHIFT + ▚ uit

12 Schakelt SHIFT + ▚ in

13 RETURN

14 Kleine letters

15 Knipperen aan

16 F9

17 ↓

18 RVS ON

19 CLR HOME

20 INST DEL

21 F10 / Vorig woord

22 F11

23 F12 / Volgend woord

24 Tab instellen/wissen

25 F13

26 F14 / Vorig tabpunt

27 ESCAPE

28 Rood

29 →

30 Groen

31 Blauw

32 SPACE

33 !

34 ″

35 #

36 $

37 %

38 &

39 ′

40 (

41)

42 *

43 +

44 ,

45 -

46 .

47 /

48 0

49 1

50 2

51 3

52 4

53 5

54 6

55 7

56 8

57 9

58 :

59 ;

60 <

61 =

62 >

63 ?

64 @

65 A

66 B

67 C

68 D

69 E

70 F

71 G

72 H
73 I
74 J
75 K
76 L
77 M
78 N
79 O
80 P
81 Q
82 R
83 S
84 T
85 U
86 V
87 W
88 X
89 Y
90 Z
91 [
92 £
93]
94 ↑
95 ←

De codes 96 tot en met 127 drukken tekens af die vergelijkbaar zijn met die van de codes 192 tot en met 223. Zij worden niet als toetsenbordinvoer teruggegeven.

128
129 Oranje
130 Onderstrepen uit
131 SHIFT RUN STOP
132 HELP
133 F1
134 F3
135 F5
136 F7
137 F2
138 F4
139 F6
140 F8
141 SHIFT RETURN
142 Hoofdletters
143 Knipperen uit
144 Zwart
145 ↑
146 RVS OFF
147 SHIFT CLR HOME
148 SHIFT INST DEL
149 Bruin
150 Lichtrood (roze)
151 Donkergrijs
152 Middengrijs
153 Lichtgroen
154 Lichtblauw
155 Lichtgrijs
156 Paars
157 ←
158 Geel
159 Cyaan
160 SPACE
161
162
163
164
165
166
167
168
169
170 
171
172 
173 
174 
175 
176 
177 
178 
179 
180 
181
182
183 
184 
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

216 ♣	**218** ♦	**220** ▌	**222** π
217 ♥	**219** ♠	**221** ▐	**223** ▄

De codes 224 tot en met 254 drukken tekens af die vergelijkbaar zijn met die van de codes 160 tot en met 190. Zij worden niet als toetsenbordinvoer teruggegeven.

De codes 126, 222 en 255 drukken alle drie het pi-symbool af (π). Wanneer u dit teken typt, is de toetsenbordinvoercode 222.

Wanneer u de kleineletters-/hoofdlettermodus gebruikt (door + SHIFT in te drukken), let dan op het volgende:

- De hoofdletters in het gebied 65-90 van bovenstaande tabel worden vervangen door kleine letters.
- De grafische tekens in het gebied 97-122 van bovenstaande tabel worden vervangen door hoofdletters.
- De kleine letters van PETSCII (65-90) en de hoofdletters (97-122) vallen in het ASCII-gebied voor respectievelijk hoofdletters (65-90) en kleine letters (97-122).
- Verscheidene symbolen hebben alternatieve symbolen in het lettertype voor kleine en hoofdletters, met name de PETSCII-codes 126, 127, 168 en 183.

HOOFDSTUK 5

Toetsen van de schermeditor

- Toetsen van de schermeditor
- Control-codes
- Shiftcodes
- Escape-sequenties

TOETSEN VAN DE SCHERMEDITOR

De volgende toetscombinaties voeren handelingen uit in de schermeditor van de MEGA65.

In sommige gevallen kan een programma de equivalente PETSCII-codes weergeven om dezelfde handelingen uit te voeren. Zo kan een programma CTRL + G, welke een beltoon voortbrengt, als CHR$(7) weergeven. Gebruik voor een ESC-sequentie CHR$(27) om de ESC-toets weer te geven, gevolgd door de volgende toets van de reeks.

CONTROL-CODES

Toetsencombinatie	Functie
Kleuren	
CTRL + 1 tot 8	Kiezen uit het eerste kleurbereik. Zie bijlage 6 op bladzijde 327 voor de lijst van kleuren in het systeempalet.
⌘ + 1 tot 8	Kiezen uit het tweede kleurbereik.
CTRL + E	Herstelt de kleur van de cursor naar de standaardkleur (wit).
CTRL + D	Schakelt de VIC-IV over naar kleurbereik 0-15 (standaardkleuren). Deze kleuren zijn bereikbaar met CTRL en de toetsen 1 tot 8 (voor de eerste 8 kleuren) of met ⌘ en de toetsen 1 tot 8 (voor de overige 8 kleuren).
CTRL + A	Schakelt de VIC-IV over naar kleurbereik 16-31 (alternatieve/regenboog-kleuren). Deze kleuren zijn bereikbaar met CTRL en de toetsen 1 tot 8 (voor de eerste 8 kleuren) of met ⌘ en de toetsen 1 tot 8 (voor de overige 8 kleuren).

Toetsencombinatie	Functie
Tabposities	
CTRL + Z	Verplaatst de cursor via een tab naar links. Indien er geen tabposities meer resteren, blijft de cursor aan het begin van de regel staan.
CTRL + I	Verplaatst de cursor via een tab naar rechts. Indien er geen tabposities meer resteren, blijft de cursor aan het einde van de regel staan.
CTRL + X	Stelt de huidige schermkolom als tabpositie in of wist deze. Gebruik CTRL + Z en I om heen en weer te springen naar alle posities die met X zijn ingesteld.
Verplaatsen	
CTRL + Q	Verplaatst de cursor telkens één regel omlaag. Gelijk aan ↓.
CTRL + J	Verplaatst de cursor één positie omlaag. Indien u zich op een lange regel BASIC-code bevindt die over twee regels doorloopt, verplaatst de cursor zich twee rijen omlaag om op de volgende regel te komen.
CTRL +]	Gelijk aan →.
CTRL + T	Wist het teken direct links van de cursor en schuift alle tekens die rechts daarvan staan één positie naar links. Dit is gelijk aan INST DEL.
CTRL + M	Voert een regelterugloop uit, gelijk aan RETURN.
Verplaatsen per woord	

Toetsencombinatie	Functie
CTRL + U	Verplaatst de cursor terug naar het begin van het vorige woord. Indien er op de huidige regel geen vorig woord staat, verplaatst de cursor zich naar de eerste kolom van de huidige regel en vervolgens naar de vorige regel, totdat een regel met een woord wordt aangetroffen.
CTRL + W	Verplaatst de cursor vooruit naar het begin van het volgende woord. Indien er op de huidige regel geen volgend woord staat, verplaatst de cursor zich naar de eerste kolom van de volgende regel, totdat een regel met een woord wordt aangetroffen.
Scrollen	
CTRL + P	Schuift de BASIC-listing één regel omlaag. Gelijk aan F9.
CTRL + V	Schuift de BASIC-listing één regel omhoog. Gelijk aan F11.
CTRL + S	Gelijk aan NO SCROLL.
Opmaak	
CTRL + B	Schakelt de onderstrepingsmodus voor tekst in. U kunt de onderstrepingsmodus uitschakelen door ESC en vervolgens O in te drukken.
CTRL + O	Schakelt de flitsmodus voor tekst in. U kunt de flitsmodus uitschakelen door ESC en vervolgens O in te drukken.
Hoofdlettergebruik	
CTRL + N	Wijzigt het hoofdlettergebruik van de tekst van uitsluitend hoofdletters naar kleine letters.

Toetsencombinatie	Functie
CTRL + K	Vergrendelt de omschakeling tussen hoofdletters en kleine letters, welke gewoonlijk met + SHIFT wordt uitgevoerd.
CTRL + L	Schakelt de omschakeling tussen hoofdletters en kleine letters in, welke met + SHIFT wordt uitgevoerd.
Diversen	
CTRL + G	Brengt een beltoon voort.
CTRL + [	Gelijk aan het indrukken van ESC.
CTRL + *	Opent de Matrix Mode Debugger.

SHIFTCODES

Toetsencombinatie	Functie
SHIFT + INST DEL	Voegt op de huidige cursorpositie een teken in en verplaatst alle tekens één positie naar rechts.
SHIFT + HOME	Wist het volledige scherm en plaatst de cursor op de startpositie.

ESCAPE-SEQUENTIES

Om een escape-sequentie uit te voeren, drukt u ESC kort in en laat u deze weer los. Druk vervolgens één van de volgende toetsen in om de sequentie uit te voeren.

Toets	Sequentie
Gedrag van de editor	
ESC X	Wist het scherm en schakelt tussen de tekstmodi 40 × 25 en 80 × 25.
ESC 4	Wist het scherm en schakelt over naar de tekstmodus 40 × 25.
ESC 8	Wist het scherm en schakelt over naar de tekstmodus 80 × 25.
ESC 5	Schakelt over naar de tekstmodus 80 × 50.
Let op dat sommige programma's verwachten dat zij in de tekstmodus 80 × 25 worden gestart en zich mogelijk niet correct gedragen wanneer zij in de tekstmodus 80 × 50 worden gestart.	
ESC @	Wist een gedeelte van het scherm, vanaf de huidige cursorpositie tot het einde van het scherm.
ESC O	Schakelt de zinnebeeldmodus, de modus voor omgekeerde tekst, de onderstrepingsmodus en de flitsmodus uit.
Scrollen	
ESC V	Schuift het hele scherm één regel omhoog.
ESC W	Schuift het hele scherm één regel omlaag.
ESC L	Schakelt het scrollen in wanneer ↓ onderaan het scherm wordt ingedrukt.
ESC M	Schakelt het scrollen uit. Wanneer ↓ onderaan het scherm wordt ingedrukt, verplaatst de cursor zich naar de bovenkant van het scherm. Wanneer ↑ echter bovenaan het scherm wordt ingedrukt, blijft de cursor op de eerste regel staan.

Toets	Sequentie
ESC N	Schakelt het “regelduwen” in: typen of afdrukken in de meest rechtse kolom schuift de daaropvolgende regels één positie omlaag.
ESC R	Schakelt het “regelduwen” uit: typen of afdrukken in de meest rechtse kolom verplaatst de cursor naar het begin van de volgende regel, maar schuift geen regels omlaag. Schakel zowel het “regelduwen” (ESC R) als het scrollen (ESC M) uit om met **PRINT** in de meest rechtse kolom af te drukken zonder de rest van de weergave te verstoren.
Invoegen en verwijderen	
ESC I	Voegt een lege regel in op de huidige cursorpositie en schuift alle daaropvolgende regels één positie omlaag.
ESC D	Verwijdert de huidige regel en schuift de regels onder de cursor één positie omhoog.
ESC P	Wist alle tekens van de cursor tot aan het begin van de huidige regel.
ESC Q	Wist alle tekens van de cursor tot aan het einde van de huidige regel.
Verplaatsen	
ESC J	Verplaatst de cursor naar het begin van de huidige regel.
ESC K	Verplaatst de cursor naar het laatste niet-witruimteteken op de huidige regel.

Toets	Sequentie
ESC ↑	Slaat de huidige cursorpositie op. Gebruik ESC ← (naast 1) om de cursor terug te verplaatsen naar de opgeslagen positie. Let op: de ↑ die hier gebruikt wordt, staat naast RESTORE.
ESC ←	Herstelt de cursorpositie naar de positie die eerder is opgeslagen door het indrukken van de toets ESC ↑ (naast RESTORE). Let op: de ← die hier gebruikt wordt, staat naast 1.
ESC HOME	Herstelt de cursorpositie naar de positie die eerder is opgeslagen door het indrukken van HOME.
Vensters	
ESC T	Stelt de linkerbovenhoek van het venstergebied in. Alle ingevoerde tekens en schermactiviteit blijven tot dit gebied beperkt. Zie ook ESC B. U kunt de venstermodus uitschakelen door CLR HOME tweemaal in te drukken.
ESC B	Stelt de rechterbenedenhoek van het venstergebied in. Alle ingevoerde tekens en schermactiviteit blijven tot dit gebied beperkt. Zie ook ESC T. U kunt de venstermodus uitschakelen door CLR HOME tweemaal in te drukken.
Gedrag van de cursor	

Toets	Sequentie
ESC A	Schakelt de automatische invoegmodus in. Elke ingedrukte toets wordt op de huidige cursorpositie ingevoegd, waarna alle tekens op de huidige regel rechts van de cursor één positie naar rechts opschuiven.
ESC C	Schakelt de automatische invoegmodus uit en keert terug naar de overschrijfmodus.
ESC E	Schakelt de cursor over naar de niet-flitsende modus.
ESC F	Schakelt de cursor over naar de gewone flitsmodus.
Gedrag van de bel	
ESC G	Schakelt de bel in. U kunt de bel laten klinken met CTRL en G.
ESC H	Schakelt de bel uit, zodat het indrukken van CTRL en G geen effect heeft.
Kleuren	
ESC U	Schakelt de VIC-IV over naar kleurbereik 0-15 (standaardkleuren). Deze kleuren zijn bereikbaar met CTRL en de toetsen 1 tot 8 (voor de eerste 8 kleuren) of met ⌘ en de toetsen 1 tot 8 (voor de overige kleuren).
ESC S	Schakelt de VIC-IV over naar kleurbereik 16-31 (alternatieve/rainbow-kleuren). Deze kleuren zijn bereikbaar met CTRL en de toetsen 1 tot 8 (voor de eerste 8 kleuren) of met ⌘ en de toetsen 1 tot 8 (voor de overige kleuren).
Tabposities	

Toets	Sequentie
ESC Y	Stelt de standaardtabposities in (om de 8 spaties) voor het volledige scherm.
ESC Z	Wist alle tabposities. Met CTRL en I verplaatst een tab de cursor naar het einde van de regel.

HOOFDSTUK 6

Systeempalet

- Systeempalet

SYSTEEMPALET

De volgende tabel beschrijft het systeempalet zoals het standaard is vastgelegd. U kunt deze paletkleuren wijzigen met het commando **PALETTE COLOR** en terugzetten op hun standaardwaarden met het commando **PALETTE RESTORE**.

Paletindexen worden gebruikt als waarden in de speciale array **C@&()** en als argumenten voor BASIC-commando's zoals **BACKGROUND**, **BORDER**, **COLOR**, **FOREGROUND**, **HIGHLIGHT**, **PEN** en **SCREEN CLR**.

Index	Rood	Groen	Blauw	Kleur
0	0	0	0	Zwart
1	15	15	15	Wit
2	15	0	0	Rood
3	0	15	15	Cyaan
4	15	0	15	Paars
5	0	15	0	Groen
6	0	0	15	Blauw
7	15	15	0	Geel
8	15	6	0	Oranje
9	10	4	0	Bruin
10	15	7	7	Lichtrood (roze)
11	5	5	5	Donkergrijs
12	8	8	8	Middengrijs
13	9	15	9	Lichtgroen
14	9	9	15	Lichtblauw
15	11	11	11	Lichtgrijs
16	14	0	0	Guru-meditatie
17	15	5	0	Rambtoetan
18	15	11	0	Wortel
19	14	14	0	Citroentaart
20	7	15	0	Pandangroen
21	6	14	6	Zeeziek groen
22	0	14	3	Soylent Groen
23	0	15	9	Slijmgroen
24	0	13	13	Het andere cyaan
25	0	9	15	Zeehemel
26	0	3	15	Smurfenblauw
27	0	0	14	Scherm des Doods
28	7	0	15	Pruimensaus
29	12	0	15	Zure druif
30	15	0	11	Kauwgom
31	15	3	6	Hete tamales

U kunt deze 32 kleuren in de schermterminal gebruiken door PETSCII-Controlcodes te typen of weer te geven. De tekenstatus van de cursor is ofwel één van de eerste 16 kleuren ofwel één van de tweede 16 kleuren, en deze status kan met Controlcodes worden gewijzigd. Elk van de 16 Controlcodes kiest de kleur van de cursor binnen die kleurenset.

Om kleurenset A te kiezen, drukt u op CTRL + D of geeft u PETSCII-code 4 weer.

Om kleurenset B te kiezen, drukt u op CTRL + A of geeft u PETSCII-code 1 weer.

Toets	PETSCII	Kleurenset A	Kleurenset B
CTRL + 1	144	Zwart	Guru-meditatie
CTRL + 2	5	Wit	Rambtoetan
CTRL + 3	28	Rood	Wortel
CTRL + 4	159	Cyaan	Citroentaart
CTRL + 5	156	Paars	Pandangroen
CTRL + 6	30	Groen	Zeeziek groen
CTRL + 7	31	Blauw	Soylent Groen
CTRL + 8	158	Geel	Slijmgroen
C= + 1	129	Oranje	Het andere cyaan
C= + 2	149	Bruin	Zeehemel
C= + 3	150	Lichtrood (roze)	Smurfenblauw
C= + 4	151	Donkergrijs	Scherm des Doods
C= + 5	152	Middengrijs	Pruimensaus
C= + 6	153	Lichtgroen	Zure druif
C= + 7	154	Lichtblauw	Kauwgom
C= + 8	155	Lichtgrijs	Hete tamales

HOOFDSTUK 7

Steuners & donateurs

- Organisaties
- Bijdragers
- Steuners

De MEGA65 zou niet tot stand zijn gekomen zonder de gulle steun van veel organisaties en individuen.

Wij stellen deze lijsten nog steeds samen; onze excuses indien u er nog niet in voorkomt. Indien u iemand kent die wij over het hoofd hebben gezien, laat het ons dan weten. Zo kunnen wij de bijdrage erkennen van iedereen die de MEGA65 mogelijk heeft gemaakt en die er het geweldige retrocomputingproject van heeft gemaakt dat de computer inmiddels is.

ORGANISATIES

Het MEGA Museum of Electronic Games & Art e.V. Duitsland
ALLES

Trenz Electronic, Duitsland
MOEDERBORD **FABRICAGE** **VERKOOP**

Hintsteiner, Oostenrijk
BEHUIZING

GMK, Duitsland
TOETSENBORD

KEVAG Telekom, Duitsland
WEBHOSTING

BIJDRAGERS

Andreas Liebeskind
(libi in paradize)
Financieel directeur MEGA eV

Thomas Hertzler
(grumpyninja)
Woordvoerder voor de VS

Russell Peake
(rdpeake)
Bugherder

Alexander Nik Petra
(n0d)
Vroeg ontwerp van de behuizing

Ralph Egas
(0-limits)
Zakelijk adviseur

Lucas Moss
PCB-ontwerp voor de MEGAphone

Daren Klamer
(Impakt)
Proeflezen van de handleiding

Daniël Mantione
(dmantione)
C64-hardwareguru

Dr. Canan Hastik
(indica)
Voorzitter MEGA eV

Simon Jameson
(Shallan)
Platformverbeteringen

Stephan Kleinert
(ubik)
Vernietiger van BASIC 10

Wayne Johnson
(sausage)
Aanvullingen op de handleiding

Raven L. A. Kleiss
(LAK132)
MegaWAT-presentatiesoftware

Maurice van Gils
(Maurice)
BASIC 65-voorbeeldprogramma's

Andrew Owen
(Cheveron)
Toetsenbord, Sinclair-ondersteuning

Adam Barnes
(amb5l)
HDMI-expert en printplaatherziening

Wayne Rittimann, Jr.
(johnwayner)
Bugverplettering op alle niveaus

STEUNERS

3c74ce64
8-Bit Classics
@11110110100
Aaron Smith
Achim Mrotzek
Adolf Nefischer
Adrian Esdaile
Adrien Guichard
Ahmed Kablaoui
Alan Bastian Witkowski
Alan Field
Alastair Paulin-Campbell
Alberto Mercuri
Alexander Haering
Alexander Kaufmann
Alexander Niedermeier
Alexander Soppart
Alfonso Ardire
Amiga On The Lake
André Kudra
André Simeit
André Wösten
Andrea Farolfi
Andrea Minutello
Andreas Behr
Andreas Freier
Andreas Grabski
Andreas Millinger
Andreas Nopper
Andreas Ochs
Andreas Wendel Manufaktur
Andreas Zschunke
Andrew Bingham
Andrew Dixon
Andrew Mondt
Andrzej Hłuchyj
Andrzej Sawiniec
Andrzej Śliwa
Anthony W. Leal
Arkadiusz Bronowicki
Arkadiusz Kwasny
Arnaud Léandre
Arne Drews
Arne Neumann
Arne Richard Tyarks
Axel Klahr
Balaz Ondrej
Barry Thompson
Bartol Filipovic
Benjamin Maas
Bernard Alaiz
Bernhard Zorn
Bieno Marti-Braitmaier
Bigby
Bill LaGrue
Bjoerg Stojalowski
Björn Johannesson
Bjørn Melbøe
Bo Goeran Kvamme
Boerge Noest
Bolko Beutner
Brett Hallen
Brian Gajewski
Brian Green
Brian Juul Nielsen
Brian Reiter
Bryan Pope
Burkhard Franke
Byron Goodman
Cameron Roberton (KONG)
Carl Angervall
Carl Danowski
Carl Stock
Carl Wall
Carlo Pastore
Carlos Silva
Carsten Sørensen
Cenk Miroglu Miroglu
Chang sik Park
Charles A. Hutchins Jr.
Chris Guthrey
Chris Hooper
Chris Stringer
Christian Boettcher
Christian Eick
Christian Gleinser
Christian Gräfe
Christian Heffner
Christian Kersting
Christian Schiller
Christian Streck
Christian Weyer
Christian Wyk
Christoph Haug
Christoph Huck
Christoph Pross
Christopher Christopher
Christopher Kalk
Christopher Kohlert
Christopher Nelson
Christopher Taylor
Christopher Whillock
Claudio Piccinini
Claus Skrepek
Collen Blijenberg
Constantine Lignos
Crnjaninja
Daniel Auger
Daniel Julien
Daniel Lobitz
Daniel O'Connor
Daniel Teicher
Daniel Tootill
Daniel Wedin
Daniele Benetti
Daniele Gaetano Capursi
Dariusz Szczesniak
Darrell Westbury
David Asenjo Raposo
David Dillard
David Gorgon
David Norwood
David Raulo
David Ross
de voughn accooe
Dean Scully
Dennis Jeschke
Dennis Schaffers
Dennis Schierholz

Dennis Schneck
denti
Dick van Ginkel
Diego Barzon
Dierk Schneider
Dietmar Krueger
Dietmar Schinnerl
Dirk Becker
Dirk Wouters
Domingo Fivoli
DonChaos
Donn Lasher
Douglas Johnson
Dr. Leopold Winter
Dusan Sobotka
Earl Woodman
Ed Reilly
Edoardo Auteri
Eduardo Gallardo
Eduardo Luis Arana
Eirik Juliussen Olsen
Emilio Monelli
EP Technical Services
Epic Sound
Erasmus Kuhlmann
ergoGnomik
Eric Hilaire
Eric Hildebrandt
Eric Hill
Eric Jutrzenka
Erwin Reichel
Espen Skog
Evangelos Mpouras
Ewan Curtis
Fabio Zanicotti
Fabrizio Di Dio
Fabrizio Lodi
FARA Gießen GmbH
FeralChild
First Choice Auto's
Florian Rienhardt
Forum64. de
Francesco Baldassarri
Frank Fechner
Frank Glaush
Frank Gulasch
Frank Haaland
Frank Hempel
Frank Koschel
Frank Linhares
Frank Sleeuwaert
Frank Wolf
FranticFreddie
Fredrik Ramsberg
Fridun Nazaradeh
Friedel Kropp
Garrick West
Gary Lake-Schaal
Gary Pearson
Gavin Jones
Geir Sigmund Straume
Gerd Mitlaender
Giampietro Albiero
Giancarlo Valente
Gianluca Girelli
Giovanni Medina
Glen Fraser
Glen R Perye III
Glenn Main
Gordon Rimac
GRANT BYERS
Grant Louth
Gregor Bubek
Gregor Gramlich
Guido Ling
Guido von Gösseln
Guillaume Serge
Gunnar Hemmerling
Günter Hummel
Guy Simmons
Guybrush Threepwood
Hakan Blomqvist
Hans Pronk
Hans-Jörg Nett
Hans-Martin Zedlitz
Harald Dosch
Harri Salokorpi
Harry Culpan
Harry Venema
Heath Gallimore
Heinz Roesner
Heinz Stampfli
Helge Förster
Hendrik Fensch
Henning Harperath
Henri Parfait
Henrik Kühn
Holger Burmester
Holger Sturk
Howard Knibbs
Hubert de Hollain
Huberto Kusters
Hugo Maria Gerardus v.d. Aa
Humberto Castaneda
Ian Cross
IDE64 Staff
Igor Ianov
Igor Kurtes
Immo Beutler
Ingo Katte
Ingo Keck
Insanely Interested Publishing
IT-Dienstleistungen Obsieger
Ivan Elwood
Jaap HUIJSMAN
Jace Courville
Jack Wattenhofer
Jakob Schönpflug
Jakub Tyszko
James Hart
James Marshburn
James McClanahan
James Sutcliffe
Jan Bitruff
Jan Hildebrandt
Jan Iemhoff
Jan Kösters
Jan Peter Borsje
Jan Schulze
Jan Stoltenberg-Lerche
Janne Tompuri
Jannis Schulte
Jari Loukasmäki
Jason Smith
Javier Gonzalez Gonzalez
Jean-Paul Lauque
Jeffrey van der Schilden
Jens Schneider

Jens-Uwe Wessling
Jesse DiSimone
Jett Adams
Johan Arneklev
Johan Berntsson
Johan Svensson
Johannes Fitz
John Cook
John Deane
John Dupuis
John Nagi
John Rorland
John Sargeant
John Traeholt
Jon Sandelin
Jonas Bernemann
Jonathan Prosise
Joost Honig
Jordi Pakey-Rodriguez
Jöre Weber
Jörg Jungermann
Jörg Schaeffer
Jörg Weese
Josef Hesse
Josef Soucek
Josef Stohwasser
Joseph Clifford
Joseph Gerth
Jovan Crnjanin
Juan Pablo Schisano
Juan S. Cardona Iguina
JudgeBeeb
Juliussen Olsen
Juna Luis Fernandez Garcia
Jürgen Endras
Jürgen Herm Stapelberg
Jyrki Laurila
Kai Pernau
Kalle Pöyhönen
Karl Lamford
Karl-Heinz Blum
Karsten Engstler
Karsten Westebbe
katarakt
Keith McComb
Kenneth Dyke
Kenneth Joensson
Kevin Edwards
Kevin Thomasson
Kim Jorgensen
Kim Rene Jensen
Kimmo Hamalainen
Konrad Buryło
Kosmas Einbrodt
Kurt Klemm
Lachlan Glaskin
Large bits collider
Lars Becker
Lars Edelmann
Lars Slivsgaard
Lasse Lambrecht
Lau Olivier
Lee Chatt
Loan Leray
Lorenzo Quadri
Lorenzo Travagli
Lorin Millsap
Lothar James Foss
Lothar Serra Mari
Luca Papinutti
Ludek Smetana
Lukas Burger
Lutz-Peter Buchholz
Luuk Spaetgens
Mad Web Skills
MaDCz
Magnus Wiklander
Maik Diekmann
Malte Mundt
Manfred Wittemann
Manuel Beckmann
Manzano Mérida
Marc "3D-vice" Schmitt
Marc Bartel
Marc Jensen
Marc Schmidt
Marc Theunissen
Marc Tutor
Marc Wink
Marcel Buchtmann
Marcel Kante
Marco Beckers
Marco Cappellari
Marco Rivela
Marco van de Water
Marcus Gerards
Marcus Herbert
Marcus Linkert
Marek Pernicky
Mario Esposito
Mario Fetka
Mario Teschke
Mariusz Tymków
Mark Adams
Mark Anderson
Mark Green
Mark Hucker
Mark Leitiger
Mark Spezzano
Mark Watkin
Marko Rizvic
Markus Bieler
Markus Bonet
Markus Dauberschmidt
Markus Fehr
Markus Fuchs
Markus Guenther-Hirn
Markus Liukka
Markus Merz
Markus Roesgen
Markus Uttenweiler
Martin Bauhuber
Martin Benke
Martin Gendera
Martin Groß
Martin Gutenbrunner
Martin Johansen
Martin Marbach
Martin Sonnleitner
Martin Steffen
Marvin Hardy
Massimo Villani
Mathias Dellacherie
Mathieu Chouinard
Matthew Adams
Matthew Browne
Matthew Carnevale
Matthew Palmer

Matthew Santos
Matthias Barthel
Matthias Dolenc
Matthias Fischer
Matthias Frey
Matthias Grandis
Matthias Guth
Matthias Lampe
Matthias Meier
Matthias Mueller
Matthias Nofer
Matthias Schonder
Maurice Al-Khaliedy
Max Ihlenfeldt
Meeso Kim
Michael Dailly
Michael Dötsch
Michael Dreßel
Michael Fichtner
Michael Fong
Michael Geoffrey Stone
Michael Gertner
Michael Grün
Michael Habel
Michael Härtig
Michael Haynes
Michael J Burkett
Michael Jensen
Michael Jurisch
Michael Kappelgaard
Michael Kleinschmidt
Michael Lorenz
Michael Mayerhofer
Michael Nurney
Michael Rasmussen
Michael Richmond
Michael Sachse
Michael Sarbak
Michael Schneider
Michael Scholz
Michael Timm
Michael Traynor
Michael Whipp
Michal Ursiny
Michele Chiti
Michele Perini
Michele Porcu
Miguel Angel Rodriguez Jodar
Mikael Lund
Mike Betz
Mike Kastrantas
Mike Pikowski
Mikko Hämäläinen
Mikko Suontausta
Mirko Roller
Miroslav Karkus
Morgan Antonsson
Moritz
Morten Nielsen
MUBIQUO APPS,SL
Myles Cameron-Smith
Neil Moore
Nelson
neoman
Nicholas Melnick
Nikolaj Brinch Jørgensen
Nils Andreas
Nils Eilers
Nils Hammerich
Nils77
Norah Smith
Norman King
Normen Zoch
Olaf Grunert
Ole Eitels
Oliver Boerner
Oliver Brüggmann
Oliver Graf
Oliver Smith
Olivier Bori
ONEPSI LLC
oRdYNe
Osaühing Trioflex
OSHA-PROS USA
Padawer
Patrick Becher
Patrick Bürckstümmer
Patrick de Zoete
Patrick Toal
Patrick Vogt
Paul Alexander Warren
Paul Gerhardt (KONG)
Paul Jackson
Paul Johnson
Paul Kuhnast (mindrail)
Paul Massay
Paul Westlake
Paul Woegerer
Pauline Brasch
Paulo Apolonia
Pete Collin
Pete of Retrohax.net
Peter Eliades
Peter Gries
Peter Habura
Peter Herklotz
Peter Huyoff
Peter Knörzer
Peter Leswell
Peter Weile
Petri Alvinen
Philip Marien
Philip Timmermann
Philipp Rudin
Pierre Kressmann
Pieter Labie
Piotr Kmiecik
Power-on.at
Przemysław Safonow
Que Labs
R Welbourn
R-Flux
Rafał Michno
Rainer Kappler
Rainer Kopp
Rainer Weninger
Ralf Griewel
Ralf Pöscha
Ralf Reinhardt
Ralf Schenden
Ralf Smolarek
Ralf Zenker
Ralph Bauer
Ralph Wernecke
Rédl Károly
Reiner Lanowski
Remi Veilleux
Riccardo Bianchi

Richard Englert
Richard Good
Richard Menedetter
Richard Sopuch
Rick Reynolds
Rico Gruninger
Rob Dean
Robert Bernardo
Robert Eaglestone
Robert Grasböck
Robert Miles
Robert Schwan
Robert Shively
Robert Tangmar
Robert Trangmar
Rodney Xerri
Roger Olsen
Roger Pugh
Roland Attila Kett
Roland Evers
Roland Schatz
Rolf Hass
Ronald Cooper
Ronald Hunn
Ronny Hamida
Ronny Preiß
Roy van Zundert
Rüdiger Wohlfromm
Ruediger Schlenter
Rutger WIllemsen
Sampo Peltonen
Sarmad Gilani
SAS74
Sascha Hesse
Scott Halman
Scott Hollier
Scott Robison
Sebastian Baranski
Sebastian Bölling
Sebastian Felzmann
Sebastian Lipp
Sebastian Rakel
Şemseddin Moldibi
Šeth Morabito
Shawn McKee
Siegfried Hartmann
Zytex Online Store
Sigurbjorn Larusson
Sigurdur Finnsson
Simon Lawrence
Simon Wolf
spreen.digital
Stefan Haberl
Stefan Kramperth
Stefan Richter
Stefan Schultze
Stefan Sonnek
Stefan Theil
Stefan Vrampe
Stefano Canali
Stefano Mozzi
Steffen Reiersen
Stephan Bielmann
Stephen Jones
Stephen Kew
Steve Gray
Steve Kurlin
Steve Lemieux
Steven Combs
Stewart Dunn
Stuart Marsh
Sven Neumann
Sven Stache
Sven Sternberger
Sven Wiegand
Szabolcs Bence
Tantrumedia Limited
Techvana Operations Ltd.
Teddy Turmeaux
Teemu Korvenpää
The Games Foundation
Thierry Supplisson
Thieu-Duy Thai
Thomas Bierschenk
Thomas Edmister
Thomas Frauenknecht
Thomas Gitzen
Thomas Gruber
Thomas Haidler
Thomas Jager
Thomas Karlsen
Thomas Laskowski
Thomas Marschall
Thomas Niemann
Thomas Scheelen
Thomas Schilling
Thomas Tahsin-Bey
Thomas Walter
Thomas Wirtzmann
Thorsten Knoll
Thorsten Nolte
Tim Krome
Tim Waite
Timo Weirich
Timothy Blanks
Timothy Henson
Timothy Prater
Tobias Butter
Tobias Heim
Tobias Köck
Tobias Lüthi
Tommi Vasarainen
Toni Ammer
Tore Olsen
Torleif Strand
Torsten Schröder
Tuan Nguyen
Uffe Jakobsen
Ulrich Hintermeier
Ulrich Nieland
Ulrik Kruse
Urban Lindeskog
Ursula Förstle
Uwe Anfang
Uwe Boschanski
Vedran Vrbanc
Verm Project
Wayne Rittimann
Wayne Sander
Wayne Steele
Who Knows
Winfried Falkenhahn
Wolfgang Becker
Wolfgang Stabla
Worblehat
www.patop69.net
Yan B
Zoltan Markus
Zsolt Zsila

Bibliografie

[1] L. Soares and M. Stumm, "Flexsc: Flexible system call scheduling with exception-less system calls." in *Osdi*, vol. 10, 2010, pp. 1–8.

[2] N. Montfort, P. Baudoin, J. Bell, I. Bogost, J. Douglass, M. C. Marino, M. Mateas, C. Reas, M. Sample, and N. Vawter, *10 PRINT CHR $(205.5+ RND (1));: GOTO 10*. MIT Press, 2012.

[3] Actraiser, "Vic-ii for beginners: Screen modes, cheaper by the dozen," 2013. [Online]. Available: http://dustlayer.com/vic-ii/2013/4/26/vic-ii-for-beginners-screen-modes-cheaper-by-the-dozen

Index

BASIC 65-arrays, 11
BASIC 65-commando's
APPEND, 24
AUTO, 27
BACKGROUND, 28
BACKUP, 29
BANK, 30
BEGIN, 31
BEND, 32
BLOAD, 33
BOOT, 35
BORDER, 36
BOX, 37
BSAVE, 39
BVERIFY, 42
CATALOG, 44
CHANGE, 46
CHAR, 48
CHARDEF, 50
CHDIR, 51
CIRCLE, 53
CLOSE, 55
CLR, 56
CLRBIT, 57
CMD, 58
COLLECT, 59
COLLISION, 60
COLOR, 61
CONCAT, 62
CONT, 63
COPY, 64
CURSOR, 67
CUT, 68
DATA, 69
DCLEAR, 71
DCLOSE, 72
DEF FN, 75
DELETE, 76
DIM, 78
DIR, 79
Directe modus, 9
DISK, 81
DLOAD, 82
DMA, 84
DMODE, 85
DO, 86
DOT, 89
DPAT, 90
DSAVE, 94
DVERIFY, 96
EDMA, 99
ELLIPSE, 102
ELSE, 104
END, 106
ENVELOPE, 107
ERASE, 110
EXIT, 112
FAST, 114
FGOSUB, 115
FGOTO, 116
FILTER, 117
FIND, 118
FONT, 121
FOR, 122
FOREGROUND, 123
FORMAT, 124
FREAD#, 126
FREEZER, 127
FWRITE#, 128
GCOPY, 129
GET, 130
GET#, 131
GETKEY, 132
GO64, 133
GOSUB, 134
GOTO, 135
GRAPHIC, 136
HEADER, 138
HELP, 140
HIGHLIGHT, 142
IF, 143
IMPORT, 145
INFO, 146
INPUT, 147

INPUT#, 149
INSTR, 151
KEY, 155
LET, 159
LINE, 160
LINE INPUT, 161
LINE INPUT#, 162
LIST, 163
LOAD, 165
LOADIFF, 167
LOCK, 169
LOOP, 173
MEM, 175
MERGE, 177
MKDIR, 179
MONITOR, 181
MOUNT, 182
MOUSE, 184
MOVSPR, 186
NEW, 188
NEXT, 189
OFF, 191
ON, 192
OPEN, 194
PAINT, 196
PALETTE, 197
PASTE, 199
PEN, 201
PLAY, 203
POLYGON, 209
PRINT, 212
PRINT USING, 214
PRINT#, 213
RCURSOR, 217
READ, 219
RECORD, 220
REM, 222
RENAME, 223
RENUMBER, 224
RESTORE, 226
RESUME, 227
RETURN, 228
RMOUSE, 231
RREG, 237
RUN, 243
SAVE, 246
SAVEIFF, 247
SCNCLR, 248
SCRATCH, 249
SCREEN, 250
SET, 253
SETBIT, 254
SLEEP, 257
SOUND, 258
SPEED, 261
SPRCOLOR, 262
SPRITE, 263
SPRSAV, 265
STEP, 268
STOP, 269
SYS, 272
TEMPO, 278
THEN, 279
TO, 282
TRAP, 283
TROFF, 284
TRON, 285
TYPE, 286
UNLOCK, 287
UNTIL, 288
USING, 289
VERIFY, 293
VIEWPORT, 294
VOL, 295
VSYNC, 296
WAIT, 297
WHILE, 298
WINDOW, 299
BASIC 65-constanten, 10
BASIC 65-functies
ABS, 22
ASC, 25
ATN, 26
BUMP, 41

CHR$, 52
COS, 66
DEC, 73
DECBIN, 74
ERR$, 111
EXP, 113
FN, 75, 120
FRE, 125
HASBIT, 137
HEX$, 141
INT, 152
JOY, 153
LEFT$, 157
LEN, 158
LOG, 170
LOG10, 171
LOG2, 172
LPEN, 174
MID$, 178
MOD, 180
PEEK, 200
PIXEL, 202
POINTER, 206
POKE, 208
POS, 210
POT, 211
RCOLOR, 216
RDISK, 218
RGRAPHIC, 229
RIGHT$, 230
RND, 232
RPALETTE, 233
RPEN, 234
RPLAY, 235
RPT$, 236
RSPCOLOR, 238
RSPEED, 239
RSPPOS, 240
RSPRITE, 241
RSPRSYS, 242
RWINDOW, 245
SGN, 255
SIN, 256
SPC, 260
SQR, 266
STR$, 270
STRBIN$, 271
TAB, 276
TAN, 277
USR, 291
VAL, 292
WPEEK, 300
WPOKE, 301
BASIC 65-operatoren, 13
AND, 23
NOT, 190
OR, 195
XOR, 302
BASIC 65-systeemcommando's
EDIT, 97
BASIC 65-systeemvariabelen
C, 43
DS, 92
DS$, 93
DT$, 95
EL, 101
ER, 108
ST, 267
&, 275
TI, 280
TI$, 281
BASIC 65-variabelen, 10
BASIC 65-voorbeelden
ABS, 22, 86, 173, 255, 298
AND, 23, 31, 41, 148, 150, 154, 190, 193, 211, 221, 259, 264, 281
APPEND, 24
ASC, 25
ATN, 26
AUTO, 27
BACKGROUND, 28
BACKUP, 29

BANK, 30, 57, 137, 168, 200, 207, 208, 254, 274, 297, 301
BEGIN, 31, 32, 105, 144, 245
BEND, 31, 32, 105, 144, 245
BLOAD, 34, 187, 218
BOOT, 35
BORDER, 36, 54, 202, 296
BOX, 37, 38, 68, 85, 89, 91, 129, 199
BSAVE, 40
BUMP, 41, 60
BVERIFY, 42
CATALOG, 44, 45
CHANGE, 47
CHAR, 49, 103
CHARDEF, 50
CHDIR, 51
CHR, 31, 52, 86, 104, 126, 128, 130, 132, 144, 156, 158, 161, 173, 207, 245, 275, 279, 288
CIRCLE, 54, 103
CLOSE, 49, 54, 55, 58, 68, 85, 89, 91, 103, 112, 129, 131, 136, 160, 168, 196, 198, 199, 201, 202, 209, 229, 233, 234, 247, 248, 252, 294
CLR, 56, 136, 168, 187, 198, 201, 229, 234, 248, 252, 259, 264, 280, 294
CLRBIT, 57, 137
CMD, 58, 194
COLLECT, 59
COLLISION, 60
COLOR, 61, 197
CONCAT, 62
CONT, 63
COPY, 65, 282
COS, 66, 75, 120, 122, 189, 268
CURSOR, 67, 217, 228, 275
CUT, 68, 199
DATA, 70, 78, 146, 168, 219, 226, 276
DCLEAR, 71
DCLOSE, 72, 126, 128, 134, 135, 150, 162, 213, 221, 267
DEC, 42, 73
DECBIN, 74
DEF, 75, 120, 136, 168, 198, 201, 229, 232, 234, 248, 252, 253, 294
DELETE, 77
DIM, 78, 122, 126, 128, 148, 150, 162, 189, 222, 267
DIR, 80, 179, 253
DISK, 81, 253
DLOAD, 83, 177, 218, 253
DMA, 84
DMODE, 85, 91
DO, 86, 112, 130, 131, 134, 135, 148, 173, 264, 288, 298
DOT, 89, 202
DPAT, 91
DSAVE, 94, 149
DVERIFY, 96
EDMA, 99, 176
ELLIPSE, 103
ELSE, 104, 105, 117, 144, 259, 279
END, 60, 70, 78, 101, 104–106, 109, 111, 116, 130, 132, 134, 135, 144, 227, 228, 239, 276, 279, 283
ENVELOPE, 107, 205
ERASE, 110
ERR, 101, 109, 111, 227, 283
EXIT, 112, 131, 148
EXP, 113, 140, 227, 283–285
FAST, 114, 253

FGOSUB, 115
FGOTO, 116
FILTER, 117
FN, 75, 120, 232
FONT, 121
FOR, 54, 60, 70, 75, 78, 89, 111, 117, 120, 122, 126, 128, 150, 162, 168, 180, 187, 189, 202, 207, 212, 213, 219, 221, 226–229, 232, 260, 264, 268, 275, 276, 278, 280–285, 295, 302
FORMAT, 124
FRE, 125
FREAD, 126, 128
FREEZER, 127
FWRITE, 126, 128
GCOPY, 129
GET, 31, 86, 112, 130, 131, 173, 202, 288
GETKEY, 49, 54, 68, 85, 89, 91, 103, 111, 129, 132, 136, 160, 196, 198, 199, 201, 209, 252, 274, 294
GO64, 133
GOSUB, 86, 105, 117, 134, 135, 154, 173, 193, 228, 298
GOTO, 31, 63, 115, 117, 130, 132, 134, 135, 148, 150, 154, 162, 193, 225, 235, 239, 255, 267, 278, 282, 295, 296
GRAPHIC, 136, 168, 198, 201, 229, 234, 248, 252
HASBIT, 137
HEADER, 139
HELP, 140
HEX, 141, 207, 281
IF, 23, 31, 32, 41, 54, 92, 93, 101, 104–106, 109, 111, 112, 117, 130–132, 134, 135, 144, 148, 150, 154, 162, 190, 193, 195, 202, 210, 221, 231, 235, 239, 245, 259, 264, 267, 269, 274, 278, 279, 295
IMPORT, 145
INFO, 146
INPUT, 104, 112, 115, 116, 131, 134, 135, 144, 148, 150, 161, 162, 221, 267, 279
INSTR, 151
INT, 152, 232
JOY, 154, 193
KEY, 156, 191, 193
LEFT, 157
LEN, 31, 86, 158, 173, 288
LET, 159
LINE, 112, 131, 136, 160–162, 196, 198, 201, 229, 247, 248, 252, 267
LIST, 58, 164, 194
LOAD, 156, 166, 264, 265
LOADIFF, 168
LOCK, 169
LOG, 113, 170
LOG10, 171
LOG2, 172
LOOP, 86, 112, 130, 131, 134, 135, 148, 173, 264, 288, 298
LPEN, 174
MEM, 176
MERGE, 177
MID, 178
MKDIR, 179
MOD, 111, 180, 275
MONITOR, 181
MOUNT, 183
MOUSE, 185, 191, 193, 202, 231, 253
MOVSPR, 60, 187, 202, 264

NEW, 188
NEXT, 54, 60, 70, 75, 78, 89, 101, 109, 111, 117, 120, 122, 126, 128, 150, 162, 168, 180, 187, 189, 202, 207, 212, 213, 219, 221, 226-229, 232, 260, 264, 268, 275, 276, 278, 280-285, 295, 302
NOT, 190
OFF, 149, 156, 183, 185, 191, 202, 228, 231
ON, 149, 154, 156, 185, 193, 202, 217, 228, 231, 253, 255
OPEN, 55, 58, 112, 131, 136, 168, 194, 198, 201, 229, 234, 248, 252
OR, 86, 134, 135, 148, 173, 195, 235, 288
PAINT, 196, 198
PALETTE, 54, 136, 168, 196-198, 201, 248, 252
PASTE, 68, 129, 199
PEEK, 78, 200, 207, 259, 281
PEN, 54, 68, 85, 91, 103, 136, 160, 196, 198, 199, 201, 229, 234, 247, 248, 252, 294
PIXEL, 202
PLAY, 107, 117, 204, 205, 235, 278, 295
POINTER, 126, 128, 207
POKE, 73, 208, 237, 274
POLYGON, 209
POS, 210
POT, 211
PRINT, 22, 23, 25, 26, 31, 32, 41, 52, 55, 56, 58, 60, 61, 63, 66, 67, 70, 73-75, 78, 92, 93, 95, 99, 101, 104-106, 109, 111-113, 115-117, 120, 122, 125, 126, 128, 130-132, 134, 135, 137, 140, 141, 144, 148, 150, 154, 157, 158, 161, 162, 170-172, 174, 178, 180, 189, 190, 193, 195, 200, 207, 210, 212, 213, 215, 217, 219, 221, 226-234, 236, 237, 239, 242, 245, 256, 260, 266-271, 275-277, 279-285, 290-292, 300, 302
RCOLOR, 216
RCURSOR, 217
RDISK, 218
READ, 70, 78, 219, 226, 276
RECORD, 221
REM, 27-32, 36, 41, 50-52, 54, 55, 57, 58, 60, 65, 67, 68, 72, 75, 77, 78, 81, 83, 84, 86, 99, 101, 103, 104, 106, 109-112, 114-117, 120, 121, 124, 127, 129-132, 134-136, 139, 144, 148-152, 154, 156, 159, 160, 162, 164, 166, 168, 169, 173, 174, 176, 179, 183, 185, 188, 191, 193, 194, 196-202, 205, 207-211, 216, 218, 221-223, 225, 228, 229, 231-235, 237-241, 244, 245, 247-249, 252-255, 257, 259-262, 265, 267, 269, 274, 279-282, 284, 285, 287, 291, 294, 296-301
RENAME, 223
RENUMBER, 225
RESTORE, 156, 168, 188, 226

RESUME, 101, 109, 111, 227, 283
RETURN, 60, 105, 115, 117, 134, 135, 154, 193, 228
RGRAPHIC, 229
RIGHT, 230
RMOUSE, 185, 202, 231
RND, 54, 202, 232
RPALETTE, 233
RPEN, 234
RPLAY, 235, 278, 295
RPT, 236
RREG, 237
RSPCOLOR, 238
RSPEED, 239
RSPPOS, 240, 264
RSPRITE, 241
RSPRSYS, 242
RUN, 56, 63, 70, 75, 78, 120, 140, 150, 161, 178, 207, 217, 221, 229, 233, 234, 244, 260, 270, 276, 281, 284, 285, 291
RWINDOW, 245
SAVE, 156, 246
SAVEIFF, 247
SCNCLR, 54, 67, 111, 136, 168, 187, 228, 229, 234, 248, 264
SCRATCH, 249
SCREEN, 49, 54, 68, 85, 89, 91, 103, 129, 136, 160, 168, 176, 196, 198, 199, 201, 202, 209, 229, 233, 234, 247, 248, 252, 294
SET, 136, 168, 198, 201, 229, 234, 248, 252, 253
SETBIT, 137, 185, 254
SGN, 255
SIN, 75, 120, 122, 189, 256, 268
SLEEP, 67, 117, 168, 187, 228, 248, 252, 257, 259, 264
SOUND, 259
SPC, 260
SPEED, 146, 261
SPRCOLOR, 262
SPRITE, 60, 187, 238, 240–242, 262, 264, 265
SPRSAV, 265
SQR, 101, 109, 111, 212, 213, 215, 266, 269, 290
STEP, 75, 117, 120, 122, 189, 221, 268, 278, 281, 295
STOP, 92, 93, 112, 131, 150, 219, 231, 267, 269
STR, 270
STRBIN, 271
SYS, 35, 133, 237, 242, 274
TAB, 276
TAN, 122, 189, 268, 277
TEMPO, 107, 205, 278, 295
THEN, 23, 31, 32, 41, 54, 92, 93, 101, 104–106, 109, 111, 112, 117, 130–132, 134, 135, 144, 148, 150, 154, 162, 190, 193, 195, 202, 210, 221, 231, 235, 239, 245, 259, 264, 267, 269, 274, 278, 279, 295
TO, 29, 40, 47, 54, 60, 62, 65, 70, 75, 78, 89, 111, 117, 120, 122, 126, 128, 150, 162, 168, 180, 187, 189, 202, 207, 212, 213, 219, 221, 223, 226–229, 232, 253, 260, 264, 268, 274–276, 278, 280–285, 295, 302
TRAP, 101, 109, 111, 227, 283
TROFF, 284, 285
TRON, 284, 285
TYPE, 150, 286

UNLOCK, 287
UNTIL, 86, 130, 134, 135, 148, 173, 264, 288
USING, 75, 111, 120, 215, 281, 290
USR, 291
VAL, 292
VERIFY, 253, 293
VIEWPORT, 89, 294
VOL, 107, 205, 278, 295
VSYNC, 296
WAIT, 297
WHILE, 86, 173, 298
WINDOW, 299
WPEEK, 207, 300
WPOKE, 291, 301
XOR, 302

Commodore 64
    GO64-modus, 133

Machinetaalmonitor
    MONITOR-commando, 181

Schermtekst- en kleurenarrays, 12

Toetsenbord
    CTRL, 315
    Cursortoetsen, 321
    Escape-reeksen, 318
    PETSCII-codes en CHR$, 52, 309
    Schermcodes, 305
    Shifttoetsen, 318
Trefwoorden, 7

Vensters
    WINDOW-commando, 299