HANDBOEK voor ZX SPECTRUM 128 + 2

I. Spital
R. Goodwins

ZX Spectrum +2

sinclair

HANDBOEK voor ZX SPECTRUM 128 + 2

I. Spital
R. Goodwins

TERMINAL SOFTWARE PUBLICATIES

Een uitgave van: Terminal Software Publicaties
Postbus 111, 5110 AC Baarle Nassau

1e druk april 1987

ISBN 90-6883-029-5

Vertaald uit het Engels door P.Pauwels

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Spital, Ivor

Handboek voor de Spectrum 128K Plus 2 / Ivor Spital,
Rupert Goodwins ; [vert. uit het Engels door P. Pauwels].
- Baarle Nassau : Terminal Software Publicaties
Vert. van: ZX Spectrum +2. - Brentwood : Sinclair Computer
Division, 1986. - Met reg.
ISBN 90-6883-029-5
SISO 525 UDC 681.31+681.3.06
Trefw.: Sinclair ZX Spectrum 128K Plus 2 (computer) /
programmeren (computer).

INHOUD

INLEIDING

Sinclair ZX Spectrum +2
128K Home Computer

EEN HOOGTEPUNT

Verderbouwend op het enorme sukses van de bestaande ZX-computers (de originele Spectrum, de Spectrum + en de Spectrum 128), kunnen we u nu, met rechtmatige trots, de ZX Spectrum +2 voorstellen. Deze machine is het resultaat van de samenwerking tussen de ingenieuze technologie van Sinclair en de ondervinding die Amstrad heeft opgedaan in integratie en de fabricage van betrouwbare machines.

UITWISSELBAARHEID VAN SOFTWARE

De +2 kan software "draaien" die werd geschreven voor vroegere machines uit de ZX-gamma. Dit houdt in dat er voor de +2 reeds een bijzonder groot aanbod aan software bestaat. Letterlijk duizenden titels zijn er, die alle mogelijke toepassingsgebieden omvatten : spelletjes, programmeerhulpen (utilities), muziek, wetenschap, opvoeding en nog veel meer.

sinclair
Quality Control

Wanneer u software koopt, let er dan op dat het "Sinclair Quality Control" logo op de verpakking zit. We raden u aan om software enkel te kopen van producenten die deel uitmaken van dit netwerk. We willen immers vermijden dat u niet-uitwisselbare software koopt, ofwel software met een misleidend label.

IETS OVER DIT BOEK ZELF

Dit boek is niet bedoeld als een allesomvattende handleiding in het programmeren van de +2. Wilt u dieper graven, dan zijn er al veel publicaties beschikbaar voor de Spectrum + en de Spectrum 128, die daar uitstekend voor geschikt zijn en die alle aspecten van de ZX Spectrum computers en de Sinclair BASIC behandelen.

Wanneer u evenwel enkel wilt weten hoe u de computer dient aan te sluiten, hoe u randapparatuur moet aansluiten, indien u de grondregels van het programmeren in BASIC wilt kennen, en software en spelletjes in wilt kunnen laden, dan kan dit boek ruimschoots volstaan.

AMSTRAD
CONSUMER ELECTRONICS PLC.

Het product dat in dit boek beschreven wordt, alsmede producten die ermee gebruikt kunnen worden, zijn het voorwerp van voortdurende ontwikkeling en verbetering. Alle informatie van technische aard en bijzonderheden over het product en het

gebruik ervan (met inbegrip van de informatie en de bijzonderheden in dit boek) worden door Amstrad in goed vertrouwen verstrekt.

Alle onderhoud en elke reparatie moet door erkende Sinclair verdelers worden uitgevoerd. Amstrad wijst elke aansprakelijkheid af voor verlies of schade, veroorzaakt door onderhoud of reparaties, uitgevoerd door onbevoegden. Deze handleiding is enkel bedoeld om de lezer behulpzaam te zijn bij het gebruik van het product. Daarom wijst Amstrad alle verantwoordelijkheid af voor verlies of schade veroorzaakt door het gebruik van informatie of bijzonderheden in dit boek of door fouten of vergetelheden erin, of door een verkeerd gebruik van het product.

Wij verzoeken de gebruikers, hun registratie- en garantiekaart terug te sturen.

Elke briefwisseling over het product dient gericht te worden aan

Terminal Software Publicaties
Postbus 111
5110 AC Baarle-Nassau

BELANGRIJK : LEES DIT IN ELK GEVAL ...

1. Sluit de voeding enkel aan op 220 V wisselspanning.

2. Trek altijd de stekker van de voeding uit het stopkontakt, wanneer u met de computer klaar bent.

3. In de de computer zit niets dat door de gebruiker gerepareerd of onderhouden kan worden. PROBEER NIET OM HET VOEDINGSBLOK OPEN TE MAKEN : DAT IS MET HET LICHTNET VERBONDEN ! Laat alle onderhoudswerkzaamheden door vaklui uitvoeren.

4. Laat de ventilatie-openingen vrij.

5. Gebruik de computer niet op een overdreven koude, warme, vochtige of stoffige plaats.

6. Sluit nooit iets aan of haal nooit iets af van de expansiebus terwijl de +2 aan staat. Meer dan waarschijnlijk beschadigt u daarmee zowel de computer als het randapparaat zelf.

7. Wacht een paar seconden met het ontkoppelen van de computer, nadat u de TV of monitor hebt uitgeschakeld.

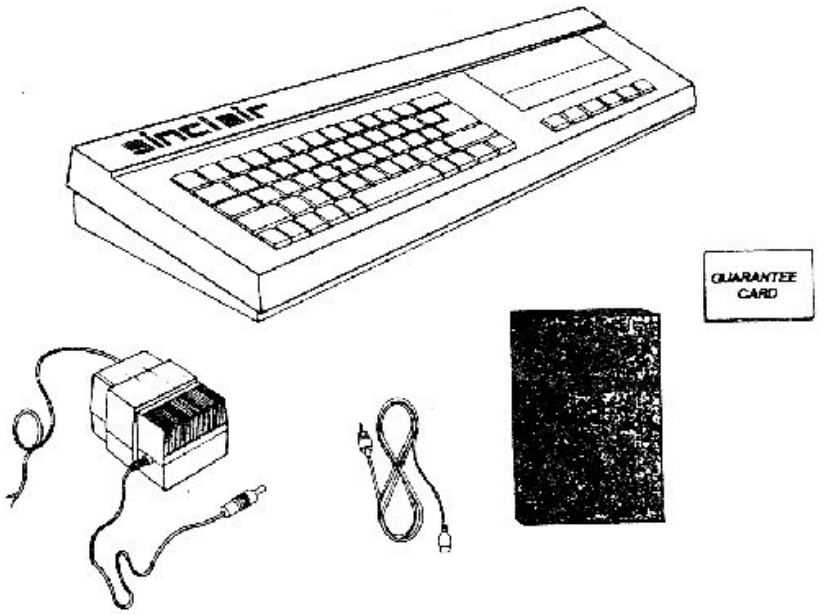
8. Schakel de +2 niet uit, terwijl er nog gegevens in het geheugen zitten die u wenst te bewaren. Zet die eerst op band. Schakel randapparatuur die met de computer verbonden is, niet aan of uit : daardoor kan de +2 "crashen", waardoor de gegevens in het geheugen verloren zijn.

HOOFDSTUK 1 UITPAKKEN

Inhoud ...
- uitpakken
- opstellen

UITPAKKEN

In de verpakking vindt u :
De Spectrum+2 computer
Het voedingsblok
De antennedraad
Dit handboek, samen met de garantiekaart



OPSTELLEN

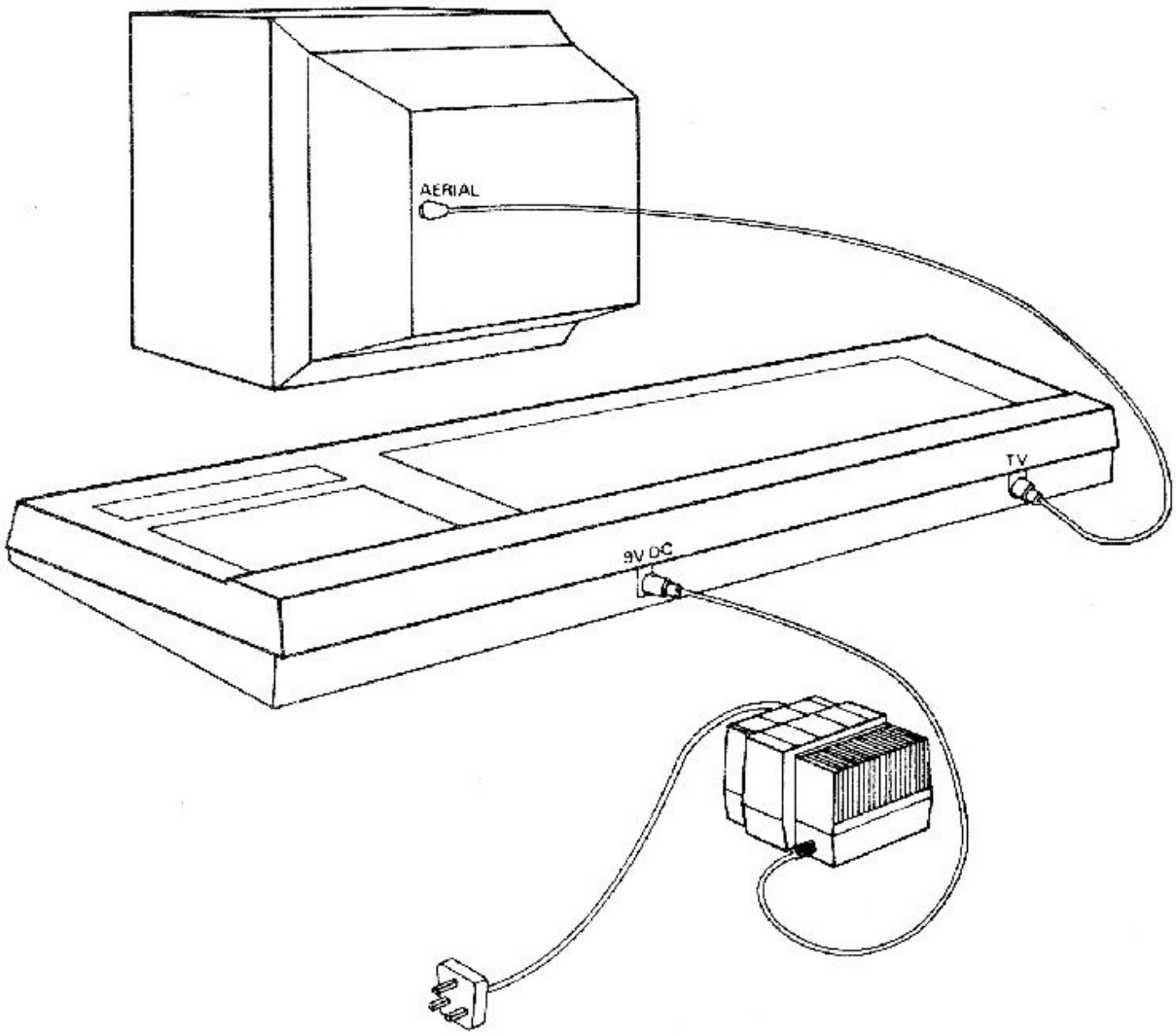
We gaan nu het standaard +2 systeem opstellen. Buiten wat u in de verpakking vond, hebt u enkel nog een gewoon UHF TV-toestel nodig, kleuren of zwart/wit. Met een zwart/wit toestel zult u uiteraard niets zien van de kleuren die uw computer produceert.

Indien u randapparaten wilt aansluiten (bv. een of twee joysticks, microdrives, een monitor, een apart numeriek toetsenbord, een geluidsversterker, een MIDI-interface, een printer of andere apparatuur) dan moet u eerst hoofdstuk 10 lezen : Randapparatuur aansluiten op uw +2.

Plaats de +2 op een horizontaal oppervlak, klaar om aan het TV-toestel te worden aangesloten. Ontkoppel de antennekabel die eventueel nog aan het toestel is aangesloten. Stop de grootste van de twee stekkers die aan de meegeleverde antennekabel zitten in de antennebus van het TV-toestel, en de kleinste stekker in de aansluiting op de computer waar "TV" bij staat.

Stop de ronde stekker die aan het voedingsblok zit, in de bus waarboven "9V DC" staat.

Nu is de +2 klaar om ingeschakeld te worden.



De standaard opstelling van het +2 systeem

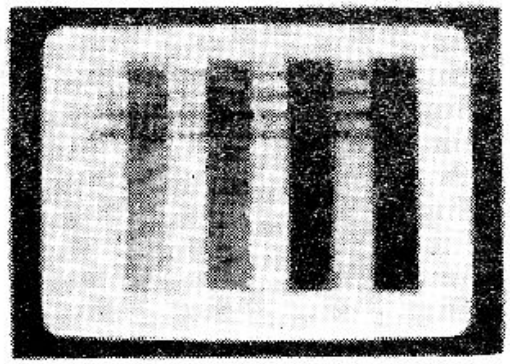
HOOFDSTUK 2 WERKEN MET DE +2

Inhoud ...

INSCHAKELEN
Stop de netstekker in het stopcontact en zet de schakelaar op de computer aan, indien nodig. Het rode lampje op de computer zou nu moeten oplichten.

Zet het TV-toestel aan. Op het scherm ziet u nu waarschijnlijk "sneeuw"; u hoort een luide ruis. Regel het geluid volgens smaak. Nu moeten we de +2 klaarzetten voor de afregeling van het TV-toestel.

VOORBEREIDING VAN DE AFSTEMMING
De +2 kan zijn eigen testbeeld produceren, waarmee u uw TV-toestel precies kunt afstemmen. Dit testbeeld bestaat uit zestien vertikale kleurenbalken (met tekst erin) op het scherm, en daarbij hoort u een toon via de luidspreker, die geregeld wordt herhaald. Indien u een zwart/wit TV gebruikt, ziet u zestien balken met verschilende grijstinten. U ziet dit testbeeld pas nadat u de afstemming hebt uitgevoerd, zoals verder wordt beschreven.



Om de +2 het testbeeld te doen produceren, houdt u de "BREAK"-toets (rechtsboven het toetsenbord) ingedrukt en drukt u op de RESET- knop, links op de +2. Houd de BREAK-toets enkele seconden ingedrukt, en laat dan los. Nu stuurt de +2 zijn testbeeld naar het TV-toestel, en kan de afstemming beginnen.

KANAALKEUZE MET DRUKTOETSEN

Indien uw TV-toestel geen druktoetsen heeft, waarmee u de zenders kiest, sla dan dit stuk tekst over en lees verder vanaf "Afstemmen met de hand".

Druk nu op een van de keuzetoetsen op de TV, die normaal niet bij een zender hoort of die niet bestemd is voor video-weergave. Indien uw TV voorzien is van een automatische fijnafstemming, dient deze uitgeschakeld te worden.

Zoek nu met de bijbehorende afstemknop, tot u het testbeeld van de computer op uw scherm ziet, zoals dit op vorige bladzijde staat afgebeeld. Zorg ervoor dat zowel beeld en geluid optimaal doorkomen.

Bent u tevreden met de afstemming, dan kunt u nu weer de automatische fijnafstemming inschakelen, indien dat op uw

toestel voorzien is.

Regel nu de helderheid, het kontrast en de kleurverzadiging zo bij, dat de tekst in de balken goed leesbaar is.

U hebt nu een van de keuzetoetsen geprogrammeerd om het signaal van de +2 te ontvangen. Later kunt u dus gewoon op uw computer "afstemmen" door op die toets te drukken.

Als alles probleemloos verloopt, kunt u het volgende overslaan en doorlezen vanaf "De +2 gebruiken !".

AFSTEMMEN MET DE HAND

Indien uw TV-toestel niet voorzien is van keuzetoetsen, dan dient u af te stemmen met de draaiknop.

Nadat u de +2 en de TV hebt ingeschakeld, zorgt u ervoor dat de computer het testbeeld uitzendt. Lees daarvoor "Voorbereiding van de afstemming".

Draai de afstemknop zolang tot u het testbeeld ziet. Stem zo af, dat u geluid en beeld optimaal ontvangt.

Regel nu de helderheid, het kontrast en de kleurverzadiging zo bij, dat de tekst in de balken goed leesbaar is.

Elke keer u later de computer met uw TV-toestel wenst te gebruiken, dient u deze handelingen te verrichten.

Als alles probleemloos verloopt, kunt u het volgende overslaan en doorlezen vanaf "De +2 gebruiken !".

PROBLEMEN ?

Als alles probleemloos is verlopen, kunt u het volgende overslaan en doorlezen vanaf "De +2 gebruiken".

Lukt het u niet om op het signaal van de computer af te stemmen, dan kan de volgende lijst u helpen met het zoeken naar de oorzaak, en een indicatie geven van wat u kunt ondernemen.

1. Probleem...

Het "ON"-lampje bovenaan de computer brandt niet.

Wat te doen :

* controleer of de voeding op de computer aangesloten is.
* controleer of de stekker van de voeding in het stopcontact zit.
* indien het stopcontact kan aan- en uitgeschakeld worden, controleer dan of de schakelaar "aan" staat.
* controleer of de stekker goed is aangesloten.

2. Probleem...

Het "ON"-lampje brandt, maar het is niet mogelijk om enig signaal op de tv te krijgen.

Wat te doen :

* controleer of het tv-toestel aan staat en goed werkt.
* controleer of uw tv-toestel UHF-kanalen kan ontvangen.
* controleer of de meegeleverde antennekabel de computer met het tv-toestel verbindt.

* indien u voorkeuzetoetsen op het toestel hebt, controleer dan of u wel de toets hebt ingedrukt die u net op de computer hebt afgestemd.

3. Probleem...

Het signaal dat u op het tv-toestel krijgt is erg zwak.

Wat te doen :

* controleer of het tv-toestel verder goed werkt.
* controleer of de meegeleverde antennekabel volledig in de stekkers van het toestel en de computer zit.
* indien op uw toestel een AFC of AFT-knop zit, controleer dan of hij uit staat.
* controleer of de afstemming wel goed gebeurd is.

4. Probleem...

Het tv-toestel ontvangt wel goed het signaal van de computer, maar het is niet het hoger beschreven test-signaal.

Wat te doen :

* controleer of u wel het test-signaal hebt ingeschakeld; hoe u dit moet doen, wordt hoger beschreven, onder "VOORBEREIDING VAN DE AFSTEMMING".

5. Probleem...

Op de tv ziet u wel de gekleurde balken, maar u hoort geen geluid (een repeterende toon).

Wat te doen :

* controleer of het volume van de tv niet dicht staat.
* controleer of u wel precies hebt afgestemd.

6. Probleem...

U hoort wel geluid (een repeterende toon), maar u ziet niet de gekleurde balken.

Wat te doen :

* controleer of de helderheid, het contrast en de kleurverzadiging niet dichtgedraaid staan.
* controleer of u wel precies hebt afgestemd.

7. Probleem...

U ziet de gekleurde balken en u hoort het geluid, maar u kunt de tekst op het scherm niet lezen.

Wat te doen :

* controleer of u wel precies hebt afgestemd.
* controleer de afstelling van de helderheid, het contrast en de kleurverzadiging.

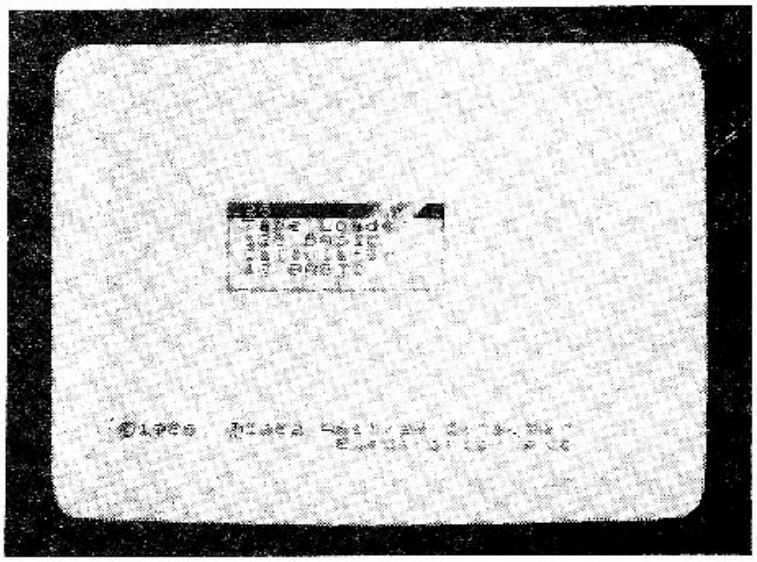
Indien u de oorzaak van een probleem niet kunt vinden, probeer dan eens de hele procedure opnieuw uit te voeren, vanaf het begin van dit hoofdstuk. Blijft alles hetzelfde, vraag dan uitleg aan de Sinclair verkoper waar u de computer kocht.

De +2 GEBRUIKEN

Het hele +2 systeem zou nu klaar moeten staan om ermee te werken, met de gekleurde balken op het scherm en de repeterende toon hoorbaar door de luidspreker van het tv-toestel.

Schakel nu het test-signaal uit : druk de "RESET"-knop in, links op de +2. Het test-signaal verdwijnt van het scherm. In de plaats ervan verschijnt het "startmenu".

HET START-MENU



Het start-menu verschijnt op het scherm, elke keer u de computer inschakelt of elke keer u de "RESET"-knop indrukt.

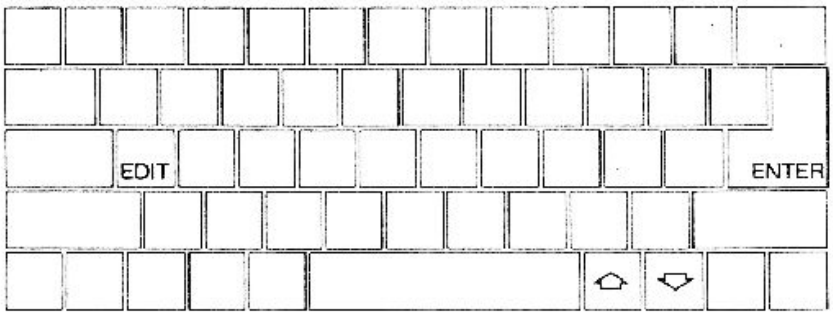
Zoals de naam al zegt, biedt het start-menu u een aantal keuzes. U kunt kiezen uit vier mogelijkheden, die in het midden van het scherm staan. Die mogelijkheden zijn :

Tape Loader	: deze keuze laat toe om software voor de Spectrum 128 in te laden
128 BASIC	: deze keuze laat toe om de +2 te gebruiken om zelf te programmeren.
Calculator	: deze keuze laat toe om de +2 als rekenmachine te gebruiken.
48 BASIC	: deze keuze laat toe om software voor de Spectrum 48K in te laden, of om de +2 als een Spectrum 48 K te gebruiken.

HOE MAAKT U EEN KEUZE ?

Op het scherm merkt u dat de keuze "Tape Loader" in een gekleurde balk staat. Dit betekent dat deze keuzemogelijkheid klaar staat om te worden uitgevoerd. De keuze werd evenwel nog niet bevestigd. Laten we aannemen dat u deze mogelijkheid NIET wenst te kiezen, maar dat u "128 BASIC" wilt kiezen. Dit houdt in, dat u de gekleurde balk zult moeten verplaatsen tot boven de

tekst "128 BASIC". Daartoe moet u op de cursor-toetsen drukken (zie hieronder), totdat de balk in de gewenste positie staat.

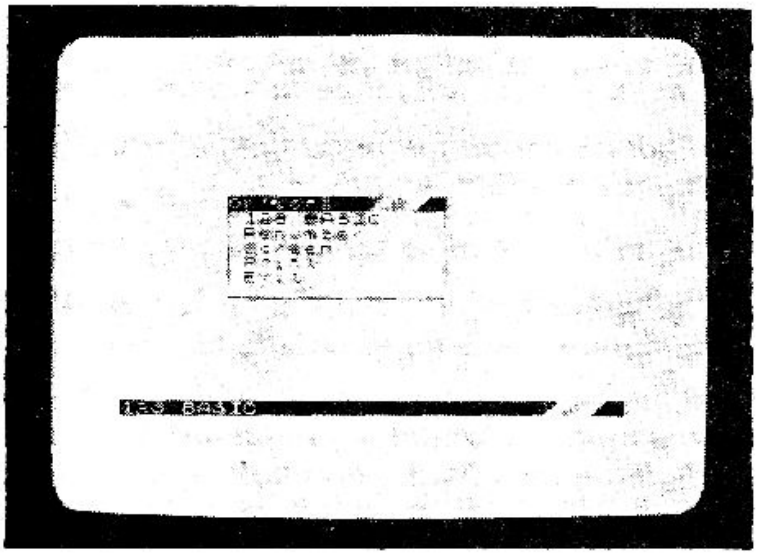


Cursor Keys

Staat de balk op de gewenste plaats, druk dan op de "ENTER"-toets om de keuze te bevestigen.

De +2 gaat nu in 128 BASIC-mode staan. U ziet onderaan het scherm een zwarte horizontale balk, en een knipperend vierkantje (de cursor) in de linker bovenhoek van het scherm.

Maak u geen zorgen, indien u nog niets over programmeren in BASIC weet : we gaan nog niet meteen programmeren. We keren gewoon terug naar het startmenu. Daartoe moeten we eerst een tweede "menu" oproepen, het "edit-menu". Dit doen we door op de "EDIT"-toets te drukken.



Dit menu werkt precies zoals het start-menu : kies door middel van de cursortoetsen, de optie "Exit", en druk op de "ENTER"-toets.

U kunt nu de keuze maken die u wenst. Afhankelijk van uw keuze, verwijzen we u naar een hoofdstuk in dit boek, waar uw keuze uitgebreid wordt besproken.

Tape Loader : zie hoofdstuk 3
128 BASIC : zie hoofdstukken 5, 6 en 8
Calculator : zie hoofdstuk 9
48 BASIC : zie hoofdstukken 4,5,7 en 8

HOOFDSTUK 3 SOFTWARE INLADEN VOOR DE SPECTRUM 128

Inhoud ...
- software inladen ("loaden")
- het inladen onderbreken
- de +2 resetten

WEES OP UW HOEDE VOOR SOFTWARE DIE NIET HET LOGO VAN "SINCLAIR QUALITY CONTROL" DRAAGT. Voor meer informatie, lees de inleiding van dit handboek.

Om software voor de Spectrum 128 te laden (een spel, een programmeerhulp ("utility") enz.), volgt u deze instructies :

1. Schakel de +2 in, zodat u het startmenu op het scherm ziet.

2. Kies in het startmenu de optie "Tape Loader". Indien u niet weet hoe u dit moet doen, lees dan eerst hoofdstuk 2.

3. Stop de cassette met de software in de datacorder en controleer of de band helemaal teruggespoeld is.

4. Laat de cassette afspelen. Tijdens het laden zult u de rand van het scherm (de "border") van kleur zien veranderen : dat is een teken dat het programma van de cassette wordt ingelezen. Indien het geluid van uw tv-toestel aan staat, zult u ook een hoge veranderlijke toon horen. Dit is eveneens een teken dat het programma wordt ingelezen.

Indien u het inladen wenst te onderbreken, drukt u op de "BREAK"-toets tot u opnieuw het startmenu ziet.

Voor de meeste software in de handel heeft de +2 enkele minuten nodig om ze in te lezen. In het begin zult u zien dat de mededeling "Program:" gevolgd door een naam, in de linker bovenhoek van het scherm verschijnt. Daarna kunnen er diverse teksten en/of tekeningen op het scherm verschijnen, afhankelijk van het programma dat wordt ingelezen.

Indien het programma ingeladen is, moet u de cassetterecorder stopzetten. U kunt nu met het programma werken.

Bent u klaar met het programma, en wilt u de +2 voor iets anders gaan gebruiken, dan drukt u kort op de "RESET"-knop, links op de computer. Denk er wel aan dat, wanneer u op deze knop drukt, de hele inhoud van het geheugen van de computer wordt gewist. Controleer daarom altijd of u wel echt helemaal klaar bent met het betreffende programma, vooraleer u op deze knop drukt.

HOOFDSTUK 4 SOFTWARE INLADEN VOOR DE SPECTRUM 48

Inhoud ...
- software inladen
- het inladen onderbreken
- de +2 resetten

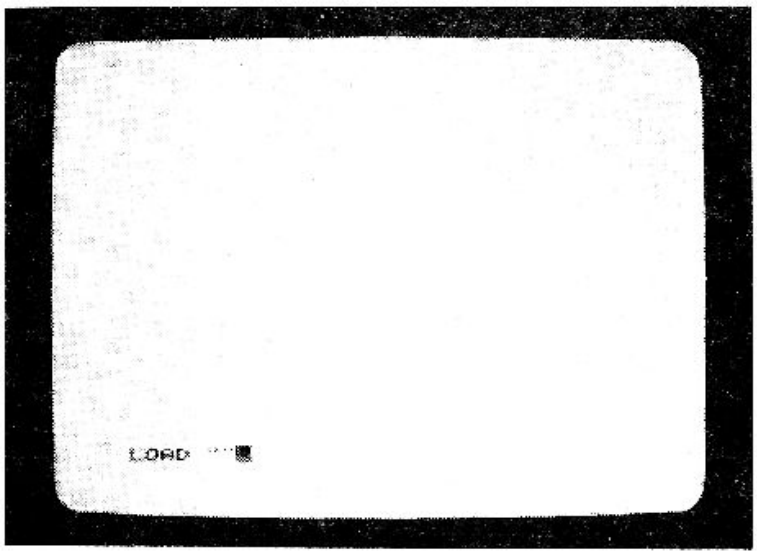
WEES OP UW HOEDE VOOR SOFTWARE DIE NIET HET LOGO VAN "SINCLAIR QUALITY CONTROL" DRAAGT. Voor meer informatie, lees de inleiding van dit handboek.

Om software voor de Spectrum 48K in te laden, volgt u deze instructies :

1. Schakel de +2 in, zodat u het startmenu op het scherm ziet.

2. Kies in het startmenu de optie "48 BASIC". Indien u niet weet hoe u dit moet doen, lees dan eerst hoofdstuk 2.

3. Het startmenu verdwijnt van het scherm, en onderaan het scherm ziet u de tekst "(c) 1982 Amstrad". Druk nu op de "J", en dan twee keer op de " (aanhalingsteken). Het scherm zou er nu moeten uitzien zoals op de foto :



Indien het scherm er niet uitziet zoals op de foto, dan hebt u ofwel de verkeerde optie gekozen, in het startmenu, ofwel hebt u op een andere toets dan de "J" gedrukt. Druk in dat geval op de "RESET"-toets, links op de +2, en herbegin bij stap 2 hierboven.

Klopt het beeld op uw tv-scherm met de foto, druk dan op de "ENTER"-toets.

4. Stop de cassette met de software in de datacorder, en controleer of de band helemaal teruggespoeld is.

5. Laat de cassette afspelen. In het begin zult u merken dat de rand van het scherm (de "border") van kleur verandert. Dit is een teken dat het programma van de cassette wordt ingelezen. Indien het geluid van uw tv-toestel aan staat, hoort u ook een hoge, veranderende toon. Ook dit is een teken dat het programma wordt ingelezen.

Indien u het inladen wilt stopzetten, drukt u op de "BREAK"-toets, tot het scherm wordt gewist. U keert dan terug naar de "48 BASIC"-mode. Wilt u terug naar het startmenu, druk dan op de "RESET"-knop.

Voor de meeste software in de handel heeft de +2 enkele minuten nodig om ze in te lezen. In het begin zult u zien dat de mededeling "Program:" gevolgd door een naam, in de linker bovenhoek van het scherm verschijnt. Daarna kunnen er diverse teksten en/of tekeningen op het scherm verschijnen, afhankelijk van het programma dat wordt ingelezen.

Indien het programma ingeladen is, moet u de cassetterecorder stopzetten. U kunt nu met het programma werken.

Bent u klaar met het programma, en wilt u de +2 voor iets anders gaan gebruiken, dan drukt u kort op de "RESET"-knop, links op de computer. Denk er wel aan dat, wanneer u op deze knop drukt, de hele inhoud van het geheugen van de computer wordt gewist. Controleer daarom altijd of u wel echt helemaal klaar bent met het betreffende programma, vooraleer u op deze knop drukt.

HOOFDSTUK 5 INLEIDING BASIC PROGRAMMEREN

De +2 "spreekt" een computertaal die "BASIC" wordt genoemd. Dit woord bestaat uit de beginletters van Beginners' All-purpose Symbolic Instruction Code. In het Nederlands : Symbolische Instructie- code voor beginners, te gebruiken voor alle doeleinden. BASIC is veruit de meest gebruikte taal voor home-computers. Bijna elke computer heeft evenwel zijn eigen "dialect" van BASIC, en daarop is de +2 geen uitzondering. De BASIC van de Spectrum werd ontwikkeld met het oog op bruikbaarheid, en is daardoor gemakkelijk om aan te leren. Hij verschilt op nogal wat punten van andere BASICs. In hoofdstuk 8 vindt u een complete handleiding voor de BASIC van de +2. Indien u nog nooit hebt geprogrammeerd, leest u het beste eerst hoofdstuk 6 : Gebruik van de 128 BASIC. Ook al bent u een doorgewinterde BASIC programmeur op andere computers, dan nog is het aangeraden om hoofdstuk 6 te lezen : daarin wordt de "editor" en andere unieke aspekten van de +2 beschreven.

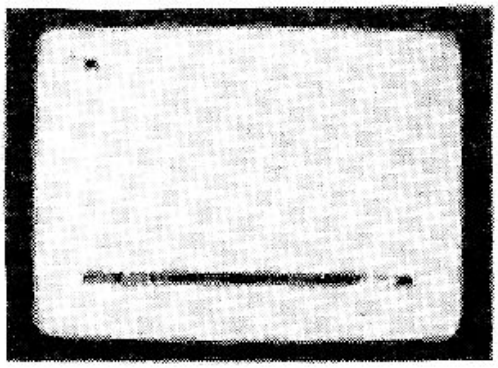
Indien u de 48K Spectrum gewend bent, dan zal veel van de inhoud van dit handboek u vertrouwd lijken. U kunt zelfs de +2 precies laten werken alsof het een "oude" Spectrum was, zelfs voor wat betreft de editor en het programmeren zelf. De nieuwsgierigen onder u kunnen dit als een geschiedenisles beschouwen. Mocht u deze "mode" toch willen uitproberen, de relevante informatie daarvoor staat in hoofdstuk 7 : Gebruik van de 48 BASIC.

HOOFDSTUK 6 HET GEBRUIK VAN 128 K BASIC

Inhoud ...
- de editor
- het edit-menu
- hernummeren van een programma
- schermen uitwisselen
- listen op de printer
- een programma intypen
- de cursor bewegen
- een programma laten uitvoeren
- commando's en instructies

De +2 beschikt over een gesofistikeerde editor, waarmee 128 BASIC programma's geschreven, gewijzigd en uitgevoerd kunnen worden. Om in de editor te komen, kiest u de optie "128 BASIC" in het startmenu, door middel van de cursortoetsen en de "ENTER"-toets. Indien u niet weet hoe u dit moet doen, lees dan eerst hoofdstuk 2.

Het scherm zou er nu zo moeten uitzien :



U kunt drie dingen zien :

Ten eerste ziet u een blauw-wit knipperend blokje links bovenaan het scherm. Dit noemen we de "cursor". Indien u een letter op het toetsenbord aanslaat, verschijnt die op de plaats waar de cursor zich bevond.

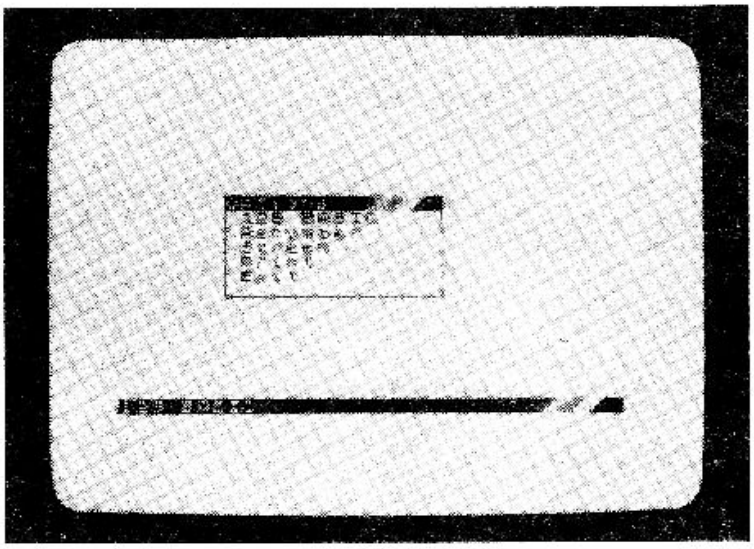
Ten tweede ziet u een zwarte balk onderaan het scherm. Die noemen we de "voetbalk". Daaraan kunt u merken welk deel van de ingebouwde software u aan het gebruiken bent. Op dit ogenblik staat daar "128 BASIC" : zo heet de editor.

Ten laatste ziet u een kleine ruimte tussen de balk en de onderkant van het scherm. Daar staat momenteel niets. In die ruimte kunnen twee regels tekst staan. Dit "kleine scherm" wordt veelal door de +2 zelf gebruikt, om melding te maken van een programmeerfout die hij heeft ontdekt in uw programma. Soms kan dit deel van het scherm ook voor andere doeleinden gebruikt worden, maar daarover hebben we het later.

Druk nu op de "EDIT"-toets. Er gebeuren twee dingen : de cursor verdwijnt en er verschijnt een menu. Dit heet het "edit-menu".

De opties in dit menu kunnen op dezelfde manier worden gekozen als in het startmenu, door middel van de cursortoetsen en de "ENTER"-toets.

Laten we de opties één voor één bekijken ...



128 BASIC : Deze optie laat gewoon het edit-menu verdwijnen en zet de cursor terug op het scherm. Op het eerste gezicht een weinig nuttige optie, maar ze laat wel toe om zonder gevolgen naar uw programma terug te keren, indien u per ongeluk op de "EDIT"-toets had gedruk

Renumber : Een BASIC programma gebruikt regelnummers, om te bepalen in welke volgorde de gegeven instructies uitgevoerd dienen te worden. Deze regelnummers typt u zelf in, aan het begin van elke programma-regel. Elk geheel getal van 1 tot 9999 kan een regelnummer zijn. De "Renumber"-optie hernummert het hele BASIC programma op zo'n manier dat de eerste regel het nummer 10 krijgt, en de daaropvolgende regels telkens 10 hoger. BASIC commando's die naar een bepaalde regel verwijzen (GO TO, GO SUB, LINE, RESTORE, RUN, LIST) worden eveneens aangepast.

Indien het om een of andere reden niet mogelijk is om het programma te hernummeren, dan produceert de +2 een lage toon, en het menu verdwijnt. Dit zal gebeuren indien er geen programma in het geheugen aanwezig is, of indien door de Renumber-opdracht sommige regelnummers hoger dan 9999 zouden worden.

Indien BASIC nieuw is voor u, slaat u het beste de volgende paragraaf over, en leest u verder vanaf "Screen".

Het is mogelijk om, via een truc, het hernummeren te laten uitvoeren met gebruik van andere waarden dan 10 als beginregel en als stapgrootte. Dit is bijvoorbeeld nodig indien het programma meer dan 1000 regels bevat : dat kan niet hernummerd worden in stappen van 10. Dit doet u door middel van de volgende instructies :

(Indien u niets kent van Spectrum BASIC, zult u vermoedelijk niet begrijpen hoe het werkt ...)

```
LET start =5: LET stepsize=2: LET histart=INT (start/256):
LET histep=INT (stepsize/256):POKE 23444,start-256*histart:
POKE 23445,histart: POKE 23446,stepsize-256*histep:
POKE 23447,histep
```

Door de waarden van "start" en "stepsize" te wijzigen, kan u doen hernummeren vanaf elke willekeurige (toegelaten) startregel en met elke willekeurige (toegelaten) stapgrootte. Typ eerst de instructies in (hierboven) en kies dan de "Renumber"-optie in het edit-menu.

Later, wanneer u BASIC programma's kunt schrijven en u hebt geleerd om ze op cassette te zetten, kunt u misschien de bovenstaande instructies in een kort programmaatje opnemen, dat u dan vaker kunt gebruiken. Een voorbeeld :

```
10 INPUT "Start op regel", start
20 INPUT "Stapgrootte", stepsize
30 LET histart=INT (start/256)
40 LET histep=INT (stepsize/256)
50 POKE 23444,start-256*histart
60 POKE 23445,histart
70 POKE 23446,stepsize-256*histep
80 POKE 23447,histep
90 PRINT "Druk op ""EDIT"" en kies dan ""Renumber"""
```

Screen : Deze optie zet de cursor in het kleine scherm, waar u vervolgens BASIC kunt intypen en wijzigen. Dat is handig, wanneer u met grafisch werk bezig bent (hoofdstuk 8, paragraaf 17). Het grote scherm blijft namelijk intact wanneer u in het kleine scherm werkt. Om opnieuw in het grote scherm te werken, kiest u (op elk willekeurig moment tijdens het "editen") de optie "Screen" een tweede keer.

Print : Indien u een printer op de +2 hebt aangesloten, wordt het programma dat in het geheugen zit, op papier afgedrukt. Na de uitvoering van deze opdracht, verdwijnt het menu en kert de cursor terug. Indien de computer om een of andere reden de opdracht niet kan uitvoeren (omdat de printer niet is aangesloten of hij niet ingeschakeld is) dan komt u terug bij de editor door twee keer op de "BREAK"-toets te drukken.

Exit : Deze optie brengt u terug naar het startmenu. Een programma in het geheugen blijft bewaard. Om terug naar het programma te gaan, kiest u de optie "128 BASIC". Indien u in het startmenu de optie "48 BASIC" kiest of indien u de computer uitschakelt of reset, wordt de geheugeninhoud gewist. Indien u de optie "Calculator" kiest, blijft een programma dat in het geheugen stond, behouden.

Reset nu de +2, kies de optie "128 BASIC", en typ onderstaande programmaregel in. Bij elke druk op een toets, ziet u het overeenkomstige karakter op het scherm verschijnen. Een "karakter" is : een letter, cijfer, spatie, enz. Om het "=" teken in te typen, drukt u op de "SYMB SHIFT"-toets en de L tegelijk. Typ deze regel in :

```
10 for f=1 to 255 step 10
```

en druk "ENTER". Wanneer u alles correct hebt ingetypt, zet de +2 de regel op het scherm, maar nu met FOR, TO en STEP in hoofdletters :

```
10 FOR f=1 TO 255 STEP 10
```

De +2 produceert een korte toon, en zet de cursor bij het begin van de volgende regel.

Indien de regel blijft staan zoals hij was, en u een korte lage toon hoorde, dan hebt u een typfout genaakt. U zal merken dat de cursor rood gekleurd is. Corrigeer eerst de fout(en) : de +2 accepteert alleen correcte regels. Om te corrigeren brengt u de cursor naar die plaats op de regel waar u een fout opmerkt, en typt u daar de juiste karakters. U kunt ook de "DELETE"-toets gebruiken om ongewenste karakters weg te halen. Bent u klaar met de correctie, druk dan weer "ENTER".

Typ nu de volgende regel in, gevolgd door "ENTER": De dubbele punt typt u als de combinatie "SYMB SHIFT" en Z. Het minteken is SYMB SHIFT en J.

```
20 plot 0,0: draw f,175:plot 255,0: draw -f,175 (en ENTER)
```

Maak u geen zorgen over het feit dat die regel niet op 1 scherm-regel kan. De computer zorgt er voor dat de tekst op een goed leesbare manier op het scherm komt. Het verschil met een schrijfmachine is, dat u nu op het einde van een regel niets moet doen. De +2 zet de cursor zelf bij het begin van een volgende regel. De laatste regel van het programma wordt nu :

```
30 next f (en "ENTER")
```

De getallen bij het begin van elke regel, zijn regelnummers. Daardoor kan de +2 elke regel identificeren. U typte dus net regel 30 in. De cursor zou nu onder en links van de 30 moeten staan. Druk nu een keer op de toets met het pijltje naar boven gericht. De cursor beweegt zicht naar regel 30. Niet recht naar boven, zoals u misschien had verwacht, omdat er recht boven de cursor niets staat. De +2 probeert te raden waar u heen wilt, en zet de cursor op die plaats. Hij vermijdt spaties die niet tussen twee woorden staan, en zoekt altijd tekst op.

Druk nu nog eens op dezelfde toets, en dan net zolang op de "cursor rechts"-toets tot de cursor op de "1" staat in "DRAW -f, 175". Wat verwacht u nu, als u op de "cursor omlaag"-toets drukt (denk aan de "angst" voor spaties) ? Probeer het maar. Zoals u wellicht had verwacht, springt de cursor naar de dichtstbijzijnde tekst, in dit geval op het einde van regel 30. Druk nu terug op "cursor omhoog". Omdat er tekst boven de cursor stond, verwachtte u misschien dat hij recht naar boven zou gaan. Maar nee, hij gaat terug op zijn vorige positie staan. Dit is een staaltje van de slimheid van de +2. Hij "weet" dat u de cursor op regel 30 helemaal niet hebt bewogen, en onthoudt waar hij voor het laatst stond. Beweeg bijvoorbeeld de cursor naar links (op de "f"), dan naar rechts en dan omhoog. Nu "denkt" de +2 dat u op regel 30 iets hebt gewijzigd, en vergeet hij zijn vorige positie op regel 20. Daarom gaat de cursor nu wél recht omhoog.

Deze manier van bewegen heet "tracking" (opsporen). In het begin kan ze verwarrend zijn, maar ze maakt het wijzigen van een programma wel gemakkelijker, een keer u er vertrouwd mee bent. Druk nu "ENTER". De computer maakt een nieuwe regel klaar om tekst in te typen. Typ nu :

```
run, en druk "ENTER".
```

Er gebeurt van alles. De voetbalk en de programmaregels worden van het scherm gewist : de editor maakt zich klaar om de controle over te dragen aan het programma dat u net intypte. Het programma begint te lopen, tekent een mooi patroon op het scherm en stopt, met de mededeling : 0 OK, 30:1

Laat voorlopig de betekenis van deze mededeling terzijde, en druk "ENTER". Het scherm wordt gewist, en de voetbalk verschijnt weer, samen met de listing van het programma. Dit hele gebeuren neemt ongeveer een seconde in beslag. Tijdens die tijd kijkt de +2 niet naar het toetsenbord. Probeer dus niets in te typen, terwijl dit gebeurt.

U hebt nu ongeveer alle belangrijke bewerkingen uitgevoerd, die nodig zijn om een computer te programmeren. Eerst en vooral hebt u de +2 een lijst met instructies gegeven. Instructies dienen om de computer te vertellen wat hij moet doen (zoals de instructie 30 NEXT f). Instructies hebben een regelnummer, en worden in het geheugen opgeslagen. Ze worden niet onmiddellijk uitgevoerd nadat u ze intypte. Na de lijst instructies, gaf u het commando "RUN", om het opgeslagen programma uit te voeren.

Commando's lijken op instructies, alleen hebben ze geen regelnummer en worden ze onmiddellijk door de +2 uitgevoerd, zodra de "ENTER"-toets wordt ingedrukt. Over het algemeen kan elke instructie als commando worden gebruikt en vice versa. Alles hangt van de omstandigheden af. Elke instructie of elk commando moet minimum 1 "keyword" bevatten. Keywords zijn een deel van de woordenschat van de computer. Sommige keywords kunnen niet op zichzelf worden gebruikt, maar moeten vergezeld gaan van parameters. In het commando "DRAW 40,100" bijvoorbeeld, is DRAW het keyword, en zijn 40 en 100 de parameters : daardoor weet de computer precies waar hij de "DRAW"-operatie moet uitvoeren. Alles wat de computer in BASIC doet, volgt deze regels.

Druk nu op "EDIT" en kies de optie "Screen". De editor plaatst het programma in het kleine scherm, en haalt de voetbalk weg. U ziet enkel regel 10 van het programma. De rest zit verborgen. Door de cursor op en neer te bewegen kunt u dit controleren.

Druk op "ENTER" en typ : run, gevolgd door "ENTER".

Het programma loopt nu, precies zoals voordien. Als u nu op de "ENTER"-toets drukt, wordt het scherm niet gewist. U kunt nu met de cursor op en neer in het programma bewegen, zonder dat het bovenste deel van het scherm aangetast wordt. Indien u op "EDIT" drukt, om het editmenu te krijgen, verwacht u misschien dat dit wel het bovenste deel van het scherm zou aantasten. Maar de +2 onthoudt wat er op het scherm stond, en zet dit weer terug op zijn plaats, wanneer het editmenu wordt weggehaald.

Als bewijs dat de editor echt in het onderste deel van het scherm werkt, druk op "ENTER". We gaan regel 10 wijzigen in :

```
10 FOR f=1 TO 255 STEP 7
```

Dat doen we, door de cursor naar het einde van regel 10 te bewegen, net rechts van STEP 10. Daarna drukken we twee keer op "DELETE" en dan typen we 7, gevolgd door een druk op "ENTER".

Typ nu : go to 10, en druk "ENTER".

De keywords "go to" laten de computer weten dat hij niet het scherm moet wissen, vooraleer hij het programma start. Het gewijzigde programma tekent een licht verschillend patroon bovenop de oude tekening. U kunt het programma verder wijzigen om meer patronen te tekenen, indien u dat wenst.

Een waarschuwing : als u in het kleine scherm werkt, probeer dan niet om regels te wijzigen die meer dan twee schermregels lang

zijn. Indien de editor een regel vindt die begint of eindigt buiten het scherm, kan hij op hol slaan. Dit geldt ook voor het bovenste schermgedeelte, maar daar is het onwaarschijnlijk dat deze beperking problemen oplevert, omdat het werkgebied aanzienlijk groter is.

Terwijl u aan het typen bent, kunt u misschien gemerkt hebben dat de combinatie van "CAPS SHIFT" en 5,6,7 en 8 de cursor laten bewegen. CAPS SHIFT met 1 roept het editmenu op, en CAPS SHIFT met 0 wist een karakter van het scherm. CAPS SHIFT met 2 doet hetzelfde als CAPS LOCK, en CAPS SHIFT met 9 selecteert de grafische mode. Al deze functies hebben een eigen, speciale toets, en er is geen enkele reden waarom u de CAPS SHIFT-combinaties zou gebruiken.

Weet u nu voldoende over de werking van de editor, lees dan door in hoofdstuk 8. Zoniet, experimenteer naar hartelust met de gegeven voorbeelden, en probeer gerust iets anders.

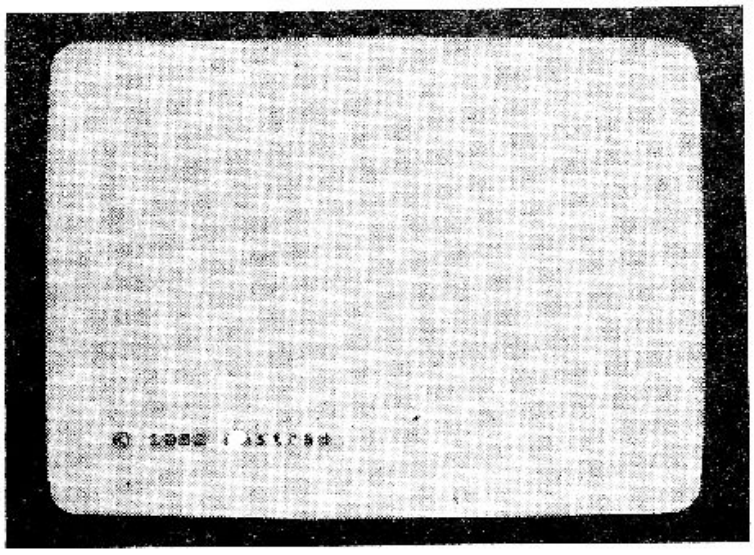
HOOFDSTUK 7 HET GEBRUIK VAN 48 K BASIC

Inhoud ...
- de +2 als Spectrum 48K gebruiken
- naar 48 BASIC mode gaan
- het toetsenbord in 48-mode
- programma's intypen
- wijzigingen in de courante regel aanbrengen

De +2 kan gebruikt worden alsof het een 48K Spectrum of Spectrum + was. Daartoe moet de optie "48 BASIC" in het startmenu gekozen worden. In deze mode kunnen de extra's van de +2 (groter geheugen, uitgebreide editor, geluid over meerdere kanalen, interface voor RS232, MIDI en numeriek toetsenbord) niet gebruikt worden. De aansluitingen voor de beide joysticks blijven in werking.

De 48 BASIC-mode is enkel voorzien om de uitwisselbaarheid van software te garanderen. Om programma's te schrijven biedt de 48-mode geen enkel voordeel tegenover de 128-mode. De 48-mode is ook niet aanbevolen. De nu volgende informatie wordt enkel als referentie gegeven, ofwel voor gebruikers die de 48K Spectrum gewend zijn en de nieuwe machine direkt willen gebruiken, zonder de 128 BASIC editor te leren kennen.

U kunt op twee manieren in 48-mode komen. De eerste bestaat erin, de optie "48 BASIC" in het startmenu te kiezen. Indien u niet weet hoe u dit moet doen, lees dan eerst hoofdstuk 2. Hebt u die keuze gemaakt, dan ziet u op het scherm dit :



De tweede manier laat u toe om, terwijl u in 128-mode bent, naar 48-mode te gaan, door gewoon het woord "spectrum" in te typen, gevolgd door "ENTER".

De +2 reageert met de boodschap "OK", en u bent in 48-mode. Een programma dat in het geheugen zat, blijft behouden. Eens u in 48-mode bent, kunt u enkel terug naar 128 BASIC door de computer te resetten of hem uit - en aan te schakelen.

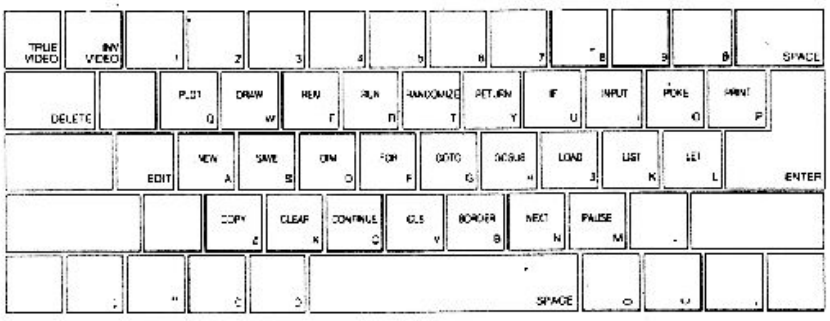
Het grootste verschil in 48-mode, is de manier waarop een programmaregel ingetypt wordt en gewijzigd. De voorbeeld-programma's in hoofdstuk 8 zullen over het algemeen in beide modes werken. De programma's die met muziek te maken hebben of die de "silicon disk" gebruiken (RAM-disk), werken enkel in

128-mode. Merk ook op dat de tokens "SPECTRUM" en "PLAY" in de plaats gekomen zijn van de grafische karakters "T" en "U" (karakter-codes 163 en 164) in 128 BASIC.

In 48-mode reageert het toetsenbord op de volgende manier :

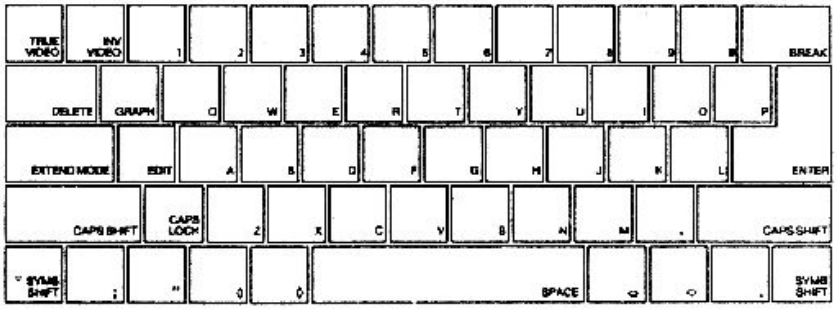
Alle BASIC commando's, functies en operators staan direct op het toetsenbord ter beschikking, en hoeven dus niet meer ingetypt te worden. Om al die funkties en commando's te kunnen onderbrengen, hebben sommige toetsen vijf of meet aparte betekenissen. Die betekenissen worden toegewezen door enerzijds de toetsen te "shiften", dat wil zeggen ofwel CAPS SHIFT ofwel SYMB SHIFT tegelijk met de gewenste toets in te drukken, en anderzijds door de machine in verschillende modes te brengen. De knipperende cursor bevat een letter (K, L, C, E of G) die aangeeft in welke mode de machine werkt.

K (eerste letter van Keywords) komt automatisch in de plaats van de L (Letters) wanneer de machine een commando of een regelnummer verwacht, in plaats van INPUT. Door de plaats op de regel waar de cursor zich bevindt, weet de +2 dat er een regelnummer of een keyword volgt. De K-mode wordt in werking gesteld, bij het begin van een regel, na een dubbele punt (behalve wanneer die in een string wordt geplaatst), of na het keyword THEN. Elke keer de K-cursor verschijnt, wordt de volgende toets die wordt ingedrukt, geïnterpreteerd als een keyword of een cijfer. Hieronder vindt u welke toets bij welk keyword hoort :



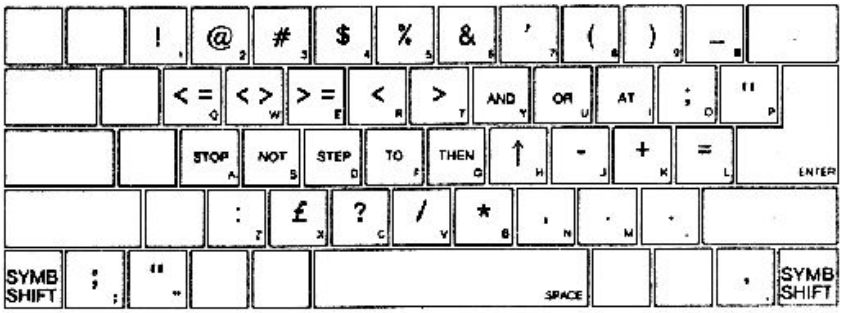
Het toetsenbord in K-mode

L (Letters)-mode is de gewone mode. Wanneer de L-cursor zichtbaar is, wordt de toets die wordt ingedrukt, geïnterpreteerd als de letter die op de toets staat, zoals in de volgende tekening duidelijk wordt gemaakt :



Het toetsenbord in L-mode

Zowel in K- als in L-mode, wordt het indrukken van een toets samen met SYMB SHIFT, als volgt geïnterpreteerd :

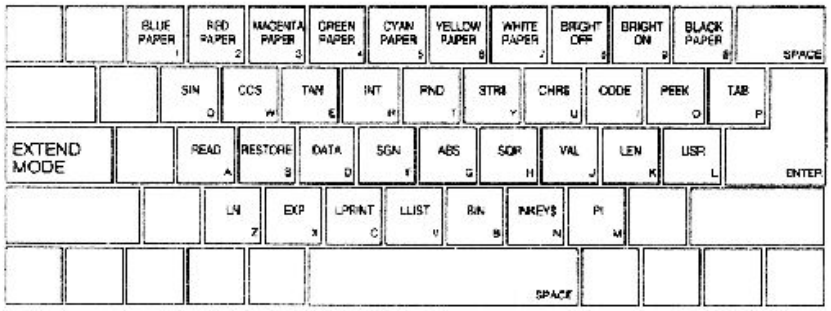


Het toetsenbord in K of L-mode, met SYMB SHIFT

Het gebruik van CAPS SHIFT in de L-mode zorgt er enkel voor dat de volgende toetsindrukken, hoofdletters opleveren. Keywords blijven altijd in hoofdletters geschreven staan.

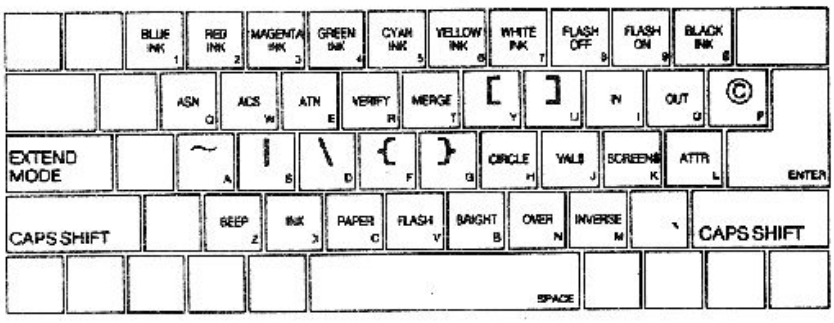
C-mode is een variante op de L-mode, waarbij alle letters als hoofdletters worden afgedrukt. CAPS LOCK schakelt heen en weer tussen hoofdletters en kleine letters.

E-mode (Extended, uitgebreide mode) wordt gebruikt om nog meer tekens te verkrijgen, veelal tokens. U komt in de E-mode door "EXTEND MODE" in te drukken. De E-mode blijft slechts gelden tot na de eerstvolgende toetsindruk. Indien de E-cursor zichtbaar is, worden de toetsen als volgt geïnterpreteerd :



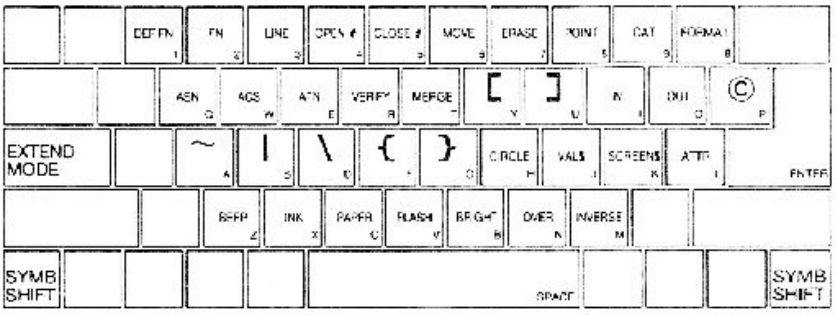
Het toetsenbord in E-mode

Indien u CAPS SHIFT indrukt in de E-mode, worden de toetsen als volgt geïnterpreteerd :



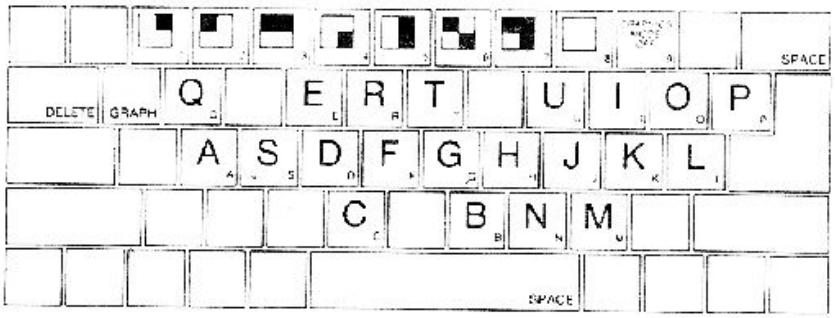
Het toetsenbord in E-mode, met CAPS SHIFT

Indien u SYMB SHIFT indrukt in de E-mode, worden de toetsen als volgt geïnterpreteerd :



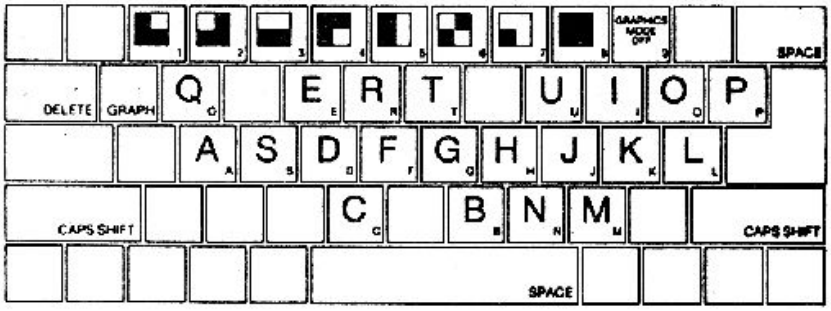
Het toetsenbord in E-mode, met SYMB SHIFT

G-mode (grafische mode) treedt in werking indien de GRAPH-toets wordt ingedrukt. Ze blijft werkzaam totdat dezelfde toets nogmaals wordt ingedrukt, of de "9"-toets wordt ingedrukt. Een cijfertoets geeft een vastliggend grafisch symbool. Een lettertoets, van A tot en met U, geeft een grafisch symbool, dat u zelf kunt bepalen. Zolang u dat niet hebt gedaan, ziet elk symbool er uit als een gewone hoofdletter. Indien de G-cursor zichtbaar is, worden de toetsen als volgt geïnterpreteerd :



Het toetsenbord in G-mode

Indien u in G-mode op CAPS SHIFT drukt, worden de vastliggende symbolen omgekeerd, dat wil zeggen dat de inkt-kleur de papierkleur wordt, en omgekeerd. De toetsen worden dan als volgt geïnterpreteerd :



Het toetsenbord in G-mode, met CAPS SHIFT

Indien een toets ingedrukt wordt gehouden gedurende meer dan twee of drie seconden, zal hij gaan repeteren. Input van het toetsenbord verschijnt onderaan het scherm, zodra een toets wordt ingedrukt. Elk karakter (enkel of keywords) wordt voor de plaats van de cursor ingevoegd. De cursor kan links en rechts bewogen worden door middel van de pijltjestoetsen links van de spatiebalk. Het karakter links van de cursor kan weggehaald worden door gebruik van de DELETE-toets.

Bij indrukken van "ENTER" wordt de regel uitgevoerd, bij het programma gevoegd of als INPUT gebruikt. Indien de regel een fout tegen BASIC bevat, wordt een knipperend vraagteken bij de fout gezet.

Een lijst van de ingevoerde programmaregels wordt getoond in het bovenste gedeelte van het scherm. De laatst ingevoerde regel wordt "courante" regel genoemd : hij wordt aangegeven voor het ">"-teken na het regelnummer. Om het even welke regel kan tot courante regel worden gemaakt (om hem te kunnen wijzigen bv.) door de cursor omhoog of omlaag te doen bewegen met de pijltjes-toetsen rechts van de spatiebalk. Om daarna de gekozen regel te wijzigen, drukt u op de "EDIT"-toets. De wijzigingen gebeuren in het onderste gedeelte van het scherm.

Wanneer een commando wordt uitgevoerd of een programma loopt, wordt het resultaat daarvan in het bovenste gedeelte van het scherm getoond. Het blijft daar staan, tot ofwel "ENTER" of een van de pijltjestoetsen wordt ingedrukt. Onderaan het scherm verschijnt een mededeling, voorafgegaan door een code (cijfer of letter). De betekenis daarvan vindt u terug in deel 28 van hoofdstuk 8. Deze mededeling blijft staan tot een toets wordt ingedrukt : dan gaat de computer naar K-mode.

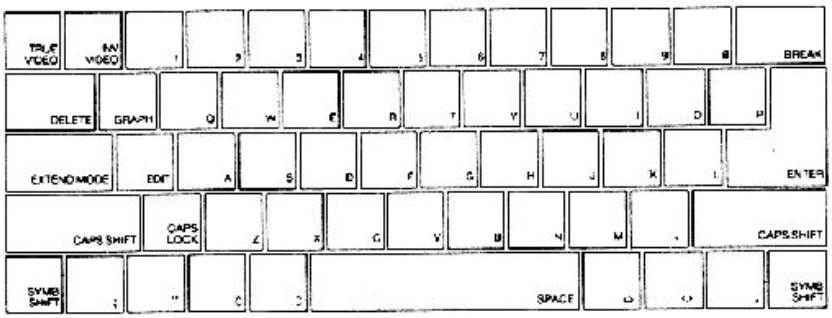
HOOFDSTUK 8 HANDLEIDING BASIC PROGRAMMEREN

DEEL 1 : INLEIDING

Ofwel hebt u hoofdstuk 6 al gelezen, ofwel begint u meteen hier. In beide gevallen moet u voor ogen houden dat :
- commando's meteen worden uitgevoerd;
- instructies met een regelnummer beginnen, en opgeslagen worden in het geheugen

Deze BASIC-programmeergids begint met een herhaling van punten die in hoofdstuk 6 al aan de orde waren (Gebruik van 128 BASIC), maar deze keer wordt er gedetailleerd op in gegaan. Er wordt precies uitgelegd wat u kunt doen. Op het einde van bepaalde stukken vindt u ook oefeningen. Het loont de moeite om die uit te voeren, omat ze veelal een illustratie geven van mogelijkheden die in de tekst werden vermeld. Lees de oefeningen door en probeer degene die u interesseren, zelf uit, zeker wanneer ze gaan over onderwerpen die u niet helemaal denkt te begrijpen. Wat u ook doet, blijf uw +2 gebruiken. Indien u zich afvraagt "Wat zou er gebeuren wanneer ik dit of dat doe ?", dan is het antwoord eenvoudig : probeer het uit, en kijk wat het resultaat is. Wat voor instructies of commando's u ook intypt, u kunt NOOIT schade toebrengen aan de +2.

Het toetsenbord.

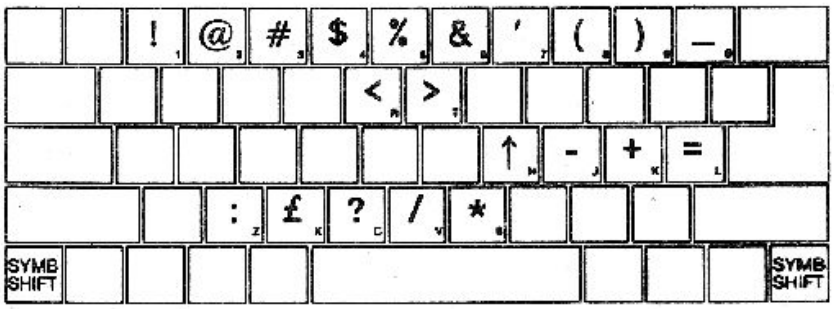


De codes die de +2 gebruikt, omvatten niet alleen enkelvoudige symbolen (letters, cijfers enz.) maar ook samengestelde symbolen, "tokens" genoemd (keywords, functies enz.). Alles moet letter per letter worden ingetypt. In de meeste gevallen doet het er niet toe of u hoofdletters dan wel kleine letters intypt. Het toetsenbord bevat drie soorten toetsen : letter- en cijfertoetsen (alfanumerieke toetsen), symbool-toetsen (leestekens) en controle-toetsen (bv. CAPS SHIFT, DELETE enz.).

In BASIC worden de alfanumerieke toetsen het meeste gebruikt. Wanneer u op een lettertoets drukt, ziet u de overeenkomstige kleine letter op het scherm verschijnen, gevolgd door een wit en blauw knipperend blokje, de cursor. Om een hoofdletter te verkrijgen, dient u eerst de CAPS SHIFT toets in te drukken, en dan de gewenste letter te typen.

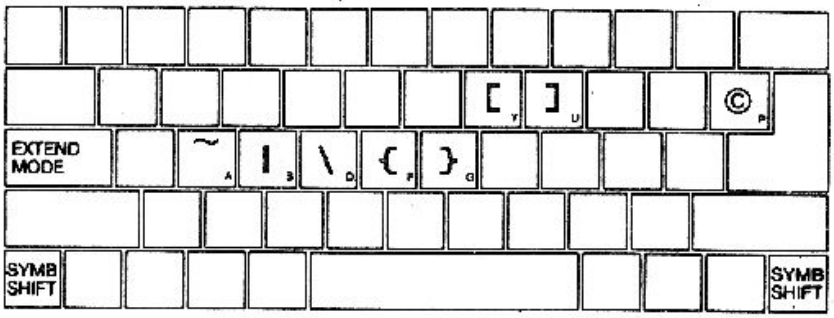
Wilt u continu hoofdletters typen, dan drukt u 1 keer op de CAPS LOCK toets. Om daarna terug kleine letters te krijgen, drukt u nogmaals op dezelfde toets.

Om de symbolen die op de alfanumerieke toetsen staan, te typen, d.w.z. : @ # $ % & ' ( ) _ < > ^ - + = ; £ ? / * houdt u SYMB SHIFT ingedrukt, en typ de letter waar het symbool op voorkomt



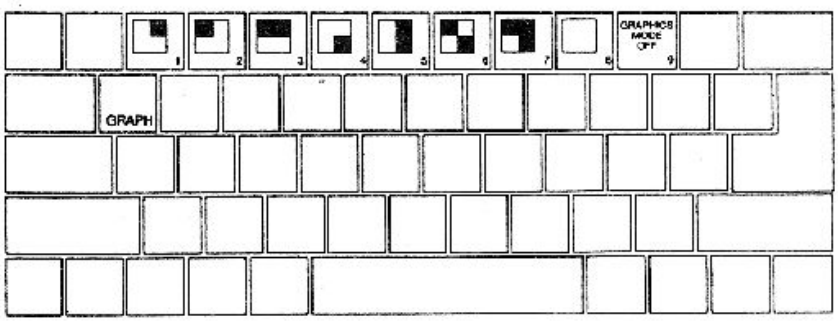
Symbolen met SYMB SHIFT

Verder kunt u de symbolen [ ] ~ | \ { } typen door eerst op de EXTEND MODE toets te drukken, dan de SYMB SHIFT ingedrukt te houden en vervolgens de overeenkomstige alfanumerieke toets in te drukken (zie tekening).



Symbolen in E-mode, met SYMB SHIFT

Om de computer in grafische mode te zetten, drukt u 1 keer op de GRAPH toets. U kunt dan de blokjes-grafieken verkrijgen door op een cijfertoets te drukken (behalve 9 en 0). Drukt u op CAPS SHIFT en dan op een cijfertoets, dan krijgt u het omgekeerde beeld van de blokgrafieken op de tekening. In grafische mode op de lettertoetsen drukken (van A tot en met S), geeft een grafisch teken op dat u zelf kunt definiëren.



Blokgrafieken in grafische mode

Wanneer een toets gedurende meer dan 2 tot 3 seconden wordt ingedrukt gehouden, gaat hij repeteren. Door het intypen van karakters, wordt een regel op het scherm opgebouwd. Ter verduidelijking : met "een regel" wordt een BASIC regel bedoeld, en die kan rustig een aantal regels op het scherm beslaan. Met de vier pijltjestoetsen kunt u de cursor doorheen de regel bewegen. Indien het gedeelte van de regel waar u de cursor naartoe stuurt, buiten het scherm valt, dan wordt de tekst op het scherm omhoog- of omlaagbewogen (gescrolld), om de bedoelde tekst te tonen. Een ingetypt karakter wordt in de tekst tussengevoegd op de plaats van de cursor. Een druk op de DELETE toets, wist het karakter links van de cursor. Zodra ENTER wordt gedrukt, of zodra de cursor tot buiten de regel wordt gestuurd, gaat de +2 controleren of de ingetypte regel in orde is wat de BASIC betreft. Is dat zo, dan produceert de computer een hoge toon en de regel wordt ofwel onmiddellijk uitgevoerd, ofwel in het geheugen opgeslagen als een deel van een programma. Indien de regel evenwel een fout bevat, dan produceert de +2 een lage toon en zet hij de cursor op de plaats waar hij denkt de fout ontdekt te hebben. De cursor wordt rood van kleur, om de fout aan te duiden. Het is niet mogelijk om de cursor te bewegen buiten een regel die een fout bevat. De +2 stuurt de cursor altijd terug.

Het scherm

Het scherm bestaat uit 24 regels van elk 32 karakters. Het is in twee stukken verdeeld. Het grootste stuk, bovenaan, is maximum 22 regels lang, en wordt gebruikt om de listing van het programma te tonen, ofwel het resultaat van een lopend programma. Dit schermgedeelte wordt meestal gebruikt om programmaregels te wijzigen. Zodra het hele bovenstuk van het scherm is volgeschreven, wordt de scherminhoud 1 regel omhoog-gescrolld. Indien dit evenwel zou meebrengen dat u een bepaalde regel niet zou kunnen zien, dan vraagt de +2 u of hij inderdaad wel mag scrollen, door onderaan het scherm de vraag : "scroll ?" te printen. Een druk op een toets (behalve N, BREAK of de spatiebalk) laat het scrollen verdergaan.

Een druk op de N, BREAK of de spatiebalk zal het scrollen onderbreken, met onderaan het scherm de mededeling :

D BREAK - CONT repeats

Het kleinste deel van het scherm, onderaan, wordt gebruikt om kortere programma's te bewerken, om gegevens in te typen voor INPUT, om commando's in te typen indien het grote scherm niet gebruikt kan worden, bv. bij grafische programma's, en voor het afdrukken van de mededelingen van de +2 zelf.

Programma's intypen

Indien het programma dat u intypt, te groot wordt om in één keer op het grote scherm te tonen, dan zal de +2 proberen om het meest interessante gebied ervan (meestal is dat de laatst ingetypte regel, met de regels daarvoor en daarna) te tonen. U kunt evenwel ook zelf bepalen welk deel van uw programma u op het scherm wilt zien, door het commando

LIST xxx

waarbij xxx een regelnummer is. Dan toont de +2 u het programma vanaf die bepaalde regel en verder.

Wanneer een commando of een programma wordt uitgevoerd, dan wordt het resultaat daarvan in het bovenste schermdeel getoond. Dat blijft daar staan, wanneer het programma stopt, tot een toets wordt ingedrukt. Indien u het programma wijzigt in het onderste schermdeel, dan blijft het bovenste deel ongewijzigd tot het wordt overschreven, van het scherm gescrolld, of een CLS-commando wordt uitgevoerd. In het onderste schermdeel verschijnt een mededeling (een "report") die een code draagt, dat is een letter of een cijfer. Die codes kunt u terugvinden in deel 28 van dit hoofdstuk. Dit report blijft staan tot een toets wordt ingedrukt.

Terwijl de +2 een BASIC programma uitvoert, wordt op geregelde tijdstippen de BREAK-toets gecontroleerd. Dit gebeurt op het einde van een statement, tijdens het gebruik van de cassette of de printer, of terwijl muziek wordt gemaakt.

Ontdekt de +2 dat de BREAK-toets wordt ingedrukt, dan stopt hij met de uitvoering van het programma en geeft het report

D... of L ...

en het programma kan weer aangepast worden.

DEEL 2 : EENVOUDIGE PROGRAMMEER-BEGRIPPEN

Inhoud ...
- programma's
- regelnummers
- programma's wijzigen met de cursortoetsen
- RUN, LIST
- GO TO, CONTINUE, INPUT, NEW, REM, PRINT
- een programma stopzetten

Typ de volgende twee programmaregels in. Het hele programma zal de som van twee getallen afdrukken.

```
20 print a      (druk ENTER)
10 let a=10    (druk ENTER)
```

Merk op dat het scherm er nu zo uitziet :

```
10 LET a=10
■
20 PRINT a
```

Voordien hebben we al gezien dat deze regels niet onmiddellijk worden uitgevoerd, maar in het geheugen opgeslagen, omdat er een regelnummer voor staat. U zal gemerkt hebben dat de regelnummers bepalen in welke volgorde de programmaregels zullen worden uitgevoerd. Zoals u op het scherm kunt merken, zet de +2 ingetypte regels zelf in de goede volgorde.

Merk ook op dat, hoewel we alles in kleine letters hadden getypt, de keywords automatisch in hoofdletters worden omgezet (PRINT en LET) van zodra de +2 een regel accepteert. Verder in dit boek zullen we alles in hoofdletters schrijven. Indien u dit wenst, kan u zelf in kleine letters blijven typen.

Tot nog toe hebt u nog maar 1 getal opgegeven. Typ nu :

```
15 LET b=15   (druk ENTER)
```

Nu dient u regel 20 te wijzigen in

```
20 PRINT a+b
```

U zou natuurlijk de bovenstaande regel voluit kunnen intypen, maar het is een stuk minder werk om de cursor tot net achter de a te bewegen, en dan te typen

```
+b     (druk nog niet op ENTER)
```

De regel zou er nu zo moeten uitzien :

```
20 PRINT a+b
```

Druk nu ENTER. De cursor gaat op de volgende regel staan. Het scherm ziet er nu zo uit :

```
10 LET a=10
15 LET b=15
20 PRINT a+b
■
```

Voer dit programma uit, door in te typen :

RUN (druk ENTER)

en u ziet de som van de getallen verschijnen.

Probeer nog een keer, en typ dan :

PRINT a,b (druk ENTER)

De variabelen zitten dus nog in het geheugen, hoewel het programma afgewerkt is.

Indien u een regel per ongeluk intypt, bijvoorbeeld

12 LET b=8

en u wilt die regel weghalen, dan typt u simpelweg

12 (druk ENTER)

en regel 12 is weg. De cursor gaat staan op de plaats waar de regel stond.

Typ nu
30 (druk ENTER)

De +2 gaat nu zoeken naar regel 30. Aangezien er geen regel 30 in het geheugen zit, "valt" hij buiten het programma. De cursor wordt gezet op de regel volgend op de laatste programmaregel. Indien u een regelnummer intypt, dat niet in het programma voorkomt, dan zet de +2 de cursor op de plaats waar die regel zou staan indien hij wel voorkwam. Dit kan handig zijn om snel door grote programma's te bewegen. Maar LET OP : het kan ook gevaarlijk zijn. Indien de regel wél bestond, voordat u het regelnummer intypte, dan wordt hij weggehaald.

Om een overzicht van het programma op het scherm te krijgen, typt u het commando

LIST (druk ENTER)

Vooral indien u met langere programma's werkt, kan het voorkomen dat u het programma wenst te bekijken vanaf een bepaalde regel. Daartoe typt u het LIST-commando, gevolgd door het nummer van de gewenste regel.

Typ bijvoorbeeld

LIST 15 (druk ENTER)

en u ziet wat we bedoelen.

Toen we het ingetypte programma aan het ontwikkelen waren, konden we gemakkelijk regel 15 tussen de andere twee regels voegen. Dit had niet gekund, indien we de regels genummerd hadden met 1 en 2, in plaats van 10 en 20. Maak er een goede gewoonte van om ruimte te laten tussen regelnummers. (Dat zijn hele getallen van 1 tot 9999).

Mocht op een bepaald ogenblik blijken dat u niet voldoende ruimte hebt gelaten tussen de regelnummers, dan kunt u het editmenu gebruiken om het programma te hernummeren. Druk daartoe op de EDIT-toets, en kies de optie "Renumber". Dan wordt de ruimte tussen alle opeenvolgende regelnummers op 10 gebracht. Probeer dit eens uit : u zult zien dat de regelnummers gewijzigd worden.

Nu gaan we het BASIC commando NEW gebruiken. Door dit commando worden alle bestaande variabelen, samen met het programma dat in het geheugen van de +2 zat, gewist. Na dit commando kunt u met een schone lei beginnen. Typ dus nu :

```
NEW
```

en druk ENTER. Vanaf nu zeggen we niet meer elke keer dat u ENTER moet indrukken. We gaan er van uit dat u het nu wel weet.

Kies in het startmenu dat nu op het scherm staat, de optie "128 BASIC".

Typ nu zorgvuldig het volgende programma in. Dit programma zet graden Fahrenheit om in graden Celsius.

```
10 REM temperatuur omrekenen
20 PRINT "gr. F","gr. C"
30 PRINT
40 INPUT "Geef graden Fahrenheit",f
50 PRINT f,(f-32)*5/9
60 GO TO 40
```

Ofschoon u de hele regel 10 in kleine letters kunt intypen, wordt enkel REM in hoofdletters omgezet, omdat dit het enige keyword is dat de +2 herkent. Ook kunt u, indien u dat wenst, het keyword GO TO, waar de +2 zelf een spatie tussen zet, als een enkel woord typen (GOTO).

Start het programma. U ziet de hoofdingen op het scherm. Daar zorgt regel 20 voor. Maar wat gebeurt er met regel 10 ? Het lijkt wel alsof de +2 die gewoon negeert. En dat is ook zo. Het keyword REM op regel 10 betekent REMark, opmerking, en dat dient enkel om uzelf, de programmeur, er aan te herinneren wat het programma precies doet. Een REM-regel bestaat uit het keyword REM, gevolgd door wat u maar wilt. De +2 houdt geen rekening met wat er na de REM komt, tot op het einde van die programmaregel.

De +2 is nu aangekomen bij het INPUT-commando op regel 40, en wacht nu tot u een waarde intypt voor de variabele f. Dat merkt u aan de knipperende cursor onderaan het scherm.

Typ een getal in. De +2 drukt dat getal af, en wacht op een tweede getal. Dit komt door de instructie GO TO 40 op regel 60. Dit betekent zoveel als : in plaats van het programma af te werken en ermee te stoppen, ga terug naar regel 40 en herbegin het programma vanaf daar. U kunt dus nog een temperatuur intypen, en nog een ...

Na een tijdje kunt u zich misschien gaan afvragen of de computer dit ooit beu wordt. Het antwoord is : nee, nooit. Bij de volgende vraag naar een getal, houdt u SYMB SHIFT ingedrukt, en drukt u op de A. U ziet het woord "STOP" verschijnen. Als u nu ENTER drukt, geeft de +2 het report

```
H STOP in INPUT,40:1
```

Daardoor weet u meteen waarom het programma stopte, en waar (in regel 40). Het cijfer 1 na het regelnummer, geeft aan dat het om de 1ste instructie op regel 40 gaat.

Wilt u nu doorgaan met het programma, typ dan

```
CONTINUE
```

en de +2 vraagt wederom een getal.

Bij het commando CONTINUE weet de +2 nog welk regelnummer hij vermeldde in zijn report (zolang het niet 0 OK was), en hij springt naar die regel, in dit geval regel 40, het INPUT-commando.

Stop het programma en vervang regel 60 door

```
60 GO TO 31
```

Dit levert geen merkbaar verschil op in de uitvoering van het programma. Dat zit zo : indien het regelnummer in een GO TO commando verwijst naar een niet-bestaande regel, dan loopt het programma verder vanaf de regel die volgt op de aangegeven regel. Datzelfde geldt ook voor het commando RUN. Eigenlijk wil RUN zeggen : RUN 0.

Typ nu verder getallen in tot het scherm vol geraakt. Wanneer het scherm vol is, zal de +2 de inhoud van het bovenscherm een regel omhoog bewegen, om plaats te maken. Daarbij gaat de bovenste regel verloren. Deze beweging heet "scrollen".

Bent u dit moe, stop dan het programma, zoals we vroeger hebben gezien, en ga naar de editor door ENTER te drukken.

Kijk naar het PRINT-statement op regel 50. De komma op die regel is erg belangrijk.

Een komma wordt gebruikt om de computer te laten printen vanaf de linker marge ofwel vanaf het midden van het scherm, afhankelijk van wat er volgt. De komma op regel 50 zorgt ervoor dat de temperatuur in graden Celsius vanaf het midden van de regel wordt geprint.

De kommapunt daarentegen wordt gebruikt om wat daarna wordt geprint, onmiddellijk te doen aansluiten op het voorgaande.

Nog een leesteken dat in PRINT-statements gebruikt kan worden is de apostrofe, het afkappingsteken. Dit zorgt ervoor dat wat daarna wordt geprint, in het begin van de volgende regel komt. Dat gebeurt automatisch op het einde van een PRINT-statement.

Indien u wilt dat wat straks nog afgedrukt moet worden, aansluit op dezelfde regel, dan moet u een komma of een kommapunt op het einde van het PRINT-statement zetten. Om te zien hoe dit werkt, vervangt u regel 50 achtereenvolgens door deze drie regels :

```
50 PRINT f,
50 PRINT f;
50 PRINT f
```

en RUN het programma om het verschil te zien.

De regel met de komma erin, drukt alles in twee kolommen af. De regel met de kommapunt schrijft alles aan elkaar. De regel zonder leesteken drukt elk getal op een nieuwe regel af. Dit laatste resultaat had u ook verkregen door PRINT f'.

Onthoud goed het verschil tussen een komma en een kommapunt in het PRINT-commando. Verwar geen van beide met de dubbele punt, die dient als scheidingsteken tussen twee commando's op een regel, zoals in deze :

```
PRINT f: GO TO 40
```

Typ nu de volgende regels in :

```
100 REM dit programma onthoudt uw naam
110 INPUT n$
120 PRINT "Dag ";n$;" !"
130 GO TO 110
```

Dit is een afzonderlijk werkend programma, maar het kan samen met het vorige in het geheugen van de +2 zitten. Om dit tweede programma te runnen, typt u

```
RUN 100
```

Dit programma verwacht dat u een "string" (een karakter of een groep karakters) intypt in plaats van een getal. Daarom zet hij, als geheugensteuntje, twee aanhalingstekens onderaan het scherm. Typ uw naam en druk ENTER.

Nu ziet u weer de aanhalingstekens. Die hoeft u niet per se te gebruiken indien u dat niet wilt. Probeer dit eens : druk twee keer op de "cursor rechts"-toets, en dan twee keer op de DELETE-toets, en typ

```
n$
```

Doordat er geen aanhalingstekens meer staan, weet de +2 dat hij iets moet gaan uitzoeken. In dit geval, moet hij de inhoud van de string-variabele n$ opzoeken. Die inhoud is wat u de vorige keer had ingetypt. Op die manier werkt het INPUT-statement als LET n$=n$. De waarde van n$ wordt niet gewijzigd.

Wilt u het programma stoppen, wis dan de aanhalingstekens uit. Druk SYMB SHIFT in en typ A, en druk ENTER.

Keren we nu eens terug naar de instructie RUN 100, die naar regel 100 springt en het programma van daar af uitvoert. U vraagt zich wellicht af, wat het verschil is tussen "RUN 100" en "GO TO 100". Het verschil is, dat RUN 100 eerst alle variabelen wist, het scherm leegmaakt en daarna hetzelfde doet als GO TO 100. Het commando GO TO 100 daarentegen wist niets. Het kan soms nodig zijn, een programma te runnen zonder variabelen te wissen. Dan moét u GO TO gebruiken : RUN kan dan rampzalig zijn. U kunt daarom beter niet de gewoonte aannemen om een programma automatisch op te starten met RUN.

Een tweede verschil tussen beide commando's is natuurlijk dat u RUN kunt intypen zonder een regelnummer, en dat het altijd naar de eerste programmaregel gaat. GO TO moet altijd gevolgd worden door een regelnummer.

Zowel dit laatste programma als het programma dat temperaturen omrekende, stopte door het indrukken van SYMB SHIFT en A in de INPUT-regel. Het kan voorkomen dat u per ongeluk een programma schrijft dat u niet op die manier kunt stoppen, en dat ook niet uit zichzelf stopt. Typ bijvoorbeeld :

```
200 GO TO 200
RUN 200
```

Ofschoon het scherm leeg blijft, is het programma wel degelijk aan het lopen. Regel 200 wordt telkens weer uitgevoerd. Het lijkt er op dat dit voor altijd zo doorgaat, tot u de RESET-knop indrukt of de stekker uittrekt. Er is evenwel een minder drastische oplossing : druk op de BREAK-toets. Het programma stopt met het report

```
L BREAK into program
```

Op het einde van elk statement controleert de computer of de BREAK-toets wordt ingedrukt. Blijkt dit zo te zijn, dan stopt hij het programma. De BREAK-toets kan ook worden gebruikt terwijl u de datacorder, de printer of een ander randapparaat van de +2 gebruikt.

In deze laatste gevallen is het report verschillend :

```
D BREAK - CONT repeats    (CONTINUE herhaalt)
```

In dit geval (en in de meeste andere gevallen) zal de instructie CONTINUE het statement waar de computer mee stopte, opnieuw uitvoeren, en dan gewoon doorgaan met de rest van het programma.

Run het tweede programma nog eens. Wanneer u om input wordt gevraagd, typt u :

```
n$ (haal eerst de aanhalingstekens weg)
```

Omdat de variabele n$ nog niet werd bepaald, geeft de computer het report :

```
2 Variable not found (variabele niet gevonden)
```

Als u nu typt :

```
LET n$="vissekop"
```

(wat het report 0 OK,0:1 oplevert), en daarna

```
CONTINUE
```

dan merkt u dat u nu wel zonder meer n$ als input kunt gebruiken.

In dit geval brengt het commando CONTINUE de computer naar het INPUT-commando op regel 110. Het report naar aanleiding van het LET-statement wordt genegeerd, omdat het "OK" was. De sprong gebeurt naar de regel die in het voorgaande report werd vermeld, namelijk regel 110. Dit kan handig te gebruiken zijn : het laat toe om een programma aan te passen, dat stopte omdat een fout werd ontdekt, en dan vanaf dat punt weer verder te gaan.

Zoals we al zegden is het report "L BREAK into program" speciaal, omdat na dit report, CONTINUE het statement waar het programma gestopt werd, niet herhaalt.

U hebt nu kennis gemaakt met de statements PRINT, LET, INPUT, RUN, LIST, GO TO, CONTINUE, NEW en REM. Die kunnen allemaal als rechtstreekse commando's ingetypt worden of in programmaregels opgenomen. Dit geldt voor zowat alle commando's in Spectrum BASIC. Hoewel RUN, LIST, CONTINUE en NEW in een programma meestal niet zoveel nuttigs kunnen doen ...

OEFENINGEN
++++++++++

1. Zet een LIST-statement in een programma zodanig dat, wanneer u het programma runt, het zichzelf list.

2. Schrijf een programma waar u prijzen moet opgeven, en dat de verschuldigde btw afdrukt (bv. 20%). Voorzie in PRINT-statements zodat de +2 zegt wat hij doet, en hij zeer beleefd naar de prijs informeert. Pas daarna het programma zo aan, dat u ook het btw-percentage kunt ingeven; zo kunt u rekening houden met toekomstige wijzigingen in btw.

3. Schrijf een programma dat het tussentotaal afdrukt van de getallen die u intypt. Een suggestie : benoem een variabele "totaal", die u 0 maakt om te beginnen. Gebruik een andere variabele "item", die dient om de waarde die u ingeeft, in op te slaan. Tel item op bij totaal, druk beide af op het scherm en ga terug om het volgende item op te halen.

4. Wat is het effect van CONTINUE en NEW in een programma ? Kunt u er een nuttig gebruik voor bedenken ?

DEEL 3 : BESLISSINGEN NEMEN

Inhoud ...
- CLS, IF, STOP
- =,<,>,<=,>=,<>

De programma's die we tot hiertoe hebben gezien, waren nogal voorspelbaar. De instructies werden één na één doorlopen, en dan herbegonnen. Dit is niet zo erg nuttig. In de praktijk willen we dat de +2 beslissingen kan nemen, en dienovereenkomstig handelen. De BASIC-instructie waarmee we dit kunnen bereiken, is IF (indien) iets juist is, of onjuist, THEN (dan) doe iets.

Een voorbeeld. Geef eerst het commando NEW om het vorige programma uit het geheugen te wissen. Kies de optie "128 BASIC". Typ nu het volgende programma in en RUN het. Het is duidelijk bedoeld om met z'n tweeën te spelen ...

```
10 REM raad een getal
20 INPUT "Geef een geheim getal op",a: CLS
30 INPUT "Raad het getal ",b
40 IF b=a THEN PRINT "Dat is juist": STOP
50 IF b<a THEN PRINT "Te laag, probeer opnieuw"
60 IF b>a THEN PRINT "Te hoog, probeer opnieuw"
70 GO TO 30
```

Het CLS-commando op regel 20 betekent : CLear Screen, wis het scherm schoon. We gebruiken het om te voorkomen dat de tweede speler het geheime getal kan lezen, nadat het werd ingetypt.

Zoals u kunt zien, heeft het IF-statement de vorm :

```
IF voorwaarde THEN xxx
```

Daarbij stelt xxx een commando of een reeks commando's voor (van elkaar gescheiden door een dubbele punt). De voorwaarde is iets, dat als juist of onjuist zal geëvalueerd worden. Blijkt ze juist te zijn, dan zullen de instructies op de rest van de regel, na het keyword THEN, uitgevoerd worden. Is de voorwaarde onjuist,

dan wordt de rest van de regel overgeslagen en gaat het programma verder op de volgende regel.

De eenvoudigste voorwaarden bestaan uit het vergelijken van twee getallen of twee strings. Daarmee kunt u controleren of ze gelijk dan wel verschillend zijn, en welke van twee strings alfabetisch voor de andere komt. Vergelijkingen maken gebruik van de symbolen = < > <= >= <>, bekend als vergelijkingstekens.

```
=   betekent : is gelijk aan
<   betekent : is kleiner dan
>   betekent : is groter dan
<=  betekent : is kleiner dan of gelijk aan
>=  betekent : is groter dan of gelijk aan
<>  betekent : is niet gelijk aan
```

Indien u de tekens > en < verwart, hier is een tip. De punt van het teken wijst altijd naar het deel van de vergelijking dat verondersteld wordt, kleiner te zijn dan het andere.

In het programma dat we net hebben ingetypt, worden a en b vergeleken op regel 40. Zijn beide gelijk, dan wordt het programma stopgezet door het STOP-commando. Het report onderaan het scherm,

```
9 STOP statement, 40:3
```

vertelt ons dat het programma gestopt werd door het derde statement op regel 40 van het programma.

Regel 50 bepaalt of b kleiner is dan a en regel 60 of b groter is. Indien aan één van beide voorwaarden wordt voldaan, dan print de +2 het bijbehorende commentaar op het scherm, en het programma gaat verder op regel 70. Daar wordt het terug naar regel 30 gestuurd.

In sommige BASICs (niet bij de Spectrum) kan een IF-statement ook zo worden geschreven :

```
IF voorwaarde THEN regelnummer
```

In Spectrum BASIC schrijven we dit zo :

```
IF voorwaarde THEN GO TO regelnummer.
```

OEFENING
++++++++
Probeer dit programma eens uit :

```
10 LET a=1
20 LET b=1
30 IF a>b THEN PRINT a;" is groter"
40 IF a<b THEN PRINT b;" is groter"
```

Probeer eerst eens te bedenken wat er op het scherm zal geprint worden, vooraleer u het programma runt.

DEEL 4 : LUSSEN LEGGEN

Inhoud ...
- FOR, NEXT
- TO, STEP

Stel dat u vijf getallen wenste in te geven, die opgeteld moeten worden.

Een manier om dit te doen, is deze (typ dit programma niet in, tenzij u echt zeer ijverig bent ...)

```
 10 LET totaal=0
 20 INPUT a
 30 LET totaal=totaal+a
 40 INPUT a
 50 LET totaal=totaal+a
 60 INPUT a
 70 LET totaal=totaal+a
 80 INPUT a
 90 LET totaal=totaal+a
100 INPUT a
110 LET totaal=totaal+a
120 PRINT totaal
```

Dit is niet bepaald een goede manier om te programmeren. Het kan nog te overzien zijn voor vijf getallen, maar stel u voor hoe het er uit zou zien voor tien getallen. Honderd getallen is al helemaal niet meer voor te stellen.

Het is beter om een variabele toe te wijzen, die tot 5 telt en daarna het programma stopt. Op deze manier (dit kunt u wel intypen ...)

```
10 LET totaal=0
20 LET tel=1
30 INPUT a
40 REM tel is aantal ingevoerde getallen
50 LET totaal=totaal+a
60 LET tel=tel+1
70 IF tel<=5 THEN GO TO 30
80 PRINT totaal
```

Merk op hoe gemakkelijk we nu regel 70 kunnen aanpassen, zodat we tien of zelfs honderd getallen kunnen optellen.

Deze methode is zo handig, dat er twee commando's bestaan die ze nog eenvoudiger te gebruiken maken : FOR en NEXT. Die gaan altijd samen. Met gebruik van die twee commando's ziet het programma dat u net intypte, er nu zo uit :

```
10 LET totaal=0
20 FOR t=1 TO 5
30 INPUT a
40 REM  t is aantal ingevoerde getallen
50 LET totaal=totaal+a
60 NEXT t
80 PRINT totaal
```

Om dit programma te verkrijgen, moet u enkel regels 20, 40 en 60 aanpassen, en regel 70 weghalen.

Merk op dat we de variabele "t" gebruiken in plaats van "aantal". Dat komt doordat de "controlevariabele" van een FOR-NEXT-lus met 1 letter benoemd moet worden.

Het effect van dit programma is dat t alle waarden van 1 tot 5 doorloopt, en voor elk van die waarden worden de regels 30, 40 en 50 uitgevoerd. Nadat t alle vijf de waarden heeft doorlopen, wordt regel 80 uitgevoerd.

Probeer nu eerst oefening 2, op het einde van dit deel. Die oefening verwijst naar het bovenstaande programma.

Een bijkomend punt van de FOR-NEXT structuur, is dat de controlevariabele niet per se met 1 moet verhogen, elke keer de lus wordt doorlopen. U kunt de controlevariabele met elke willekeurige waarde doen ophogen, door het STEP-gedeelte toe te voegen. De meest algemene vorm van een FOR-commando is :

```
FOR controlevariabele = startwaarde TO eindwaarde STEP stap
```

waar de controlevariabele een naam heeft die uit 1 letter bestaat, en waarbij de startwaarde, de eindwaarde en de stapwaarde alles kunnen zijn wat de +2 als een getal kan uitrekenen, niet alleen getallen, maar bijvoorbeeld sommen of de naam van numerieke variabelen. Indien u bijvoorbeeld regel 20 wijzigt in :

```
FOR t=1 TO 5 STEP 3/2
```

dan wordt de controlevariabele met 3/2 verhoogd, elke keer de lus wordt doorlopen. We hadden ook kunnen schrijven : STEP 1.5 of we hadden een variabele s de waarde 1.5 kunnen toekennen en schrijven : STEP s.

Na de bovenstaande wijziging zal t de waarden 1, 2.5 en 4 aannemen. Merk op dat de STEPwaarde niet beperkt is tot gehele getallen, en dat de controlevariabele niet precies de eindwaarde hoeft te bereiken : de lus wordt doorlopen zolang de controlevariabele een waarde heeft die kleiner is dan of gelijk aan de eindwaarde.

Probeer nu eerst oefening 3 aan het einde van dit deel. Die oefening verwijst naar het bovenstaande programma.

De STEP-waarde kan ook negatief zijn in plaats van positief. Probeer dit programma, dat de getallen van 1 tot 10 afdrukt in omgekeerde volgorde. Denk eraan, dat u steeds het commando NEW gebruikt vooraleer u een nieuw programma intypt.

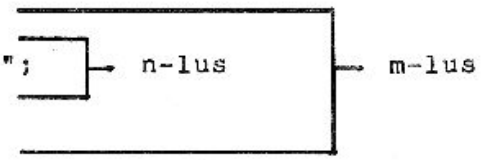
```
10 FOR n=10 TO 1 STEP -1
20 PRINT n
30 NEXT n
```

We zegden al dat het programma loopt zolang de controlevariabele kleiner is dan of gelijk aan de eindwaarde. Als u bedenkt wat dat in dit geval zou betekenen, dan weet u dat het in dit geval niet waar is. De regel wordt dus gewijzigd in die zin dat, indien de STEP-waarde negatief is, de lus wordt doorlopen zolang de controlevariabele groter is dan of gelijk is aan de eindwaarde.

Probeer nu eerst oefeningen 4 en 5 op het einde van dit deel. Die verwijzen naar het bovenstaande programma.

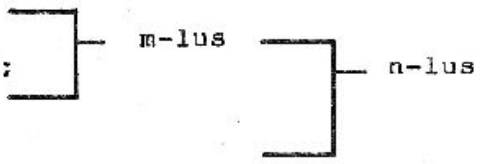
Voorzichtigheid is wel geboden, indien u twee FOR-NEXT lussen door elkaar wilt gebruiken. Probeer dit programma eens : het drukt de getallen af die op een spel dominostenen staan.

```
10 FOR m=0 TO 6
20 FOR n=0 TO m
30 PRINT m;":";n;" ";
40 NEXT n
50 PRINT
60 NEXT m
```



U merkt wel dat de n-lus helemaal binnen de m-lus ligt. Dit wil zeggen dat ze "genest" zijn zoals het hoort. Wat u absoluut moet vermijden, zijn twee FOR-NEXT lussen die mekaar overlappen, waarbij geen van beide volledig binnen de andere ligt, zoals dit programma :

```
 5 REM Dit programma is fout
10 FOR m=0 TO 6
20 FOR n=0 TO m
30 PRINT m;":";n;" ";
40 NEXT m
50 PRINT
60 NEXT n
```



Twee FOR-NEXT lussen moeten ofwel volledig afzonderlijk staan, ofwel moet een van beide binnen de andere gevat zijn.

Nog iets wat u dient te vermijden, is naar het midden van een FOR-NEXT lus te springen van buiten de lus. De controlevariabele wordt enkel goed toegekend door het FOR-statement, en indien dat niet voorkomt, raakt de +2 in de war. U zal in dat geval waarschijnlijk een van deze twee reports krijgen : NEXT without FOR (NEXT zonder FOR) of Variable not found (variabele niet gevonden).

Er is niets tegen het gebruik van een FOR-NEXT lus in een direct commando. Probeer dit eens :

```
FOR m=0 TO 10: PRINT m: NEXT m
```

Soms is dit bruikbaar als een (vergezochte) manier om de beperking te omzeilen, dat u niet met een GO TO binnen in een commando kunt springen, omdat een commando geen regelnummer heeft. Zo bijvoorbeeld :

```
FOR m=0 TO 1 STEP 0: INPUT a: PRINT a: NEXT m
```

De STEPwaarde 0 zorgt ervoor dat het commando steeds weer wordt herhaald.

Dit soort trucjes is evenwel niet aan te bevelen : indien een fout gebeurt, is het commando verdwenen en moet u alles opnieuw intypen. En CONTINUE werkt niet, in dit geval ...

OEFENINGEN
++++++++++

1. Overtuig u ervan dat u terdege begrijpt dat een controlevariabele niet enkel een naam en een waarde heeft, zoals elke variabele, maar ook een eindwaarde, een stapgrootte en een referentie naar het statement volgend op het FOR-statement. Wanneer het FOR-statement wordt uitgevoerd, staat al deze informatie ter beschikking (waarbij de eerste waarde die de variabele aanneemt, de startwaarde is). Die informatie is voor het NEXT-statement voldoende om te weten met hoeveel de variabele verhoogd of verlaagd moet worden, en of terug in het programma gesprongen moet worden en zo ja, waarheen.

2. Run het derde programma in dit deel, en typ, wanneer het

klaar is :

```
PRINT t
```

Waarom is het antwoord 6, en niet 5 ?

Antwoord : het NEXT-commando op regel 60 wordt vijf keer uitgevoerd. Elke keer wordt 1 bij t opgeteld. De laatste keer wordt t dus 6, waarop het NEXT-commando beslist om de lus niet nog een keer te doen, omdat t voorbij de grenswaarde ligt.

Wat gebeurt er wanneer u op het einde van regel 20 STEP 2 zet ?

3. Wijzig het programma, zodat het niet langer automatisch vijf getallen bij elkaar optelt, maar eerst vraagt hoeveel getallen u wenst op te tellen. Wat gebeurt er, wanneer u dit programma laat runnen en u geeft 0 op als aantal (d.w.z. u wenst geen getallen op te tellen) ? Waarom zou u kunnen verwachten dat de +2 daar moeite mee heeft, hoewel het toch duidelijk is wat u bedoelt ? (De +2 moet zoeken naar het commando NEXT t, wat gewoonlijk niet nodig is). Dat is allemaal voorzien.

4. Wijzig op regel 10 van het vierde programma in dit deel, het getal 10 in 100, en run het programma. De getallen van 100 tot 79 worden op het scherm gezet, en dan vraagt de +2 onderaan het scherm "scroll?". De bedoeling is, dat u daardoor de kans krijgt om de inhoud van het scherm te zien, vooraleer die van het scherm gescrolld wordt. Drukt u nu op N, op BREAK of op de spatiebalk, dan stopt het programma met het report "D BREAK - CONT repeats". Drukt u op een andere toets dan de genoemde, dan worden opnieuw 22 regels afgedrukt, en vraagt de +2 opnieuw of hij kan scrollen.

5. Haal regel 30 uit het vierde programma weg. Run nu het ingekorte programma. Het eerste getal wordt afgedrukt, en het programma stopt met de mededeling "0 OK". Als u nu typt :

```
NEXT n
```

dan zal het programma de lus nog een keer uitvoeren, en het volgende getal printen.

DEEL 5 : SUBROUTINES

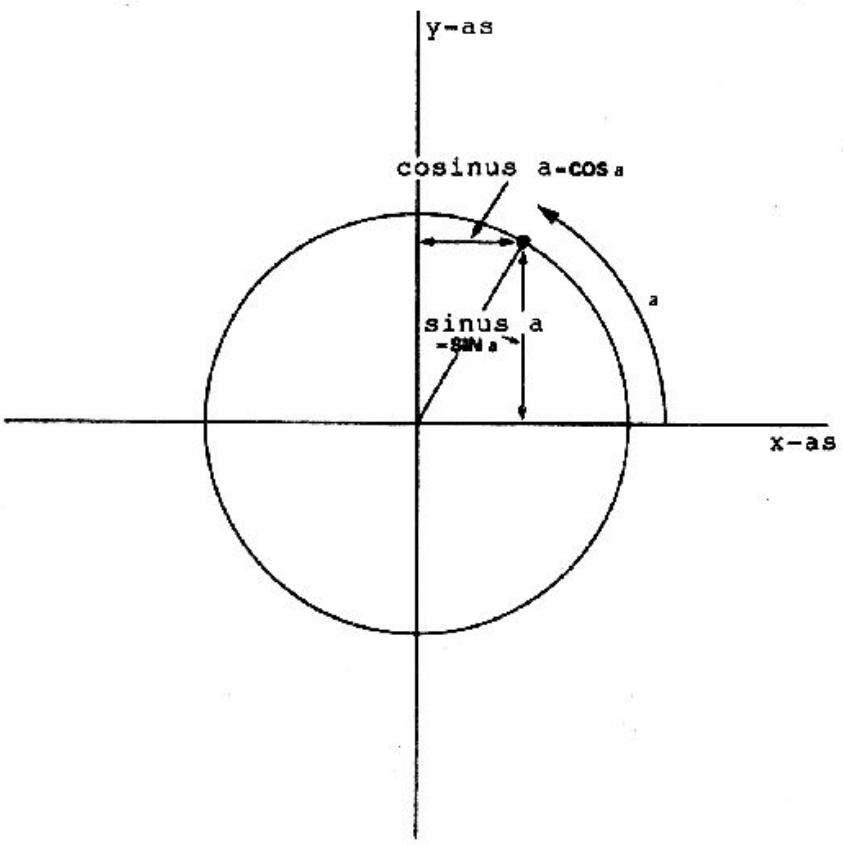
Inhoud ...
- GO SUB, RETURN

Soms komt het voor dat verschillende delen van een programma, gelijkaardige taken moeten uitvoeren. Dan kunt u natuurlijk twee of meer keren dezelfde programmaregels intypen, maar dat is niet nodig. U typt die regels 1 keer in (dat blok wordt dan een "subroutine" genoemd), en u laat de subroutine op elk gewenst tijdstip in het programma uitvoeren. Dat gebeurt, door middel van de statements GO SUB (GO to SUBroutine) en RETURN (keer terug), in de vorm :

```
GO SUB xxx
```

waarbij xxx het regelnummer is van de eerste regel van de subroutine. Dit commando werkt precies zoals GO TO xxx, maar bij GO SUB xxx onthoudt de +2 ook nog waar het GO SUB-statement staat, zodat hij na de subroutine daarheen terug kan keren.

Ter informatie : de +2 doet dit, door het adres te onthouden van de plaats waar hij het GO SUB-statement zag, en van waar af hij dus straks verder moet, en dit RETURN-adres te bewaren in een



Het kan voorkomen dat we het omgekeerde van deze functies moeten berekenen, bijvoorbeeld de waarde van a berekenen voor een gegeven sinus, cosinus of tangens. De functies op de +2 die dit doen, heten ASN (arcsinus), ACS (arccosinus) en ATN (arctangens).

Kijk nog eens naar de tekening : er staat een straal getekend vanaf het middelpunt naar het punt dat op de omtrek beweegt. U kunt zien dat de afstand die we a noemden (de afstand die het punt op de omtrek heeft afgelegd) gebruikt kan worden om de hoek te meten die de straal met de x-as maakt. Wanneer a=pi/2, dan is de hoek 90 graden. Wanneer a=pi dan is de hoek 180 graden, en wanneer a= 2*pi, is de hoek 360 graden. U kunt rustig het rekenen in graden vergeten, en de hoek meten in termen van a. We zeggen dan dat we de hoek in radialen meten. Dat houdt in dat pi/2 radialen = 90 graden, en zo verder.

Denk er aan dat de functies COS, SIN enz. op de +2 werken in radialen, en niet in graden. Om graden in radialen om te zetten, deelt u door 180 en vermenigvuldigt u met pi. Om radialen in graden om te zetten, deelt u door pi en vermenigvuldigt u met 180.

DEEL 6 : GEGEVENS IN EEN PROGRAMMA

Inhoud...
- READ, DATA, RESTORE

In een paar programma's zagen we dat informatie, ook "data" genoemd, rechtstreeks aan de +2 verschaft kan worden, door middel van het INPUT statement. Soms kan dit erg lastig zijn, vooral indien een grote hoeveelheid data elke keer dat het programma wordt gerund, terug wordt gebruikt. U kunt een hoop tijd besparen, door de commando's READ, DATA en RESTORE te gebruiken. Neem bijvoorbeeld dit programma :

```
10 READ a,b,c
20 PRINT a;b;c
30 DATA 1,2,3
```

Een READ-statement bestaan uit het keyword READ, gevolgd door een lijst van namen van variabelen, gescheiden door een komma. Dit werkt net zoals een INPUT-statement, maar in plaats van alle waarden voor de variabelen in te moeten typen, zoekt de +2 ze nu zelf op in de DATA-regel(s).

Elk DATA-statement bestaat uit een lijst van uitdrukkingen (numerieke of string-uitdrukkingen), gescheiden door een komma. U kunt die statements op een willekeurige plaats in het programma zetten, omdat de +2 ze negeert, indien hij geen READ-statement uitvoert. Stelt u zich voor dat alle uitdrukkingen in alle DATA-statements in het programma, samen een grote lijst vormen, de DATA-lijst. De eerste keer dat de +2 een READ-statement uitvoert, leest hij de eerste uitdrukking die in de lijst voorkomt. De volgende keer, leest hij de tweede; op die manier doorloopt hij bij herhaalde READ-statements de hele DATA-lijst. Moet hij na het einde van de lijst nog doorlezen, dan geeft hij een foutmelding.

Het is nutteloos om DATA in een rechtstreeks commando te typen : READ zal ze niet kunnen lezen. DATA moeten in een programma worden opgenomen.

Laten we nu eens kijken hoe dit allemaal in zijn werk gaat, in het programma dat u net hebt ingetypt. Regel 10 doet de +2 drie stukjes DATA lezen, en de gelezen waarden aan de variabelen a, b en c toekennen. Op regel 20 krijgt hij de instructie om die variabelen op het scherm te printen. Het DATA-statement op regel 30 verschaft de waarden voor a, b en c, die door regel 10 gelezen kunnen worden.

De informatie op een DATAregel kan ook midden in een FOR-NEXT lus staan. Typ dit eens in :

```
10 FOR n=1 TO 6
20 DATA 2,4,6,8,10,12
30 READ d
40 PRINT d
50 NEXT n
```

Uit de twee bovenstaande programma's blijkt dat een DATA-statement echt overal in een programma kan staan, zowel voor als na het READ-statement.

Bij het runnen van het bovenstaande programma leest het READ-statement elke keer de FOR-NEXT lus doorlopen wordt, één element uit de DATA-lijst.

De reeks getallen die u op deze manier verkrijgt, is niet echt willekeurig. Dat komt doordat RANDOMIZE gebruik maakt van de tijd die verlopen is sinds u de +2 inschakelde. Er verloopt even veel tijd tussen twee opeenvolgende uitvoeringen van RANDOMIZE en daardoor zal het RND-getal ook elke keer met min of meer hetzelfde bedrag worden opgehoogd. De reeks wordt willekeuriger indien u GO TO 10 vervangt door GO TO 20.

Hier volgt een programma dat muntstukken opgooit, en het aantal keren optelt dat kruis of munt wordt gegooid.

```
10 LET kruis=0: LET munt=0
20 LET worp=INT (RND*2)
30 IF worp=0 THEN LET kruis=kruis+1
40 IF worp=1 THEN LET munt=munt+1
50 PRINT kruis;",";munt
60 IF munt<>0 THEN PRINT kruis/munt
70 PRINT: GO TO 20
```

De verhouding kruis/munt zou 1 moeten worden indien u maar lang genoeg doorgaat, omdat te verwachten is dat na lange tijd het aantal keren kruis hetzelfde is als het aantal keren munt.

OEFENING
++++++++
1. Kies een getal tussen 1 en 872, en typ :
   RANDOMIZE (uw getal)
Controleer dat de eerstvolgende waarde van RND gelijk is aan
   (75*(uw getal+1)-1)/65536

DEEL 12 : ARRAYS

Inhoud ...
- arrays (de +2 behandelt string-arrays op een manier die licht afwijkt van de standaard)
- DIM

Veronderstel dat u een lijst getallen hebt, bijvoorbeeld de punten van tien leerlingen in een klas. Om die lijst te verwerken met de +2 kunt u de variabelen m1, m2, m3 enz. tot m10 gebruiken. Het programma om die variabelen toe te wijzen zou wel saai worden om in te typen :

```
 10 LET m1=75
 20 LET m2=44
 30 LET m3=90
 40 LET m4=38
 50 LET m5=55
 60 LET m6=64
 70 LET m7=70
 80 LET m8=12
 90 LET m9=75
100 LET m10=60
```

Voor dat soort toepassingen bestaat er een mechanisme, bekend als een "array", dat er in bestaat, een variabele te specificeren die (in plaats van één enkele waarde, zoals een gewone variabele) een aantal afzonderlijke "cellen" bevat, die elk verschillende waarden kunnen bevatten. Elk van die elementen wordt aangeduid met een index (het subscript), die tussen haakjes wordt geschreven na de naam van de variabele. In het bovenstaande voorbeeld zou de naam van het array m kunnen zijn

aantal bewerkingen na elkaar te laten doen. In het gegeven voorbeeld betekent de uitdrukking "som*25/100" : zoek de waarde op van de variabele "som"; vermenigvuldig die met 25, en deel dit resultaat door 100.

Een volledige lijst van de "voorrangsregels" bij rekenkundige (en logische) bewerkingen kunt u vinden in deel 30 van dit hoofdstuk.

In een uitdrukking die *, /, +, - bevat, worden de vermenigvuldiging en deling het eerst uitgevoerd : ze hebben een hogere voorrang dan optelling en aftrekking. Vermenigvuldiging en deling hebben dezelfde voorrang. Dit betekent dat ze worden uitgevoerd in de volgorde zoals ze in de uitdruking voorkomen (van links naar rechts). De volgende bewerkingen zijn optelling en aftrekking. Ook deze twee hebben gelijke voorrang, en worden dus ook uitgevoerd zoals ze in de uitdrukking voorkomen, van links naar rechts.

In de uitdrukking : 8-12/4+2*2, zal eerst de deling 12/4 worden uitgevoerd. Het resultaat is 3. We kunnen de uitdrukking dus schrijven als : 8-3+2*2.

De volgende bewerking die wordt uitgevoerd, is de vermenigvuldiging 2*2. Dit resultaat is 4. We kunnen de uitdrukking dus herschrijven als 8-3+4.

De volgende bewerking is de aftrekking 8-3. Het resultaat is 5. De uitdrukking wordt dus : 5+4. Tenslotte wordt de optelling uitgevoerd, met als resultaat 9.

Probeer dit eens, door in te typen :

```
PRINT 8-12/4+2*2
```

U kunt evenwel de voorrang van een bewerking in een uitdrukking wijzigen, door het gebruik van haakjes. Bewerkingen tussen haakjes worden het eerst uitgevoerd. Indien u dus wilde dat in de bovenstaande uitdrukking, de optelling 4+2 eerst werd uitgevoerd, dan had u die tussen haakjes moeten schrijven. Probeer het uit door te typen :

```
PRINT 8-12/(4+2)*2
```

wat als resultaat 4 oplevert, in plaats van 9.

Uitdrukkingen zijn nuttig, omdat u de +2 een uitdrukking kunt opgeven, wanneer hij een getal verwacht, en hij zelf de waarde ervan zal uitrekenen.

U kunt ook strings of stringvariabelen samentellen in een enkelvoudige uitdrukking, bijvoorbeeld :

```
10 LET a$="friet"
20 LET b$="mayonnaise"
30 PRINT a$;" met ";b$
```

Nu horen we u eigenlijk te vertellen wat u wel en niet kunt gebruiken als namen voor variabelen. We hebben al gezegd dat de naam van een stringvariabele uit één letter moet bestaan, gevolgd door een $-teken. U weet ook al dat de naam van de controlevariabele van een FOR-NEXT lus eveneens uit 1 letter kan bestaan. De namen van gewone variabelen bieden een veel ruimere keuze. Die kunnen een onbeperkt aantal letters hebben, of letters en cijfers, zolang het eerste karakter maar een letter is. De naam kan ook spaties bevatten, om hem makkelijker

array-variabele, kunnen er niet tegelijkertijd twee arrays dezelfde naam hebben, ook al hebben ze andere dimensies.

Er bestaan ook string-arrays. De strings in een array verschillen van gewone strings, doordat ze een vaste lengte hebben en ze op de al eerder genoemde Procrusteaanse manier worden ingevuld (ingekort of opgevuld met spaties).

De naam van een string-array is 1 letter gevolgd door het $-teken. In tegenstelling tot numerieke arrays, kan een string-array niet dezelfde naam hebben als een gewone string.

Veronderstel dat u een array a$ wilt, die vijf strings kan bevatten. U moet op voorhand beslissen hoe lang die strings zullen zijn. Stel, dat u 10 karakters lang genoeg vindt. Het array maakt u klaar door :

```
DIM a$(5,10) (typ dit in)
```

Dit commando maakt een array klaar van 5*10 karakters, maar u kunt zich dat array ook voorstellen alsof elke rij een string was, op deze manier :

```
a$(1) = a$(1,1)a$(1,2)a$(1,3) ... a$(1,10)
a$(2) = a$(2,1)a$(2,2)a$(2,3) ... a$(2,10)
a$(3) = a$(3,1)a$(3,2)a$(3,3) ... a$(3,10), enz.
```

Wanneer u evenveel indexen opgeeft als er dimensies voorkwamen in het DIM-statement, twee in dit geval, dan is het resultaat één karakter. Indien u de laatste dimensie weglaat, krijgt u een string met een vaste lengte. Bijvoorbeeld : a$(2,7) is het zevende karakter uit de string a$(2). In slicing-notatie kunnen we dit ook schrijven als a$(2)(7). Typ nu in :

```
LET a$(2)="1234567890"
```

en dan :

```
PRINT a$(2),a$(2,7)
```

Dit levert het volgende op :

```
1234567890   7
```

In plaats van de laatste index (die u kunt weglaten), kunt u ook een slicer schrijven. Zo is bijvoorbeeld :

```
a$(2,4 TO 8) gelijk aan a$(2)(4 TO 8) = "45678"
```

Onthoud dat in een string-array, alle strings even lang zijn.

Het DIM-statement bevat een extra getal, het laatste, die deze vaste lengte aangeeft . Wanneer u een geïndexeerde variabele in een string-array aanduidt, kunt u een getal extra opgeven (of een slicer), dat dan overeenkomt met het laatste getal in het DIM-statement.

U kunt ook een string-array zonder extra dimensies aanmaken, bijvoorbeeld :

```
DIM a$(10)
```

dat ervoor zorgt dat a$ zich als een gewone string-variabele gedraagt, maar dat zijn lengte altijd 10 blijft, en dat de invulling ervan altijd Procrusteaans gebeurt.

De string "" zonder enig karakter erin, heet de lege string of de nulstring. Denk er aan dat spaties een betekenis hebben en dat een lege string niet hetzelfde is als een string met alleen maar spaties.

Probeer eens :

```
PRINT "Hebt u vandaag "De Krant" al gelezen ?"
```

Bij indrukken van ENTER komt er een rood knipperende cursor staan, om aan te geven dat er een fout in de regel zit. Wanneer de +2 de aanhalingstekens voor "De Krant" ziet, neemt hij aan dat die het einde van de string "Hebt u vandaag al" aangeven, en dan weet hij zich geen raad met : De Krant" en verder.

Er bestaat een truuk om dit te omzeilen. Indien u in een string een aanhalingsteken wilt schrijven, moet u dit twee keer doen :

```
PRINT "Hebt u vandaag ""De Krant"" al gelezen ?"
```

Op het scherm ziet u maar één aanhalingsteken voor De, maar u moet er wel twee typen, opdat de +2 het als dusdanig zou herkennen.

DEEL B : STRINGS

Inhoud ...
- strings in stukken delen (slicing) met gebruik van TO

Een substring van een gegeven string, bestaat uit een aantal opeenvolgende karakters van de oorspronkelijke string. Zo is bijvoorbeeld "string" wél een substring van "andere string", maar "derest" of "nadere" zijn dat niet.

Om substrings te omschrijven, bestaan de begrippen "slice" en "slicing", die respectievelijk betekenen : "een stuk van een string", en "een string in stukken verdelen". Slicing kan op elke string-uitdrukking worden toegepast. De algemene vorm van slicing is :

```
string-uitdrukking (begin TO einde)
```

Dat wil bijvoorbeeld zeggen dat

```
"abcdef"(2 TO 5)
```

gelijk is aan "bcde"

Wordt het begin niet vermeld, dan wordt 1 genomen. Wordt het einde niet vermeld, dan wordt de lengte van de string als einde genomen. Enkele voorbeelden :

```
"abcdef"( TO 5) is "abcde"
"abcdef"(2 TO ) is "bcdef"
"abcdef"( TO ) is "abcdef"
```

Die laatste kunt u ook schrijven als : "abcdef"().

Er is ook een licht verschillende schrijfwijze, zonder het keyword TO, met één getal :

```
"abcdef"(3) is gelijk aan "abcdef"(3 TO 3) is gelijk aan "c"
```

NOT a<>b is hetzelfde als a=b

Ga voor uzelf na dat >= en <= de negaties zijn van < en >, in die volgorde. U kunt dus altijd een NOT weghalen uit een uitdrukking, door de vergelijking aan te passen.

De uitdrukking

NOT (uitdrukking 1 AND uitdrukking 2)

betekent hetzelfde als

NOT (uitdrukking 1) OR NOT (uitdrukking 2)

en de uitdruking

NOT (uitdrukking 1 OR uitdrukking 2)

betekent hetzelfde als

NOT (uitdrukking 1) AND NOT (uitdrukking 2)

Op die manier kunt u alle NOTs buiten haakjes zetten, tot ze allemaal op vergelijkingen slaan, en dan kunt u ze weghalen. Logisch bekeken, is NOT onnodig, hoewel een programma er soms wel doorzichtiger door wordt.

Wat nu volgt is nogal ingewikkeld. Mensen met een zwak hart kunnen dit stuk het beste maar overslaan...

Typ dit in :

PRINT 1=2, 1<>2

Misschien had u een foutmelding verwacht ? De computer kent geen logische waarden. Hij gebruikt gewoon getallen, volgens een aantal regels :

1) =, <, >, <=, >= en <> geven een numeriek resultaat : 1 voor juist, 0 voor onjuist. Daarom gaf het eerste PRINT-commando hierboven (1=2) het resultaat 0, omdat de uitdrukking 1=2 onjuist is. De uitdrukking 1<>2 gaf het resultaat 1, omdat ze juist is.

2) In het statement "IF voorwaarde THEN ... " kan de voorwaarde ook een numerieke uitdrukking zijn. Is de waarde van die uitdrukking 0, dan wordt ze als onjuist gerekend. Elke andere waarde wordt als "juist" gerekend, met inbegrip van de waarde 1, als resultaat van een juiste vergelijking. Het IF-statement betekent dus precies hetzelfde als : IF voorwaarde <>0 THEN ...

3) AND, OR en NOT krijgen ook een numerieke waarde :

x AND y wordt x, indien y juist is (van 0 verschilt)
0 (onjuist) indien y onjuist is (nul)

x OR y wordt 1 (juist) indien y juist is (van nul verschilt)
x indien y onjuist is (nul)

NOT x wordt 0 (onjuist) indien x juist is (van nul verschilt)
1 (juist) indien x onjuist is (nul)

Let op : "juist" betekent "niet nul" indien we een bepaalde, gegeven waarde bekijken, maar het betekent "1" indien er een nieuwe waarde wordt berekend, als resultaat van een uitdrukking.

Ingewikkelde string-uitdrukkingen moeten ook tussen haakjes worden gezet, voordat ze gesliced kunnen worden. Bijvoorbeeld :

```
"abc"+"def"(1 TO 2) is "abcde"
("abc"+"def")(1 TO 2) is "ab"
```

OEFENING
++++++++

1. Probeer een programma te schrijven dat de dag van de week afdrukt, door middel van slicing. Een tip : Stel de string gelijk aan "ZonMaaDinWoeDonVryZat".

DEEL 9 : FUNCTIES

Inhoud...
- DEF
- LEN, STR$, VAL, SGN, ABS, INT, SQR
- FN

Denk even aan een machine die worsten draait. Aan de ene kant wordt er een stuk vlees in gestopt, en aan de andere kant komt de worst er uit. Varkensvlees geeft varkensworst, kalfsvlees geeft kalfsworst, en rundsvlees geeft rundsworst.

Hoewel functies nauwelijks verschillen van worstmachines, is er toch een verschil. Functies werken met getallen en strings in plaats van vlees. U levert een waarde (het argument) en de functie bewerkt die tot een andere waarde, het resultaat.

vlees in → worstmachine → worst uit
argument in → functie → resultaat uit

Verschillende argumenten leveren verschillende resultaten op, en indien het argument niet bruikbaar is, stopt de functie met een foutmelding.

Net zoals verschillende machines, verschillende producten maken (worsten, theedoeken of fish-sticks) voeren verschillende functies ook verschillende bewerkingen uit. Elke functie levert een verschillende waarde op, die ze van de andere functies onderscheidt.

Een functie gebruikt u in uitdrukkingen door de naam van de functie te typen, gevolgd door haar argument. Nadat de uitdrukking werd uitgewerkt, wordt de rest van de functie uitgevoerd.

Er is bijvoorbeeld een functie die LEN heet. Ze berekent de lengte van een string. Het argument is de string waarvan u de lengte wenst te kennen; het resultaat is de lengte van de string. Als u dus typt :

```
PRINT LEN "Spectrum +2"
```

dan print de +2 het antwoord : 11, dat is het aantal karakters (met inbegrip van spaties) in de string "Spectrum +2".

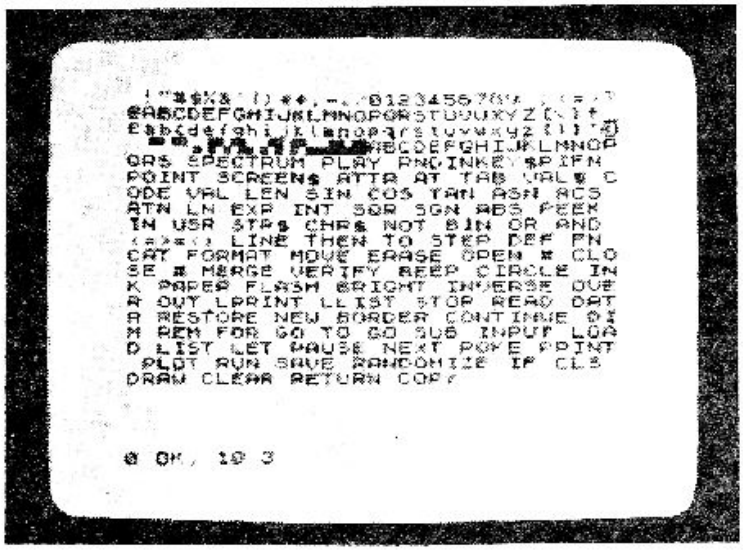
Indien u in een uitdrukking functies en bewerkingen door elkaar gebruikt, worden eerst de functies uitgewerkt, en dan pas de bewerkingen uitgevoerd. U kunt evenwel alweer die regel omzeilen

CHR$ kan op een getal worden toegepast, en levert het karakter op waarvan het opgegeven getal de code is.

Dit programma laat de hele karakterset zien :

```
10 FOR a=32 TO 255: PRINT CHR$ a;: NEXT a
```

Op het scherm ziet u nu dit :



De karakterset

Zoals u ziet, bestaat de karakterset uit een spatie, vijftien symbolen en leestekens, de tien cijfers, nog eens zeven symbolen, de hoofdletters, nogmaals zes symbolen, de kleine letters en dan nog vijf symbolen. Al deze tekens, behalve het pond-teken en het copyright-teken, maken een deel uit van een algemeen gebruikte groep karakters die bekend staat als de ASCII code (zeg : "aski") (American Standard Codes for Information Interchange) (Amerikaanse standaard-codes voor de uitwisseling van informatie). De ASCII-standaard kent elk karakter een code toe; deze code wordt ook door de +2 gebruikt.

Het overige deel van de karakterset van de +2 maakt geen deel uit van de ASCII-codes : die is eigen aan de computers uit de ZX-gamma. De eerste groep bestaat uit een spatie en vijftien patronen met zwarte/witte blokjes. Dat zijn de grafische symbolen; die worden gebruikt om tekeningen te maken. U kunt deze tekens met het toetsenbord invoeren, door de "grafische mode" te gebruiken. U zet de +2 in grafische mode door op de "GRAPH"-toets te drukken. De toetsen 1 tot en met 8 geven dan deze symbolen :

Dit kan verwarrend worden, indien u het hoofd niet koel houdt. Kijk bijvoorbeeld hier eens naar :

```
PRINT VAL"VAL""VAL""""2""""""""
```

Denk er aan dat in een string, aanhalingstekens twee keer geschreven moeten worden. Gaat u verder in een string, dan zult u merken dan ze vier keer, zelfs acht keer geschreven moeten worden.

Er is nog een andere functie, die op VAL gelijkt maar minder nuttig is : VAL$. Het argument ervan is een string, maar het resultaat ervan is ook een string. Om te kunnen begrijpen hoe dit werkt, moet u er aan denken hoe VAL in twee stappen werkt. Eerst wordt het argument als een string geëvalueerd, daarna worden de aanhalingstekens weggehaald en wat overblijft wordt als getal geëvalueerd. Bij VAL$ is de eerste stap gelijk, de tweede ook, maar wat nu overblijft wordt als een string geëvalueerd, op deze manier :

```
VAL$"""Ursula""" is gelijk aan "Ursula"
```

Merk op hoe de aanhalingstekens welig bloeien...

Typ nu : LET a$="99"

en probeer dan de volgende functies te laten afdrukken : VAL a$, VAL "a$", VAL """a$""", VAL$ a$, VAL$"a$" en VAL$"""a$""". Sommige uitdrukkingen leveren een resultaat op, andere niet. Probeer alle antwoorden te verklaren. Houd het hoofd koel...

SGN is de teken-functie (soms "signum" genoemd). Het is de eerste functie die we zien, die niets met strings doet : argument en resultaat zijn allebei getallen. Het resultaat is +1 indien het argument positief is, 0 indien het argument nul is, en -1 indien het argument negatief is.

ABS is nog een functie waarvan het argument en het resultaat allebei getallen zijn. De functie zet het argument om in een positief getal (het resultaat), door het teken weg te laten. Zo betekent

```
ABS-3.2
```

hetzelfde als

```
ABS 3.2
```

namelijk : 3.2

INT wil zeggen : geheel gedeelte, integer. Een integer kan ook negatief zijn. Deze functie zet een gebroken getal om in een geheel getal door de cijfers na de komma weg te halen, zodat

```
INT 3.9
```

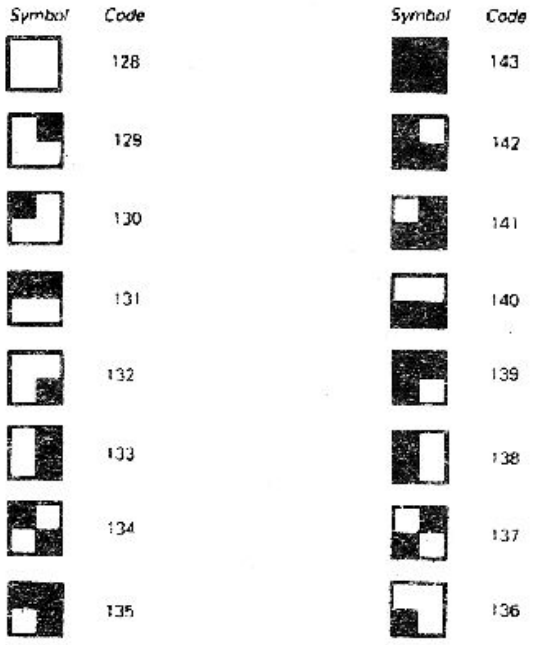
gelijk is aan 3.

Let op wanneer u de functie INT op negatieve getallen toepast : de functie rondt altijd af naar beneden. Zo is

```
INT-3.1
```

gelijk aan -4.

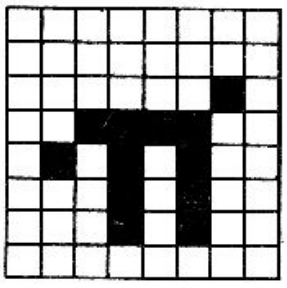
Hieronder vindt u de zestien grafische symbolen met hun code :



Na de grafische symbolen in de karakterset, ziet u opnieuw een reeks hoofdletters, van A tot S. Die reeks karakters kunt u zelf definiëren. Elke keer u de +2 inschakelt, worden ze als hoofdletters ingevuld. Deze groep staat bekend als UDG's : User Defined Graphics (door de gebruiker zelf gedefinieerde grafische symbolen). U kunt die intypen, door de +2 in grafische mode te zetten, en dan op de toetsen A tot en met S te drukken.

Om zelf een karakter te definiëren, volgt u de richtlijnen hieronder. Daarin wordt de Griekse letter pi getekend.

(1) Bepaal het uitzicht van het karakter. Elk karakter wordt getekend binnen een raster van 8 op 8 punten. Elk van die 64 punten kan "aan" of "uit" zijn, zwart of wit. Het is makkelijk om een tekening te maken, zoals hieronder. Daarop stellen de zwarte vierkantjes, de puntjes voor die "aan" zijn.



De zwarte puntjes, tekent de +2. in de INK-kleur. De witte puntjes worden in PAPER-kleur getekend. Die beide termen, INK en PAPER, worden in deel 16 van dit hoofdstuk uitgelegd.

INT rondt altijd af naar beneden. Om af te ronden naar het dichtstbije gehele getal, telt u eerst 0.5 op bij het argument. U zou een functie kunnen definiëren om dit automatisch te doen :

```
DEF FN r(x)=INT(x+0.5): REM x afgerond tot dichtstbije getal
```

Op die manier wordt

```
FN r(2.9) gelijk aan 3     FN r(2.4) gelijk aan 2
FN r(-2.9) gelijk aan -3   FN r(-2.4) gelijk aan -2
```

Vergelijk deze resultaten met wat u krijgt, indien u gewoon INT gebruikt in plaats van FN r().

Typ nu het volgende programmaatje in :

```
10 LET x=0: LET y=0: LET a=10
20 DEF FN p(x,y)=a+x*y
30 DEF FN q()=a+x*y
40 PRINT FN p(2,3),FN q()
```

In dit programmaatje zitten een aantal subtiele punten. Ten eerste, blijkt een functie niet beperkt te zijn tot één argument. Ze kan er meerdere hebben, of geen enkel. Maar in alle gevallen moeten de haakjes er staan.

Ten tweede maakt het niets uit waar in het programma de DEF-statements voorkomen. Nadat de +2 regel 10 heeft uitgevoerd, slaat hij gewoon regels 20 en 30 over, en voert hij regel 40 uit. De definities moéten evenwel ergens in het programma staan, en kunnen niet via een commando worden ingegeven.

Ten derde kunnen x en y tegelijk gebruikt worden als variabelen in het programma én als argument voor dé functie FN p. FN p "vergeet" op dat ogenblik de waarden van de variabelen x en y, maar omdat er geen argument "a" is, neemt de functie de waarde van de variabele "a". Op het ogenblik dat de functie FN p(2,3) wordt geëvalueerd, heeft a de waarde 10, omdat het om de variabele a gaat, heeft x de waarde 2 omdat x het eerste argument is, en heeft y de waarde 3 omdat y het tweede argument is. Het resultaat is dus 10+2*3 = 16. Bij de evaluatie van FN q() worden er geen argumenten gevonden, zodat a, x en y alledrie naar de variabelen met die namen refereren, en dus respectievelijk de waarden 10, 0 en 0 krijgen. In dit geval is het resultaat dus 10+0*0 = 10.

Wijzig nu regel 20 in :

```
20 DEF FN p(x,y)=FN q()
```

Nu zal FN p(2,3) ook de waarde 10 hebben, omdat FN q opnieuw de waarde van de variabelen x en y gebruikt, in plaats van de argumenten van FN p.

In sommige BASICs (niet bij de Spectrum) bestaan de functies LEFT$, RIGHT$, MID$ en TL$.

LEFT$(a$,n) geeft de substring van a$, die bestaat uit de eerste n karakters van a$.

RIGHT$(a$,n) geeft de substring van a$, die bestaat uit de karakters van a$, vanaf het nde karakter tot het einde.

MID$(a$,n1,n2) geeft de substring van a$, die bestaat uit n2 karakters van a$, en waarvan het eerste karakter het n1de karakter van a$ is. TL$(a$) geeft de substring van a$, die

aan die inhoud te veranderen. PEEK en POKE worden verder beschreven in deel 24 van dit hoofdstuk.

Na de UDG's volgen de "tokens". U zult gemerkt hebben, dat we de eerste tweeëndertig karakters, met codes 0 tot 31) niet afgedrukt hebben. Dat zijn allemaal controle-karakters. Ze drukken zelf niets af, maar dienen enkel om te controleren wat er op het scherm komt, of een andere functie van de +2.

Indien u probeert om de controlekarakters af te drukken, dan reageert de +2 met een "?" om aan te geven dat hij ze niet kan ontcijferen. In deel 27 van dit hoofdstuk vindt u meer uitleg over de controlekarakters.

De drie controlecodes die door het scherm worden gebruikt, zijn de codes 6, 8 en 13. Daarover hebben we het nu. Al bij al, is CHR$ 8 de enige code die u nuttig zou kunnen vinden.

CHR$ 6 drukt spaties af, op dezelfde manier van een komma in een PRINT-statement. Bijvoorbeeld :

```
PRINT 1; CHR$6; 2    doet net hetzelfde als    PRINT 1,2
```

Dit is natuurlijk niet zo'n handige manier om de code te gebruiken. Subtieler kan het ook :

```
LET a$="1"+CHR$ 6+"2"
PRINT a$
```

CHR$ 8 betekent "1 positie terug". Deze code plaatst de cursor 1 print-positie meer naar links. Probeer dit eens :

```
PRINT "1234"; CHR$ 8;"5"
```

en u ziet op het scherm staan :

```
1235
```

CHR$ 13 is "nieuwe regel". Deze code plaatst de cursor aan het begin van de volgende regel.

Het scherm wordt ook gecontroleerd door de codes 16 tot 23. Die worden uitgelegd in deel 15 en 16 van die hoofdstuk. Een lijst van alle codes vindt u in deel 27.

Met behulp van de codes van de karakters, kunnen we het begrip "alfabetisch rangschikken" uitbreiden, zodat strings kunnen worden gerangschikt, die niet alleen letters en cijfers bevatten maar willekeurige karaktercodes. Indien we niet langer denken aan een alfabet van 26 letters, maar aan een uitgebreid alfabet van 256 letters, in dezelfde volgorde als hun codes, dan blijft het principe gelijk. De onderstaande strings staan voor de Spectrum in alfabetische volgorde. Het kan wellicht raar lijken dat kleine letters nà de hoofdletters komen : a is "groter dan" Z. Ook hier hebben spaties een betekenis.

```
CHR$3+"ZOO"                          "Aardworm"
CHR$8+"AARDVARK"                     "Eglantier"
" AAAAA"                             "PRINT"
"(Opmerking tussen haakjes)"         "Zoo"
"100"                                "[interpolatie]"
"129.95 incl. BTW"                   "aardworm"
"AASVOGEL"                           "zoo"
```

te vinden die ook voor andere waarden van b opgaat, vertrekken we van de volgende regel :

```
a↑(b+c) = a↑b*a↑c
```

(Merk op dat we aan ↑ een hogere voorrang geven dan aan * en /. Indien er dus meerdere bewerkingen in een uitdrukking voorkomen, zal ↑ vóór * en / worden geëvalueerd.) U kunt zelf makkelijk zien dat deze regel opgaat indien b en c beide positieve gehele getallen zijn. Maar, indien we ook willen dat de regel opgaat voor andere getallen, dan moeten we aannemen dat :

```
a↑0 =1
a↑(-b) = 1/a↑b
a↑(1/b) = de b-de wortel van a, dat wil zeggen het getal dat
we b keer met zichzelf moeten vermenigvuldigen, om het getal
a opnieuw te krijgen.
```

Er geldt eveneens dat

```
a↑(b*c) = (a↑b)↑c
```

Als u dit nog nooit eerder zag, probeer dan niet om het van de eerste keer te onthouden. Onthoud alleen dat :

```
a↑(-1) = 1/a
```

en dat

```
a↑(1/2) = SQR a
```

en eens u met die twee vertrouwd bent, zal de rest ook wel duidelijker worden.

Probeer dit eens uit, met behulp van dit programma :

```
10 INPUT a,b,c
20 PRINT a↑(b+c),a↑b*a↑c
30 GO TO 10
```

Indien de regel die we opgaven, ook juist is, dan zullen de twee getallen die de +2 afdrukt, elke keer gelijk zijn. Let op : omwille van de manier waarop de computer de machtsverheffing uitwerkt, mag het getal links van het symbool, a in dit geval, nooit negatief zijn.

Een kenmerkend voorbeeld van het gebruik van deze functie, is samengestelde intrest berekenen. Stel dat u geld hebt belegd tegen een intrest van 15 % per jaar. Na een jaar hebt u dus niet enkel de 100 % die u voordien had, maar ook 15 % intrest, wat dus samen 115 % van het oorspronkelijke bedrag is. Anders gezegd, het bedrag is met 1.15 vermenigvuldigd, onverschillig om welk bedrag het ging. Na een tweede jaar, gebeurt weer hetzelfde. Nu hebt u dus 1.15 * 1.15, met andere woorden 1.15↑2, met andere woorden 1.3225 keer het oorspronkelijke bedrag. Algemeen gezegd : na y jaar hebt u 1.15↑y keer het oorspronkelijk bedrag.

Probeer dit :

```
FOR y=0 TO 100: PRINT y,10*1.15↑y: NEXT y
```

en u zult zien dat, zelfs al begint u met een klein bedrag (10), het vrij snel aangroeit en, wat meer is, het sneller en sneller groeit naarmate de tijd vordert. Hoewel, misschien stijgt de inflatie toch nog sneller...

```
100 REM loper
110 DATA "b",0,d, BIN 00101000, BIN 01000100
120 DATA BIN 01101100,c,b,0
130 REM koning
140 DATA "k",0,d,c,d
150 DATA c, BIN 01000100,c,0
160 REM toren
170 DATA "r",0, BIN 01010100,b,c
180 DATA c,b,b,0
190 REM koningin
200 DATA "q",0, BIN 01010100, BIN 00101000,d
210 DATA BIN 01101100,b,b,0
220 REM pion
230 DATA "p",0,0,d,c
240 DATA c,d,b,0
250 REM paard
260 DATA "n",0,d,c, BIN 01111000
270 DATA BIN 00011000,c,b,0
```

Merk op dat we in dit programma, in de DATA-statements gewoon 0 hebben gebruikt in plaats van BIN 00000000. RUN dit programma, en bekijk daarna de stukken door de +2 in grafische mode te zetten en dan op de toetsen B, K, R, Q, P of N te drukken.

OEFENINGEN
++++++++++

1. Stel u het raster van een karakter voor als verdeeld in vier gelijke parten. Elk part kan wit of zwart zijn, dus er zijn 2^4 = 16 combinaties. Zoek die allemaal op in de karakterset.

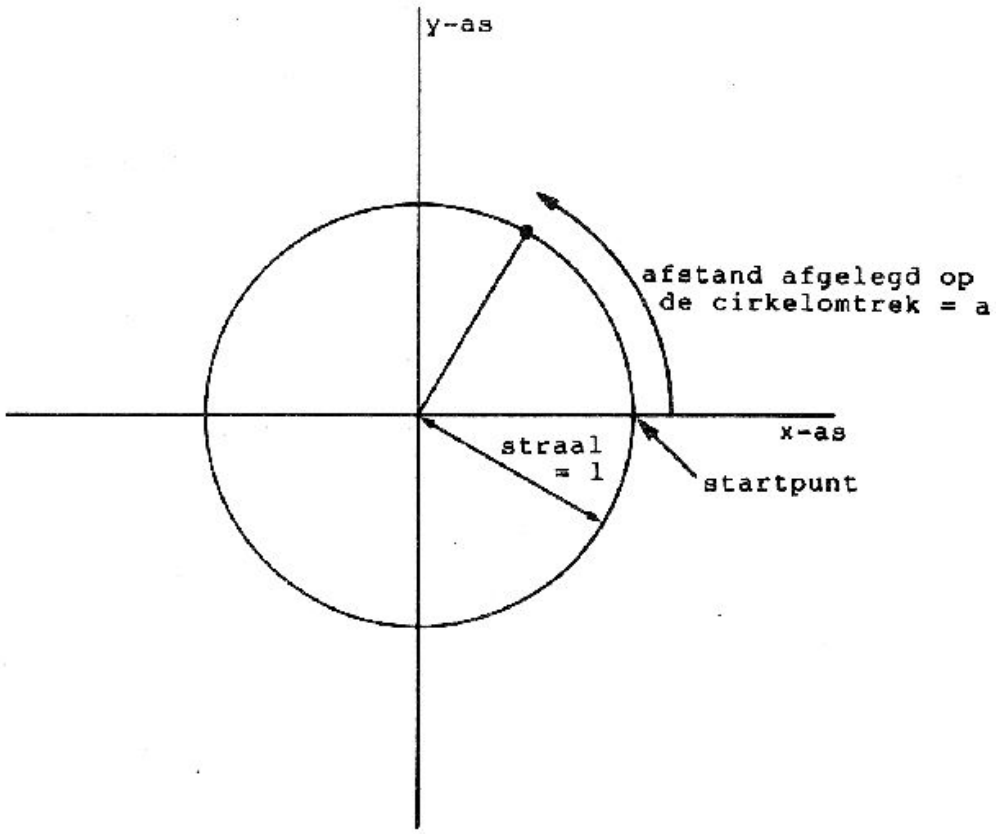
2. Run dit programma :

```
10 INPUT a
20 PRINT CHR$ a;
30 GO TO 10
```

Indien u wat experimenteert, zult u ontdekken dat CHR$ a altijd wordt afgerond tot het dichtstbije geheel getal. Indien a niet tussen 0 en 255 ligt, stopt het programma met de foutmelding : B Integer out of range".

3. Welke string is de "kleinste" : "KWAAD" of "kwaad" ?

vraagt u nu "straal 1 wat ?". Dat maakt niets uit, zolang we maar dezelfde eenheidsmaat aanhouden voor alle berekeningen. Het punt start op de "3 uur"-stand, en beweegt in tegenwijzerzin.



Door het middelpunt van de cirkel tekenden we ook twee lijnen, die we "assen" noemen. De horizontale as heet "x-as", de vertikale as heet "y-as".

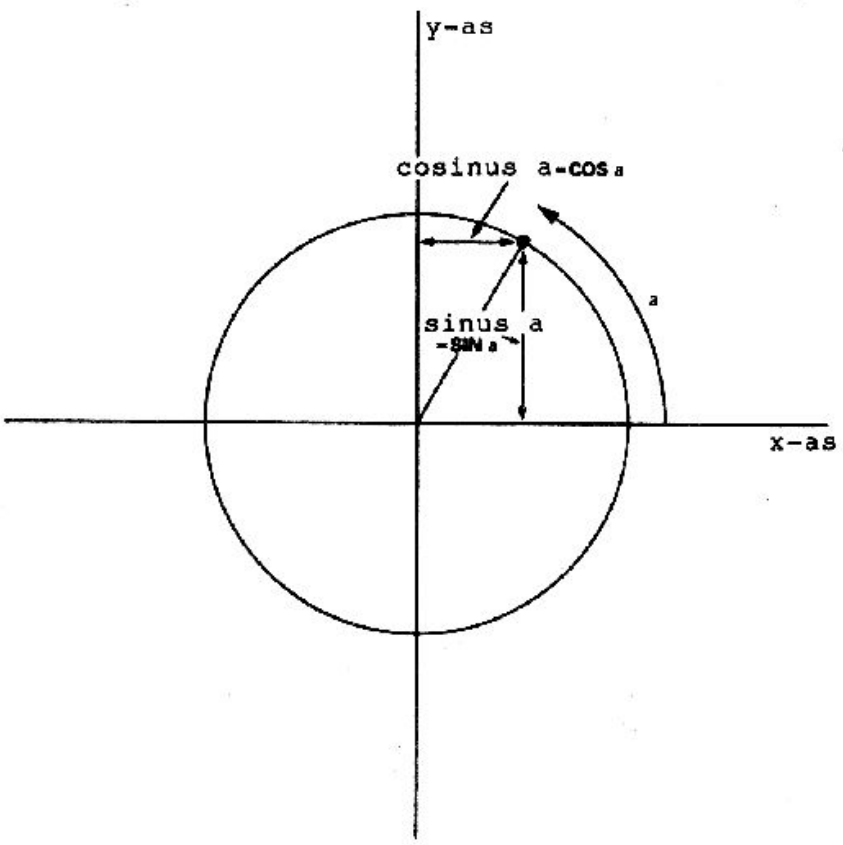
Om aan te geven waar het punt zich bevindt, geven we de afstand die het punt heeft afgelegd op de omtrek, te rekenen vanaf het 3 uur-punt. Die afstand noemen we a. We weten dat de omtrek van de cirkel gelijk is aan 2*pi (omdat de straal 1 is, en de diameter dus 2). Indien het punt dus een kwart van de omtrek heeft afgelegd, dan is a=pi/2. Is het punt een halve omtrek ver, dan is a=pi, en heeft het punt de hele omtrek beschreven, dan is a=2*pi.

Indien a gegeven is als de boog-afstand op de omtrek, dan kunnen we nog twee andere afstanden berekenen : hoever het punt zich rechts van de y-as bevindt, en hoever boven de x-as. Die beide afstanden heten respectievelijk de cosinus en de sinus. De functies COS en SIN op de +2 berekenen die afstanden.
(zie tekening op volgende bladzijde)

Merk op dat indien het punt zich links van de y-as bevindt, de cosinus begatief wordt en indien het punt onder de x-as gaat, wordt de sinus negatief.

Nog een eigenschap is dat, wanneer a de waarde 2*pi bereikt, het punt terug op de startpositie staat, en de sinus en cosinus opnieuw dezelfde waarden aannemen. Dat wil zeggen dat SIN(a+2*PI) = SIN a en dat COS(a+2*PI) = COS a.

De tangens van a wordt gedefinieerd als het quotient van sinus en cosinus. De overeenkomstige functie op de +2 heet TAN.



Het kan voorkomen dat we het omgekeerde van deze functies moeten berekenen, bijvoorbeeld de waarde van a berekenen voor een gegeven sinus, cosinus of tangens. De functies op de +2 die dit doen, heten ASN (arcsinus), ACS (arccosinus) en ATN (arctangens).

Kijk nog eens naar de tekening : er staat een straal getekend vanaf het middelpunt naar het punt dat op de omtrek beweegt. U kunt zien dat de afstand die we a noemden (de afstand die het punt op de omtrek heeft afgelegd) gebruikt kan worden om de hoek te meten die de straal met de x-as maakt. Wanneer a=pi/2, dan is de hoek 90 graden. Wanneer a=pi dan is de hoek 180 graden, en wanneer a= 2*pi, is de hoek 360 graden. U kunt rustig het rekenen in graden vergeten, en de hoek meten in termen van a. We zeggen dan dat we de hoek in radialen meten. Dat houdt in dat pi/2 radialen = 90 graden, en zo verder.

Denk er aan dat de functies COS, SIN enz. op de +2 werken in radialen, en niet in graden. Om graden in radialen om te zetten, deelt u door 180 en vermenigvuldigt u met pi. Om radialen in graden om te zetten, deelt u door pi en vermenigvuldigt u met 180.

DEEL 11 : WILLEKEURIGE GETALLEN

Inhoud ...
- RANDOMIZE
- RND

Dit deeltje handelt over de keywords RND en RANDOMIZE. In bepaalde opzichten is RND een functie : ze voert berekeningen uit en geeft een resultaat. Het ongewone is, dat er geen argument nodig is. Elke keer RND wordt gebruikt, levert dit een getal op tussen 0 en 1. Soms is het 0, maar nooit 1.

Typ dit in :

```
10 PRINT RND
20 GO TO 10
```

en zie hoe u elke keer verschillende getallen krijgt. Kunt u er een patroon in herkennen ? Dat hoort niet te kunnen : "willekeurig" houdt in dat er geen patroon in zit.

In feite is RND niet echt willekeurig, omdat er een vaste volgorde van 65536 getallen wordt gevolgd. Die getallen staan echter in een dergelijke wanorde, dat er op zijn minst geen duidelijk patroon in te herkennen valt. Daarom zeggen we dat RND pseudo-willekeurig is.

RND levert een willekeurig getal tussen 0 en 1 op, maar het is ook mogelijk om getallen binnen andere grenzen te verkrijgen. Bijvoorbeeld, 5*RND ligt tussen 0 en 5, en 1.3+0.7*RND ligt tussen 1.3 en 2. Om gehele getallen te verkrijgen, gebruikt u INT (denk er aan dat INT altijd naar beneden afrondt), bijvoorbeeld 1+INT(RND*6), dat we gebruiken in een programma dat teerlingen werpt. RND*6 ligt tussen 0 en 6, maar omdat het nooit 6 wordt, is INT(RND*6) dus 0, 1, 2, 3, 4 of 5.

Dit is het programma :

```
10 REM teerlingen gooien
20 CLS
30 FOR n=1 TO 2
40 PRINT 1+INT (RND*6);" ";
50 NEXT n
60 INPUT a$: GO TO 20
```

Druk ENTER telkens u wilt "werpen".

Het commando RANDOMIZE zorgt ervoor dat RND op een bepaalde plaats in de reeks getallen begint. Dit programma toont dat aan:

```
10 RANDOMIZE 1
20 FOR n=1 TO 5: PRINT RND,: NEXT n
30 PRINT: GO TO 10
```

Na de uitvoering van RANDOMIZE 1, start de RND-reeks telkens weer met 0.0022735596. U kunt andere getallen (tussen 1 en 65535) in het RANDOMIZE-statement gebruiken, waardoor de RND-reeks op een andere plaats start. In een programma met RND, dat fouten blijft maken die u niet kunt vinden, kan het nuttig zijn om RANDOMIZE op die manier te gebruiken : zo doet het programma elke keer hetzelfde, wanneer het gerund wordt.

RANDOMIZE alleen (of RANDOMIZE 0) werkt anders : daardoor wordt RND echt willekeurig. Kijk maar :

```
10 RANDOMIZE
20 PRINT RND: GO TO 10
```

De reeks getallen die u op deze manier verkrijgt, is niet echt willekeurig. Dat komt doordat RANDOMIZE gebruik maakt van de tijd die verlopen is sinds u de +2 inschakelde. Er verloopt even veel tijd tussen twee opeenvolgende uitvoeringen van RANDOMIZE en daardoor zal het RND-getal ook elke keer met min of meer hetzelfde bedrag worden opgehoogd. De reeks wordt willekeuriger indien u GO TO 10 vervangt door GO TO 20.

Hier volgt een programma dat muntstukken opgooit, en het aantal keren optelt dat kruis of munt wordt gegooid.

```
10 LET kruis=0: LET munt=0
20 LET worp=INT (RND*2)
30 IF worp=0 THEN LET kruis=kruis+1
40 IF worp=1 THEN LET munt=munt+1
50 PRINT kruis;",";munt
60 IF munt<>0 THEN PRINT kruis/munt
70 PRINT: GO TO 20
```

De verhouding kruis/munt zou 1 moeten worden indien u maar lang genoeg doorgaat, omdat te verwachten is dat na lange tijd het aantal keren kruis hetzelfde is als het aantal keren munt.

OEFENING
++++++++
1. Kies een getal tussen 1 en 872, en typ :
   RANDOMIZE (uw getal)
Controleer dat de eerstvolgende waarde van RND gelijk is aan
   (75*(uw getal+1)-1)/65536

DEEL 12 : ARRAYS

Inhoud ...
- arrays (de +2 behandelt string-arrays op een manier die licht afwijkt van de standaard)
- DIM

Veronderstel dat u een lijst getallen hebt, bijvoorbeeld de punten van tien leerlingen in een klas. Om die lijst te verwerken met de +2 kunt u de variabelen m1, m2, m3 enz. tot m10 gebruiken. Het programma om die variabelen toe te wijzen zou wel saai worden om in te typen :

```
 10 LET m1=75
 20 LET m2=44
 30 LET m3=90
 40 LET m4=38
 50 LET m5=55
 60 LET m6=64
 70 LET m7=70
 80 LET m8=12
 90 LET m9=75
100 LET m10=60
```

Voor dat soort toepassingen bestaat er een mechanisme, bekend als een "array", dat er in bestaat, een variabele te specificeren die (in plaats van één enkele waarde, zoals een gewone variabele) een aantal afzonderlijke "cellen" bevat, die elk verschillende waarden kunnen bevatten. Elk van die elementen wordt aangeduid met een index (het subscript), die tussen haakjes wordt geschreven na de naam van de variabele. In het bovenstaande voorbeeld zou de naam van het array m kunnen zijn

(die naam kan slechts uit 1 letter bestaan), en de tien variabelen zouden dan zijn : m(1), m(2), m(3) enzovoort.

De elementen van een array heten "geïndexeerde variabelen", ter onderscheiding van de gewone variabelen die u al kent.

Vooraleer u een array kunt gebruiken, moet u er ruimte voor voorzien in het geheugen van de +2. Dit doet u met het keyword DIM (dimensie). Het statement :

```
DIM m(10)
```

organiseert een array dat m heet, dat 10 dimensies heeft (d.w.z. er zijn 10 geïndexeerde variabelen). Het DIM-statement zet meteen ook elk element in het array op 0. Het wist ook een vooraf bestaande array met dezelfde naam uit het geheugen. Een gewone variabele m blijft evenwel bestaan : een array kan dezelfde naam dragen als een gewone variabele, omdat een array-variabele altijd met een index wordt aangeduid.

De indexen van de array-elementen mogen opgegeven worden door elke numerieke uitdrukking die een geldige index oplevert. Dit houdt in dat een array in een FOR-NEXT lus bewerkt kan worden. In plaats van het hogervermelde saaie programma, kunnen we de variabelen m(1) tot m(10) dus invullen als volgt :

```
10 DIM m(10)
20 FOR n=1 TO 10
30 READ m(n)
40 NEXT n
50 DATA 75,44,90,38,55,64,70,12,75,60
```

Let er op dat het DIM-statement moet uitgevoerd worden voordat er iets met het array kan gebeuren.

Indien u dat wilt, kunt u de inhoud van het array kennen door :

```
PRINT m(1)
PRINT m(2)
PRINT m(3), enzovoort
```

U kunt ook arrays opzetten met meer dan 1 dimensie. In een array met twee dimensies moet u een element aanduiden met twee indexen, op dezelfde manier waarop een positie op het scherm wordt aangeduid door een regelnummer en een kolom. Indien u dan nog stelt dat de regel- en kolomnummers op een bepaalde bladzijde betrekking hebben, dan kunt u nog een derde dimensie toevoegen, voor het nummer van de bladzijde. We hebben het hier wel over numerieke arrays : de elementen bevatten dus geen karakters, maar getallen. De elementen van een driedimensionaal array v worden dus aangeduid door v(b,r,k) waarbij b = bladzijde, r = regel en k = kolom.

Om bijvoorbeeld een tweedimensionaal array c op te zetten met dimensies 3 en 6, gebruikt u het statement :

```
DIM c(3,6)
```

Daardoor hebt u de beschikking over 3*6 = 18 geïndexeerde variabelen :

```
c(1,1) c(1,2) ... c(1,6)
c(2,1) c(2,2) ... c(2,6)
c(3,1) c(3,2) ... c(3,6)
```

Datzelfde principe gaat op voor elk aantal dimensies.

Hoewel een numerieke variabele dezelfde naam kan hebben als een

array-variabele, kunnen er niet tegelijkertijd twee arrays dezelfde naam hebben, ook al hebben ze andere dimensies.

Er bestaan ook string-arrays. De strings in een array verschillen van gewone strings, doordat ze een vaste lengte hebben en ze op de al eerder genoemde Procrusteaanse manier worden ingevuld (ingekort of opgevuld met spaties).

De naam van een string-array is 1 letter gevolgd door het $-teken. In tegenstelling tot numerieke arrays, kan een string-array niet dezelfde naam hebben als een gewone string.

Veronderstel dat u een array a$ wilt, die vijf strings kan bevatten. U moet op voorhand beslissen hoe lang die strings zullen zijn. Stel, dat u 10 karakters lang genoeg vindt. Het array maakt u klaar door :

```
DIM a$(5,10) (typ dit in)
```

Dit commando maakt een array klaar van 5*10 karakters, maar u kunt zich dat array ook voorstellen alsof elke rij een string was, op deze manier :

```
a$(1) = a$(1,1)a$(1,2)a$(1,3) ... a$(1,10)
a$(2) = a$(2,1)a$(2,2)a$(2,3) ... a$(2,10)
a$(3) = a$(3,1)a$(3,2)a$(3,3) ... a$(3,10), enz.
```

Wanneer u evenveel indexen opgeeft als er dimensies voorkwamen in het DIM-statement, twee in dit geval, dan is het resultaat één karakter. Indien u de laatste dimensie weglaat, krijgt u een string met een vaste lengte. Bijvoorbeeld : a$(2,7) is het zevende karakter uit de string a$(2). In slicing-notatie kunnen we dit ook schrijven als a$(2)(7). Typ nu in :

```
LET a$(2)="1234567890"
```

en dan :

```
PRINT a$(2),a$(2,7)
```

Dit levert het volgende op :

```
1234567890   7
```

In plaats van de laatste index (die u kunt weglaten), kunt u ook een slicer schrijven. Zo is bijvoorbeeld :

```
a$(2,4 TO 8) gelijk aan a$(2)(4 TO 8) = "45678"
```

Onthoud dat in een string-array, alle strings even lang zijn.

Het DIM-statement bevat een extra getal, het laatste, die deze vaste lengte aangeeft . Wanneer u een geïndexeerde variabele in een string-array aanduidt, kunt u een getal extra opgeven (of een slicer), dat dan overeenkomt met het laatste getal in het DIM-statement.

U kunt ook een string-array zonder extra dimensies aanmaken, bijvoorbeeld :

```
DIM a$(10)
```

dat ervoor zorgt dat a$ zich als een gewone string-variabele gedraagt, maar dat zijn lengte altijd 10 blijft, en dat de invulling ervan altijd Procrusteaans gebeurt.

OEFENING
++++++++
1. Maak een array m$ met READ en DATA. Het array bevat twaalf strings, waarvan m$(n) de naam van de nde maand bevat. Een tip : het DIM-statement is DIM m$(12,9). Controleer het, door de inhoud van alle m$(n) te laten afdrukken (gebruik een lus).

DEEL 13 : VOORWAARDEN

Inhoud...
- AND, OR
- NOT

In deel 3 van dit hoofdstuk zagen we al dat een IF-statement er zo uitziet :

```
If voorwaarde THEN ...
```

De voorwaarden die we daar gebruikten bestonden uit de vergelijkingen =, < , > , <=, >= en <>, waarmee twee getallen of twee strings vergeleken werden. U kunt ook een voorwaarde stellen, die een aantal van deze vergelijkingen combineert, door middel van de logische operatoren : AND, OR, NOT.

De combinatie (vergelijking AND vergelijking) is juist indien beide vergelijkingen tegelijk juist zijn. Het is dus mogelijk om een regel te schrijven zoals :

```
IF a$="ja" AND x>0 THEN PRINT x
```

waarbij x enkel afgedrukt wordt indien a$ "ja" bevat én indien x tegelijk groter is dan nul. Hier ligt BASIC zo dicht bij onze eigen taal, dat verdere uitleg eigenlijk niet hoeft. Net zoals in het Nederlands, kunnen hele reeksen vergelijkingen aan elkaar worden geregen met AND. De hele reeks is dan juist, indien alle vergelijkingen tegelijk juist zijn.

De combinatie (vergelijking OR vergelijking) is juist, indien tenminste één van de vergelijkingen juist is. De combinatie is ook juist, indien beide vergelijkingen juist zijn. Dit laatste gaat in het Nederlands niet altijd op...

De NOT-functie draait de zaken om. De NOT-betrekking is juist, wanneer de vergelijking onjuist is, en onjuist wanneer de vergelijking juist is.

Logische uitdrukingen kunnen combinaties bevatten van AND, OR en NOT, net zoals numerieke uitdrukkingen combinaties kunnen bevatten van +, -, * en /. Indien nodig, kunt u ook met haakjes werken. Logische bewerkingen hebben ook voorrangsregels, net zoals rekenkundige bewerkingen. OR heeft de laagste voorrang, dan komt AND en dan NOT.

NOT is eigenlijk een functie, met een argument en een resultaat, maar ze heeft een voorrang die veel lager ligt dan die van andere functies. Daarom moet het argument van NOT niet tussen haakjes, tenzij het AND of OR bevat (of beide). NOT a=b betekent hetzelfde als NOT (a=b) (en uiteraard hetzelfde als a<>b).

<> is de negatie van = in die zin, dat de uitdrukking enkel juist is indien = niet juist is. Met andere woorden :

```
a<>b is hetzelfde als NOT a=b
```

NOT a<>b is hetzelfde als a=b

Ga voor uzelf na dat >= en <= de negaties zijn van < en >, in die volgorde. U kunt dus altijd een NOT weghalen uit een uitdrukking, door de vergelijking aan te passen.

De uitdrukking

NOT (uitdrukking 1 AND uitdrukking 2)

betekent hetzelfde als

NOT (uitdrukking 1) OR NOT (uitdrukking 2)

en de uitdruking

NOT (uitdrukking 1 OR uitdrukking 2)

betekent hetzelfde als

NOT (uitdrukking 1) AND NOT (uitdrukking 2)

Op die manier kunt u alle NOTs buiten haakjes zetten, tot ze allemaal op vergelijkingen slaan, en dan kunt u ze weghalen. Logisch bekeken, is NOT onnodig, hoewel een programma er soms wel doorzichtiger door wordt.

Wat nu volgt is nogal ingewikkeld. Mensen met een zwak hart kunnen dit stuk het beste maar overslaan...

Typ dit in :

PRINT 1=2, 1<>2

Misschien had u een foutmelding verwacht ? De computer kent geen logische waarden. Hij gebruikt gewoon getallen, volgens een aantal regels :

1) =, <, >, <=, >= en <> geven een numeriek resultaat : 1 voor juist, 0 voor onjuist. Daarom gaf het eerste PRINT-commando hierboven (1=2) het resultaat 0, omdat de uitdrukking 1=2 onjuist is. De uitdrukking 1<>2 gaf het resultaat 1, omdat ze juist is.

2) In het statement "IF voorwaarde THEN ... " kan de voorwaarde ook een numerieke uitdrukking zijn. Is de waarde van die uitdrukking 0, dan wordt ze als onjuist gerekend. Elke andere waarde wordt als "juist" gerekend, met inbegrip van de waarde 1, als resultaat van een juiste vergelijking. Het IF-statement betekent dus precies hetzelfde als : IF voorwaarde <>0 THEN ...

3) AND, OR en NOT krijgen ook een numerieke waarde :

x AND y wordt x, indien y juist is (van 0 verschilt)
0 (onjuist) indien y onjuist is (nul)

x OR y wordt 1 (juist) indien y juist is (van nul verschilt)
x indien y onjuist is (nul)

NOT x wordt 0 (onjuist) indien x juist is (van nul verschilt)
1 (juist) indien x onjuist is (nul)

Let op : "juist" betekent "niet nul" indien we een bepaalde, gegeven waarde bekijken, maar het betekent "1" indien er een nieuwe waarde wordt berekend, als resultaat van een uitdrukking.

Probeer dit programma eens :

```
10 INPUT a
20 INPUT b
30 PRINT (a AND a>=b)+(b AND a<b)
40 GO TO 10
```

Elke keer wordt het grootste getal van beide afgedrukt.

Probeer zover te komen dat u de uitdrukking

x AND y ziet als : x indien y (anders is het resultaat 0)

en x OR y als : x tenzij y (in dit geval is het resultaat 1)

Een uitdrukking die AND en OR bevat, wordt "voorwaardelijke uitdrukking" genoemd. Een voorbeeld van het gebruik van OR is :

```
LET totaal = nettoprijs *(1.20 OR v$="taksvrij")
```

Merk op, dat AND meer met optellingen wordt geassocieerd (omdat de standaardwaarde 0 is), en OR met vermenigvuldigingen (omdat de standaardwaarde 1 is).

U kunt ook voorwaardelijke uitdrukkingen maken die strings bevatten, maar enkel met AND.

```
x$ AND y wordt x$ indien y verschilt van nul
              "" indien y nul is
```

en betekent dus : x$ indien y (zoniet, de nulstring)

Probeer eens dit programma. Het vraagt twee strings, en zet ze in alfabetische volgorde.

```
10 INPUT"Geef twee strings"'a$,b$
20 IF a$>b$ THEN LET c$=a$:LETa$=b$: LET b$=c$
30 PRINT a$;" ";("<" AND a$<b$)+("=" AND a$=b$);" ";b$
40 GO TO 10
```

DEEL 14 : DE KARAKTERSET

Inhoud...
- CODE, CHR$
- POKE, PEEK
- USR
- BIN

De letters, cijfers, spaties en leestekens die in een string kunnen voorkomen, noemen we "karakters". Samen vormen ze de karakterset van de +2. De meeste van die karakters bestaan uit één symbool. Sommige karakters, die we "tokens" noemen, stellen een heel woord voor, of een groep symbolen : PRINT, STOP, <> enzovoort.

De +2 beschikt over 256 karakters, die elk een code hebben van 0 tot 255. In hoofdstuk 27 vindt u een complete lijst van die karakters. Om codes in karakters te vertalen en omgekeerd, beschikken we over twee functies : CODE en CHR$.

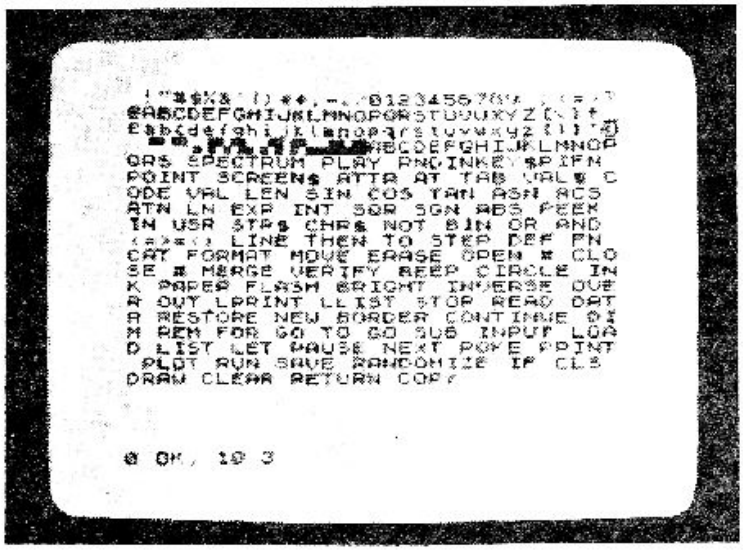
CODE kan op een string worden toegepast, en levert de code op van het eerste karakter in de string, of 0 voor de nulstring.

CHR$ kan op een getal worden toegepast, en levert het karakter op waarvan het opgegeven getal de code is.

Dit programma laat de hele karakterset zien :

```
10 FOR a=32 TO 255: PRINT CHR$ a;: NEXT a
```

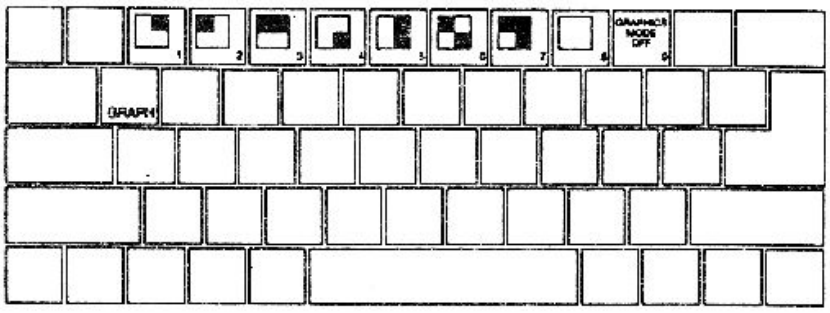
Op het scherm ziet u nu dit :



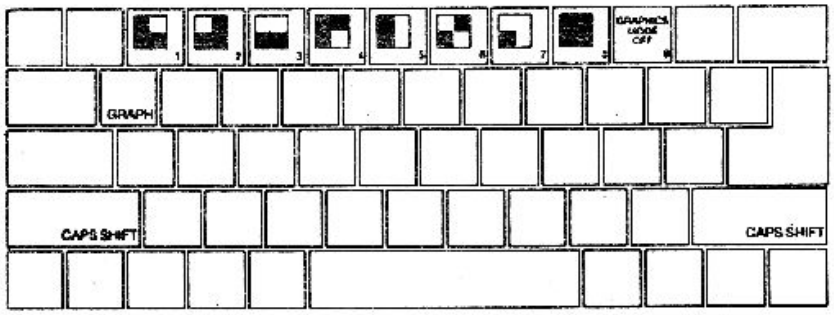
De karakterset

Zoals u ziet, bestaat de karakterset uit een spatie, vijftien symbolen en leestekens, de tien cijfers, nog eens zeven symbolen, de hoofdletters, nogmaals zes symbolen, de kleine letters en dan nog vijf symbolen. Al deze tekens, behalve het pond-teken en het copyright-teken, maken een deel uit van een algemeen gebruikte groep karakters die bekend staat als de ASCII code (zeg : "aski") (American Standard Codes for Information Interchange) (Amerikaanse standaard-codes voor de uitwisseling van informatie). De ASCII-standaard kent elk karakter een code toe; deze code wordt ook door de +2 gebruikt.

Het overige deel van de karakterset van de +2 maakt geen deel uit van de ASCII-codes : die is eigen aan de computers uit de ZX-gamma. De eerste groep bestaat uit een spatie en vijftien patronen met zwarte/witte blokjes. Dat zijn de grafische symbolen; die worden gebruikt om tekeningen te maken. U kunt deze tekens met het toetsenbord invoeren, door de "grafische mode" te gebruiken. U zet de +2 in grafische mode door op de "GRAPH"-toets te drukken. De toetsen 1 tot en met 8 geven dan deze symbolen :



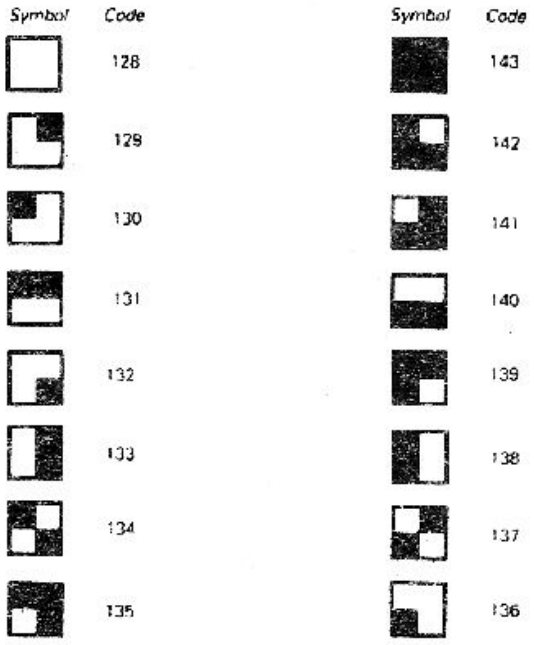
Indien u in grafische mode, de CAPS SHIFT-toets indrukt samen met een van de toetsen 1 tot 8, dan krijgt u het omgekeerde van diezelfde grafische symbolen. Dat wil zeggen : zwart wordt wit, en wit wordt zwart :



In grafische mode zullen de cursortoetsen niet werken, omdat de +2 ze interpreteert als de combinatie van een cijfer + CAPS SHIFT. Ze leveren dus ook grafische symbolen op bij het indrukken ervan.

Een druk op de "9" schakelt de grafische mode weer uit. Hetzelfde gebeurt door een tweede druk op de GRAPH-toets. In grafische mode kunt u het karakter links van de cursor weghalen, door op de "0" te drukken.

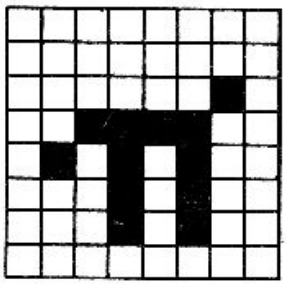
Hieronder vindt u de zestien grafische symbolen met hun code :



Na de grafische symbolen in de karakterset, ziet u opnieuw een reeks hoofdletters, van A tot S. Die reeks karakters kunt u zelf definiëren. Elke keer u de +2 inschakelt, worden ze als hoofdletters ingevuld. Deze groep staat bekend als UDG's : User Defined Graphics (door de gebruiker zelf gedefinieerde grafische symbolen). U kunt die intypen, door de +2 in grafische mode te zetten, en dan op de toetsen A tot en met S te drukken.

Om zelf een karakter te definiëren, volgt u de richtlijnen hieronder. Daarin wordt de Griekse letter pi getekend.

(1) Bepaal het uitzicht van het karakter. Elk karakter wordt getekend binnen een raster van 8 op 8 punten. Elk van die 64 punten kan "aan" of "uit" zijn, zwart of wit. Het is makkelijk om een tekening te maken, zoals hieronder. Daarop stellen de zwarte vierkantjes, de puntjes voor die "aan" zijn.



De zwarte puntjes, tekent de +2. in de INK-kleur. De witte puntjes worden in PAPER-kleur getekend. Die beide termen, INK en PAPER, worden in deel 16 van dit hoofdstuk uitgelegd.

Rond het karakter hebben we een witte rand van 1 blokje gelaten, omdat dat bij alle andere karakters ook zo is. Alleen de letters met een "staart" gaan helemaal tot op de bodem van het vakje.

(2) Bepaal bij welke letter het symbool moet horen. Stel, dat pi door de letter P wordt voorgesteld, zodat een druk op "P" in grafische mode, de nieuwe letter pi zal opleveren.

(3) Zet het patroon van de nieuwe letter in het geheugen. Elke UDG staat in het geheugen als een reeks van acht getallen, één getal per horizontale rij. U kunt die getallen in een programma opnemen als een reeks van BIN-getallen : het keyword BIN, gevolgd door acht nullen of enen (0 voor PAPER, 1 voor INK). De acht getallen voor het pi-teken zijn :

```
BIN 00000000  bovenste rij
BIN 00000000  tweede rij
BIN 00000010  derde rij
BIN 00111100  vierde rij
BIN 01010100  vijfde rij
BIN 00010100  zesde rij
BIN 00010100  zevende rij
BIN 00000000  onderste rij
```

Indien u iets kent van binaire getallen, dan kan het nuttig zijn om te weten dat BIN gebruikt kan worden om een getal in zijn binaire vorm te schrijven, in plaats van als decimaal. Indien u met halfgesloten ogen naar het patroon van nullen en enen kijkt, dan is het nieuwe symbool goed te onderscheiden.

De genoemde acht getallen worden op acht plaatsen (bytes) in het geheugen opgeslagen. Elk van die acht plaatsen heeft een eigen "adres". Het adres van de eerste byte (de eerste groep van acht cijfers) is USR "P" (omdat we hoger de leter P gekozen hadden om het nieuwe teken in op te slaan). Het adres van de tweede byte is USR "P" +1, en zo verder tot de achtste byte, die het adres USR "P" +7 heeft.

In dit geval is USR een functie, die een string-argument omrekent in het adres van de eerste byte van de overeenkomstige UDG. Het string-argument moet bestaan uit één karakter. Dat kan ofwel de UDG zelf zijn, ofwel de overeenkomstige letter, in hoofdletters of kleine letters. USR kan ook nog met een getal als argument gebruikt worden, maar daarover zien we later meer.

Ook al begrijpt u het niet zo goed, typ het volgende programma in. Het definieert de grafische "P" als het symbool pi.

```
 10 FOR n=0 TO 7
 20 READ rij: POKE USR "P"+n,rij
 30 NEXT n
 40 BIN 00000000
 50 BIN 00000000
 60 BIN 00000010
 70 BIN 00111100
 80 BIN 01010100
 90 BIN 00010100
100 BIN 00010100
110 BIN 00000000
```

Het POKE-statement zet een getal direkt in het geheugen, zonder verdere tussenkomst van de technieken die BASIC gewoonlijk gebruikt. Het tegenovergestelde van POKE is PEEK. Met PEEK kunnen we de inhoud van een geheugenplaats bekijken, zonder iets

aan die inhoud te veranderen. PEEK en POKE worden verder beschreven in deel 24 van dit hoofdstuk.

Na de UDG's volgen de "tokens". U zult gemerkt hebben, dat we de eerste tweeëndertig karakters, met codes 0 tot 31) niet afgedrukt hebben. Dat zijn allemaal controle-karakters. Ze drukken zelf niets af, maar dienen enkel om te controleren wat er op het scherm komt, of een andere functie van de +2.

Indien u probeert om de controlekarakters af te drukken, dan reageert de +2 met een "?" om aan te geven dat hij ze niet kan ontcijferen. In deel 27 van dit hoofdstuk vindt u meer uitleg over de controlekarakters.

De drie controlecodes die door het scherm worden gebruikt, zijn de codes 6, 8 en 13. Daarover hebben we het nu. Al bij al, is CHR$ 8 de enige code die u nuttig zou kunnen vinden.

CHR$ 6 drukt spaties af, op dezelfde manier van een komma in een PRINT-statement. Bijvoorbeeld :

```
PRINT 1; CHR$6; 2    doet net hetzelfde als    PRINT 1,2
```

Dit is natuurlijk niet zo'n handige manier om de code te gebruiken. Subtieler kan het ook :

```
LET a$="1"+CHR$ 6+"2"
PRINT a$
```

CHR$ 8 betekent "1 positie terug". Deze code plaatst de cursor 1 print-positie meer naar links. Probeer dit eens :

```
PRINT "1234"; CHR$ 8;"5"
```

en u ziet op het scherm staan :

```
1235
```

CHR$ 13 is "nieuwe regel". Deze code plaatst de cursor aan het begin van de volgende regel.

Het scherm wordt ook gecontroleerd door de codes 16 tot 23. Die worden uitgelegd in deel 15 en 16 van die hoofdstuk. Een lijst van alle codes vindt u in deel 27.

Met behulp van de codes van de karakters, kunnen we het begrip "alfabetisch rangschikken" uitbreiden, zodat strings kunnen worden gerangschikt, die niet alleen letters en cijfers bevatten maar willekeurige karaktercodes. Indien we niet langer denken aan een alfabet van 26 letters, maar aan een uitgebreid alfabet van 256 letters, in dezelfde volgorde als hun codes, dan blijft het principe gelijk. De onderstaande strings staan voor de Spectrum in alfabetische volgorde. Het kan wellicht raar lijken dat kleine letters nà de hoofdletters komen : a is "groter dan" Z. Ook hier hebben spaties een betekenis.

```
CHR$3+"ZOO"                          "Aardworm"
CHR$8+"AARDVARK"                     "Eglantier"
" AAAAA"                             "PRINT"
"(Opmerking tussen haakjes)"         "Zoo"
"100"                                "[interpolatie]"
"129.95 incl. BTW"                   "aardworm"
"AASVOGEL"                           "zoo"
```

Om de volgorde van twee strings te bepalen, is er een eenvoudige regel. Vergelijk eerst de beide eerste karakters van de strings. Zijn die verschillend, dan is één van beide kleiner dan de anders, zodat de string waar die code in voorkomt, vóór de andere komt (kleiner is dan de andere). Waren de codes gelijk, dan gaat u verder met de vergelijking van de volgende karakters. Wordt het einde van een string bereikt terwijl de andere nog meer karakters bevat, dan is de eerste string de kleinste. In het andere geval, moeten ze gelijk zijn.

De vergelijkingstekens =, < , >, <=, >= en <> kunnen ook voor strings gebruikt worden. "<" betekent dan "komt vóór" en ">" betekent "komt nà". Zo zijn de vergelijkingen

```
"AA man"<"AARDVARKEN"
"AARDVARKEN">"AA man"
```

allebei juist.

Ook <= en >= werken met strings zoals met getallen, zodat

```
"Dezelfde string" <= "Dezelfde string"
```

juist is, maar

```
"Dezelfde string" < "Dezelfde string"
```

onjuist.

Probeer dit alles eens uit met het volgende programma. Dit vraagt u om twee strings in te typen, en zet ze in alfabetische volgorde.

```
10 INPUT "Geef twee strings:",a$,b$
20 IF a$>b$ THEN LET c$=a$: LET a$=b$: LET b$=c$
30 PRINT a$;" ";
40 IF a$<b$ THEN PRINT "<";: GO TO 60
50 PRINT "=";
60 PRINT " ";b$
70 GO TO 10
```

In het bovenstaande programma (en in het programma op het einde van deel 13) moeten we c$ ter hulp roepen op regel 20, wanneer we de inhoud van a$ en b$ willen verwisselen. Kunt u begrijpen waarom de instructies

```
LET a$=b$: LET b$=a$
```

niet het gewenste effect zouden hebben ?

Het nu volgende programma definieert UDG' als schaakstukken. De volgende letters worden gebruikt : B (Bishop) voor de loper, K (King) voor de koning, R (Rook) voor de toren, Q (Queen) voor de koningin, P (Pawn) voor de pion en N (kNight) voor het paard.

Schaakstukken maken ...

```
 5 LET b=BIN 01111100: LET c= BIN 0011100
   LET d=BIN 00010000
10 FOR n=1 TO 6: READ p$: REM 6 stukken
20 FOR f=0 TO 7: REM lees stuken in 8 bytes
30 READ a: POKE USR p$+f,a
40 NEXT f
50 NEXT n
```

```
100 REM loper
110 DATA "b",0,d, BIN 00101000, BIN 01000100
120 DATA BIN 01101100,c,b,0
130 REM koning
140 DATA "k",0,d,c,d
150 DATA c, BIN 01000100,c,0
160 REM toren
170 DATA "r",0, BIN 01010100,b,c
180 DATA c,b,b,0
190 REM koningin
200 DATA "q",0, BIN 01010100, BIN 00101000,d
210 DATA BIN 01101100,b,b,0
220 REM pion
230 DATA "p",0,0,d,c
240 DATA c,d,b,0
250 REM paard
260 DATA "n",0,d,c, BIN 01111000
270 DATA BIN 00011000,c,b,0
```

Merk op dat we in dit programma, in de DATA-statements gewoon 0 hebben gebruikt in plaats van BIN 00000000. RUN dit programma, en bekijk daarna de stukken door de +2 in grafische mode te zetten en dan op de toetsen B, K, R, Q, P of N te drukken.

OEFENINGEN
++++++++++

1. Stel u het raster van een karakter voor als verdeeld in vier gelijke parten. Elk part kan wit of zwart zijn, dus er zijn 2^4 = 16 combinaties. Zoek die allemaal op in de karakterset.

2. Run dit programma :

```
10 INPUT a
20 PRINT CHR$ a;
30 GO TO 10
```

Indien u wat experimenteert, zult u ontdekken dat CHR$ a altijd wordt afgerond tot het dichtstbije geheel getal. Indien a niet tussen 0 en 255 ligt, stopt het programma met de foutmelding : B Integer out of range".

3. Welke string is de "kleinste" : "KWAAD" of "kwaad" ?

DEEL 15 : OVER PRINT EN INPUT

Inhoud ...
- CLS
- PRINT items : niets
- Uitdrukkingen (numeriek of met strings) : TAB num. uitdrukking en AT num. uitdrukking
- PRINT-scheidingstekens.
- INPUT variabelen, numeriek of strings
- LINE string-variabele
- PRINT items die niet met een letter beginnen (tokens beginnen niet met een letter)
- scrollen van het scherm
- SCREEN$

We hebben het PRINT-commando al heel vaak gebruikt. U zult waarschijnlijk wel reeds een idee hebben hoe het gebruikt kan worden. Uitdrukkingen waarvan de waarde wordt afgedrukt, noemen we PRINT-items. Ze kunnen van elkaar gescheiden worden door komma's, kommapunten of afkappingstekens. Die scheidingstekens noemen we PRINT-scheidingstekens. Een PRINT-item kan ook uit helemaal niets bestaan. Dit verklaart wat er gebeurt, wanneer u twee komma's na elkaar gebruikt.

Er zijn nog twee PRINT-items, die de +2 niet vertellen wàt hij moet afdrukken, maar wel wààr op het scherm. Zo zal bijvoorbeeld

```
10 PRINT AT 11,16;"*"
```

een sterretje in het midden van het scherm afdrukken. Dat komt doordat het PRINT-item

```
AT regel,kolom
```

de print-positie (d.w.z. de plaats op het scherm waar het eerstvolgende print-item zal worden afgedrukt) instelt op de aangegeven regel en kolom. De regels op het scherm zijn genummerd van 0 (bovenaan) tot 21 (onderaan). De kolommen gaan van 0 (links op de regel) tot 31 (rechts op de regel).

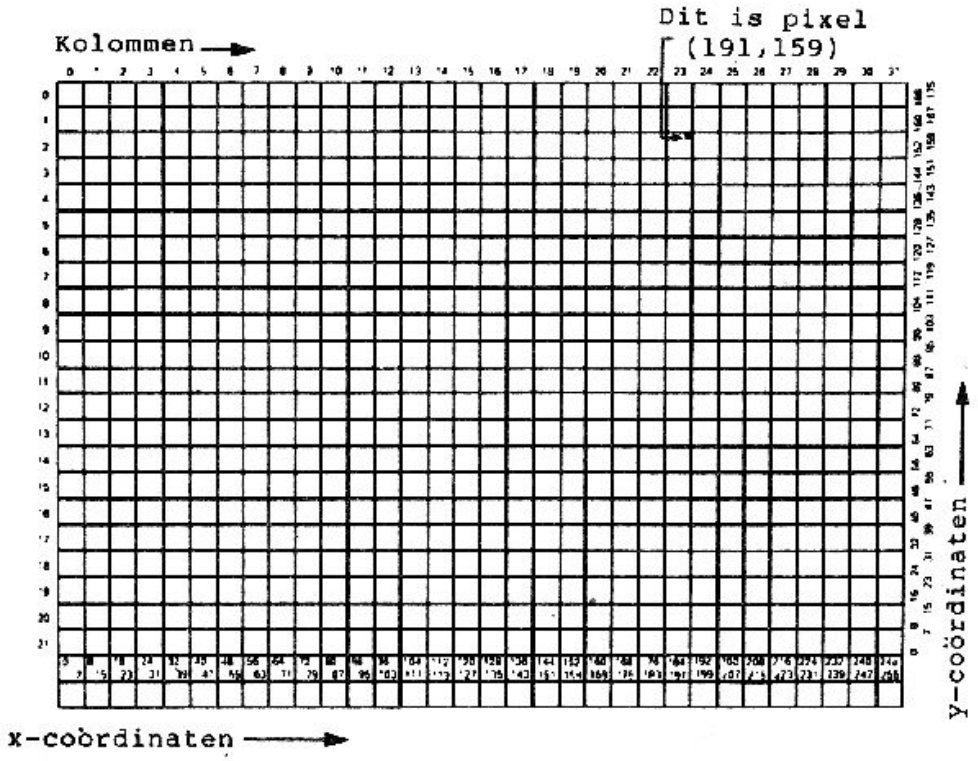
SCREEN$ is het omgekeerde van PRINT AT. Deze functie "leest", binnen bepaalde grenzen, het karakter dat op een bepaalde plaats op het scherm staat. SCREEN$ gebruikt dezelfde regel- en kolom-coördinaten als PRINT AT, maar nu tussen haakjes. De instructie

```
20 PRINT AT 0,0; SCREEN$(11,16)
```

zal het sterretje in het midden van het scherm "lezen" en dit vervolgens op de positie 0,0 (linksboven het scherm) printen.

Karakters die deel uitmaken van tokens worden eveneens gelezen, als enkelvoudige karakters; spaties worden als spaties gelezen. Indien u evenwel probeert om UDG's, te lezen, grafische karakters of lijnen die door PLOT, DRAW of CIRCLE zijn getekend, dan is het resultaat de nulstring. Dit geldt ook indien u het OVER-commando had gebruikt om een samengesteld karakter af te drukken. De keywords PLOT, DRAW, CIRCLE en OVER worden in deel 16 en 17 van dit hoofdstuk beschreven.

Op de volgende bladzijde ziet u een raster van PRINT- en PLOT-coördinaten. Het is niet mogelijk om met gewone middelen vanuit BASIC op de onderste twee regels te PRINTen noch te PLOTten.



De functie

TAB kolom

drukt zoveel spaties af als nodig, om de PRINT-positie op de aangegeven kolom in te stellen. Indien het commando zou inhouden dat de positie "achteruit" zou moeten bewegen, wordt naar de volgende regel bewogen. Merk op dat de +2 het kolom-nummer modulo 32 berekent. Dat wil zeggen dat hij het getal deelt door 32 en de rest behoudt : TAB 33 is hetzelfde als TAB 1.

Een voorbeeld :

```
PRINT TAB 30;1; TAB 12;"Inhoud"; AI 3,1;"Hoofdstuk"; TAB 24;"
Blz."
```

zou de hoofding kunnen afdrukken van de inhouds-opgave van een boek.

Probeer dit eens :

```
10 FOR n=0 TO 20
20 PRINT TAB 8*n;n;
30 NEXT n
```

Daaruit blijkt wat er bedoeld wordt met "modulo 32".

Om het mooier te maken, kunt u op regel 20 de 8 in 6 veranderen.

Een paar punten om te onthouden :

(1) TABs en PRINT-items worden het beste gevolgd door kommapunten, zoals in het bovenstaande programma. Het is toegestaan om komma's (of helemaal niets, op het einde van het statement) te gebruiken, maar dat houdt in dat u eerst precies de PRINT-positie hebt bepaald, en ze direct daarna weer verzet. Niet zo erg nuttig ...

(2) U kunt niets afdrukken op de onderste twee regels (22 en 23) omdat die gereserveerd worden voor het intypen van commando's, INPUT van gegevens, foutmeldingen en zo verder. Wanneer er dus over "de onderste regel" wordt gesproken, wordt daarmee veelal regel 21 bedoeld.

(3) Met AT kunt u de print-positie ook bepalen op een plaats waar al iets werd afgedrukt. Wat er stond, wordt gewoon overschreven.

Nog een statement dat met PRINT verband houdt, is CLS. Dit commando wist het hele scherm schoon.

Indien bij het afdrukken, de onderste regel wordt bereikt, dan scrollt het scherm omhoog, op dezelfde manier als het papier in een schrijfmachine. Dit kunt u uitproberen door in het kleine scherm te werken, via de optie "Screen" in het edit-menu (zie hoofdstuk 6), en daar te typen :

```
CLS: FOR n=1 TO 30: PRINT n: NEXT n
```

Nadat hij een aantal regels heeft geprint, stopt de +2 en vraagt hij onderaan het scherm "scroll ?". U kunt nu op uw gemak de eerste 22 getallen bekijken. Bent u klaar, dan drukt u "y" (yes, ja) en de +2 gaat verder met het afdrukken. In werkelijkheid kunt u op elke toets drukken, behalve N (neen), BREAK of de spatiebalk. Deze drie doen de +2 ophouden, en geven het report "D BREAK - CONT repeats".

Het INPUT-statement kan veel meer doen dan we tot nu toe hebben vermeld. U bent al vertrouwd met INPUT op deze manier :

```
INPUT "Hoe oud ben je ?";leeftijd
```

waarbij de +2 de tekst "Hoe oud ben je?" onderaan het scherm afdrukt, en dan wacht tot u uw leeftijd intypt. In feite kan een INPUT-statement ook items en scheidingstekens bevatten, net zoals een PRINT-statement. Dit wil zeggen dat "Hoe oud ben je?" en "leeftijd" allebei INPUT-items zijn. Over het algemeen zijn INPUT-items hetzelfde als PRINT-items, hoewel er een paar zeer belangrijke verschillen zijn.

Ten eerste is er duidelijk een extra INPUT-item, namelijk de variabele waaraan de ingetypte waarde zal worden toegewezen, "leeftijd" in het bovenstaande voorbeeld. De regel luidt, dat indien een INPUT-item met een letter begint, het een variabele is waarvan de waarde ingetypt dient te worden.

Dit lijkt in te houden dat u in de tekst onderaan, geen waarden van variabelen kunt laten afdrukken. Dit kunt u wel, door de variabele tussen haakjes te zetten. Een uitdrukking die met een letter begint, moet tussen haakjes gezet worden, indien het de bedoeling is dat ze onderaan het scherm wordt afgedrukt, samen met de INPUT-tekst.

Elk PRINT-item dat niet aan hogergenoemde regels onderhevig is, kan ook een INPUT-item zijn. Dit voorbeeld illustreert de

bedoeling :

```
LET mijnlftd = INT (RND*100): INPUT ("Ik ben "; mijnlftd;".")
;"Hoe oud ben jij? ",jouwlftd
```

De variabele "mijnlftd" staat tussen haakjes, zodat de waarde ervan wordt afgedrukt. De variabele "jouwlftd" staat niét tussen haakjes, zodat de waarde ervan ingetypt dient te worden.

Alles wat een INPUT-statement afdrukt, wordt in het onderste deel van het scherm afgedrukt. Dit staat min of meer los van het bovenste deel. Wat meer is, de regels in dat onderste deel worden genummerd, te beginnen bij de bovenste regel van dat deel. Dit blijft zo, ook al overschrijft dit deel het eigenlijke scherm. Dat kan gebeuren, indien u erg veel INPUT-data intypt. Wat er ook gebeurt met het onderste deel van het scherm, het wordt altijd weer teruggebracht tot twee regels, wanneer het programma stopt en u aan het programma gaat werken.

Om het resultaat van AT in INPUT-statements te zien, kunt u dit uitproberen :

```
10 INPUT "Dit is regel 1.",a$; AT 0,0;"Dit is regel 0.";a$;
AT 2,0;"Dit is regel 2.",a$; AT 1,0;"Dit is nog steeds regel
1.",a$
```

Run het programma. Druk ENTER, elke keer het stopt. Wanneer "Dit is regel 2." wordt afgedrukt, wordt het onderste deel van het scherm groter gemaakt, om plaats te maken. Maar tegelijk wordt de nummering van de regels aangepast, zodat de regels hun oorspronkelijke nummers behouden.

Probeer nu dit eens :

```
10 FOR n=0 TO 19: PRINT AT n,0;n;: NEXT n
20 INPUT AT 0,0;a$; AT 1,0;a$; AT 2,0;a$; AT 3,0;a$;
   AT 4,0;a$; AT 5,0;a$;
```

Naarmate het onderste deel van het scherm groter wordt, blijft het bovenste deel zoals het is, tot op het ogenblik dat het onderste deel op dezelfde regel moet gaan PRINTen als het PRINT-commando. Dan gaat het bovenste deel scrollen, om dit conflict te vermijden.

Een extra mogelijkheid van INPUT, die we nog niet hebben besproken, heet LINE input. Dat is een andere manier om string-variabelen in te typen. Indien u LINE gebruikt, voor de naam van een stringvariabele, zoals in:

```
INPUT LINE a$
```

dan drukt de +2 niet, zoals gewoonlijk, de aanhalingstekens af. Toch doet hij alsof ze er wel stonden. Indien u dus als antwoord gewoon het woord "kat" intypt, dan krijgt a$ de inhoud "kat". Doordat de aanhalingstekens er niet staan, kunt u ze ook niet weghalen om als input een of andere string-uitdruking in te typen. Denk er ook aan dat LINE niet gebruikt kan worden voor numerieke variabelen.

INPUT heeft nog een interessant neveneffect. Terwijl u data intypt als antwoord op een INPUT-vraag, komt het oude systeem van de Spectrum voor een kort ogenblik terug tot leven, voordat u het met een druk op ENTER weer het zwijgen op legt. Probeer het volgende programma eens, indien het u interesseert :

```
10 INPUT getallen
20 PRINT getallen
30 GO TO 10
```

Typ een paar getallen in, en ze worden getrouw op het scherm afgedrukt. Druk nu op EXTEND MODE, en dan op de "M". U ziet het woord "PI" verschijnen. Indien u nu ENTER drukt, komt als bij toverslag op het scherm het getal 3.1415927 te staan. Indien u evenwel het woord PI als twee letters ingeeft, zonder de hulp van EXTEND MODE, dan stopt de +2 met de foutmelding :

```
2 Variable not found, 10:1
```

De verklaring voor dit verschijnsel is niet eenvoudig. Onthoud enkel dat het kàn gebeuren, indien u bepaalde toetsen combineert tijdens INPUT. Mocht u willen experimenteren, voor een of andere reden, dan kunt u in hoofdstuk 7 vinden welke toets wat oplevert.

De controlekarakters CHR$ 22 en CHR$ 23 hebben hetzelfde effect als AT en TAB. Indien u die beide kodes laat printen door de +2, moeten ze gevolgd worden door nog twee karakters, die niet het gewone effect hebben, maar in plaats daarvan als getallen worden geïnterpreteerd (hun codes) die de regel en de kolom aangeven (bij AT) of de kolom (bij TAB). Het zal bijna altijd eenvoudiger zijn om gewoon AT en TAB te gebruiken, in plaats van controle-karakters. Hoewel ze soms wel nuttig kunnen zijn. Het controlekarakter voor AT is CHR$ 22. Het eerste karakter erna, geeft de regel aan en het tweede de kolom. Zo heeft het commando

```
PRINT CHR$ 22+CHR$ 1+CHR$ c;
```

precies hetzelfde effect als

```
PRINT AT 1,c;
```

Ook al zouden CHR$ 1 of CHR$ c normaal een andere betekenis hebben (bijvoorbeeld indien c=13), dan wordt die teniet gedaan door het CHR$ 22 ervoor.

Het controlekarakter voor TAB is CHR$ 23, en de twee karakters die erop volgen, geven in combinatie een getal tussen 0 en 65535, dat het kolom-nummer aangeeft zoals u het bij TAB zou gebruiken. Het commando

```
PRINT CHR$ 23+CHR$ a+CHR$ b;
```

heeft hetzelfde resultaat als

```
PRINT TAB a+256*b;
```

Met POKE kunt u de "scroll ?"-vraag uitstellen, door met geregelde tussenpozen te typen :

```
POKE 23692,255
```

Het scherm zal nu 255 regels scrollen vooraleer de vraag "scroll?" verschijnt. Probeer het volgende voorbeeld, en zie de getallenmolen draaien...

```
10 FOR n=0 TO 1000
20 PRINT n: POKE 23692,255
30 NEXT n
```

OEFENING
++++++++

1. Probeer dit programma eens uit met behulp van enkele kinderen, om de tafels van vermenigvuldiging te oefenen :

```
 10 LET m$=""
 20 LET a= INT (RND*12)+1: LET b=INT (RND*12)+1
 30 INPUT (m$)''"Hoeveel is ";(a);" x ";(b);"?";c
100 IF c=a*b THEN LET m$="Juist.": GO TO 20
110 LET m$="Fout. Probeer opnieuw": GO TO 30
```

Als ze slim zijn, kunnen ze ontdekken dat ze zelf niets hoeven uit te rekenen. Indien de +2 hen bijvoorbeeld vraagt naar het resultaat van 2 x 3, dan hoeven ze enkel letterlijk "2*3" in te typen als antwoord.

DEEL 16 : KLEUREN

Inhoud ...
INK, PAPER, FLASH, BRIGHT, INVERSE, OVER
BORDER

Run het volgende programma :

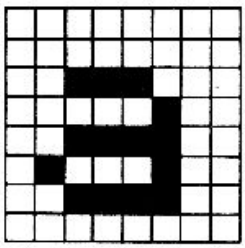
```
10 FOR m=0 TO 1: BRIGHT m
20 FOR n=1 TO 10
30 FOR c=0 TO 7
40 PAPER c: PRINT "    ";: REM 4 gekleurde spaties
50 NEXT c: NEXT n: NEXT m
60 FOR m=0 TO 1: BRIGHT m: PAPER 7
70 FOR c=0 TO 3
80 INC c: PRINT c;"  ";
90 NEXT c: PAPER 0
100 INK c: PRINT c;"  ";
120 NEXT c: NEXT m
130 PAPER 7: INK 0: BRIGHT 0
```

Het programma laat de acht kleuren zien (wit en zwart inbegrepen) en de twee niveaus in helderheid, die de +2 op een kleurentelevisie kan produceren. Indien uw toestel zwart/wit is, dan ziet u alleen een aantal grijstinten. U kunt een gelijkaardig resultaat sneller bereiken door de +2 te resetten, en tegelijk de BREAK-toets in te drukken, maar dat is nogal drastisch. Hieronder volgt een lijstje met de codes die bij de kleuren horen.

0 zwart
1 blauw
2 rood
3 magenta (roze)
4 groen
5 cyaan (lichtblauw)
6 geel
7 wit

Op een zwart/wit-toestel geven de codes de grijstinten in volgorde van helderheid aan. Om deze kleuren goed te kunnen gebruiken, dient u een inzicht te hebben in de structuur van het beeldscherm.

Het beeldscherm bestaat uit 768 posities of cellen (24 regels van 32 kolommen) waarin karakters afgedrukt kunnen worden.



Een voorbeeld van een karaktercel

Elke karaktercel bestaat uit een raster van 8 x 8 (zoals op de tekening). Dit zou u moeten herinneren aan de UDG's in deel 14, waar een 0 een wit vakje en een 1 een zwart vakje voorstelde.

Bij een karakter horen twee kleuren : de INK of voorgrondkleur (dat is de kleur voor de zwarte punten in het raster) en de PAPER of achtergrondkleur (dat is de kleur van de witte punten in het raster). Bij het inschakelen van de computer wordt elke cel ingesteld op zwarte inkt en wit papier, zodat op het scherm alles zwart op wit afgedrukt wordt.

Een karakter kan ook helder of gewoon afgedrukt worden, en het kan knipperen of niet. Dit knipperen ontstaat door voortdurend de kleuren van INK en PAPER om te wisselen. Al die informatie kan in getallen omgezet worden. Bij een karakter hoort dus de volgende informatie :

(1) een raster van 8 x 8 waarin nullen en enen de vorm van het karakter bepalen; een nul geeft PAPER-kleur, een 1 INK-kleur
(2) de kleuren van inkt en papier, beide gecodeerd in een getal van 0 tot 7
(3) een helderheidscode : 0 voor gewoon, 1 voor helder
(4) een "knipper"-code : 0 voor gewoon, 1 voor knipperend

Merk op dat de kleuren van de inkt en het papier gelden voor de hele oppervlakte van de karaktercel. Daarom is het niet mogelijk om in een bepaald blok van 64 punten, méér dan twee kleuren te bepalen. Dat geldt ook voor de helderheid en het knipper-effect. Beide gelden voor de hele karaktercel, en niet voor individuele punten binnen die cel. De kleur, de helderheid en het knipperen van een bepaalde karaktercel noemen we de "attributen".

Door iets op het scherm te printen, wijzigt u in wezen het punten-patroon van die bepaalde karaktercel. Minder voor de hand liggend, maar niet minder waar, is dat u ook de atributen van die cel wijzigt. In het begin merkt u daar niets van, omdat alles zwart op wit wordt afgedrukt, met normale helderheid en niet knipperend. Daar kunt u verandering in brengen, door de INK, PAPER, BRIGHT en FLASH statements te gebruiken. Kies de optie "Screen" in het editmenu, ga naar het onderste deel van het scherm en typ in :

```
PAPER 5
```

en print dan iets op het scherm. U zult merken dat ze op lichtblauw papier worden afgedrukt. Dat komt doordat bij het printen, de papierkleur voor de karaktercellen waarin geprint wordt, lichtblauw wordt gemaakt, dat is code 5.

De rest werkt op dezelfde manier. U kunt dus de volgende commando's geven :

```
PAPER   (geheel getal van 0 tot 7)
INK     (geheel getal van 0 tot 7)
BRIGHT  (geheel getal van 0 tot 1)
FLASH   (geheel getal van 0 tot 1)
```

en elke daaropvolgende print-opdracht zal de overeenkomstige karaktercellen voorzien van de opgegeven attributen.

Probeer een paar van die commando's. U zou nu moeten begrijpen hoe het programma bij het begin van dit deel werkte. Denk eraan, dat een spatie, een karakter is waarvan de inkt en het papier dezelfde kleur hebben.

In die statements kunt u nog twee getallen gebruiken, die een minder direkt zichtbaar effect hebben.

Het getal 8 kan in alle vier de statements worden gebruikt, en betekent "transparant", dat wil zeggen dat het vorige attribuut

zichtbaar blijft. Stel dat u intypt :

```
PAPER 8
```

Geen enkele karakterpositie zal ooit de papierkleur 8 hebben, omdat die kleur niet bestaat. Wat er gebeurt, is dat indien er op een bepaalde positie wordt geprint, de papierkleur die er stond, behouden blijft. INK 8, BRIGHT 8 en FLASH 8 werken op dezelfde manier als voor de overige attributen-getallen.

De code 9 kan enkel met PAPER en INK gebruikt worden. Ze betekent : kontrast. De kleur waarmee u de code 9 gebruikt (inkt of papier) wordt kontrasterend gemaakt met de andere van beide. Indien de andere een donkere kleur heeft (zwart, blauw, rood of magenta) wordt de ene wit, en indien de andere een lichte kleur heeft (groen, cyaan, geel of wit) wordt de ene zwart gemaakt. Probeer dit uit door de volgende instructie :

```
INK 9: FOR c=0 TO 7: PAPER c: PRINT c: NEXT c
```

Nog indrukwekkender kan het, door het programma bij het begin van dit deel te runnen, om gekleurde strepen te maken (zorg ervoor dat u in het onderste schermdeel werkt wanneer u RUN intypt), en daarna te typen :

```
INK 9: PAPER 8: PRINT AT 0,0;: FOR n=1 TO 1000: PRINT n;:
NEXT n
```

Nu wordt de inktkleur voortdurend in kontrast gebracht met de bestaande papierkleur, voor elke karaktercel.

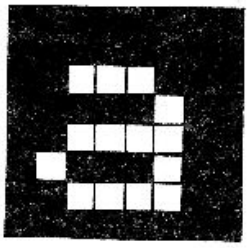
Kleurentelevisie is gebaseerd op het vrij eigenaardige feit dat het menselijk oog eigenlijk maar drie kleuren kan zien : rood, groen en blauw. De overige kleuren zijn mengkleuren van die drie. Magenta bijvoorbeeld, ontstaat door rood met blauw te mengen - daarom is de code van magenta 3, de som van de codes voor rood en blauw.

Om te zien hoe de acht kleuren in elkaar passen, moet u zich drie rechthoekige spots voorstellen, een rode, een groene en een blauwe. Die spots werpen hun licht op een vel wit papier, maar niet precies op dezelfde plaats. Waar de lichtbundels elkaar overlappen, zult u de mengkleuren zien. Dit wordt door het volgende programma nagebootst. Merk op, dat massieve inkt-vakjes ingetypt kunnen worden door in grafische mode te gaan (druk op GRAPH), dan CAPS SHIFT ingedrukt te houden, en op "8" te drukken. Om uit grafische mode te gaan, drukt u op de "9".

```
  10 BORDER 0: PAPER 0: INK 7: CLS
  20 FOR a=1 TO 6
  30 PRINT TAB 6; INK 1;"[redacted]": REM 18
     "volle" vakjes
  40 NEXT a
  50 LET dataregel=200
  60 GO SUB 1000
  70 LET dataregel=210
  80 GO SUB 1000
  90 STOP
1000 FOR a=1 TO 6
1010 RESTORE dataline
1020 FOR b=1 TO 5
1030 READ c: PRINT INK c;"[redacted]";: REM 6
     "volle" vakjes
1040 NEXT b: PRINT: NEXT a
1050 RETURN
```

Er bestaat een functie, ATTR, die berekent wat de attributen op een bepaalde plaats op het scherm zijn. De functie is vrij complex; daarom wordt ze pas op het einde van dit deel besproken.

De twee statements INVERSE en OVER bepalen niet zozeer de attributen als wel het puntjes-patroon dat op het scherm komt. Ze kunnen de codes 0 (uit) en 1 (aan) meekrijgen. Indien u INVERSE 1 gebruikt, dan wordt het gewone patroon van een karaktercel omgekeerd : de punten in het raster die gewoonlijk de papierkleur vertoonden, krijgen nu de inktkleur, en vice versa. De karaktercel die een "a" bevat (zie de vorige tekening) zou er dan zo uitzien :



Indien we dus, zoals bij het inschakelen van de machine, wit papier en zwarte inkt hadden, zou de "a" wit op zwart worden afgedrukt.

Het statement

```
OVER 1
```

schakelt een bepaalde manier van printen in. Gewoonlijk overschrijft iets dat op een bepaalde plaats wordt geprint, datgene wat daar voordien stond. Indien u evenwel OVER 1 gebruikt, dan wordt het nieuwe karakter gewoon gevoegd bij wat er reeds op die plaats stond. Dit kan erg handig zijn om samengestelde karakters te printen, bijvoorbeeld een onderstreepte letter, zoals in het volgende programma. Reset de +2 en kies "128 BASIC". Het streepje wordt geprint door SYMB SHIFT samen met de 0 in te drukken.

```
10 OVER 1
20 PRINT "w"; CHR$8;"_";
```

Merk op dat we het controlekarakter CHR$ 8 heben gebruikt, om de cursor een plaats terug te zetten, alvorens de w met het _ te over-printen.

INK, PAPER enzoverder kunnen nog op een andere manier worden gebruikt. Wellicht zult u die manier nuttiger vinden dan ze enkel als statements te gebruiken. U kunt ze als items in een PRINT-statement gebruiken, gevolgd door een kommapunt. Op die manier doen ze exact hetzelfde als indien u ze als aparte statements gebruikt. Er is één verschil : het effect ervan is tijdelijk, en duurt net zolang tot het PRINT-statement is afgewerkt, waarin ze zijn opgenomen. Indien u dus typt :

```
PRINT PAPER 6;"x";: PRINT "y"
```

dan zal alleen de "x" op geel papier afgedrukt worden.

Wanneer INK enzoverder als statements worden gebruikt, hebben ze geen invloed op de kleuren in het onderste deel van het scherm, waar INPUT wordt ingetypt en foutmeldingen worden afgedrukt. Dat deel van het scherm gebruikt de kleur van de schermrand, de "border" als papierkleur, code 9 voor de inktkleur, knippert niet en is van gewone helderheid. U kunt de kleur van de border wijzigen in één van de acht gewone kleuren (niet 8 of 9) door :

```
BORDER kleur
```

Bij het intypen van INPUT-gegevens wordt in de regel kontrasterende inkt gebruikt. U kunt evenwel de kleur van de tekst wijzigen die u via een INPUT laat afdrukken, door INK en PAPER en zoverder te gebruiken, net zoals in een gewoon PRINT-statement. Het effect ervan duurt ofwel tot op het einde van het statement, ofwel tot INPUT-gegevens worden ingetypt, afhankelijk wat er het eerste komt. Probeer maar :

```
INPUT FLASH 1; INK 1;"Geef een getal op:";n
```

De +2 let goed op uw gezondheid : welke combinatie van kleuren en effecten u ook in een programma weet te verzinnen, de editor gebruikt altijd zwarte inkt op wit papier.

Er is nog een manier om de kleuren te wijzigen : door middel van kontrolekarakters, op dezelfde manier als die voor AT en TAB die we in deel 15 beschreven.

```
CHR$ 16 komt overeen met INK
CHR$ 17 komt overeen met PAPER
CHR$ 18 komt overeen met FLASH
CHR$ 19 komt overeen met BRIGHT
CHR$ 20 komt overeen met INVERSE
CHR$ 21 komt overeen met OVER
```

Deze codes dienen gevolgd te worden door een karakter, dat een kleurcode aangeeft. Zo heeft, bijvoorbeeld,

```
PRINT CHR$ 16+CHR$ 9;"iets"
```

hetzelfde effect als

```
PRINT INK 9;"iets"
```

Over het algemeen zult u die kontrolekarakters niet gebruiken, omdat u net zo goed de statements INK, PAPER enzoverder kunt gebruiken. Het kan evenwel gebeuren, indien u nog oude 48K BASIC programma's op cassette hebt, dat u die codes terugvindt in de listing van het programma. Over het algemeen zal de editor die straal negeren en ze bij de eerste gelegenheid weghalen. Het is niet mogelijk, in tegenstelling tot de oude 48 K Spectrum, om ze in listings aan te brengen.

De functie ATTR heeft de vorm :

```
ATTR (regel,kolom)
```

De twee argumenten van de functie ATTR zijn de regel en de kolom, zoals u die gebruikt bij AT. Het resultaat van de functie, is een getal dat aangeeft welke attributen op die bepaalde positie op het scherm gebruikt werden. U kunt deze functie vrijelijk gebruiken in om het even welke uitdrukking.

Het getal dat de functie ATTR berekent, is de som van vier andere getallen, op deze manier :

```
128 indien de cel knippert, 0 indien niet
 64 indien de cel helder is, 0 indien gewoon
  8 maal de code van de papierkleur
  1 maal de code van de inktkleur
```

Indien bijvoorbeeld een karaktercel knippert, normaal helder is, geel papier en blauwe inkt heeft, dan moeten we de getalen 128, 0, 8*6 en 1 bij elkaar optellen, dat is 177. Probeer maar uit met deze instructie :

```
PRINT AT 0,0; FLASH 1; PAPER 6; INK 1;" ";ATTR (0,0)
```

OEFENINGEN
++++++++++

1. Probeer dit : PRINT "B"; CHR$ 8; OVER 1;"/";

Waar de "/" over de "B" loopt, staat een wit puntje. Op die manier werkt het overprinten op de +2 : twee keer papierkleur of twee keer inktkleur, geeft papierkleur; één van elk geeft inktkleur. Interessant daaraan is, dat indien u nadien nogmaals hetzelfde overprint, u het oorspronkelijke patroon terugkrijgt. Typ nu dit : PRINT CHR$ 8; OVER 1;"/". Waarom wordt de "B" plots terug een gewone "B" ?

2. Run dit programma :

```
10 POKE 22527+RND*704, RND*127
20 GO TO 10
```

(Breek er uw hoofd niet over, hoe het werkt)

Het programma wijzigt de kleuren van karaktercellen op het scherm. Het RND-statement zorgt ervoor dat dit op willekeurige plaatsen gebeurt. De diagonale strepen die u na een tijd begint te onderscheiden zijn een zichtbaar bewijs voor het verborgen patroon in de RND-functie : het is niet écht willekeurig, maar pseudo-willekeurig.

DEEL 17 : GRAFISCH WERK

Inhoud ...
- PLOT, DRAW, CIRCLE
- pixels

Indien u de programma's in dit deel wilt proberen, dient u het programma zelf, eventuele bijkomende commando's en RUN, in het onderste deel van het scherm in te typen. Kies daartoe de optie "Screen" in het editmenu.

In dit deel leren we tekeningen maken met de +2. Het deel van het scherm dat u daarbij kunt gebruiken, telt 22 regels van 32 kolommen, dat is 22*32=704 karaktercellen. U herinnert zich wellicht nog uit deel 16, dat elke karaktercel bestaat uit een raster van 8 x 8 punten. Die punten heten "pixels", van het Engels "picture element", "beeld-element".

Een pixel wordt bepaald door twee getallen, zijn coördinaten. Het eerste getal, het x-coördinaat, geeft de afstand vanaf de linkerkant van het scherm. Het tweede getal, het y-coördinaat, geeft de afstand vanaf de onderkant van het scherm. Gewoonlijk worden deze coördinaten als een paar opgegeven, tussen haakjes. Zo zijn (0,0), (255,0), (0,175) en 255,175) de coördinaten van de vier hoeken van het scherm.

Het statement

```
PLOT x-coördinaat, y-coördinaat
```

zet de pixel op de aangegeven plaats in inkt-kleur. Het volgende korte "mazelen"-programma :

```
10 PLOT INT (RND*256), INT (RND*176): INPUT a$: GO TO 10
```

zet bij elke druk op ENTER, een puntje op een willekeurige plaats op het scherm.

Het volgende programma is interessanter. Het tekent de grafiek van de functie SIN (een sinuscurve) voor alle waarden tussen 0 en 2*pi.

```
10 FOR n=0 TO 255
20 PLOT n,88+80*SIN(n/128*PI)
30 NEXT n
```

En dit programma tekent de grafiek van SQR (een deel van een parabool) tussen 0 en 4 :

```
10 FOR n=0 TO 255
20 PLOT n,80*SQR(n/64)
30 NEXT n
```

Merk op dat pixel-coördinaten verschillen van de regel- en kolomnummers bij AT. Het raster in deel 15 kan u helpen bij het berekenen van pixel-coördinaten of regel- en kolomnummers.

Om uw grafisch werk gemakkelijker te maken, tekent de +2 rechte lijnen, cirkels en cirkelbogen, door middel van de statements DRAW en CIRCLE.

Het statement DRAW om rechte lijnen te tekenen, ziet er zo uit :

```
DRAW x,y
```

De lijn die wordt getekend, start op de plaats waar de laatste pixel door een PLOT, DRAW of CIRCLE-statement werd ingevuld. Die plaats heet "de PLOT-positie". De commando's RUN, CLEAR, CLS en NEW zetten de PLOT-positie op 0,0 : in de hoek linksonder het scherm. De lijn die wordt getekend, stopt op de plaats, x pixels rechts of links van de PLOT-positie en y pixels hoger of lager dan die positie. Het DRAW statement bepaalt dus de lengte en de richting van de lijn, maar niet het startpunt ervan.

Probeer eens een paar PLOT en DRAW-commando's, bijvoorbeeld :

```
PLOT 0,100: DRAW 80,-35
PLOT 90,150: DRAW 80,-35
```

Merk op dat de getallen in een DRAW-statement negatief kunnen zijn, maar getallen in een PLOT-statement niet.

U kunt ook PLOTten en DRAWen in kleur. Toch moet u daarbij letten op het feit dat kleuren altijd een volledige karaktercel vullen, en niet voor bepaalde pixels alleen kunnen gelden. Indien een pixel wordt ingevuld, krijgt hij de geldende inktkleur, en wordt tegelijk de karaktercel waarin die pixel zich bevindt, met diezelfde kleur gevuld. Het volgende programma illustreert dit :

```
10 BORDER 0: PAPER 0: INK 7: CLS: REM scherm zwart maken
20 LET x1=0: LET y1=0: REM start van de lijn
30 LET c=1: REM inktkleur begint als blauw
40 LET x2=INT (RND*256): LET y2=INT (RND*176): REM wille-
   keurige eindpunten van de lijn
50 DRAW INK c;x2-x1,y2-y1
60 LET x1=x2: LET y1=y2: REM volgende lijn begint waar de
   vorige eindigt
70 LET c=c+1: IF C=8 THEN LET c=1: REM nieuwe kleur
80 GO TO 40
```

De lijnen lijken breder te worden naarmate het programma langer loopt. Dit komt doordat een lijn de kleuren wijzigt van alle "1"-pixels van alle karaktercellen die door die lijn worden doorkruist. U kunt ook PAPER, INK, FLASH, BRIGHT, INVERSE en OVER in een PLOT of DRAW-statement gebruiken, net zoals bij PRINT en INPUT. U schrijft ze tussen het keyword en de coördinaten; ze worden van elkaar gescheiden door kommapunten of komma's.

Een extraatje bij DRAW is, dat u met dit commando cirkelbogen kunt tekenen in plaats van rechte lijnen, door een derde parameter toe te voegen, die de hoek aangeeft die de boog omspant. De vorm van het commando is :

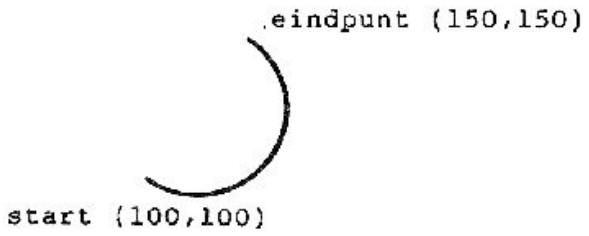
```
DRAW x,y,a
```

X en y geven het eindpunt van de lijn aan, zoals bij het gewone DRAW-statement. A geeft het aantal radialen aan die de boog omspant. Indien a positief is, draait de boog linksom; is a negatief, dan draait de boog rechtsom. U kunt het ook anders bekijken : a geeft aan, welk deel van een cirkel getekend moet worden. Een volledige cirkel is 2*pi radialen, zodat een halve cirkel wordt getekend indien a = pi, een kwart cirkel indien a = pi/2, en zo verder.

Stel dat a = pi. Welke waarden x en y ook aannemen, er zal altijd een halve cirkel getekend worden. Probeer maar :

```
10 PLOT 100,100: DRAW 50,50, PI
```

en u krijgt dit :



De boog start in zuid-oostelijke richting, en eindigt in noord-westelijke richting. Tussendoor is hij 180 graden (dat is pi radialen) gedraaid, dat is dus de waarde van a.

Laat dit programma een aantal keren runnen, en wijzig de waarde van a (PI) in andere waarden, bijvoorbeeld -PI, PI/2, 3*PI/2, PI/4, 1, 0 enzovoort.

Het laatste statement dat hier behandeld wordt, is CIRCLE. Dit commando tekent een volledige cirkel. U geeft de coördinaten op van het middelpunt, en de straal van de cirkel, op deze manier :

```
CIRCLE x-coördinaat, y-coördinaat, straal
```

Ook bij CIRCLE kunt u de diverse kleur-items gebruiken.

De functie POINT laat zien of een pixel in inktkleur dan wel in papierkleur staat. De twee argumenten van de functie zijn de coördinaten van de pixel, die tussen haakjes moeten staan. Het resultaat is 0 indien de pixel in papierkleur staat, of 1 indien hij in inktkleur staat. Probeer dit uit :

```
CLS : PRINT POINT (0,0): PLOT 0,0: PRINT POINT (0,0)
```

Typ nu :

```
PAPER 7: INK 0
```

en laten we nu eens kijken hoe INVERSE en OVER bij een PLOT statement werken. Beide hebben ze enkel invloed op de pixel waar het PLOT statement op slaat, en niet op de rest van de karaktercel. Gewoonlijk zijn deze functies niet werkzaam bij een PLOT-statement. U moet ze dus enkel vermelden indien u ze wilt in werking stellen.

Hier volgt een volledige lijst van de mogelijkheden :

PLOT; - dit is de gebruikelijke vorm. Het zorgt ervoor dat een pixel in inktkleur wordt gezet.

PLOT INVERSE 1; - dit wist een pixel uit, dat wil zeggen dat de pixel in papierkleur wordt getekend.

PLOT OVER 1; - dit wisselt de kleur van een pixel om : indien hij inktkleur had, wordt hij papierkleur en vice versa.

PLOT INVERSE 1; OVER 1; - dit laat de pixel totaal ongemoeid. Het wijzigt wel de PLOT-positie, dus u kunt dit commando daarvoor gebruiken.

Nog een voorbeeld van het gebruik van OVER ziet u, als u eerst een en ander op het scherm laat afdrukken, en dan typt :

```
PLOT 0,0: DRAW OVER 1; 255,175
```

Dit levert een vrij rechte lijn op, met hier en daar een opening, waar de lijn de tekst op het scherm doorkruist. Typ nu nogmaals hetzelfde commando. De lijn verdwijnt zonder een spoor. Dit is het grote voordeel van OVER 1. Had u de lijn getekend met

```
PLOT 0,0: DRAW 255,175
```

en ze uitgewist door

```
PLOT 0,0: DRAW INVERSE 1; 255,175
```

dan was u ook stukjes van de scherminhoud kwijt.

Probeer nu eens :

```
PLOT 0,0: DRAW OVER 1; 250,175
```

en wis die lijn uit door :

```
DRAW OVER 1;-250,-175
```

Dit lukt niet helemaal, doordat de tweede lijn niet precies over dezelfde pixels loopt als de eerste lijn. U moet dus een lijn wissen in dezelfde richting als u ze tekende.

Eén manier om ongewone kleuren te verkrijgen is, twee kleuren te mengen in een UDG. Probeer dit eens :

```
1000 FOR n=0 TO 6 STEP 2
1010 POKE USR "a"+n, BIN 01010101:
     POKE USR "a"+n+1, BIN 10101010
1020 NEXT n
```

Dit programma maakt een UDG met een schaakbord-patroon. Indien u dit karakter op het scherm zet (druk GRAPH en dan "A") in rode inkt op geel papier, dan zult u zien dat het resultaat goed op oranje lijkt.

OEFENINGEN
++++++++++

1. Experimenteer met PAPER, INK, FLASH en BRIGHT in een PLOT-statement. Die items beïnvloeden de hele karaktercel waarin de getekende pixel zich bevindt. Het PLOT-statement zonder meer, is in feite dit :

```
PLOT PAPER 8; FLASH 8; BRIGHT 8; enz.
```

zodat enkel de inktkleur van een karaktercel wordt beïnvloed wanneer in die cel een pixel wordt gePLOT. U kunt dit wijzigen zoals u dat wilt.

Wees wel voorzichtig indien u INVERSE 1 gebruikt bij kleuren : door dit statement krijgt de pixel de papierkleur, en kan de inktkleur veranderen, wat soms onverwachte resultaten oplevert.

2. Probeer cirkels te tekenen door middel van SIN en COS. Indien u deel 10 gelezen hebt, probeer dan zelf te bedenken hoe dat moet. Run dit programma eens :

```
10 FOR n=0 TO 2*PI STEP PI/180
20 PLOT 100+80*COS n,87+80*SIN n
30 NEXT n
40 CIRCLE 150,87,80
```

Dit toont aan dat, hoewel sneller, het CIRCLE-statement minder accuraat werkt.

3. Probeer dit :

```
CIRCLE 100,87,80: DRAW 50,50
```

Dit toont aan dat de PLOT-positie door het CIRCLE-statement op een onvoorspelbare plaats wordt gezet, ongeveer halverwege de rechterkant van de cirkel. Veelal is dus een PLOT-statement nodig, indien u na een CIRCLE verder wilt tekenen.

DEEL 18 : BEWEGING

Inhoud ...
PAUSE, INKEY$, PEEK

Vrij vaak zult u willen dat een programma gedurende een bepaalde tijd wacht; daarvoor is het PAUSE-statement handig te gebruiken.

```
PAUSE n
```

laat de computer ophouden met werken, en nog enkel het tv-beeld weergeven, gedurende de tijd die nodig is om n beelden op het scherm te tekenen (50 beelden per seconde in Europa, 60 in de Verenigde Staten). De waarde van n kan maximaal 65535 zijn. Dat geeft een wachttijd van iets minder dan 22 minuten. Is n=0, dan wil dat voor de computer zeggen "pauzeer een onbepaalde tijd".

Elke pauze kan onderbroken worden door op een willekeurige toets te drukken.

Dit programma tekent de secondewijzer van een klok :

```
 10 REM eerst de wijzerplaat tekenen
 20 FOR n=1 TO 12
 30 PRINT AT 10-10*COS(n/6*PI),16+10*SIN(n/6*PI);n
 40 NEXT n
 50 REM zet de klok in gang
 60 FOR t=0 TO 200000: REM t is de tijd in seconden
 70 LET a=t/30*PI: REM a is de hoek van de seconden-
    wijzer in radialen
 80 LET sx=80*SIN a: LET sy=80*COS a
200 PLOT 128,88: DRAW OVER 1;sx,sy: REM teken sec.wijzer
210 PAUSE 42
220 PLOT 128,88: DRAW OVER 1;sx,sy: REM wis sec. wijzer
400 NEXT t
```

Regel 60 zorgt ervoor, dat de klok na ongeveer 55 en een half uur stopt. U kunt die tijd langer maken als u dat wilt. De gelijkloop wordt gecontroleerd door regel 210. U had wellicht PAUSE 50 verwacht, omdat die precies 1 seconde duurt, maar voor de verwerking van het programma heeft de +2 ook wat tijd nodig, en die moet van de PAUSE-waarde worden afgetrokken. U kunt het beste die waarde uitproberen, door dit programma te laten lopen en de tijd te vergelijken met een echte klok. Pas regel 210 dan aan, tot beide klokken gelijk lopen. Dit kan niet heel precies gebeuren : een wijziging van 1 in de PAUSE-waarde is 2 % of een half uur per dag.

Er bestaat een veel preciezere manier om de tijd te meten. Die manier gebruikt de inhoud van bepaalde geheugenadressen, die kan opgevraagd worden door PEEK. In deel 25 wordt daar dieper op ingegaan. De uitdrukking die gebruikt wordt, is :

```
(65536*PEEK 23674 + 256*PEEK 23673 + PEEK 23672)/50
```

Die uitdrukking berekent het aantal seconden sinds de +2 werd ingeschakeld of geRESET (tot maximum ongeveer 3 dagen en 21 uur, waarna de tijd doortelt vanaf 0).

Dit gewijzigde klok-programma maakt daar gebruik van :

```
 10 REM eerst wijzerplaat tekenen
 20 FOR n=1 TO 12
 30 PRINT AT 10-10*COS(n/6*PI),16+10*SIN(n/6*PI);n
 40 NEXT n
 50 DEF FN t()=INT ((65536*PEEK 23674+256*PEEK 23673+
```

```
    PEEK 23672)/50): REM seconden sinds aanzetten
100 REM zet de klok in gang
110 LET t1= FN t()
120 LET a=t1/30*PI: REM a is de hoek van de seconden-
    wijzer in radialen
130 LET sx=72*SIN a: LET sy=72*COS a
140 PLOT 131,91: DRAW OVER 1;sx,sy: REM wijzer
200 LET t=FN t()
210 IF t<=t1 THEN GO TO 200: REM wacht tot het tijd is
    om de wijzer opnieuw te tekenen
220 PLOT 131,91: DRAW OVER 1;sx,sy: REM wis wijzer uit
230 LET t1=t: GO TO 120
```

De interne klok die hier wordt gebruikt, zou op 0.01 % precies gelijk moeten lopen (ongeveer 10 seconden per dag), zolang de +2 alleen maar een programma verwerkt. Indien u evenwel het BEEP-statement gebruikt (zie deel 19) of de datacorder, de printer of een ander randapparaat, dan stopt de interne klok, waardoor de juiste tijd verloren gaat.

De getallen PEEK 23674, PEEK 23673 en PEEK 23672 worden in de +2 gebruikt om in 50ste seconden te tellen. De drie genoemde getallen liggen tussen 0 en 255, en worden geleidelijk met 1 opgehoogd tot 255, waarna ze weer van 0 af beginnen.

Het getal dat het meest frekwent wordt opgehoogd, is PEEK 23672. Elk 50ste seconden wordt er 1 bijgeteld. Indien het 255 bereikt, wordt het weer 0, maar tegelijk wordt PEEK 23673 met 1 opgehoogd. Wanneer dan (elke 256/50 seconden) PEEK 23673 van 255 naar 0 gaat, wordt PEEK 23674 met 1 opgehoogd. Dit zou voldoende de werking van de hoger gebruikte uitdrukking duidelijk moeten maken.

Let nu goed op. Stel dat de drie genoemde getallen 0 (voor PEEK 23674), 255 (voor PEEK 23673) en 255 (voor PEEK 23672) zijn. Dat wil zeggen, ongeveer 21 minuten na het inschakelen van de computer. De genoemde uitdrukking zou dan als resultaat moeten geven :

```
(65536*0 + 256*255 + 255)/50 = 1310.7
```

Maar daar zit een addertje onder het gras : de volgende keer dat er een 50ste seconde wordt geteld, worden de drie getallen dus 1, 0 en 0. Die wijziging kan zich voltrekken, terwijl de computer de uitdrukking aan het evalueren is. PEEK 23674 zou dan 0 opleveren, maar nog voordat de +2 kon PEEKen, zou hij de andere twee ook 0 maken, zodat het resultaat zou worden :

```
(65536*0 + 256*0 +0)/50 = 0.
```

wat duidelijk fout is.

Een eenvoudige manier om dit probleem te omzeilen is, de uitdrukking twee keer uit te werken en het grootste resultaat van beide bewerkingen als eindresultaat te nemen.

Indien u het vorige programma goed bestudeert, zult u merken dat dit impliciet gebeurt.

U kunt ook deze truc gebruiken, om de regel toe te passen. Definieer functies als volgt :

```
10 DEF FN m(x,y)=(x+y+ABS(x-y))/2: REM grootste van twee
20 DEF FN u()=(65536*PEEK 23674+256*PEEK 23673+PEEK
   23672)/50: REM tijd (kan fout zijn)
30 DEF FN t()=FN m(FN u(), FN u()): REM tijd (juist)
```

U kunt ook de drie getallen van de interne klok wijzigen, zodanig dat ze de reële tijd aangeven in plaats van de tijd sinds het inschakelen van de +2. Om bijvoorbeeld de klok op 10:00 uur te zetten, rekent u eerst uit dat dit gelijk is aan : 10x60x60x50 = 1800000 50ste seconden, en dat

```
1800000 = 65536*27 + 256*119 + 64
```

Om de drie getallen nu in te vullen als 27, 119 en 64, typt u :

```
POKE 23674,27: POKE 23673,119: POKE 23672,64
```

In landen met een netfrekwentie van 60 Hz moet in alle programma's "50" door "60" worden vervangen waar dat nodig is.

De functie INKEY$ (die geen argument heeft) leest het toetsenbord uit. Indien u 1 toets indrukt (of CAPS SHIFT en een andere toets), dan is het resultaat van de functie INKEY$, het karakter dat die toets gewoonlijk oplevert bij indrukken. Drukt u op geen enkele toets, dan is het resultaat de nulstring.

Probeer dit programma eens. Het werkt zoals een schrijfmachine.

```
10 IF INKEY$<>"" THEN GO TO 10
20 IF INKEY$="" THEN GO TO 20
30 PRINT INKEY$;
40 GO TO 10
```

Regel 10 wacht tot u de toetsen loslaat, en regel 20 wacht tot u op een nieuwe toets drukt.

Denk eraan dat, in tegenstelling tot INPUT, INKEY$ niet wacht tot u een toets indrukt. U moet dus niet op ENTER drukken. Anderzijds, als u op geen enkele toets drukt, hebt u uw kans verspeeld.

OEFENINGEN

1. Wat gebeurt er indien u regel 10 in het "schrijfmachine"-programma weglaat ?

2. Door gebruik van INKEY$ in combinatie met PAUSE, kunt u een tweede schrijfmachine construeren :

```
10 PAUSE 0
20 PRINT INKEY$;
30 GO TO 10
```

Opdat dit zou werken is het essentieel dat een pauze niet onderbroken wordt indien bij de start ervan, reeds op een toets wordt gedrukt. Waarom ?

3. Breid het klok-programma uit, zodat ook om de minuut een minutenwijzer en een uur-wijzer wordt getekend. Hebt u nog meer moed, probeer het dan zo te maken, dat er om het kwartier iets gebeurt - bijvoorbeeld dat het Big Ben-melodietje wordt ten gehore gebracht. Dat kunt u met het PLAY-statement maken (zie deel 19).

DEEL 19 : GELUID

Inhoud ...
BEEP, PLAY

U zult ondertussen al gemerkt hebben, dat de +2 diverse geluiden kan voortbrengen. Voor een goede geluidskwaliteit dient uw tv-toestel optimaal afgestemd te staan (zie hoofdstuk 2). Indien u in plaats van een tv-toestel, een videomonitor gebruikt (die het geluid van de +2 niet weergeeft), dan staat het geluidssignaal ter beschikking op de SOUND-aansluiting achteraan de +2. Dit signaal kunt u aansluiten op een geluidsversterker. U mag niet rechtstreeks een koptelefoon aansluiten op deze uitgang.

In hoofdstuk 10 staat beschreven hoe u aansluitingen op de SOUND-uitgang kunt maken.

Om de muzikale talenten van de +2 ten volle te gebruiken, is het wel handig dat u enige muzikale termen kent.

Let op : in de volgende voorbeelden is het van belang, dat u de string-uitdrukingen precies zo intypt als ze in dit boek staan, met behoud van hoofd- en kleine letters. De string "ga" mag bijvoorbeeld niet als "Ga", "gA" of "GA" worden ingetypt.

Typ dit commando in (maak u voorlopig geen zorgen over de betekenis ervan) :

```
PLAY "ga"
```

U hoorde twee noten, de tweede noot iets hoger dan de eerste. Het verschil tussen de twee noten heet een "toon". Probeer nu :

```
PLAY "g$a"
```

U hoorde opnieuw twee noten. De eerste klonk hetzelfde als in het eerste voorbeeld, maar de tweede noot was niet meer zoveel hoger dan de eerste, als in het vorige voorbeeld. Als u het verschil niet goed hoorde, probeer dan beide voorbeelden eens vlak na elkaar uit. In het tweede voorbeeld liggen de twee noten een "halve toon" uit elkaar.

Als u genoeg hebt van de halve tonen, probeer dit dan :

```
PLAY "gD"
```

Het verschil tussen deze twee noten heet een "quint". U hoort het vaak in allerlei muziek. Typ nu :

```
PLAY "gG"
```

Hopelijk hoorde u meer verschil tussen de twee noten, dan in het vorige voorbeeld. Maar toch klonken beide noten ongeveer gelijk. Dit verschil heet een "octaaf". Op dit punt begint de muziek "zichzelf te herhalen". Maak u daar verder niet teveel zorgen over, en probeer enkel te onthouden hoe een octaaf klinkt.

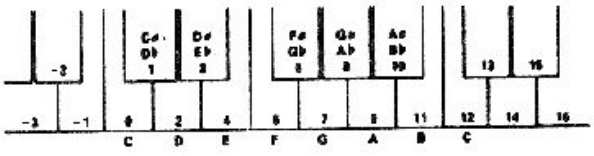
Met de +2 kunt u op twee manieren geluid en muziek maken. De simpelste manier is : het nogal Spartaanse commando BEEP. Dit wordt geschreven in de vorm :

```
BEEP lengte, hoogte
```

Zoals gewoonlijk stellen "lengte" en "hoogte" numerieke uitdrukkingen voor. De lengte wordt opgegeven in seconden. De toonhoogte wordt opgegeven in halftonen boven de middelste do (C),

waarbij negatieve getallen, noten aangeven onder die do (C).

Op het onderstaande diagram staan de toonhoogtes voor alle noten binnen 1 octaaf, voor het BEEP-commando.



Om bijvoorbeeld de A (la) boven de middenste do, een halve seconde lang te laten klinken, typt u :

```
BEEP 0.5,9
```

en om een toonladder (bijvoorbeeld C (do) majeur) te laten horen, is een kompleet programma nodig, ook al is het niet zo lang :

```
10 FOR f=1 TO 8
20 READ noot
30 BEEP 0.5,noot
40 NEXT f
50 DATA 0,2,4,5,7,9,11,12
```

Om hogere of lagere tonen te horen, trekt u 12 af of telt u 12 op bij de toonhoogte, per octaaf dat u hoger of lager wilt.

BEEP is enkel voorzien om nog te kunnen werken met programma's voor de oudere Spectrums. Het kan ook nog gebruikt worden voor korte of snelle geluidseffecten. Wanneer u nieuwe programma's ontwikkelt, kunt u beter de tweede manier gebruiken, om geluid te maken : met het commando PLAY. Als u de vorbeelden bij het begin van dit deel hebt uitgeprobeerd, dan herinnert u zich nog wel dat we dit commando gebruikten.

PLAY is veel flexibeler dan BEEP. Het laat toe om driestemmig geluid te maken, met een hele reeks geluidseffecten erbij. De geluidskwaliteit is veel beter, en het commando is ook stukken gemakkelijker te gebruiken. Om bijvoorbeeld A (la) boven de middelste C (do) voor een halve seconde te laten klinken, typt u gewoon :

```
PLAY "a"
```

en om de toonladder in C (do) majeur te spelen, waar bij BEEP een heel programma nodig was, typt u :

```
PLAY "cdefgabC"
```

Merk op dat de laatste C in dit voorbeeld, een hoofdletter is. Daardoor weet het PLAY-commando dat ze een octaaf hoger moet klinken dan de c met kleine letter. Overigens, een toonladder is een groep noten die precies een octaaf bestrijken. Het voorbeeld hierboven geeft de toonladder in C (do) majeur, omdat het de groep noten is tussen twee c's. Waarom "majeur" ? Er bestaan twee soorten van toonladders, majeur en mineur. Dat is gewoon een muzikale notatie om twee groepen van noten te omschrijven. Ter illustratie, de toonladder in C (do) mineur klinkt zo :

```
PLAY "cd$efg$a$bC"
```

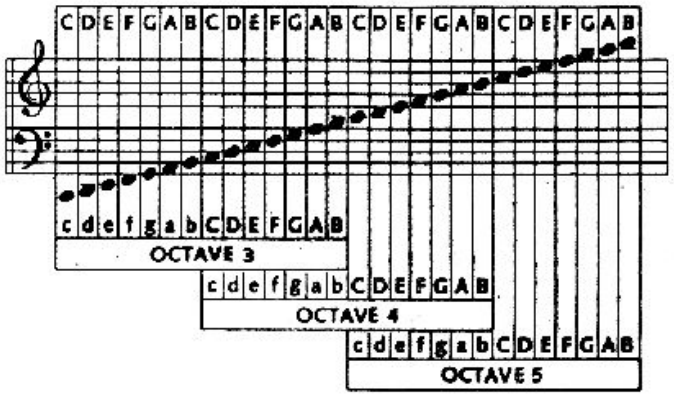
Een $-teken voor een noot, maakt ze een halve toon lager (verlaagt ze), terwijl een #-teken ervoor de noot een halve toon hoger maakt (ze verhoogt). Het PLAY-commando bestrijkt 9 octaven. U geeft op, welk octaaf als basis gebruikt wordt, door de hoofdletter O, gevolgd door een getal, op te nemen in de string waarin de noten worden geschreven. Probeer dit eens :

```
10 LET o$="O5"
20 LET n$="DECcg"
30 LET a$=o$+n$
40 PLAY a$
```

In dit programma vallen een aantal nieuwe dingen op. Ten eerste blijkt PLAY net zo goed te werken met string-variabelen als met string-constanten. Met andere woorden, PLAY a$ werkt net zo goed als PLAY "O5DECcg", in de veronderstelling dat a$ vooraf werd ingevuld. Meer nog, het gebruik van variabelen bij PLAY heeft duidelijke voordelen. Daarom doen we dit voortaan altijd.

Merk ook op dat de string a$ opgebouwd werd uit twee kleinere strings, o$ en n$. Op dit niveau heeft dit niet zoveel belang, maar PLAY kan strings aan, die uit duizenden karakters bestaan. De meest verstandige manier om dergelijke lange strings samen te stellen en te wijzigen in BASIC, is : diverse kleine strings combineren.

Run nu het bovenstaande programma. Wijzig regel 10, zodat "O5" nu "O7" wordt, en run het nogmaals of, indien u een groot ruimteschip wilt horen, zet "O2" in de plaats. Als u geen octaaf-nummer opgeeft, dan neemt de +2 aan dat u octaaf 5 wilt. Hieronder ziet u een diagram met de noten en de octaaf-nummers, die overeenkomen met de standaard gelijkzwevende toonladder.



Er is nogal wat overlapping : "O3D" is hetzelfde als "O4d". Dat maakt het schrijven van muziek makkelijker, omdat u dan niet de hele tijd van octaaf moet veranderen. Om technische redenen kunnen bepaalde noten in de laagste octaven (0 en 1) niet erg nauwkeurig worden geproduceerd. De +2 probeert die tonen zo goed als mogelijk te benaderen.

PLAY kan ook verschillende lengtes van noten aan. Wijzig regel 10 van het vorige programma in

```
10 LET o$="2"
```

en run het programma. Wijzig daarna de inhoud van o$ tussen "1" en "9". De lengte van een noot kan op elke plaats in de string

worden gewijzigd, door een getal tussen 1 en 9 in de string op te nemen. Die waarde geldt dan voor alle volgende noten, tot opnieuw een getal wordt gelezen. Elk van die negen lengtes heeft in de muziek een bepaalde naam, en ziet er ook anders uit indien ze op papier wordt gezet in muzikale notatie. De volgende tabel maakt het duidelijker :

getal	naam	symbool
1	zestiende noot	
2	zestiende punt	
3	achtste noot	
4	achtste punt	
5	kwartnoot	
6	kwart punt	
7	halve noot	
8	halve noot punt	
9	hele noot	

PLAY kan ook "triolen" verwerken, drie noten die in de tijd van twee worden gespeeld. In tegenstelling tot de regel voor gewone noten, geldt de code die de lengte van een triool aangeeft, enkel voor de drie daarop volgende noten. Daarna wordt de vorige lengte weer aangehouden. De codes voor triolen zijn als volgt :

getal	triool van	symbool
10	zestiende noten	
11	achtste noten	
12	kwartnoten	

PLAY kan ook tijdelijk tot zwijgen worden gebracht. De periode waarin geen noot wordt gespeeld, heet "rust". Die wordt aangegeven door het &-teken. De lengte van de rust wordt bepaald door de lengte van de noten op dat ogenblik. Wijzig, ter demonstratie, regels 10 en 20 zo :

```
10 LET o$="O4"
20 LET n$="DEC&cg"
```

Twee noten die aan elkaar moeten worden gespeeld, zonder onderbreking ertussen, heten "verbonden noten". In het PLAY-commando wordt dit aangegeven door een "_". Een kwart en een halve C (do) aan elkaar schrijft u dus zo : "5_7c". De tweede waarde wordt dan verder als lengte aangehouden.

Soms kan het onduidelijk worden. Stel dat u in een muziekstuk octaaf nummer 6 en lengte 2 nodig hebt. U denkt misschien dat :

```
10 LET o$="O62"
```

het gewenste effect zal hebben. Toch is dat niet zo. De computer ziet de O en probeert dan het erop volgende getal te lezen. Hij leest "62" en stopt met de foutmelding "n Out of range" (valt buiten het bereik). In deze situaties gebruikt u een "loze" noot, N, die alleen maar dient als scheidingsteken. Regel 10 moet dus worden :

```
10 LET o$="O6N2"
```

Het volume kan ingesteld worden tussen 0 (minimum) en 15 (maximum), door de letter "V" gevolgd door een getal. In de praktijk zullen enkel de getallen 10 tot 15 nut hebben. De volumes 1 tot 9 klinken te zacht, tenzij de +2 aan een versterker is gekoppeld. Zoals reeds gezegd, klinkt BEEP luider dan één kanaal van PLAY. Indien alledrie de kanalen op volume 15 een toon weergeven, klinkt dit even luid als een BEEP-toon.

Meer dan één kanaal tegelijk aansturen doet u, door gewoon een komma tussen de strings voor elk kanaal te zetten. Probeer dit :

```
10 LET a$="O4cCcCgGgG"
20 LET b$="O6CaCe$bd$bD"
30 PLAY a$,b$
```

Over het algemeen is er geen enkel verschil tussen de drie kanalen. Elke string kan door elk kanaal verwerkt worden. De algemene snelheid van de muziek, het tempo, moet evenwel in de string staan, die door kanaal A wordt verwerkt (de string die volgt op het PLAY-commando) zoniet wordt hij genegeerd. Om het tempo aan te geven (in kwartnoten per minuut), gebruikt u "T" gevolgd door een getal tussen 60 en 240. De standaardwarde is 120, dat wil zeggen twee kwartnoten per seconde. Wijzig het programma zo :

```
 5 LET t$="T120"
10 LET a$=t$+"O4cCcCgGgG"
20 LET b$="O6CaCe$bd$bD"
30 PLAY a$,b$
```

en laat het een paar keer runnen, met telkens een andere waarde op regel 5, voor verschillende tempi.

In een muziekstukje komt het vaak voor dat een groep noten herhaald dient te worden. Een willekeurig gedeelte van een string wordt herhaald door het tussen haakjes te zetten. Indien u dus regel wijzigt in :

```
10 LET a$=t$+"O4(cC)(gG)"
```

dan zal die string even goed door PLAY worden verwerkt. Indien u een sluit-haakje in de string zet, zonder een bijbehorend open-haakje, dan wordt de string tot op dat punt continu herhaald. Dit kan handig zijn voor ritme-effecten en begeleiding. Typ, ter demonstratie, het volgende in (om te stoppen moet u BREAK drukken) :

```
PLAY "O4N2cdefgfed)"
```

en dan

```
PLAY "O4N2cd(efgf)ed)"
```

Indien u een repeterende begeleiding laat horen, en daarna een melodie, zou het wel mooi zijn als de begeleiding stopte wanneer de melodie ten einde is. Ook daarin is voorzien. Indien PLAY een "H" leest ("Halt") in om het even welke string die hij verwerkt,

dan stopt hij direkt alle geluid. Laat het volgende programma runnen (u dient weer BREAK te drukken om het te stoppen) :

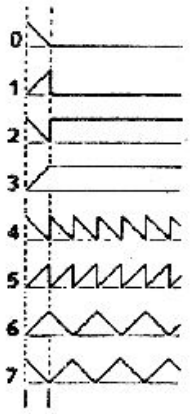
```
10 LET a$="cegbdfac"
20 LET b$="O4cC)"
30 PLAY a$,b$
```

Wijzig nu regel 10 in :

```
10 LET a$="cegbdfaCH"
```

en run opnieuw.

Tot hiertoe hebben we enkel noten gebruikt die beginnen en eindigen met hetzelfde volume. De +2 kan ook het volume van een noot veranderen terwijl die noot weerklinkt. Het geluid kan bijvoorbeeld luid beginnen en wegsterven, zoals bij een piano, of aanzwellen en verminderen, zoals een huilende hond. Om deze effecten te bereiken, gebruikt u "W", gevolgd door een getal tussen 0 en 7, samen met "U" voor elk kanaal waarop dat effect van toepassing moet zijn. Een kanaal waarvan het volume werd ingesteld met "V", negeert de "U". Onderstaande tabel geeft grafisch het verloop weer van het volume, voor elke code die bij "W" gebruikt kan worden.



0 1x wegsterven en uit
1 1x aanzetten en uit
2 1x wegsterven en aanhouden
3 1x aanzetten en aanhouden
4 herhaald wegsterven
5 herhaald aanzetten
6 herhaald aanzetten/wegsterven
7 herhaald wegsterven/aanzetten

Dit programma laat dezelfde noot horen, met alle mogelijke effecten na elkaar. Zo kunt u het effect vergelijken met het bovenstaande diagram.

```
10 LET a$="UX1000W0C&W1C&W2C&W3C&W4C&W5C&W6C&W7C"
20 PLAY a$
```

De "U" schakelt de effecten in, en de "W" kiest één bepaald effect. De string bevat ook "X1000". Met "X" regelt u hoe lang het effect duurt (tussen 0 en 65535). Indien u de "X" weglaat, dan neemt de +2 de langste duur. Golfvormen die op een bepaald niveau stabiliseren (0 tot 3 in de tabel) werken het beste met een X-waarde rond 1000. Repeterende effecten (4 tot 7) hebben meer resultaat met kortere tijden, rond 300. Probeer eens verschillende X-waarden in het bovenstaande programma, om een idee te krijgen van het resultaat.

Het PLAY-commando kan meer dan louter muziek spelen. U beschikt ook over drie "witte ruis"-generatoren. Witte ruis is het geluid dat u hoort, indien u de radio tussen twee FM-stations in, afstemt. Elk van de drie kanalen kan noten, witte ruis of een mengsel van de twee voortbrengen. Om een mengsel te verkrijgen, gebruikt u "M", gevolgd door een getal van 1 tot 63. Dat getal kunt u berekenen aan de hand van de volgende tabel.

	Toonkanaal			Ruiskanaal		
	A	B	C	A	B	C
Getal	1	2	4	8	16	32

Schrijf de getallen op die horen bij het effect dat u wilt, en tel ze op. Indien u bijvoorbeeld kanaal A als ruiskanaal, kanaal B als muziekkanaal en kanaal C als meng-kanaal wilt gebruiken, tel dan 8,2,4 en 32 bij elkaar op, en u krijgt 46. De volgorde van de kanalen is dezelfde als de volgorde van de strings die volgen na het PLAY-commando. Met het A-kanaal kunt u de beste effecten bereiken. Probeer gerust allerlei uit.

U zult nu wel al symfonieën kunnen schrijven. Het kan soms wel moeilijk worden, om naderhand uit te zoeken welk stuk muziek door een bepaald gedeelte van een string wordt voortgebracht. Om dit probleem wat minder lastig te maken, kunt u in een muziek-string "commentaar" opnemen, tussen twee uitroeptekens, bv. :

```
1098 LET z$=z$+"CDcE3Ge4_6f!eind van de 75ste maat!egeA"
```

Het PLAY-commando slaat het commentaar gewoon over.

Indien u over een electronisch muziekinstrument beschikt, met MIDI, dan kan de +2 dat besturen door middel van het PLAY-commando. Tot 8 muziekkanalen kunnen naar synthesisers, drums of sequencers worden gestuurd. Het PLAY-commando ziet er uit zoals gewoonlijk, alleen moet nu elke string een "Y" bevatten, gevolgd door een getal tusen 1 en 16. Dit getal bepaalt aan welk kanaal de muziek-data toegewezen wordt. U kunt acht strings gebruiken. De eerste drie worden nog wel naar de tv gestuurd, dus u kunt beter het geluid uitzetten. U kunt ook programmeercodes voor de MIDI doorsturen met het PLAY-commando, door middel van "Z", gevolgd door een code. Toets-snelheden (volume) worden berekend en doorgestuurd tegen 8 keer de "V"-waarde. "V6" wordt dus als 48 doorgeseind.

Tot slot volgt hier een overzicht van de parameters die in een PLAY-string gebruikt kunnen worden, samen met de waarde die ze kunnen aannemen.

karakter functie

a-g	toonhoogte van een noot binnen een octaaf
A-G	idem
$	verlagen van de volgende noot
#	verhogen van de volgende noot
O	octaafnummer (gevolgd door 0-8)
1-12	lengte van de noot
&	rust
	gebonden noot
N	scheidt twee getallen
V	volume-waarde (gevolgd door 0-15)
W	golfvorm van het geluid (gevolgd door 0-7)
U	schakelt de effecten in
X	geeft lengte aan van effecten (gevolgd door 0-65535)
T	geeft tempo van muziek aan (gevolgd door 60-240)
()	ingesloten frase moet herhaald worden
!	ingesloten commentaar moet overgeslagen worden
H	stopt het PLAY-commando, alle kanalen
M	geeft te gebruiken kanaal/kanalen aan (gevolgd door 1-63)
Y	geeft aan dat het MIDI-kanaal gebruikt wordt (gevolgd door 1-16)
Z	signaleert een programmeercode voor de MIDI (gevolgd door een codegetal)

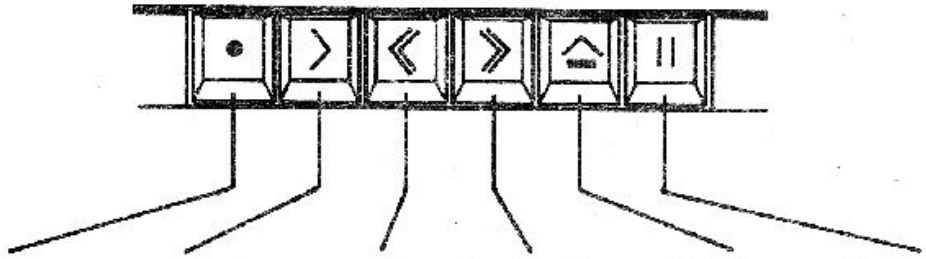
DEEL 20 : DE DATACORDER

Inhoud ...
LOAD, SAVE, VERIFY, MERGE

De eenvoudigste manier om de datacorder te gebruiken, om software in te laden, wordt in hoofdstuk 3 en 4 beschreven.

U kunt de datacorder ook gebruiken om uw eigen programma's op cassette op te slaan (te "saven"), zodat u ze later opnieuw in het geheugen kunt inlezen ("loaden"). Zonder die mogelijkheid zou u elke keer opnieuw het programma moeten intypen, als u het wilde gebruiken.

Maak uzelf eerst vertrouwd met de zes toetsen op de datacorder :



opnemen weergeven terugspoelen opspoelen stop/cas.uit pauze

Om te zien hoe de datacorder een programma opslaat, dient u eerst dit korte programma in te typen dat u op het einde van deel 16 al zag. Het zet gekleurde blokjes op het scherm :

```
10 POKE 22527+RND*704, RND*127
20 GO TO 10
```

Dit programma gaan we nu op cassette zetten. Elke cassette zou goed moeten zijn, hoewel de "Low Noise"-soort de beste is.

Typ nu in :

```
SAVE"blokjes"
```

"blokjes" is gewoon een willekeurige naam, om later het programma terug te kunnen vinden. Die naam kan maximum tien karakters bevatten.

De +2 zet deze boodschap op het scherm :

Press REC & PLAY, then any key.

(druk REC en PLAY in, en druk daarna op een toets)

We zullen eerst een proef doen, zodat u kunt zien wat er gebeurt wanneer we écht een programma saven. Druk daarom deze keer niét de REC en PLAY-toetsen in, maar druk alleen op een toets van de computer, bijvorbeeld ENTER, en let goed op de rand van het tv-scherm. U zult verschillende gekleurde patronen zien, in deze volgorde :

- vijf seconden lang, rode en lichtblauwe strepen die langzaam naar boven toe bewegen, gevolgd door een kort stukje met gele en donkerblauwe strepen

- een korte pauze

- twee seconden lang, opnieuw rode en lichtblauwe strepen, gevolgd door nog een kort stukje geel en blauw.

Terwijl u de strepen op het scherm ziet, kunt u ook de data "horen" via de luidspreker.

Probeer dit uit, net zolang tot u die patronen kunt herkennen. De informatie wordt op cassette gezet in twee blokken. Beide blokken worden voorafgegaan door een aanlooptoon (de rode en lichtblauwe strepen). Dan pas komt de eigenlijke informatie (de gele en blauwe strepen). Het eerste blok schrijft de naam en diverse informatie over het programma op de band, en het tweede blok bevat het eigenlijke programma samen met eventueel aanwezige variabelen. De pauze tussendoor heeft geen doel.

Nu gaan we het programma écht saven.

1. Spoel de cassette zover, dat er een blanco stuk band klaar staat, of een stukje dat overschreven mag worden.

2. Typ het commando : SAVE"blokjes"

3. Volg de aanwijzingen van de computer

4. Kijk naar het scherm. Wanneer de +2 klaar is (dan print hij "0 OK" op het scherm), stopt u de datacorder.

Wanneer u nu met goed gevolg een programma hebt gesaved, kunt u met een gerust gemoed de computer uitschakelen of resetten, of na NEW een volgend programma intypen : u kunt het zojuist weggeschreven programma altijd terug in de computer laden, als u dat wenst. Vooraleer u evenwel het programma dat u wegschreef, uit het geheugen wist, is het sterk aan te raden om na te gaan of het wel foutloos op de band werd gezet. U kunt controleren of het signaal op de cassette precies overeenkomt met de geheugeninhoud, door het commando "VERIFY" :

1. Spoel de cassette terug, tot ze staat zoals bij het begin van het SAVE-commando

2. Typ in : VERIFY "blokjes"

De rand van het scherm zal nu afwisselend rood en lichtblauw knipperen, totdat de +2 het programma vindt waarvan u de naam opgaf. Daarna ziet u op het scherm hetzelfde patroon als tijdens het saven. Tijdens de pauze tussen de twee blokken, wordt op het scherm de boodschap : "Program: blokjes" afgedrukt. Indien u de +2 iets op cassette laat zoeken, geeft hij de naam op van alles wat hij op die cassette leest. Wanneer, na het tweede blokje geel/blauwe strepen, de boodschap "0 OK" wordt geprint, dan wil dat zeggen dat uw programma veilig en wel op band staat, en dan kunt u de volgende vijf paragrafen overslaan. In het andere geval is er iets fout gelopen. Volg deze procedure om uit te zoeken wát er foutliep :

Wordt de naam van het programma niet op het scherm gezet, dan zijn er twee mogelijkheden. Ofwel was het programma niet op de band gezet zoals het hoort, ofwel werd het niet correct "teruggelezen". Om uit te zoeken wat er aan de hand is, spoelt u de band terug tot waar u begon te saven, en speelt u de cassette af terwijl u naar het geluid luistert. De rood/lichtblauwe strepen moeten gepaard gaan met een heldere hoge toon, en de blauw/gele strepen met een minder aangenaam raspend geluid.

Hoort u deze geluiden niet, dan werd het programma vermoedelijk niet op band gezet. Ga na of u niet probeerde, het programma op de plastic aanloopstrook van de cassette te saven. Probeer nu nog eens te saven.

Indien u de geluiden hoort, dan was het saven waarschijnlijk in orde, en ligt het probleem bij het inlezen.

Het is mogelijk dat u de naam van het programma verkeerd intypte, bij het saven. In dat geval zal de +2 die naam op het scherm zetten, wanneer hij hem op de band vindt. U kunt ook de naam verkeerd getypt hebben, in het VERIFY-commando. In dat geval zal de +2 het goed weggeschreven programma negeren, en op de band verder zoeken naar de foute naam, waarbij de rand van het scherm rood en lichtblauw blijft knipperen.

Staat er echt iets fouts op de cassette, dan geeft de +2 de boodschap : "R Tape loading error". In dit geval betekent die boodschap dat de geheugeninhoud niet overeenstemt met wat er op de band staat. Merk op dat een klein foutje in de band zelf, dat misschien bij muziek niet eens hoorbaar is, voor een computer-programma fataal kan zijn. Probeer het programma nogmaals te saven, misschien op een ander stuk van de band.

Laten we nu aannemen dat u het programma met goed gevolg op de cassette hebt gezet, en geverifieerd. Terug in het geheugen laden werkt precies zoals verifiëren, alleen typt u nu :

```
LOAD"blokjes"
```

in plaats van VERIFY "blokjes".

Aangezien u geverifieerd hebt, zou u geen problemen mogen hebben met het inladen.

Het commando LOAD wist een programma en de variabelen uit, die in het geheugen zitten, wanneer een nieuw programma van cassette wordt ingeladen.

Na het inladen van een programma, kan het gestart worden door het commando RUN.

In hoofdstuk 3 en 4 zegden we al dat u programma's op cassette (software noemen we dat) kunt kopen. Let er op dat wat u koopt, speciaal voor de ZX Spectrum-gamma geschreven werd (de Spectrum, de Spectrum +, de Spectrum 128 of de Spectrum +2). Verschillende merken van computers gebruiken verschillende manieren om programma's weg te schrijven. Ze kunnen dus elkaars cassettes niét lezen.

Indien er op uw cassette meerdere programma's staan, dan hebben die allemaal een naam. U kunt beslissen welk programma ingeladen zal worden, door de naam die u in het LOAD-commando opgeeft. Indien u bijvoorbeeld een programma wilt inladen, dat "helicopter" heet, dan typt u :

```
LOAD"helicopter"
```

Het commando LOAD "" betekent dat de +2 het eerste programma dat hij op de cassette vindt, in het geheugen zal laden. Dit kan van pas komen indien u niet meer weet onder welke naam u een bepaald programma hebt weggeschreven.

De optie "Tape Loader" in het startmenu, doet precies hetzelfde als LOAD "" en is veel eenvoudiger te bedienen : schakel de +2 in en druk op ENTER.

We zegden al dat LOAD een aanwezig programma met eventuele variabelen, uit het geheugen wist bij het inladen van een ander programma. Er bestaat evenwel nog een commando, MERGE, dat op LOAD lijkt maar dat enkel een programmaregel of een variabele

wist, indien de +2 op de cassette een regel leest met hetzelfde nummer, of een variabele met dezelfde naam. Typ het teerling-programma uit deel 11 in, en zet het op cassette onder de naam "teerling". Typ nu het volgende in en run het :

```
 1 PRINT 1
 2 PRINT 2
10 PRINT 10
20 LET x=20
```

Spoel de cassette terug, klaar om "teerling" in te laden, en typ

```
MERGE"teerling"
```

Doe nu precies alsof u het programma inlaadde. Indien u nu een LIST opvraagt, zult u merken dat regels 1 en 2 nog in het geheugen zitten, en dat regels 10 en 20 overschreven werden door de regels met dezelfde nummers uit het programma op de band. De variabele x bestaat eveneens nog (vraag PRINT x).

U kent nu de eenvoudige vorm van de vier commando's die met de datacorder te maken hebben :

SAVE - zet programma plus variabelen op cassette

VERIFY - vergelijkt wat op de cassette staat met de inhoud van het geheugen

LOAD - wist programma en variabelen uit het geheugen en vervangt die door wat op de cassette staat

MERGE - zoals LOAD, maar wist niets tenzij het moet, omdat hetzelfde regelnummer of dezelfde variabele wordt ingelezen.

Bij deze vier commando's wordt het keyword gevolgd door een string. Bij het SAVE-commando bevat deze string de naam waaronder een programma op cassette wordt bewaard. Bij de andere drie commando's weet de computer door de inhoud van de string, wat hij moet zoeken op de band. Tijdens dat zoekproces, drukt de +2 de namen af van elk programma dat hij op de band leest. Er zijn ook een paar afwijkingen mogelijk :

Bij VERIFY, LOAD en MERGE kunt u de nulstring opgeven. Dan neemt de +2 gewoon het eerste programma dat hij vindt op de cassette.

Een variante op het SAVE-commando is :

```
SAVE string LINE getal
```

Een programma dat op deze manier werd op band gezet, wordt zo opgeslagen dat bij het inladen met LOAD (maar niet met MERGE) het programma automatisch begint te lopen op het aangegeven regelnummer. Dit staat bekend als "auto-run".

Tot hier toe hebben we enkel programma's met de bijbehorende variabelen op cassette gezet. We kunnen ook andere informatie op band zetten, namelijk arrays en bytes.

Om arrays op cassette te zetten, gebruikt u het keyword DATA in het SAVE-statement :

```
SAVE string DATA array-naam ()
```

Daarbij bevat de string weer de naam waaronder de informatie op band wordt gezet, precies zoals voor een programma (of bytes).

De array-naam bepaalt welk array u op band zet. Dat is dus gewoon een letter (of een letter gevolgd door $). Denk er wel aan, dat de haakjes er dienen te staan.

Onthoud goed het verschil tussen de string en de array-naam. Indien u bijvoorbeeld het commando geeft :

```
SAVE "Naam" DATA b()
```

dan wordt de array b() uit het geheugen gelezen en op cassette gezet, onder de naam "Naam".

Wanneer u typt :

```
VERIFY "Naam" DATA b()
```

dan zoekt de computer op de cassette naar een numeriek array met de naam "Naam". Vindt hij dit, dan zet hij op het scherm : Number array: Naam", en vergelijkt hij de informatie op cassette met de inhoud van het array b in de computer.

Het commando

```
LOAD "Naam" DATA b()
```

zoekt het array op de cassette en indien er plaats voor is in het geheugen, wist het een bestaand array b uit het geheugen en laadt het nieuwe array in het geheugen, onder de naam "b".

MERGE kan met arrays niet gebruikt worden.

Een array met karakters (strings) kan op precies dezelfde manier gesaved worden. Indien de computer zo'n string-array op de band vindt, dan zet hij "Character array:" op het scherm, gevolgd door de naam ervan. Wanneer u een karakter-array inlaadt, wordt niet alleen een al bestaand array met dezelfde naam uit het geheugen gewist, maar ook een gewone stringvariabele met diezelfde naam.

Binaire opslag wordt gebruikt om informatie op te slaan, zonder verwijzing naar de aard ervan. Het kan gaan om de inhoud van een televisiescherm, of een set UDG's, of iets wat u zelf had gemaakt. Binaire opslag wordt aangegeven met het woord CODE :

```
SAVE "beeld" CODE 16384,6912
```

De opslag-eenheid in het geheugen heet een "byte" (een getal tussen 0 en 255). Elk byte heeft een adres (een getal tussen 0 en 65535). Het eerste getal na CODE is het adres van het eerste byte in het geheugen dat op band gezet moet worden; het tweede getal geeft aan, hoeveel opeenvolgende bytes er moeten opgeslagen worden. In het voorbeeld, is 16384 het adres van het eerste byte in de display file (waarin de inhoud van het tv-beeld zit), en 6912 is de lengte van die file. We zetten dus in feite een tv-scherm op de band. Probeer het bovenstaande SAVE-commando uit. U hoeft niet per se de naam "beeld" te gebruiken. Dat is enkel bedoeld als geheugensteuntje, om te onthouden wat er op de band staat.

Om het beeld terug in te laden, typt u

```
LOAD "beeld" CODE
```

Bij CODE kunt u getallen gebruiken, in deze vorm :

```
LOAD naam CODE start, lengte
```

Daarbij dient "lengte" enkel als een beveiliging. Wanneer de computer de informatie op de band gevonden heeft, onder de opgegeven naam, gaat hij na of de opgegeven lengte klopt met die van de informatie op de band, en hij zal weigeren om de bytes in te laden indien er meer op de band staan dan aangegeven in het commando. Daardoor wordt vermeden dat eventuele overtollige bytes een stuk van het geheugen overschrijven, dat u had willen bewaren. In zo'n geval geeft de +2 de foutmelding : "R Tape loading error".

Indien u in het commando de lengte niet vermeldt, dan worden alle bytes ingelezen, hoeveel het er ook zijn.

Het startadres geeft aan waar het eerste byte dient geladen te worden. Dit adres kan verschillen van het adres waar het oorspronkelijk vandaan werd gesaved. Indien beide adressen gelijk zijn, kunt u "start" weglaten bij het LOAD-commando.

De uitdrukking "CODE 16384,6912" slaat op een stukje geheugen dat vaak gesaved en ingeladen wordt, namelijk het scherm. Daarom is er een speciale functie voorzien, die deze uitdrukking vervangt. U kunt dus typen :

```
SAVE "beeld" SCREEN$
```

en op dezelfde manier

```
LOAD "beeld" SCREEN$
```

Dit is overigens een van de zeldzame gevallen waarbij VERIFY nooit zal werken. Het commando VERIFY schrijft de naam van wat op de cassette wordt gevonden, op het scherm. Daardoor klopt de scherminhoud niet meer met wat er op de band staat, waardoor de VERIFY niet lukt.

Alles wat u met SAVE, LOAD, MERGE en de cassette kan doen, is ook mogelijk met de "silicon disk", de RAM disk die in de +2 zit. Die werkt ongeveer zoals een cassette (met enkele commando's meer), maar hij is ongeveer 64 K groot, werkt zeer snel en verliest alles wanneer u de +2 reset of de stroom uitschakelt. NEW heeft evenwel geen invloed. Gebruik alle commando's precies zoals u zou doen voor de cassetterecorder, maar zet er voor de RAM disk een uitroepteken bij, tussen het commando en de parameters. Om iets op cassette te saven, zou u bijvoorbeeld typen

```
SAVE "blokjes"
```

en om datzelfde te saven op de RAMdisk, typt u

```
SAVE ! "blokjes"
```

Voor de RAM disk bestaan nog twee speciale commando's. Het eerste is

```
CAT !
```

waardoor u een overzicht krijgt van alles wat op de RAM disk staat gesaved. Het tweede commando is

```
ERASE ! "naam"
```

waardoor u een bepaald programma van de RAM disk kunt verwijderen.

De meest voor de hand ligende manier om de RAM disk te gebruiken, is misschien om stukken van BASIC programma's in op te slaan, die met een MERGE ! commando om beurten in een kleiner programma opgenomen kunnen worden. Op die manier kunt u een programma van ongeveer 90 K BASIC schrijven, dat u kompleet in het geheugen van de +2 kunt houden. Dat veronderstelt wel, dat de structuur van het programma zeer duidelijk vast ligt.

Een interessant gebruik van de RAM disk is het animeren van beelden. Een "traag" BASIC programma kan dan de beelden tekenen, ze op de RAM disk wegschrijven, waarvan ze later tegen een hoge snelheid terug kunnen gehaald worden. Het volgende programma geeft daarvan een voorproefje. Zonder twijfel kunt u zelf heel wat meer realiseren ...

```
 10 INK 5: PAPER 0: BORDER 0: CLS
 20 FOR f=1 TO 10
 30 CIRCLE f*20,150,f
 40 SAVE ! "bal"+STR$ f CODE 16384,2048
 50 CLS
 60 NEXT f
 70 FOR f=1 TO 10
 80 LOAD ! "bal"+STR$ f CODE
 90 NEXT f
100 BEEP 0.01, 0.01
110 FOR f=0 TO 2 STEP -1
120 LOAD ! "bal"+STR$ f CODE
130 NEXT f
140 BEEP 0.01, 0.01
150 GO TO 70
160 REM met GO TO 160 wist u de beelden van de disk
170 FOR f=10 TO 1 STEP -1
180 ERASE ! "bal"+STR$ f
190 NEXT f
```

Merk op dat op regel 40 van dit programma, de twee getallen na CODE, het adres aangeven van het beeldscherm, en de lengte van het bovenste derde ervan. Doordat enkel dit bovenste deel van het scherm wordt gesaved en ingeladen, gaat alles een stuk sneller. Regels 160 tot 190 worden gebruikt indien u het programma stopt met BREAK, de teken-routine wijzigt en de nieuwe tekeningen wilt saven. Vooraleer u dit doet, typt u "GO TO 160" om de RAM disk leeg te maken. Probeer er steeds voor te zorgen dat u de RAM disk van achter naar voren leegmaakt, zodat de laatste informatie die werd gesaved, de eerste is die gewist wordt. Daardoor bespaart u de computer een hoop rekenwerk, en gebeurt alles ook veel sneller.

Tot slot van dit deel, geven we hieronder een volledig overzicht van de vier commando's die met de datacorder werken.

De parameter "naam" kan een willekeurige string-uitdrukking zijn. Hij slaat op de naam waaronder de informatie op de band wordt gezet. Hij moet uit ASCII-karakters bestaan, waarvan alleen de eerste tien gebruikt worden.

U kunt vier soorten informatie op cassette of RAM disk zetten : programma en variabelen (samen), numerieke arrays, string-arrays en bytes.

Wanneer, na een VERIFY, LOAD of MERGE-commando, de computer de cassette doorzoekt naar informatie met een bepaalde naam en van een bepaalde soort, dan zet hij het type en de naam op het scherm van alle informatie die hij vindt. Het type wordt aangegeven als "Program:", "Number array:", "Character array:", of "Bytes:". Indien hierbij "naam" de nulstring is (""), dan

neemt de computer het eerste blok informatie aan, wat de naam ook zij.

SAVE

1. Programma en variabelen

   SAVE (!) naam LINE regelnummer

schrijft het programma en de variabelen weg, op zo'n manier dat LOAD nadien automatisch "GO TO regelnummer" uitvoert.

2. Bytes

   SAVE (!) naam CODE start, lengte

schrijft "lengte" bytes weg, te beginnen op adres "start"

Onthoud dat

   SAVE (!) naam SCREEN$

hetzelfde betekent als

   SAVE (!) naam CODE 16384,6912

en de scherminhoud wegschrijft.

3. Arrays

   SAVE (!) naam DATA letter ()

of ook nog

   SAVE (!) naam DATA letter $()

schrijft een numeriek array of een string-array weg, met de opgegeven naam

VERIFY

1. Programma en variabelen

   VERIFY naam

gaat na of het programma en de variabelen die onder die naam op cassette staan, kloppen met de geheugeninhoud.

2. Bytes

   VERIFY naam CODE start,lengte

Indien blijkt dat het aantal bytes dat onder de opgegeven naam op cassette staan, niet groter is dan "lengte", wordt nagegaan of wat op band staat overeenkomt met wat er in het geheugen staat, te beginnen vanaf adres "start".

   VERIFY naam CODE

controleert of de bytes met die naam op cassette, overeenkomen

met wat er in het geheugen staat, te beginnen op het adres waar het eerste byte oorspronkelijk stond.

3. Arrays

```
VERIFY naam DATA letter ()
```

of evengoed

```
VERIFY naam DATA letter $()
```

controleert of het numeriek (string-) array met die naam op band, hetzelfde is als het array met die naam in het geheugen.

LOAD

1. Programma en variabelen

```
LOAD (!) naam
```

wist een bestaand programma en variabelen uit het geheugen, en laadt het programma en de variabelen die onder die naam op band staan, in het geheugen. Indien dat programma gesaved was met LINE regelnummer, dan wordt na het inladen automatisch een GO TO regelnummer uitgevoerd. Indien het inladen niet kan gebeuren, wordt het bestaande programma en de variabelen in het geheugen bewaard.

2. Bytes

```
LOAD (!) naam CODE start, lengte
```

Indien het aantal bytes dat onder die naam op band staat, niet groter is dan "lengte", worden ze in het geheugen geladen, te beginnen op adres "start", waarbij de vorige inhoud van de opeenvolgende adressen wordt overschreven.

```
LOAD (!) naam CODE start
```

laadt de bytes met die naam onvoorwaardelijk in het geheugen, te beginnen op adres "start". Daarbij wordt de vorige inhoud van de opeenvolgende adressen overschreven.

```
LOAD (!) naam CODE
```

laadt de bytes met die naam in het geheugen, te beginnen op het adres vanwaar het eerste byte werd gesaved aarbij wordt de vorige inhoud van dat deel van het geheugen overschreven.

3. Arrays

```
LOAD (!) naam DATA leter ()
```

of ook

```
LOAD (!) naam DATA letter $()
```

wist een numeriek (string-) array die "letter" heet, uit het geheugen, en maakt een nieuw array met de informatie die van cassette wordt gelezen.

MERGE

1. Programma en variabelen

   MERGE (1) naam

zorgt ervoor dat het programma met die naam wordt vermengd met een programma in het geheugen. Daarbij worden programmaregels in het geheugen overschreven door programmaregels met hetzelfde regelnummer, die van cassette worden gelezen. Ook variabelen met dezelfde namen worden vervangen door de variabelen die op de band stonden.

2. Bytes

MERGE bytes is niet mogelijk.

3. Arrays

MERGE arrays is niet mogelijk.

OEFENING

Probeer enkele keren de SAVE, LOAD en MERGE-commando's met programma's en data, zowel op cassette als op de RAM disk.

DEEL 21 : WERKEN MET DE PRINTER

Inhoud ...
LPRINT, LLIST, COPY

De +2 heeft een ingebouwde seriële poort met de bijbehorende software, waarmee u een printer kunt gebruiken. Dat kan evenwel enkel in de 128 BASIC mode.

De printer die u wilt gebruiken, dient een RS232 (seriële) interface te hebben. Indien u de scherminhoud op papier wilt kunnen afdrukken, dient de printer te beschikken over een quadruple density grafische instelling, die Epson compatibel is.

ndien u twijfelt of u het juiste verbindingssnoer hebt tussen de computer en de printer, raadpleeg uw Sinclair verkoper.

Opdat de printer en de computer met elkaar zouden kunnen communiceren, moeten ze beide dezelfde "baud rate" hebben. De baud rate is de snelheid waarmee gegevens tussen computer en printer worden overgebracht. Misschien is het wel mogelijk om uw printer op diverse snelheden in te stellen, maar het is alleszins makkelijker om de snelheid van de computer aan te passen. Ergens in het handboek van uw printer, wordt de baud rate opgegeven. Stel de +2 op dezelfde snelheid in met het commando :

```
FORMAT "p"; (baud rate)
```

Dit is niet nodig indien uw printer een baud rate van 9600 heeft omdat dit de standaard snelheid van de +2 is.

Nadat alles is ingesteld, kunt u drie BASIC commando's gebruiken om iets af te drukken op papier. De eerste twee, LPRINT en LLIST doen precies hetzelfde als PRINT en LIST, maar ze gebruiken de printer in plaats van het scherm. Merk op dat de "Print" optie in het editmenu hetzelfde effect heeft als LLIST, alleen werkt het iets makkelijker.

Probeer bijvoorbeeld dit programma eens :

```
10 PRINT "Dit programma ..."
20 LLIST
30 LPRINT'"... drukt de karakterset af, d.w.z. :"
40 FOR n=32 TO 255
50 LPRINT CHR$ n;
60 NEXT n
```

Een belangrijk punt : LPRINT en LLIST filteren zorgvuldig alle kleurcodes met hun parameters uit, alvorens er iets geprint of gelist wordt. Die kleurcodes zijn nog een nalatenschap van de 48 K Spectrum. De kleurcodes kon je toen ook in een string opnemen, om PAPER, INK enzovoort te bepalen bij het printen. Over het algemeen betekenen die codes totaal iets anders voor printers (schuinschrift, onderstrepen of iets dergelijks), zodat het nogal gevaarlijk zou zijn om die kleurcodes door te sturen in de hoop dat er niets raars zou gebeuren. Een neveneffect hiervan is dat het in BASIC niet mogelijk is om bepaalde speciale effecten op een printer in te stellen, die gecontroleerd worden middels een groep codes die met ESCAPE (karaktercode 27) beginnen of een gelijkaardige groep.

Het derde commando, COPY, tekent een afbeelding van het scherm op de printer. Om dit te zien, gaat u naar het kleine scherm, typ LIST, waardoor u een listing van het bovenstaande programma op het scherm krijgt, en typ dan : COPY

Het commando COPY wacht een 15 tot 30 seconden vooraleer het van start gaat. Geen paniek dus, indien er niet direkt iets gebeurt. U zult na een tijd nog een listing op de printer zien verschijnen, maar deze listing lijkt erg op die op het scherm. Indien u na een COPY, alleen maar een hoop rare tekens op het papier ziet verschijnen, dan kunt u aannemen dat uw printer niet helemaal geschikt is voor de computer.

U kunt de printer op elk moment doen ophouden, door op BREAK te drukken. Sommige printers beschikken over een buffer, dat is een tussengeheugen waarin de tekst wordt opgeslagen alvorens hij wordt afgedrukt. Indien uw printer een buffer heeft, zal hij niet onmiddellijk stoppen met printen nadat u BREAK drukte, hoewel de +2 die toets wel registreerde.

Indien u een van de printer-commando's gebruikt zonder dat er een printer is aangesloten, dan blijft de +2 geduldig wachten tot de (niet-bestaande) printer hem signaleert dat hij klaar is om gegevens te ontvangen. Zoals gewoonlijk zal ook hier de BREAK-toets de +2 weer tot leven brengen.

Probeer dit programma :

```
10 FOR n=31 TO 0 STEP -1
20 PRINT AT 31-n,n; CHR$ (CODE "0"+n);
30 NEXT n
```

U ziet een reeks karakters die diagonaal vanaf de hoek rechtsboven worden afgedrukt op het scherm, tot onderaan : daar vraagt de computer u "scroll ?".

Wijzig nu op regel 20, AT 31-n in TAB n. Het programma doet exact hetzelfde als voordien.

Wijzig nu PRINT op regel 20 in LPRINT. Nu vraagt de computer niet of hij mag scrollen, omdat dit bij een printer geen nut heeft.

Wijzig nu TAB n terug in AT 31-n, en laat LPRINT staan. Nu wordt er maar één regel karakters afgedrukt. Het verschil zit hierin : de informatie die LPRINT moet verwerken, wordt niet meteen geprint, maar wordt in een buffer opgeslagen tot ofwel voldoende is opgeslagen om een regel te vullen, ofwel iets de buffer leeg maakt. Er wordt daarom enkel geprint :

1. indien de buffer vol is
2. na een LPRINT statement dat niet eindigt op een komma of een kommapunt
3. wanneer een komma, afkappingsteken of TAB-item een nieuwe regel vereist
4. op het einde van een programma, indien er nog iets overblijft dat geprint moet worden
5. (afhankelijk van uw printer) indien u de printer "off line" schakelt.

Nummer 3 verklaart waarom het programma met TAB werkt zoals we zagen. Bij AT wordt het regelnummer genegeerd, en de LPRINT-positie (zoals de PRINT-positie) wordt naar het kolomnummer gebracht. Een AT-uitdrukking zal er nooit voor zorgen dat er geprint wordt.

OEFENING
++++++++

Maak op papier een grafiek van een SIN-curve, met behulp van het programma dat we in deel 17 schreven, en het commando COPY.

DEEL 22 : ANDERE RAND-APPARATUUR

Inhoud ...

Er zijn vele randapparaten beschikbaar, die u op de +2 kunt aansluiten. In hoofdstuk 10 wordt dieper ingegaan op de aansluiting en bediening ervan.

De ZX microdrive is een veelzijdig opslagmedium dat tegen hoge snelheid werkt. Het werkt niet enkel met SAVE, VERIFY, LOAD en MERGE, maar ook met PRINT, LIST, INPUT en INKEY$.

Een netwerk wordt gebruikt om verscheidene computers met elkaar te verbinden, zodat ze met elkaar kunnen communiceren. U kunt dit bijvoorbeeld gebruiken om met slechts 1 microdrive een hele groep computers te bedienen.

De RS232 interface is een manier om verbindingen tot stand te brengen, die aan een bepaalde standaard beantwoordt. Hij laat toe om de computer met toetsenborden, printers en diverse andere apparaten te verbinden, ook al waren die niet speciaal voor de +2 computer ontworpen.

Het aparte toetsenbordje kan handig gebruikt worden om in 128 BASIC het aanpassen van programma's te vergemakkelijken. Het is eveneens goed om snel gegevens in te typen.

DEEL 23 : IN EN OUT

Inhoud ...
OUT, IN

De processor kan geheugen uitlezen (ROM en RAM) of in geheugen schrijven (enkel RAM) door middel van PEEK en POKE. In feite maakt het de processor niets uit, of hij in ROM of RAM werkt. Hij ziet enkel 65536 geheugenadressen, waaruit hij even zoveel bytes kan lezen - ook al leest hij nonsens - en waarin hij even zoveel bytes kan schrijven - ook al heeft dat geen effect.

Kompleet in overeenstemming daarmee, heeft de processor ook 65536 zogenaamde I/O-poorten (In/Out-poorten). Via die poorten kan hij communiceren met bijvoorbeeld het toetsenbord of de printer, maar controleert hij ook het extra geheugen en de geluidschip. Enkele van die poorten kunnen ook rustig vanuit BASIC worden gebruikt, met behulp van de IN-functie en het OUT-commando. Er zijn evenwel geheugenlocaties waarvan u de inhoud in BASIC niet mag gebruiken, omdat u anders een crash riskeert, waardoor programma en gegevens onherroepelijk verloren gaan.

IN is een functie die op PEEK lijkt. Ze ziet er zo uit :

IN adres

De functie heeft één argument : het poort-adres. Het resultaat van de functie is, het byte wat die poort bevatte.

OUT is een commando dat op POKE lijkt. Het werkt zo :

OUT adres, waarde

en het schrijft de gegeven waarde naar de poort met het opgegeven adres. Hoe dit adres wordt geïnterpreteerd, hangt sterk van de rest van de computer af. Vaak hebben verschillende adressen dezelfde betekenis. Bij de +2 is het nog het beste om u het adres als een binair getal voor te stellen, omdat elk bit de neiging heeft om onafhankelijk te werken. Elk bit kan 1 of 0 zijn. Er zijn 16 bits, die we (met de A van Adres) als volgt benoemen :

A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0

Daarbij is A0 het eerste bit, A1 het tweede bit, A2 het derde, en zo verder. Bits A0, A1, A2, A3 en A4 zijn het belangrijkste. Normaal zijn ze 1. Indien één ervan 0 is, weet de computer dat hij iets bepaalds moet ondernemen. De computer kan slechts één ding tegelijk doen, dus mag er maar 1 van dit bits tegelijk 0 zijn. Bits A6 en A7 worden genegeerd. Indien u erg goed bent in electronica, kunt u die zelf voor iets gebruiken. Het beste kunt u adressen gebruiken die 1 minder zijn dan een veelvoud van 32 : dan zijn bits A0 tot A4 allemaal 1. Bits A8, A9 en zo verder worden soms gebruikt om extra informatie over te brengen, meestal in verband met het extra geheugen of het geluid.

Het byte dat wordt gelezen of geschreven, telt 8 bits. Die noemen we (met de D van Data) :

D7,D6,D5,D4,D3,D2,D1,D0

Op de volgende bladzijde ziet u een lijst van de poort-adressen die door de +2 worden gebruikt .

Eén groep adressen wordt gebruikt voor het toetsenbord en de datacorder.

Het toetsenbord is verdeeld in acht halve rijen van vijf toetsen elk, namelijk :

```
IN 65278 leest de halve rij van CAPS SHIFT tot V
IN 65022 leest de halve rij van A tot G
IN 64510 leest de halve rij van Q tot T
IN 63486 leest de halve rij van 1 tot 5 (en JOYSTICK 2)
IN 61438 leest de halve rij van 0 tot 6 (en JOYSTICK 1)
IN 57342 leest de halve rij van P tot Y
IN 49150 leest de halve rij van ENTER tot H
IN 32766 leest de halve rij van SPATIEBALK tot B
```

Die adressen worden gevormd door 254 + 256*(255-2^n), waarbij n varieert tussen 0 en 7.

In het byte dat wordt ingelezen, vertegenwoordigen de bits D0 tot D4 de vijf toetsen in de gegeven rij. D0 is de buitenste toets, D4 de toets in het midden van het toetsenbord. Het bit is 0 wanneer de toets wordt ingedrukt, en 1 als dat niet zo is. D6 wordt door de datacorder geset. Indien er geen gegevens van de cassette gelezen worden, heeft dit bit een willekeurige waarde.

Voor JOYSTICK 1 is bit 0 de schietknop, bit 1 omhoog, bit 2 omlaag, bit 3 rechts en bit 4 links. Voor joystick 2 is bit 0 links, bit 1 rechts, bit 2 omlaag, bit 3 omhoog en bit 4 de schietknop. Vanuit BASIC worden die uitgelezen als de cijfertoetsen.

Poortadres 254 controleert (bij OUTput) het geluid (D4) en het SAVE-signaal naar de datacorder (D3). Het bestuurt ook de kleur van de border (D2, D1 en D0).

Poort-adressen 254, 247 en 239 verzorgen ook de communicatie met de randapparaten die in deel 22 werden vermeld.

Poort-adres 32765 schakelt het extra geheugen. Een OUT-commando naar deze poort, vanuit BASIC, zal bijna altijd voor een crash zorgen, waardoor programma en gegevens onherroepelijk verloren gaan. Deze poort wordt in meer detail beschreven in deel 24 van dit hoofdstuk, onder de kop "Geheugenbeheer". Deze poort kan enkel beschreven worden. Het is niet mogelijk om de momentele toestand van het geheugen (welke "bladzijde" van het geheugen staat ingeschakeld) te bepalen door een IN-instructie.

Poort-adres 49149 bestuurt de dataregisters van de geluidschip. Poort-adres 65533 schrijft, in output, een register-adres en leest, in input, een register uit. Door deze registers oordeelkundig te gebruiken, kunnen geluiden gemaakt worden, terwijl BASIC verder gaat met andere taken. Denk er wel aan, dat deze poorten ook de RS232, het los toetsenbord (keypad) en de MIDI besturen.

Run dit programma, en kijk hoe het toetsenbord werkt :

```
10 FOR n=0 TO 7: REM nummer van halve rij
20 LET a=254+256*(255-2^n)
30 PRINT AT 0,0; IN a: GO TO 30
```

Duw op een aantal willekeurige toetsen. Wanneer u met een bepaalde groep (halve rij) klaar bent, druk dan op BREAK en typ

```
NEXT n
```

De controle-, data- en adres-bus zijn toegankelijk achteraan de +2, op de "EXPANSION I/O"-stekkerbus. U kunt dus bijna alles doen met een +2 wat u met een Z80 kunt, hoewel de hardware van de computer u soms parten kan spelen.

In hoofdstuk 10 vindt u een tekening met de aansluitingen van de EXPANSION I/O-stekkerbus.

DEEL 24 : HET GEHEUGEN

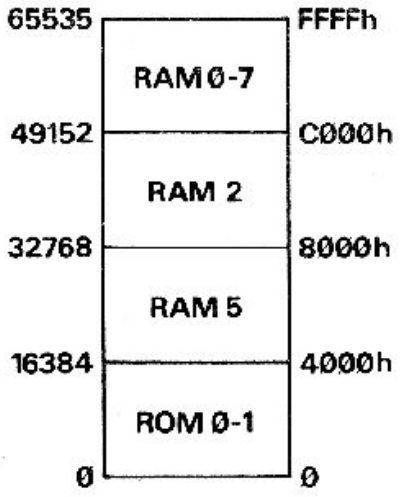
Inhoud ...
   CLEAR

Binnen in de +2 wordt alles in de vorm van bytes opgeslagen. Dat zijn getallen tussen 0 en 255. Misschien denkt u dat de +2 de prijs van aardappelen of de naam van uw vriend Pieters onthoudt, maar dat is niet zo : alle informatie wordt omgezet in een verzameling bytes. Bytes, daar werkt de computer mee.

Elke plek waar een byte kan opgeslagen worden, heeft een adres. Dat is een getal tussen 0 en FFFFh (de h op het einde van een getal geeft aan dat het een hexadecimaal getal is), dat wil zeggen dat een adres in twee bytes kan opgeslagen worden.

Stel u het geheugen voor als een lange rij dozen, die elk één byte kunnen bevatten. Niet alle dozen zijn gelijk : de dozen genummerd van 4000h tot FFFFh zijn RAM-dozen. Die kunt u openmaken, en de inhoud ervan wijzigen. De dozen genummerd van 0 tot 3FFFh zijn ROM-dozen : die hebben een glazen deksel, dat niet open kan. We kunnen alleen naar de inhoud kijken, maar hem niet veranderen. Die inhoud werd er in gestopt, toen de computer gefabriceerd werd.

In de +2 hebben we twee keer zoveel geheugen gestopt, als er normalerwijs in kan zitten. De processor kan 65536 bytes adresseren, maar er zijn in werkelijkheid 131072 bytes RAM en 32768 bytes ROM aanwezig, dat is samen 163840 bytes of 160 K. Dit wordt voor de processor verborgen gehouden, omdat de hardware werkt volgens een procédé dat "paging" heet, dat wil zeggen dat hij het geheugen kan inschakelen "per bladzijde". Daardoor ziet BASIC (en de processor) het geheugen altijd als een combinatie van 16K ROM en 48K RAM.



De indeling van het geheugen van de +2

Om de inhoud van een doos te bekijken, gebruiken we de functie PEEK. Het argument van die functie, is het adres van de doos; het resultaat ervan is de inhoud van de doos. Dit programma geeft bijvoorbeeld de inhoud van de eerste eenentwintig bytes van de ROM, met hun adres :

```
10 PRINT "Adres"; TAB 8; "Byte"
20 FOR a=0 TO 20
30 PRINT a; TAB 8; PEEK a
40 NEXT a
```

Waarschijnlijk zeggen al die bytes u niets, maar de processor begrijpt ze als instructies, waardoor hij weet wat hij moet doen.

Om de inhoud van een doos te wijzigen (indien het een RAM-doos is), gebruiken we het commando POKE, in de vorm :

```
POKE adres, inhoud
```

waarbij "adres" en "inhoud" numerieke uitdrukkingen zijn. Indien u bijvoorbeeld typt :

```
POKE 31000,57
```

dan wordt het byte op adres 31000 de waarde 57 gegeven. Typ nu

```
PRINT PEEK 31000
```

en u ziet het resultaat. Probeer gerust andere waarden, dan zult u zien dat het echt zo is. De waarde van "inhoud" moet liggen tussen -255 en +255. Als ze negatief is, wordt er 256 bij opgeteld.

De POKE-instructie geeft u erg veel macht over de computer, indien u ze goed weet te gebruiken, maar kan tegelijk zeer destructief zijn indien u dat niet weet. Het is erg gemakkelijk, door een verkeerde waarde op een verkeerd adres te POKEN, om een programma kwijt te spelen waaraan u uren hebt gewerkt. Gelukkig kan u door een POKE geen schade toebrengen aan de computer.

We gaan nu dieper in op de organisatie van de RAM. U kunt dit deel rustig overslaan, indien u niet echt geïnteresseerd bent.

Het geheugen is opgedeeld in een aantal gebieden (cfr. de tekening hiervoor), waarin verschillende soorten informatie worden opgeslagen. Elk gebied is precies groot genoeg om alles te bevatten wat er op een gegeven ogenblik in zit. Indien u op een bepaald moment méér erin wilt stoppen, bijvoorbeeld door een programmaregel of een variabele toe te voegen, dan wordt eerst ruimte gemaakt door alles wat hoger in het geheugen ligt, naar boven te schuiven. Indien u informatie weghaalt, gebeurt het tegenovergestelde.

De display file bevat het beeld op het televisiescherm. Het is op een nogal eigenaardige manier georganiseerd, zodat u er waarschijnlijk niet in zult willen PEEKen of POKEn. Elke karaktercel bestaat uit een raster van 8 x 8 puntjes. Elk van die puntjes kan papierkleur (0) of inktkleur (1) zijn. Op die manier kunnen we het patroon van een karaktercel binair als 8 bytes schrijven, een byte per rij. Die acht bytes worden evenwel niet samen opgeslagen.

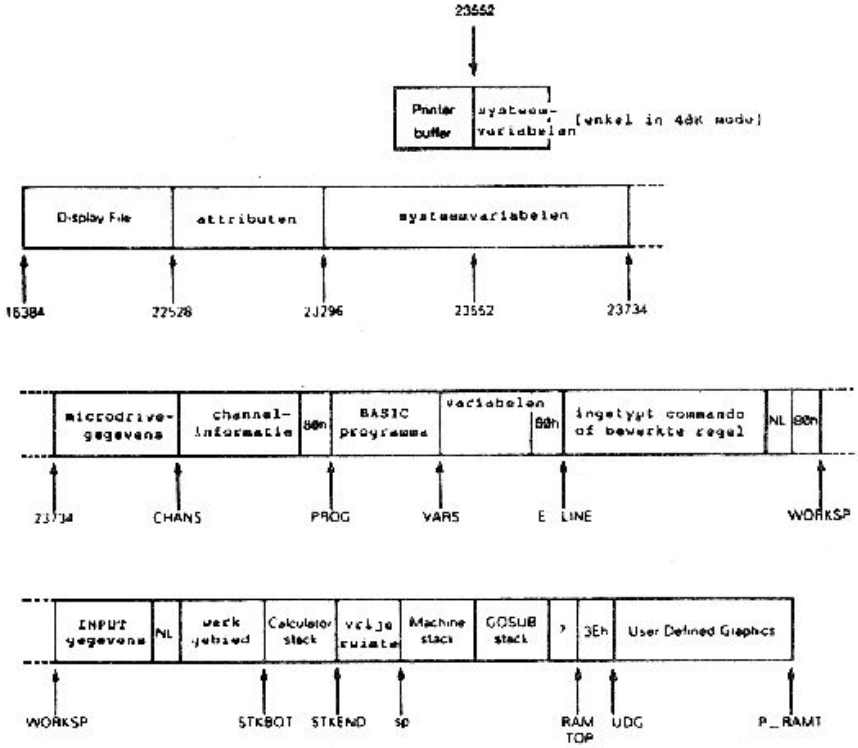
De pixelrijen met eenzelfde rangnummer, in de 32 karakters van een regel op het scherm, worden samen opgeslagen als één lijn van 32 bytes, omdat die gebruikt kan worden door de electronenstraal van de tv, die van links naar rechts over het scherm zwiept.

Het volledige beeld bevat 24 regels van elk 8 rijen, zodat u zou kunnen verwachten dat de 192 rijen in volgorde zouden opgeslagen worden. Niets daarvan ! Eerst komen de bovenste rijen van regels 0 tot 7, dan de volgende rijen van regels 0 tot 7, en zo door tot de achtste rijen van regels 0 tot 7. Daarna hetzelfde voor de regels 8 tot 15, en dan nog eens voor de regels 16 tot 23. Het komt er op neer dat, wanneer u een computer gewend bent die

PEEK en POKE voor het scherm gebruikt, u zult moeten wennen aan SCREEN$ en PRINT AT (of PLOT en POINT).

De attribute file bevat de kleur-informatie voor elke karaktercel, op de manier zoals we verklaarden bij ATTR. Deze informatie wordt wél regel per regel opgeslagen, zoals we zouden verwachten.

De manier waarop de computer werkt in 48 BASIC en 128 BASIC is lichtjes verschillend. Op de plaats van de printer-buffer in 48 mode, staan in 128 mode extra systeemvariabelen.



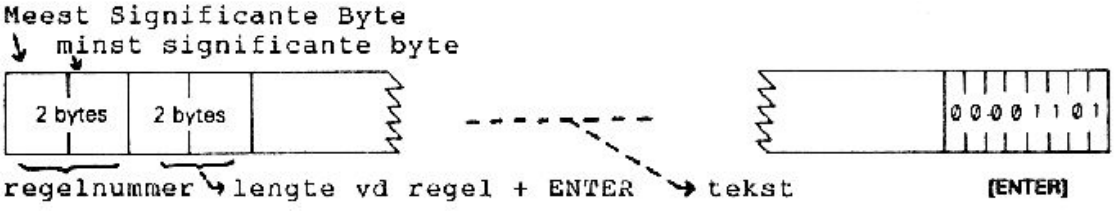
Geheugenindeling in BASIC

De systeemvariabelen bevatten informatie waaruit de computer kan opmaken wat hij moet doen. Een volledige lijst van die variabelen vindt u in deel 25 van dit hoofdstuk. Onthoud voor het ogenblik alleen dat sommige ervan (CHANS, PROG, VARS, E_LINE enzovoort) het adres bevatten van de grens tussen verschillende geheugengebieden. Het zijn geen BASIC variabelen, en worden door de +2 niet bij hun naam herkend.

De index van de microdrive-cartridge wordt enkel gebruikt indien er een microdrive is aangesloten. Normaal is dit gebied leeg.

De channel-informatie slaat op input en output van diverse rand-apparatuur zoals het toetsenbord (samen met het onderste deel van het scherm), het bovenste deel van het scherm, de printer.

Een BASIC programmaregel wordt als volgt opgeslagen :

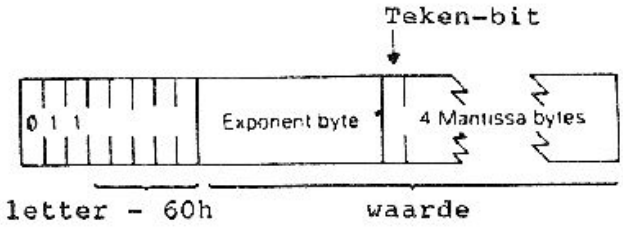


Merk op dat, in tegenstelling tot alle andere Z80-getallen, het regelnummer hier opgeslagen wordt met het belangrijkste byte eerst, dat wil zeggen zoals we het opschrijven.

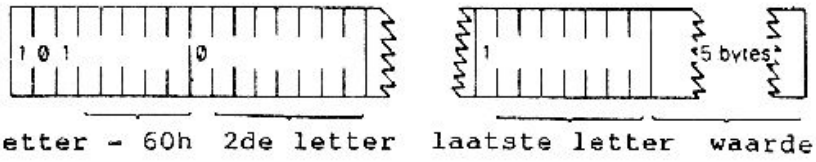
Een numerieke konstante in het programma wordt gevolgd door de binaire vorm ervan, dat wil zeggen eerst het karakter 14, en dan vijf bytes waarin het getal zelf staat.

Variabelen worden, afhankelijk van hun aard, op verschillende manieren opgeslagen. De letters in de naam van variabelen worden als kleine letters opgeslagen.

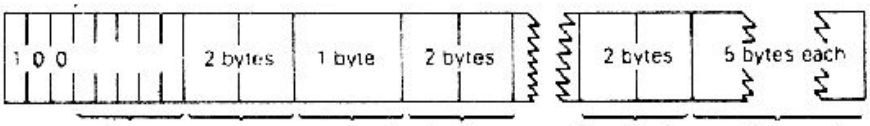
Getal dat met één letter wordt benoemd :



Getal dat met meerdere letters wordt benoemd :



Numeriek array :



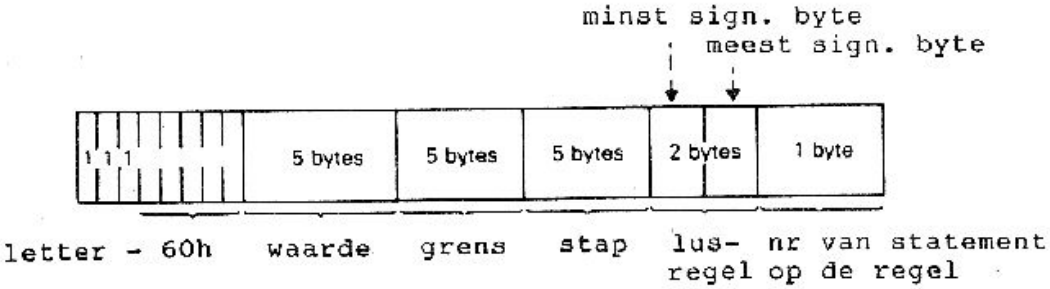
De elementen van het array staan in deze volgorde :
- eerst de elementen die als eerste index 1 hebben
- dan de elementen die als eerste index 2 hebben
- dan de elementen die als eerste index 3 hebben

en zo verder voor alle mogelijke waarden van de eerste index

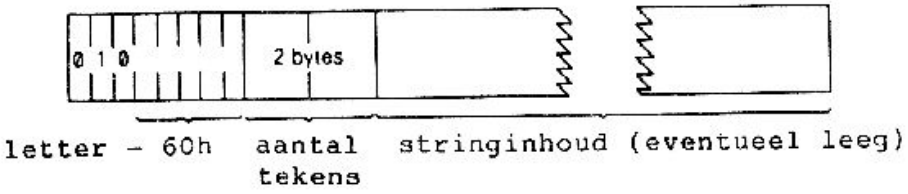
De elementen met een bepaalde eerste index, worden op dezelfde manier geordend volgens de tweede index, en zo verder tot de laatst voorkomende index.

Een voorbeeld : de elementen van het array c(3,6) dat in deel 12 van dit hoofdstuk ter sprake kwam, worden opgeslagen in de orde c(1,1) c(1,2) c(1,3) c(1,4) c(1,5) c(1,6) en c(2,1) c(2,2) ... c(2,6) en c(3,1) c(3,2) ... c(3,6)

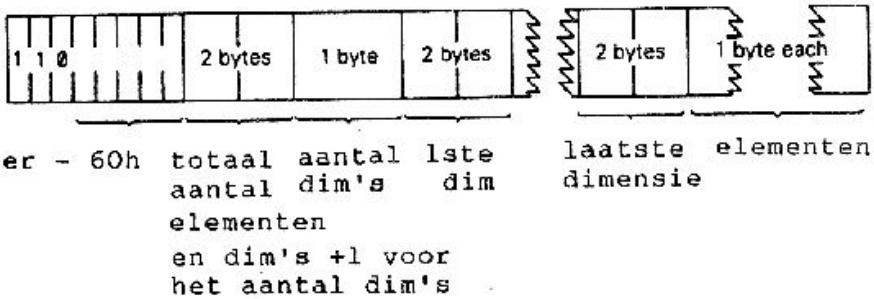
Controlevariabele van een FOR/NEXT-lus :



Stringvariabele:



String-array :



De calculator is het deel van het BASIC systeem, dat de rekenkundige bewerkingen verzorgt. De getallen die verwerkt worden, bevinden zich veelal op de stack van de calculator zelf.

Het lege gedeelte bevat de ruimte die tot dan toe nog niet werd gebruikt.

De machine-stack is de stack die door de Z80-processor wordt gebruikt om RETURN-adressen en dergelijke op te slaan.

De GO SUB-stack kwam in deel 5 van dit hoofdstuk aan de orde.

Het byte waar RAMTOP naar verwijst, staat op het hoogste adres dat bij het BASIC systeem gebruikt kan worden. Zelfs NEW,

waardoor alle RAM wordt gewist, werkt maar tot op dit adres. Dat houdt in dat de UDG's door NEW niét worden beïnvloed. U kunt het adres van RAMTOP wijzigen door een getal op te nemen in een CLEAR-commando :

```
CLEAR nieuwe RAMTOP
```

Dit commando heeft het volgende effect :
1. alle variabelen worden gewist
2. de display file wordt leeggemaakt (zoals bij CLS)
3. de PLOT-positie wordt in de hoek linksonder gezet
4. de DATA-pointer wordt bij het begin van het programma gezet (zoals door RESTORE)
5. de GO SUB-stack wordt gewist en onder de nieuwe RAMTOP gezet, indien die tussen de calculator en het fysieke eindadres van RAM ligt; anders wordt RAMTOP niet gewijzigd.

Het commando RUN voert ook een CLEAR uit, maar wijzigt RAMTOP niet.

Door CLEAR op deze manier te gebruiken, kunt u ofwel RAMTOP hoger zetten, om meer plaats te maken voor BASIC (waarbij de UDG's wel worden overschreven), ofwel RAMTOP lager zetten, om een groter gedeelte van RAM veilig te stellen voor NEW.

Typ NEW en dan CLEAR 23825. Dan zult u gauw merken wat er gebeurt wanneer de computer "vol" geraakt.

Indien u nu probeert om de +2 te laten rekenen (typ bijvoorbeeld PRINT 1+1) dan geeft hij de foutmelding "4 Out of memory". Daarmee geeft hij aan dat hij geen plaats meer heeft om informatie op te slaan. Indien u tijdens het intypen van een groot programma, deze foutmelding te zien krijgt, dan moet u een beetje van het geheugen opnieuw vrij maken (een paar regels weghalen) om terug controle te krijgen over de computer.

Geheugenbeheer

We hebben al gezegd dat de computer meer geheugen bevat dan de processor aankan. De processor kan weliswaar slechts 64 K geheugen tegelijk adresseren (overzien), maar de rest van het beschikbare geheugen kan naar believen binnen dat 64K-bereik geschakeld worden. Dat werkt ongeveer zoals bij een tv. Het toestel kan slechts één zender tegelijk verwerken (zoals uzelf), maar toch zijn er daarnaast nog een aantal zenders beschikbaar, waarop u kunt afstemmen door op de juiste toets te drukken. Hoewel er dus veel meer informatie aanwezig is dan u tegelijk kunt verwerken, kunt u zelf kiezen welk stuk informatie u op een bepaald moment wilt zien.

Zo ongeveer werkt de processor. Door de juiste bits in een I/O-poort te wijzigen, kan de processor kiezen welk deel van het 160K geheugen hij wil bewerken. In BASIC wordt voor het grootste deel van de tijd, het merendeel van het geheugen genegeerd. Maar voor spelletjes is het wel handig om drie keer zoveel RAM ter beschikking te hebben.

Kijk nog eens naar het overzicht van het geheugen van de +2, in het begin van dit deel. RAM 2 en RAM 5 staan altijd op de plaats zoals aangegeven in de tekening, hoewel er geen enkele reden is waarom ze niet ook in het gebied van de "geheugenbanken" zouden

staan (C000h tot FFFFh) - maar dat zou geen enkel nuttig effect opleveren. Er zijn twee soorten RAM-banken. RAM 4 tot RAM 7 is een soort die door de video- schakelingen én door de processor wordt gebruikt. RAM 0 tot RAM 3 worden uitsluitend door de processor gebruikt. Een machinetaal programma dat met kritische tijdslussen werkt (muziek of communicatie bijvoorbeeld) werkt het beste met de laatstgenoemde soort.

De schakelaar voor die RAM-bank zit op I/O-adres 7FFDh (32765). Het bitpatroon voor dit adres ziet er zo uit :

D0-D2 : RAM-keuze
D3 : schermkeuze
D4 : ROM-keuze
D5 : 48K vergrendeling

D2-D0 vormen een getal van drie bits, dat bepaalt welke RAM in het gebied van C000h tot FFFFh geschakeld wordt. In BASIC staat daar normaal RAM 0. Bij het aanpassen van een programma, wordt RAM 7 gebruikt voor diverse buffers en als "kladblok" voor de computer. D3 wisselt schermen. Scherm 0 staat in RAM 5 (vanaf 4000h) en wordt door BASIC gebruikt. Scherm 1 staat in RAM 7 (vanaf C000h) en kan enkel door machinetaal-programma's gebruikt worden. Het is absoluut te doen, om een scherm in RAM 7 te tekenen en het dan uit te schakelen. Daardoor komt de volle 48K vrij voor data en programma. D4 bepaalt of ROM 0 (de editor) in het gebied tussen 0000h tot 3FFFh staat.

D5 is een veiligheid. Wanneer dit bit 1 is, wordt de computer geblokkeerd in een standaard 48 K-configuratie. Het is dan niet meer mogelijk om geheugenbanken om te schakelen. De enige manier om er terug een 128 K machine van te maken, is de computer uit te schakelen of de RESET-knop in te drukken. De geluidschip kan evenwel nog steeds aangestuurd worden door het OUT-commando.

DEEL 25 : DE SYSTEEMVARIABELEN

Inhoud ...

POKE, PEEK

Het geheugengebied van 23296 tot 23733 wordt door het systeem voor zichzelf gereserveerd. Het bevat enkele korte programma's (voor het geheugenbeheer) en een aantal adressen die we systeem-variabelen noemen. U kunt die PEEKen, om informatie te verkrijgen over het systeem. Sommige adressen kunt u ook met goed gevolg POKEn. Hieronder volgt een lijst ervan, met kort commentaar over hun gebruik.

Zoals u had kunnen verwachten, is er een duidelijk verschil tussen de systeemvariabelen in 48 BASIC mode en in 128 BASIC. In 48 BASIC mode bestaan alle variabelen en routines onder het adres 23552 niet. un plaats wordt ingenomen door een buffer voor de printer (tussen 23296 en 23552). Bij de 48 K Spectrum werd die buffer vaak gebruikt om kleine programmaatjes in machinetaal in op te slaan. Wanneer u dat soort routines in 128 BASIC mode probeert te runnen, loopt de computer onvermijdelijk vast. Een oud programma dat PEEK, POKE en USR gebruikt, kunt u derhalve, om zeker te spelen, het beste in 48 BASIC mode runnen. U kunt het wel in 128 BASIC intypen, en overbrengen door het commando SPECTRUM.

De systeemvariabelen hebben een naam. Die mag evenwel niet verward worden met de woorden en namen die in BASIC gebruikt worden. De computer zal niet begrijpen dat u die namen gebruikt om naar systeemvariabelen te verwijzen. Ze dienen enkel als een geheugensteuntje voor ons.

De afkortingen in de eerste kolom betekenen :

X : niet POKEn in de variabele, u riskeert een crash
N : POKE heeft geen blijvend effect
R : startadres van een routine; is geen variabele

Het getal in de eerste kolom geeft het aantal bytes aan dat de variabele of de routine telt. Van twee bytes is het eerste het minst belangrijke, misschien het omgekeerde van wat u had verwacht. Om dus een bepaalde waarde w in een variabele van twee bytes te POKEn op adres n, moet u :

```
POKE n, w-256*INT(w/256) en
POKE n+1,INT(w/256)
```

en om de waarde ervan te PEEKen, typt u :

```
PEEK n + 256*PEEK(n+1)
```

De systeemvariabelen

Opm.	Adres	Naam	Inhoud/functie
R20	23296	SWAP	subroutine voor paging
R9	23316	YOUNGER	subroutine voor paging
R18	23325	ONERR	subroutine voor paging
R5	23343	PIN	voorbereiding input RS232
R22	23348	POUT	voorbereiding output tokens via RS232; kan gewijzigd worden om uitfilteren van controlecodes te voorkomen
R14	23370	POUT2	voorbereiding output karakters via RS232
N2	23384	TARGET	adres subroutine in ROM 1
X2	23386	RETADDR	RETURN-adres in ROM 0
X1	23388	BANKM	laatste byte dat naar een bank werd geschreven
X1	23389	RAMRST	RST 8-instructie
N1	23390	RAMERR	Fout-code ROM 1
2	23391	BAUD	bitperiode voor RS232 in T-states/26
N2	23393	SERFL	vlag "2de karakter ontvangen" plus data
N1	23395	COL	huidige kolom, van 1 tot WIDTH
1	23396	WIDTH	breedte in kolommen voor printer
1	23397	TVPARS	aantal opeenvolgende parameters die de RS232 verwacht
1	23398	FLAGS3	diverse vlaggen
N10	23399	N_STR1	naam van de file
1	23409	HD_00	code voor soort file
2	23410	HD_0B	lengte van het blok
2	23412	HD_0D	start van het blok
2	23414	HD_0F	lengte van het programma
2	23416	HD_11	regelnummer
1	23418	SC_00	tweede groep, code voor soort file
2	23419	SC_08	tweede groep, lengte van het blok
2	23421	SC_0D	tweede groep, start van het blok
2	23423	SC_0F	tweede groep, lengte van het programma
X2	23425	OLDSP	vorige Stack Pointer, indien TSTACK in gebruik is
X2	23427	SFNEXT	pointer naar eerste lege plaats in de cataloog van de RAM disk
X3	23429	SFSPACE	aantal vrije bytes (17 bits-getal)
N1	23432	ROW01	los toetsenbord, vlaggen en rij 1
N1	23433	ROW23	los toetsenbord, rijen 2 en 3
N1	23434	ROW45	los toetsenbord, rijen 4 en 5
X2	23435	SYNRET	RETURN-adres voor ONERR
5	23437	LASTV	laatste waarde die calculator printte
2	23442	RNLINE	regel die nu wordt hernummerd
2	23444	RNFIRST	eerste regelnummer voor RENUMBER
2	23446	RNSTEP	regelafstand voor RENUMBER
N8	23448	STRIP1	bitmap voor eerste strook
N8	23456	STRIP2	bitmap voor tweede strook
X	23551	TSTACK	hoogste punt van tijdelijke stack
N8	23552	KSTATE	gebruikt om toetsenbord uit te lezen
N1	23560	LASTK	bevat de laatst ingedrukte toets
1	23561	REPDEL	tijd in 50ste sec. (60ste in de USA) dat een toets ingedrukt moet worden, alvorens hij begint te repeteren. Normaal 35, maar u kunt andere waarden POKEn
1	23562	REPPER	tijd in 50ste sec. (60ste in de USA) tussen twee repetities van een toets; oorspronkelijk 5
N2	23563	DEFADD	adres van de argumenten van een zelf gedefinieerde functie, indien ze op dat ogenblik wordt geëvalueerd; anders 0

N1	23565	KDATA	2de byte van de kleur-codes die worden ingetypt
N2	23566	TVDATA	bevat de bytes die horen bij de kleur-codes, AT en TAB die naar de TV gaan
X38	23568	STRMS	adressen van kanalen die aan streams verbonden zijn
2	23606	CHARS	256 minder dan het adres van de karakterset (beginnend bij "spatie", tot en met het (c) symbool). Normaal in ROM, maar u kunt zelf een karakterset maken en CHARS er naar laten verwijzen
1	23608	RASP	lengte van de waarschuwingstoon
1	23609	PIP	lengte van de toets-klik
1	23610	ERR_NR	1 minder dan de foutmeldings-code. Begint bij 255 (-1), zodat PEEK 23610, 255 oplevert.
X1	23611	FLAGS	diverse vlaggen voor het BASIC systeem
X1	23612	TVFLAG	vlaggen i.v.m. de TV
X2	23613	ERR_SP	adres op de stack, waar adres staat dat gebruikt zal worden bij een foutmelding
N2	23615	LISTSP	adres van het RETURN-adres na een automatische listing
N1	23617	MODE	bepaalt de cursor (K,L,C,E of G)
2	23618	NEWPPC	volgende uit te voeren regel
1	23620	NSPPC	hoeveelste statement in de volgende uit te voeren regel. Eerst NEWPPC POKEn en dan NSPPC, forceert een sprong naar een bepaald statement op een bepaalde regel
2	23621	PPC	regelnummer waarop het statement staat dat momenteel wordt uitgevoerd
1	23623	SUBPPC	nummer van het statement uit PPC
1	23624	BORDCR	borderkleur x8; ook de attributen die die voor het onderste deel van het scherm worden gebruikt
2	23625	E_PPC	nummer van courante regel (met cursor)
X2	23627	VARS	adres van variabelen
N2	23629	DEST	adres van variabele bij toewijzing
X2	23631	CHANS	adres van gegevens over kanalen
X2	23633	CURCHL	adres van informatie die momenteel voor input/output wordt gebruikt
X2	23635	PROG	adres van BASIC programma
X2	23637	NXTLIN	adres van volgende regel in programma
X2	23639	DATADD	adres van teken na het laatst gelezen DATA-item
X2	23641	E_LINE	adres van commando dat wordt ingetypt
2	23643	K_CUR	adres van de cursor
X2	23645	CH_ADD	adres van volgende karakter dan moet geïnterpreteerd worden - het karakter na het argument van PEEK, of het ENTER-karakter op het einde van een POKE
2	23647	X_PTR	adres van het karakter na het knipperende "?" bij een fout
X2	23649	WORKSP	adres van tijdelijk werkgeheugen
X2	23651	STKBOT	adres van bodem van calculator-stack
X2	23653	STKEND	adres van begin van vrije ruimte
N1	23655	BREG	het B-register van de calculator
N2	23656	MEM	adres van gebied, dat de calculator als geheugen gebruikt. Gewoonlijk MEMBOT, maar niet altijd
1	23658	FLAGS2	meer vlaggen
X1	23659	DF_SZ	aantal regels (inclusief 1 lege) in het onderste deel van het scherm
2	23660	STOP	nummer van de programmaregel die bij een automatische listing bovenaan het scherm moet komen
2	23662	OLDPPC	regelnummer waar CONTINUE verdergaat

1	23664	OSPCC	nummer van het statement op de regel waar CONTINUE verdergaat
N1	23665	FLAGX	diverse vlaggen
N2	23666	STRLEN	lengte van doelstring bij toewijzing van een string-variabele
N2	23668	T_ADDR	adres van volgende item in de syntax-tabel (kunt u waarschijnlijk niets mee)
2	23670	SEED	grondtal voor RND. Deze variabele wordt ingevuld door RANDOMIZE
3	23672	FRAMES	teller van 3 bytes, wordt elke 20 msec. opgehoogd; minst belangrijke byte eerst (zie deel 18 van dit hoofdstuk)
2	23675	UDG	adres van eerste UDG. Dit kunt u wijzigen om ruimte te winnen door het aantal UDG's kleiner te maken
1	23677	COORDS	x-coördinaat van laatst geplotte pixel
1	23678		y-coördinaat van laatst geplotte pixel
1	23679	P_POSN	33-koloms getal voor printer-positie
1	23680	PR_CC	minst belangrijke byte van het adres van de volgende positie in de printer-buffer waar LPRINT kan printen
1	23681		niet gebruikt
2	23682	ECHO_E	kolomnummer (33 kolommen) en regel-nummer (24 regels) van het einde van de input-buffer (onderste deel van het scherm)
2	23684	DF_CC	adres in de display file voor de PRINT-positie
2	23686	DF_CCL	zoals DF_CC maar voor het onderste deel van het scherm
X1	23688	S_POSN	kolomnummer (33 koloms) voor PRINT
X1	23689		regelnummer (24 regels) voor PRINT
X2	23690	SPOSNL	zoals S_POSN, voor het onderste deel
1	23692	SCR_CT	scroll-teller. Bevat altijd 1 meer dan het aantal regels die gescrolld worden, vooraleer de vraag "scroll?" te stellen. Door dit adres met een getal groter dan 1 (bv. 255) te POKEn, zal het scherm blijven scrollen zonder de vraag te stellen.
1	23693	ATTR_P	permanent ingestelde kleuren enz., zoals ze door kleur-statements werden ingegeven
1	23694	MASK_P	masker voor transparante kleuren enz. Elk bit dat 1 is, geeft aan dat het overeenkomstige attribuut niet uit ATTR_P gehaald moet worden, maar uit de informatie die al op het scherm staat.
N1	23695	ATTR_T	tijdelijk ingestelde kleuren enz., zoals opgegeven door kleur-commando's in een PRINT-statement
N1	23696	MASK_T	zoals MASK_P, maar tijdelijk
1	23697	P_FLAG	meer vlaggen
N30	23698	MEMBOT	gebied dat door de calculator als geheugen wordt gebruikt, om getallen op te slaan die niet op de calculator-stack geplaatst kunnen worden zonder hinderlijk te zijn
2	23728		niet gebruikt
2	23730	RAMTOP	adres van laatste byte dat door het BASIC systeem dan gebruikt worden
2	23732	P_RAMT	adres van laatste byte van de aanwezige en bruikbare RAM

OEFENING
++++++++

1. Dit programma laat u 22 bytes zien uit het variabelen-geheugen (vanaf de inhoud van de systeemvariabele VARS en hoger)

```
10 FOR n=0 TO 21
20 PRINT PEEK (PEEK 23627+256*PEEK23628+n)
30 NEXT n
```

Probeer of u de controlevariabele n kunt terugvinden, aan de hand van de beschrijving die we voordien gaven. Wijzig dan regel 20 in :

```
20 PRINT PEEK (23755+n)
```

en u ziet de eerste 22 bytes van het programma-geheugen. Probeer te begrijpen wat u ziet.

DEEL 26 : MACHINETAAL GEBRUIKEN

Inhoud ...

Dit deel is bestemd voor lezers die Z80 machinetaal begrijpen, dat is de instructieset die de Z80 processor zelf gebruikt. Indien u niets van machinetaal begrijpt, maar dat wel zou willen, dan dient u n van de vele beschikbare boeken aan te schaffen. Neem een boek met een titel in de aard van "Z80 machinetaal (of Assembler) voor beginnelingen". Indien het boek dan nog over de +2 of een andere Spectrum gaat, des te beter.

Gewoonlijk worden programma's in machinetaal, ontwikkeld in Assembler. Dat ziet er geheimzinnig uit, hoewel het met een beetje oefenen niet zo moeilijk te begrijpen is. In deel 27 van dit hoofdstuk vindt u een overzicht van de instructies in Assembler. Om die programma's op de +2 te runnen, moet u ze coderen in een reeks van bytes : in deze vorm heet het programma "machinecode" of "machinetaal". Die omzetting in bytes wordt gewoonlijk door de computer zelf uitgevoerd, met behulp van een programma dat "assembler" heet. De +2 beschikt niet over een ingebouwde assembler, maar u kunt er verscheidene kopen op cassette. Zonder een assembler, zult u zelf voor de omzetting moeten zorgen, wanneer het programma niet te lang is.

Nemen we bijvoorbeeld het programma :

```
ld bc,99
ret
```

Dat programma laadt het BC registerpaar met 99, em keert daarna terug. In machinecode worden dit vier bytes : 1, 99, 0 (voor de instructie ld bc,99) en 201 (voor ret). Indien u de codes 1 en 201 opzoekt in deel 27, dan zult u zien dat 1 staat voor ld bc,NN (NN is een willekeurig getal van twee bytes) en dat 201 staat voor ret.

Nu hebt u dus een machinetaal-programma. De volgende stap is, het in de computer brengen. Een assembler doet dit automatisch. U moet eerst beslissen wáár in het geheugen het programma zal staan. U kunt het beste ruimte ervoor vrijmaken tussen het BASIC-gebied en de UDG's.

Typ daarom :

```
CLEAR 65267
```

waardoor u 100 bytes vrij maakt (dat is ruim genoeg) vanaf adres 65268.

Om de machinecode in te voeren, zou u een programma kunnen gebruiken zoals dit :

```
10 LET a=65268
20 READ n: POKEa,n
30 LET a=a+1: GO TO 20
40 DATA 1,99,0,201
```

Dat programma zal stoppen met de foutmelding "E Out of DATA", nadat het de vier bytes heeft ingevoerd.

Om de machinecode te laten uitvoeren, gebruikt u de functie USR, maar dit keer met een numeriek argument, dat wil zeggen het

start-adres. Het resultaat van de functie is dan de waarde van het bc-registerpaar, bij de terugkeer naar BASIC uit het machinetaal-programma. Indien u dus typt :

```
PRINT USR 65268
```

krijgt u als antwoord : 99.

Het RETURN-adres naar BASIC wordt op de gewone manier op de stack gezet, zodat u met een RET-instructie kunt terugkeren.

In een machinetaal-routine die met het BASIC interrupt-mechanisme zal werken, mag u niet de IY of I registers gebruiken. U mag evenmin I laden met waarden tussen 40h en 7Fh (ook al gebruikt u IM2 niet). Indien u RAM 4 tot 7 wilt inschakelen in het gebied tussen C000h en FFFFh, dient u voor I ook de waarden tussen C0h en FFh te vermijden. De reden daarvoor is een interactie tussen de video-controller en het "refresh"-mechanisme van de Z80. Anders kunt u onverklaarbare crashes krijgen, verstoring van het tv-beeld of andere ongewenste verschijnselen. In IM2 kunt u dus enkel de interrupts doorsturen naar het gebied tussen 8000h en BFFFh, tenzij u op elk moment precies weet hoe de geheugenbanken geschakeld staan.

Het programmeren van een systeem met banken (zoals de +2) in machinecode, kent een aantal klassieke valkuilen. Indien u problemen hebt, controleer dan of uw stack niet "weggebladerd" wordt door het paging-mechanisme, tijdens interrupts, of dat uw interrupt-routine wel altijd dáár is waar u ze verwacht. Het is aan te raden om tijdens de paging de interrupts uit te schakelen. We bevelen ook aan om de stand van de bankregisters ergens bij te houden in RAM die nit geschakeld wordt, omdat u die poort alleen máar kunt beschrijven, niet uitlezen. BASIC en de editor gebruiken de systeemvariabele BANK_M.

U kunt uw machinetaal-programma gemakkelijk saven :

```
SAVE "naam" CODE 65268,4
```

Op het eerste gezicht is er geen manier om dit programma op zo een manier te saven, dat het bij inladen automatisch opstart. Dit kunt u omzeilen door een kort BASIC programma, zoals dit :

```
10 LOAD "" CODE 65268,4
20 PRINT USR 65268
```

op de band te zetten vóór de code, met bijvoorbeeld het commando

```
SAVE "loader" LINE 0
```

waarna u de machinecode kunt saven met bijvoorbeeld

```
SAVE "m code" CODE 65268,4
```

Daarna kunt u de machinecode vanuit BASIC láten runnen door het ene commando :

```
LOAD "loader"
```

dat het BASIC programma inlaadt en het automatisch opstart, waardoor de machinetaal wordt ingeladen en gerund.

DEEL 27 : KARAKTERSET VAN DE SPECTRUM

Inhoud ...
- Controlecodes
- Karakters
- Z80 assembler-instructies

Hieronder volgt de volledige karakterset van de Spectrum, met de codes in decimale en in hexadecimale notering. Indien u zich de codes voorstelt als machinecode-instructies voor de Z80, dan vindt u in de rechtse kolommen de overeenkomstige assembler-instructies. U weet wellicht dat bepaalde Z80-instructies samengestelde instructies zijn, die beginnen met CBh of EDh. Die vindt u in de uiterst rechtse kolommen. Indien er een verschil bestaat tussen de code van een karakter in 48 BASIC en in 128 BASIC, dan wordt de betekenis in 48 BASIC tussen haakjes gegeven na de betekenis in 128 BASIC.

DEC	HEX	KARAKTER	Z80 assembler	NA 203/CB	NA 237/ED
0	00	niet gebruikt	NOP	RLC B	
1	01	niet gebruikt	LD BC,nn	RLC C	
2	02	niet gebruikt	LD(BC),A	RLC D	
3	03	niet gebruikt	INC BC	RLC E	
4	04	niet gebruikt	INC B	RLC H	
5	05	niet gebruikt	DEC B	RLC L	
6	06	PRINT komma	LD B,n	RLC(HL)	
7	07	EDIT	RLCA	RLC A	
8	08	cursor links	EX AF,AF'	RRC B	
9	09	cursor rechts	ADD HL,BC	RRC C	
10	0A	cursor omhoog	LD A,(BC)	RRC D	
11	0B	cursor omlaag	DEC BC	RRC E	
12	0C	DELETE	INC C	RRC H	
13	0D	ENTER	DEC C	RRC L	
14	0E	getal-marker	LD C,n	RRC (HL)	
15	0F	niet gebruikt	RRCA	RRC A	
16	10	INK-contr.code	DJNZ DIS	RL B	
17	11	PAPER-code	LD DE,nn	RL C	
18	12	FLASH-code	LD(DE),A	RL D	
19	13	BRIGHT-code	INC DE	RL E	
20	14	INVERSE-code	INC D	RL H	
21	15	OVER-cont.code	DEC D	RL L	
22	16	AT-contr.code	LD D,n	RL (HL)	
23	17	TAB-contr.code	RLA	RL A	
24	18	niet gebruikt	JR DIS	RR B	
25	19	niet gebruikt	ADD HL,DE	RR C	
26	1A	niet gebruikt	LD A, (DE)	RR D	
27	1B	niet gebruikt	DEC DE	RR E	
28	1C	niet gebruikt	INC E	RR H	
29	1D	niet gebruikt	DEC E	RR L	
30	1E	niet gebruikt	LD E,n	RR (HL)	
31	1F	niet gebruikt	RRA	RR A	
32	20	spatie	JR NZ,DIS	SLA B	

DEC	HEX	KARAKTER	Z80 assembler	NA 203/CB	NA 237/ED
33	21	!	LD HL,nn	SLA C	
34	22	"	LD(nn),HL	SLA D	
35	23	#	INC HL	SLA E	
36	24	$	INC H	SLA H	
37	25	%	DEC H	SLA L	
38	26	&	LD H,n	SLA (HL)	
39	27	'	DAA	SLA A	
40	28	(	JR Z,DIS	SRA B	
41	29	)	ADD HL,HL	SRA C	
42	2A	*	LD HL,(nn)	SRA D	
43	2B	+	DEC HL	SRA E	
44	2C	,	INC L	SRA H	
45	2D	-	DEC L	SRA L	
46	2E	.	LD L,n	SRA (HL)	
47	2F	/	CPL	SRA A	
48	30	0	JR NC,DIS		
49	31	1	LD SP,nn		
50	32	2	LD(nn),A		
51	33	3	INC SP		
52	34	4	INC(HL)		
53	35	5	DEC(HL)		
54	36	6	LD(HL),n		
55	37	7	SCF		
56	38	8	JR C,DIS	SRL B	
57	39	9	ADD HL,SP	SRL C	
58	3A	:	LD A,(nn)	SRL D	
59	3B	;	DEC SP	SRL E	
60	3C	<	INC A	SRL H	
61	3D	=	DEC A	SRL L	
62	3E	>	LD A,n	SRL (HL)	
63	3F	?	CCF	SRL A	
64	40	@	LD B,B	BIT 0,B	IN B,(C)
65	41	A	LD B,C	BIT 0,C	OUT(C),B
66	42	B	LD B,D	BIT 0,D	SBC HL,BC
67	43	C	LD B,E	BIT 0,E	LD(nn),BC
68	44	D	LD B,H	BIT 0,H	NEG
69	45	E	LD B,L	BIT 0,L	RETN
70	46	F	LD B,(HL)	BIT 0,(HL)	IM 0
71	47	G	LD B,A	BIT 0,A	LD I,A
72	48	H	LD C,B	BIT 1,B	IN C,(C)
73	49	I	LD C,C	BIT 1,C	OUT(C),C
74	4A	J	LD C,D	BIT 1,D	ADC HL,BC
75	4B	K	LD C,E	BIT 1,E	LD BC,(nn)
76	4C	L	LD C,H	BIT 1,H	
77	4D	M	LD C,L	BIT 1,L	RETI
78	4E	N	LD C,(HL)	BIT 1,(HL)	
79	4F	O	LD C,A	BIT 1,A	LD R,A
80	50	P	LD D,B	BIT 2,B	IN D,(C)
81	51	Q	LD D,C	BIT 2,C	OUT(C),D
82	52	R	LD D,D	BIT 2,D	SBC HL,DE
83	53	S	LD D,E	BIT 2,E	LD(nn),DE
84	54	T	LD D,H	BIT 2,H	
85	55	U	LD D,L	BIT 2,L	
86	56	V	LD D,(HL)	BIT 2,(HL)	IM 1
87	57	W	LD D,A	BIT 2,A	LD A,I
88	58	X	LD E,B	BIT 3,B	IN E,(C)
89	59	Y	LD E,C	BIT 3,C	OUT(C),E
90	5A	Z	LD E,D	BIT 3,D	ADC HL,DE
91	5B	[	LD E,E	BIT 3,E	LD DE,(nn)
92	5C	\	LD E,H	BIT 3,H	
93	5D	]	LD E,L	BIT 3,L	

DEC	HEX	KARAKTER	Z80 assembler:	-NA 203/CB	-NA 237/ED
94	5E	↑	LD E,(HL)	BIT 3,(HL)	IM 2
95	5F		LD E,A	BIT 3,A	LD A,R
96	60	£	LD H,B	BIT 4,B	IN H,(C)
97	61	a	LD H,C	BIT 4,C	OUT(C),H
98	62	b	LD H,D	BIT 4,D	SBC HL,HL
99	63	c	LD H,E	BIT 4,E	LD(nn),HL
100	64	d	LD H,H	BIT 4,H	
101	65	e	LD H,L	BIT 4,L	
102	66	f	LD H,(HL)	BIT 4,(HL)	
103	67	g	LD H,A	BIT 4,A	RRD
104	68	h	LD L,B	BIT 5,B	IN L,(C)
105	69	i	LD L,C	BIT 5,C	OUT(C),L
106	6A	j	LD L,D	BIT 5,D	ADC HL,HL
107	6B	k	LD L,E	BIT 5,E	LD HL,(nn)
108	6C	l	LD L,H	BIT 5,H	
109	6D	m	LD L,L	BIT 5,L	
110	6E	n	LD L,(HL)	BIT 5,(HL)	
111	6F	o	LD L,A	BIT 5,A	RLD
112	70	p	LD (HL),B	BIT 6,B	IN F,(C)
113	71	q	LD (HL),C	BIT 6,C	
114	72	r	LD (HL),D	BIT 6,D	SBC HL,SP
115	73	s	LD (HL),E	BIT 6,E	LD (nn),SP
116	74	t	LD (HL),H	BIT 6,H	
117	75	u	LD (HL),L	BIT 6,L	
118	76	v	HALT	BIT 6,(HL)	
119	77	w	LD (HL),A	BIT 6,A	
120	78	x	LD A,B	BIT 7,B	IN A,(C)
121	79	y	LD A,C	BIT 7,C	OUT (C),A
122	7A	z	LD A,D	BIT 7,D	ADC HL,SP
123	7B	{	LD A,E	BIT 7,E	LD SP,(nn)
124	7C	|	LD A,H	BIT 7,H	
125	7D	}	LD A,L	BIT 7,L	
126	7E	~	LD A,(HL)	BIT 7,(HL)	
127	7F	©	LD A,A	BIT 7,A	
128	80	▯	ADD A,B	RES 0,B	
129	81	▯	ADD A,C	RES 0,C	
130	82	▯	ADD A,D	RES 0,D	
131	83	▯	ADD A,E	RES 0,E	
132	84	▯	ADD A,H	RES 0,H	
133	85	▯	ADD A,L	RES 0,L	
134	86	▯	ADD A,(HL)	RES 0,(HL)	
135	87	▯	ADD A,A	RES 0,A	
136	88	▯	ADC A,B	RES 1,B	
137	89	▯	ADC A,C	RES 1,C	
138	8A	▯	ADC A,D	RES 1,D	
139	8B	▯	ADC A,E	RES 1,E	
140	8C	▯	ADC A,H	RES 1,H	
141	8D	▯	ADC A,L	RES 1,L	
142	8E	▯	ADC A,(HL)	RES 1,(HL)	
143	8F	▯	ADC A,A	RES 1,A	
144	90	UDG (A)	SUB B	RES 2,B	
145	91	UDG (B)	SUB C	RES 2,C	
146	92	UDG (C)	SUB D	RES 2,D	
147	93	UDG (D)	SUB E	RES 2,E	
148	94	UDG (E)	SUB H	RES 2,H	
149	95	UDG (F)	SUB L	RES 2,L	
150	96	UDG (G)	SUB (HL)	RES 2,(HL)	
151	97	UDG (H)	SUB A	RES 2,A	
152	98	UDG (I)	SBC B	RES 3,B	
153	99	UDG (J)	SBC C	RES 3,C	
154	9A	UDG (K)	SBC D	RES 3,D	
155	9B	UDG (L)	SBC E	RES 3,E	
156	9C	UDG (M)	SBC H	RES 3,H	

DEC	HEX	KARAKTER	Z80 assembler	NA 203/CB	NA 237/ED
157	9D	UDG (N)	SBC L	RES 3,L	
158	9E	UDG (O)	SBC (HL)	RES 3,(HL)	
159	9F	UDG (P)	SBC A	RES 3,A	
160	A0	UDG (Q)	AND B	RES 4,B	LDI
161	A1	UDG (R)	AND C	RES 4,C	CPI
162	A2	UDG (S)	AND D	RES 4,D	INI
163	A3	SPECTRUM of (T)	AND E	RES 4,E	OUTI
164	A4	PLAY of (U)	AND H	RES 4,H	
165	A5	RND	AND L	RES 4,L	
166	A6	INKEY$	AND (HL)	RES 4,(HL)	
167	A7	PI	AND A	RES 4,A	
168	A8	FN	XOR B	RES 5,B	LDD
169	A9	POINT	XOR C	RES 5,C	CPD
170	AA	SCREEN$	XOR D	RES 5,D	IND
171	AB	ATTR	XOR E	RES 5,E	OUTD
172	AC	AT	XOR H	RES 5,H	
173	AD	TAB	XOR L	RES 5,L	
174	AE	VAL$	XOR (HL)	RES 5,(HL)	
175	AF	CODE	XOR A	RES 5,A	
176	B0	VAL	OR B	RES 6,B	LDIR
177	B1	LEN	OR C	RES 6,C	CPIR
178	B2	SIN	OR D	RES 6,D	INIR
179	B3	COS	OR E	RES 6,E	OTIR
180	B4	TAN	OR H	RES 6,H	
181	B5	ASN	OR L	RES 6,L	
182	B6	ACS	OR (HL)	RES 6,(HL)	
183	B7	ATN	OR A	RES 6,A	
184	B8	LN	CP B	RES 7,B	LDDR
185	B9	EXP	CP C	RES 7,C	CPDR
186	BA	INT	CP D	RES 7,D	INDR
187	BB	SQR	CP E	RES 7,E	OTDR
188	BC	SGN	CP H	RES 7,H	
189	BD	ABS	CP L	RES 7,L	
190	BE	PEEK	CP (HL)	RES 7,(HL)	
191	BF	IN	CP A	RES 7,A	
192	C0	USR	RET NZ	SET 0,B	
193	C1	STR$	POP BC	SET 0,C	
194	C2	CHR$	JP NZ,nn	SET 0,D	
195	C3	NOT	JP nn	SET 0,E	
196	C4	BIN	CALL NZ,nn	SET 0,H	
197	C5	OR	PUSH BC	SET 0,L	
198	C6	AND	ADD A,n	SET 0,(HL)	
199	C7	<=	RST 0	SET 0,A	
200	C8	>=	RET Z	SET 1,B	
201	C9	<>	RET	SET 1,C	
202	CA	LINE	JP Z,nn	SET 1,D	
203	CB	THEN		SET 1,E	
204	CC	TO	CALL Z,nn	SET 1,H	
205	CD	STEP	CALL nn	SET 1,L	
206	CE	DEF FN	ADC A,n	SET 1,(HL)	
207	CF	CAT	RST 8	SET 1,A	
208	D0	FORMAT	RET NC	SET 2,B	
209	D1	MOVE	POP DE	SET 2,C	
210	D2	ERASE	JP NC,nn	SET 2,D	
211	D3	OPEN#	OUT (n),A	SET 2,E	
212	D4	CLOSE#	CALL NC,nn	SET 2,H	
213	D5	MERGE	PUSH DE	SET 2,L	
214	D6	VERIFY	SUB n	SET 2,(HL)	
215	D7	BEEP	RST 16	SET 2,A	
216	D8	CIRCLE	RET C	SET 3,B	
217	D9	INK	EXX	SET 3,C	
218	DA	PAPER	JP C,nn	SET 3,D	

DEC	HEX	KARAKTER	Z80 assembler	NA 203/CB	NA 237/ED
219	DB	FLASH	IN A,(n)	SET 3,E	
220	DC	BRIGHT	CALL C,nn	SET 3,H	
221	DD	INVERSE	PREFIX voor IX	SET 3,L	
222	DE	OVER	SBC A,n	SET 3,(HL)	
223	DF	OUT	RST 24	SET 3,A	
224	E0	LPRINT	RET PO	SET 4,B	
225	E1	LLIST	POP HL	SET 4,C	
226	E2	STOP	JP PO,nn	SET 4,D	
227	E3	READ	EX (SP),HL	SET 4,E	
228	E4	DATA	CALL PO,nn	SET 4,H	
229	E5	RESTORE	PUSH HL	SET 4,L	
230	E6	NEW	AND n	SET 4,(HL)	
231	E7	BORDER	RST 32	SET 4,A	
232	E8	CONTINUE	RET PE	SET 5,B	
233	E9	DIM	JP (HL)	SET 5,C	
234	EA	REM	JP PE,nn	SET 5,D	
235	EB	FOR	EX DE,HL	SET 5,E	
236	EC	GOTO	CALL PE,nn	SET 5,H	
237	ED	GOSUB		SET 5,L	
238	EE	INPUT	XOR n	SET 5,(HL)	
239	EF	LOAD	RST 40	SET 5,A	
240	F0	LIST	RET P	SET 6,B	
241	F1	LET	POP AF	SET 6,C	
242	F2	PAUSE	JP P,nn	SET 6,D	
243	F3	NEXT	DI	SET 6,E	
244	F4	POKE	CALL P,nn	SET 6,H	
245	F5	PRINT	PUSH AF	SET 6,L	
246	F6	PLOT	OR n	SET 6,(HL)	
247	F7	RUN	RST 48	SET 6,A	
248	F8	SAVE	RET M	SET 7,B	
249	F9	RANDOMIZE	LD SP,HL	SET 7,C	
250	FA	IF	JP M,nn	SET 7,D	
251	FB	CLS	EI	SET 7,E	
252	FC	DRAW	CALL M,nn	SET 7,H	
253	FD	CLEAR	PREFIX voor IY	SET 7,L	
254	FE	RETURN	CP n	SET 7,(HL)	
255	FF	COPY	RST 56	SET 7,A	

TIJDENTABEL

NIET alle instructies duren even lang. Hieronder staat een tijdentabel met de tijden in klokpulsen.
De Z80 chip in de spectrum krijgt 4 miljoen klokpulsen per seconde, dus U moet de hieronderstaande tijden uit de tabel met 1/4000000 vermenigvuldigen om de tijd in seconden te krijgen.

LD A,r	4	INC r	4	
LD r,n	7	INC (HL)	11	
LD r,(HL)	7	RLA	4	
LD A,(nn)	13	JP nn	10	
LD p,nn	10	JR e	12	
LD HL,(nn)	16	JR c,e	12	(niet springen:7)
PUSH p	11	CALL nn	17	
POP p	10	RET	10	
ADD A,r	4			

Gelijksoortige instructies hebben dezelfde tijdsduur.

DEEL 28 : FOUTMELDINGEN EN MEDEDELINGEN

Inhoud ...

Elke keer de +2 ophoudt met het uitvoeren van BASIC, zet hij een mededeling onderaan op het scherm. Deze mededelingen geven aan, waarom hij stopte : om een voor de hand liggende reden, of omdat hij een fout ontdekte.

De mededeling bestaat uit een code (een getal of een letter, die verwijzen naar de volgende tabel), een korte tekst die uitlegt wat er aan de hand is, en het regelnummer waar de +2 stopte (en het nummer van het statement op die regel). Een direkt commando heeft regelnummer 0. Statement 1 staat in het begin van een regel, statement 2 komt na de eerste dubbele punt (of THEN), en zo verder.

Hoe CONTINUE reageert, hangt sterk af van de mededeling in kwestie. Gewoonlijk herbegint CONTINUE de uitvoering van het programma op de regel, vanaf het statement die in de mededeling werden opgegeven. Uitzonderingen zijn de mededelingen met code 0, 9 en D.

Nu volgt een tabel met alle mededelingen die kunen voorkomen. In de tabel staat ook, in welke omstandigheden die mededeling gegeven kan worden; meer daarover vindt u dan in deel 30 van dit hoofdstuk. De foutmelding "A Invalid argument" (ongeldig argument) bijvoorbeeld, kan voorkomen bij SQR, IN, ACS en ASN. In deel 30 vindt u onder deze vier hoofdingen welke argumenten ongeldig zijn.

CODE	Tekst & Betekenis	Situatie
0	OK	overal
	Programma correct afgewerkt, of een sprong naar een regelnummer dat hoger is dan het hoogste dat in het programma voorkomt. Het regelnummer en het statement waar CONTINUE verdergaat, wordt niet gewijzigd.	
1	Next without FOR	NEXT
	De controlevariabele bestaat niet (werd niet door een FOR-statement geïnitialiseerd), maar er bestaat wel een gewone variabele met die naam	
2	Variable not found	overal
	Bij een gewone variabele wil dit zeggen dat hij wordt gebruikt vooraleer hij werd toegewezen door LET, READ of INPUT of voordat hij van cassette werd ingeladen, of door een FOR werd geïnitialiseerd. Bij een array-variabele wil dit zeggen dat hij nog niet gedimensioneerd werd door DIM, of van cassette werd ingeladen.	

3	Subscript wrong De opgegeven index ligt buiten het bereik van het array, of het aantal opgegeven indexen klopt niet. Indien de index negatief is of groter dat 65535, wordt foutmelding B gegeven	variabele met index of een substring
4	Out of memory Dit wil zeggen dat er niet voldoende plaats vrij is in het geheugen voor wat u wilt doen. Blijkt de computer echt vast te zitten, dan kan het nodig zijn om de commando-regel te wissen met DELETE, en dan een paar programmaregels weg te halen (met de bedoeling ze nadien wer terug te typen), om zodoende de nodige ruimte te verkrijgen.	LET, INPUT, FOR,DIM, GOSUB, LOAD, MERGE. Ook bij de evaluatie van een uitdrukking
5	Out of Screen Een INPUT-statement probeerde meer dan 23 regels in het onderste deel van het scherm te zetten. Kan ook voorkomen bij PRINT AT 22,xx.	INPUT, PRINT AT
6	Number too big Een of andere bewerking leverde een resultaat op dat groter is dan +/- 10^38	rekenkundige bewerkingen
7	RETURN without GO SUB Er werd een RETURN mr gevonden dan er GO SUBs werden opgegeven.	RETURN
8	End of file	Microdrive-bewerkingen
9	STOP statement Hierna gaat CONTINUE door met het statement dat volgt op het STOP-statement, en herhaalt niet datzelfde statement.	STOP
A	Invalid argument Om een of andere reden kan het argument niet gebruikt worden bij die functie.	SQR, LN, ASN, ACS, USR met een string-argument
B	Integer out of range Wanneer een geheel getal moet berekend worden, wordt het argument dat met vlottende komma werd genoteerd, afgerond tot het dichtstbije geheel getal. Indien dit buiten bepaalde, voor die functie of dat commando toegestane grenzen ligt, dan wordt foutmelding B gegeven. Wat indexen betreft, zie ook foutmelding nummer 3	RUN, RANDOMIZE, POKE, DIM, GO SUB, GO TO, LIST, LLIST, PAUSE, PLOT, CHR$, PEEK, USR met numeriek argument, indexen van arrays

C	Nonsense in Basic De tekst van het string-argument vormt geen geldige uitdrukking. Ook, indien het argument voor een functie of een commando overdreven duidelijk fout is.	VAL, VAL$
D	BREAK - CONT repeats Komt voor indien BREAK werd ingedrukt tijdens de werking van een of ander rand-apparaat. Het commando CONTINUE na deze foutmelding is normaal, in die zin dat het onderbroken statement herhaald wordt. Vergelijk met code L.	LOAD, SAVE, VERIFY, MERGE. Ook indien na "scroll?" de N, BREAK of de spatiebalk wordt ingedrukt
E	Out of DATA U probeerde verder te lezen dan de DATA-lijst lang was	READ
F	Invalid file name Komt voor bij SAVE, gevolgd door de nulstring of een naam die meer dan 10 karakters bevatte.	SAVE
G	No room for line Er is in het geheugen niet meer voldoende ruimte vrij om de regel die u intypte, onder te brengen.	Bij intypen van een programma-regel
H	STOP in INPUT U begon een INPUT met STOP. In tegenstelling tot code 9, zal CONTINUE bij code H wel normaal werken, en het INPUT-statement herhalen.	INPUT
I	FOR without NEXT Een bepaalde lus moest niet uitgevoerd worden (bv. FOR n=1 TO 0), en het overeenkomstige NEXT-statement werd niet gevonden	FOR
J	Invalid I/O device (Ongeldig I/O apparaat)	Microdrive bewerkingen
K	Invalid colour Het opgegeven getal is geen geldige waarde voor dat bepaalde kleur-commando.	INK, PAPER, BORDER, FLASH, BRIGHT, OVER, INVERSE en na de overeenkomstige controle codes
L	Break into program De BREAK-toets werd ingedrukt. Dit wordt ontdekt tussen elke twee statements. Het regelnummer en het statement-nummer in de melding verwijzen naar het statement voor het indrukken van BREAK. CONTINUE gaat	overal

door vanaf het daaropvolgende statement.

M	RAMTOP no good Het getal dat voor RAMTOP werd opgegeven is ofwel te hoog, ofwel te laag.	CLEAR, of soms RUN
N	Statement lost Een sprong naar een statement dat niet meer bestaat.	RETURN, NEXT, CONTINUE,
O	Invalid stream	Microdrive bewerkingen
P	FN without DEF De functie die werd gebruikt, werd niet eerst in het programma gedefinieerd.	FN
Q	Parameter error Aantal argumenten klopt niet, of een ervan is niet het juiste type (string in plaats van getal of andersom)	FN
R	Tape loading error Er werd een bestand op de cassette gevonden, maar dat kon om een of andere reden niet worden ingelezen of geverifieerd.	VERIFY, LOAD, MERGE
a	MERGE error Om een of andere reden kon MERGE ! niet uitgevoerd worden : grootte klopte niet of type klopte niet	MERGE !
b	Wrong file type Bij een RAMdisk-bewerking werd de naam opgegeven van een bestand van een verkeerd type, bv. een CODE-bestand in het commando LOAD ! "naam"	MERGE !, LOAD!,
c	CODE error Indien de code werd ingeladen, zou ze boven het hoogste adres uitkomen.	LOAD ! naam CODE
d	To many brackets In een van de argumenten staan rond een te herhalen frase teveel haakjes.	PLAY
e	File already exists De naam die u bij SAVE ! opgaf, werd al gebruikt.	SAVE !
f	Invalid name De naam die u opgaf, is de nulstring, of hij telt meer dan 10 karakters.	ERASE !

h	File does not exist	LOAD !, ERASE !
	Op de RAMdisk staat geen bestand onder de naam die u opgaf.	
i	Invalid device	FORMAT
	De naam die u na FORMAT opgaf, bestaat niet of komt niet overeen met een bestaand randapparaat.	
j	Invalid baud rate	FORMAT
	U zette de baudrate voor de RS232 op nul.	
k	Invalid note name	PLAY
	PLAY las een noot of een commando dat het niet herkende, ofwel een commando in kleine letters geschreven	
l	Number too big	PLAY
	Een bepaalde parameter bij een commando is een orde van grootte t groot.	
m	Note out of range	PLAY
	Een reeks kruisen of mollen heeft een noot buiten het bereik van de geluids-chip gebracht.	
n	Out of range	PLAY
	Een parameter van een commando is te groot of te klein. Indien de afwijking rg groot is, dan wordt melding 1 gegeven.	
o	To many tied notes	PLAY
	U probeerde om teveel noten aan elkaar te binden.	

DEEL 29 : ALGEMENE TECHNISCHE INFORMATIE

Inhoud ...
   Hardware

De +2 werd ontworpen rond de Z80A microprocessor, die werkt met een snelheid van 3.54 MHz (3.54 miljoen cycli per seconde).

Het geheugen van de +2 bestaat uit 32K ROM en 128 K RAM, georganiseerd in bladzijden van 16 K. De twee bladzijden ROM (0 en 1) liggen in de onderste 16 K (0-3FFFh) van het geheugen. De acht bladzijden RAM (0 tot 7) liggen in de bovenste 16 K van het geheugen (C000h-FFFFh). Bladzijde 5 van RAM ligt in het gebied 4000h-7FFFh, en bladzijde 2 van RAM ligt van 8000h tot BFFFh.

In realiteit is de ROM n enkele chip van 32K (zoals een 27256) die door het systeem als twee chips van 16K wordt behandeld. De RAM bestaat uit zestien 64K x 1 bit chips (4164). Enkele daarvan (banken 4 tot 7) worden tegelijk door het video-circuit (dat het TV-beeld produceert, meer daarover later) en de Z80A gebruikt. De overige acht (banken 0 tot 3) worden uitsluitend door de Z80A gebruikt, net zoals de ROM.

De ULA (Uncommitted Logic Array) verzorgt het merendeel van de I/O-bewerkingen, zoals toetsenbord, datacorder en scherm. De ULA zet bytes in het geheugen om in patronen en kleuren op het scherm, en maakt het mogelijk dat de Z80A het toetsenbord kan aftasten, en data van cassette kan lezen of erop schrijven.

Het driekanaals-geluid wordt geproduceerd door de AY-3-8912. Dat is een veelgebruikte geluidschip. Die bestuurt ook de poorten van de RS232, de MIDI en het losse toetsenbord. De werking ervan is vrij gecompliceerd. Indien u wilt experimenteren, zorg er dan voor dat u de technische gegevens ervan bij de hand hebt. Om te beginnen kan wellicht de volgende informatie volstaan.

De geluidschip bevat zestien registers. Die worden geselecteerd door eerst het registernummer naar de adrespoort te schrijven (I/O adres FFFDh, 65533 decimaal), en daarna de waarde van het register uit te lezen (zelfde adres) of een andere waarde erin te schrijven via het daartoe gereserveerde adres (adres BFFDh, 49149 decimaal). Wanneer een register eenmaal geselecteerd is, kan een willekeurig aantal keren gelezen of geschreven worden. De schrijf-poort moet enkel een nieuwe waarde krijgen, indien een ander register gewenst wordt.

De klokfrekwentie van de schakeling is 1.7734 MHz (+/- 0,01%).

De registers hebben de volgende functie :

R0 : fijnregeling van de toon van kanaal A
R1 : grove instelling van de toon van kanaal A
R2 : fijnregeling van de toon van kanaal B
R3 : grove instelling van de toon van kanaal B
R4 : fijnregeling van de toon van kanaal C
R5 : grove instelling van de toon van kanaal C

De toon van een bepaald kanaal, is een 12-bits waarde, die wordt samengesteld uit de som van D3-D0 van het "grove" register plus D7-D0 van het "fijne" register. De eenheid van toon is de klokfrekwentie gedeeld door 16 (d.w.z. 110.83 kHz). Met een tellerbereik van 12 bits, kunnen dus frekwenties tussen 27Hz en 110 kHz worden geproduceerd.

R6 : besturing van de ruisgenerator, D4-D0

De periode van de ruisbron wordt berekend door het getal dat wordt gevormd door de laagste 5 bits van het ruisregister met 1 te verminderen, bij elke klokperiode van het geluid, gedeeld door 16.

R7 : Menger en I/O-besturing

D7 : niet gebruikt
D6 : 1=input-poort, 0 = output-poort
D5 : ruis kanaal C
D4 : ruis kanaal B
D3 : ruis kanaal A
D2 : toon kanaal C
D1 : toon kanaal B
D0 : toon kanaal A

Dit register bestuurt zowel het mengen van ruis en toon in elk kanaal, als de richting waarin de 8-bits I/O-poort werkt. Indien een meng-bit in dit register 0 is, betekent dit, dat die bepaalde functie kan uitgevoerd worden.

R8 : volumecontrole kanaal A
R9 : volumecontrole kanaal B
RA : volumecontrole kanaal C

D4 : 1 = gebruik omhullende-generator
0 = gebruik D3-D0 voor amplitudo
D3-D0 : amplitudo

Deze drie registers besturen het volume van elk kanaal, en bepalen tegelijk of het al dan niet door de omhullende-registers word gemoduleerd.

RB : grove instelling van periode van omhullende
RC : fijnregeling van periode van omhullende

De acht-bits waarden van RB en RC worden opgeteld, waardoor een 16-bits getal ontstaat, dat naar nul wordt afgeteld. Daarbij is de eenheid gelijk aan 256 x de klokfrekwentie van het geluid. De frekwentie van de omhullende kan liggen tussen 0,1Hz en 6 kHz.

RD : besturing van de vorm van de omhullende

D3 : doorlopend
D2 : aanzet
D1 : afwisselend
D0 : permanent

De tekening van de omhullenden (zie deel 19 van dit hoofdstuk) illustreert grafisch op welke manieren dit register ingesteld kan worden.

DEEL 30 : DE BASIC

Inhoud ...

Getallen worden opgeslagen tot op 9 of 10 cijfers nauwkeurig. Het grootste getal dat nog verwerkt wordt, is ongeveer 10^38, en het kleinste (positieve) getal is ongeveer 4*10^-39.

De +2 slaat een getal op in de vorm van een binair getal met vlottende komma, met één byte voor de exponent, e (1`=e`=255) en vier bytes voor de mantisse (1/2≤m<1). Het getal wordt dus geschreven in de vorm : m*2^(e-128).

Aangezien 1/2≤m<1, zal het meest significante bit van de mantisse m, altijd 1 zijn. Daarom kunnen we het in de realiteit vervangen door een bit dat het teken aangeeft : 0 voor positief, 1 voor negatief.

Kleine gehele getallen worden op een speciale manier vorgesteld. Het eerste byte is 0, het tweede geeft het teken aan (0 of FFh) en het derde en vierde byte stellen het getal zelf voor, in twee-complements vorm, met het minst significante byte eerst.

Numerieke variabelen kunnen een naam hebben van willekeurige lengte, die moet beginnen met een letter, die gevolgd kan worden door letters of cijfers. Spaties worden genegeerd, en voor intern gebruik worden alle letters omgezet in kleine letters.

Controlevariabelen voor een FOR/NEXT-lus hebben een naam van maximum n letter lang.

Numerieke arrays worden met n letter benoemd, die ook nog in hetzelfde programma voor een gewone variabele gebruikt kan worden. Ze kunnen een willekeurig aantal dimensies hebben, van willekeurige grootte. De indexering start bij 1.

Strings kunnen om het even welke lengte hebben. De naam van een string is n letter, gevolgd door $.

String-arrays kunnen vele dimensies hebben, van willekeurige grootte. De naam ervan is n letter, gevolgd door $. Die naam kan niet meer gebruikt worden voor een gewone stringvariabele. Alle strings in een bepaald string-array hebben dezelfde lengte, die bepaald wordt door de laatste dimensie in het DIM-statement. De indexering start bij 1.

Slicing : een substring van een string kan worden beschreven door gebruik van slicers. Als slicer kan gebruikt worden :

1. de lege slicer
2. een numerieke uitdruking
3. (facultatieve numerieke uitdr.) TO (fac. num. uitdr.)

Een slicer wordt gebruikt om een substring aan te duiden, op een van de volgende manieren :

a) string uitdrukking (slicer)
b) string-array variabele (index,..,index, slicer)
   wat hetzelfde is als
   string-array variabele (index,..index)(slicer)

Stel, bij a), dat de string-uitdrukking de waarde s$ heeft. Indien de slicer dan leeg is, is het resultaat s$ (die beschouwd wordt als een substring van zichzelf).

Indien de slicer een numerieke uitdrukking is, met de waarde m, dan is het resultaat het mde karakter van s$ (een substring met lengte 1).

Indien de slicer van de derde vorm is, stel dan dat de eerste numerieke uitdrukking de waarde m heeft (standaardwaarde is 1) em de tweede, de waarde n (de standaardwaarde is de lengte van s$). Indien 1<=m<=n<=lengte van s$, dan is het resultaat : de substring van s$ die begint met het mde karakter en eindigt met het nde karakter.

Indien 0<=n<m, dan is het resultaat de nulstring. In andere gevallen wordt foutmelding 3 gegeven.

Slicen gebeurt voordat functies of bewerkingen worden gevalueerd, tenzij dit door haakjes anders werd geregeld.

Substrings kunnen ook apart worden ingevuld (cfr. LET). Indien in een string een aanhalingsteken moet opgenomen worden, dan moet het twee keer getypt worden.

Functies

Het argument van een functie moet niet tussen haakjes worden geschreven indien het een constante of een variabele is (eventueel een array-variabele of een slice).

Functie	soort argument	resultaat
ABS	getal	absolute grootte
ACS	getal	arccosinus in radialen. Foutmelding A indien x niet tussen -1 en +1
AND	binaire bewerking, rechts altijd een getal.	
	links numeriek :	a AND b => a indien b<>0 0 indien b=0
	links string :	a$ AND b => a$ indien b<>0 "" indien b=0
		AND heeft voorrang 3
ASN	getal	arcsinus in radialen. Foutmelding A indien x niet tussen -1 en 1
ATN	getal	arctangens in radialen
ATTR	twee argumenten, x en y, beide een getal; haakjes	een getal, waarvan de binaire vorm de attributen bevat van regel x, kolom y op het scherm. Bit 7 (meest significante) is 1 voor knipperend, 0 voor gewoon. Bit 6 is 1 voor hel-

		der, 0 voor gewoon. Bits 5 tot 3 bevatten de papierkleur. Bits 2 tot 0 bevatten de inktkleur. Foutmelding B tenzij 0<=x<=23 en tegelijk 0<=y<=31
BIN		In feite geen functie, maar een alternatief om getallen te noteren. BIN gevolgd door een reeks van 0 en 1 stelt het getal voor dat er in binaire notatie zo uit ziet.
CHR$	getal	het karakter waarvan x de code is, afgerond tot het dichtstbijzijnde gehele getal
CODE	string	de code van het eerste karakter in de string, of 0 indien de string de nulstring is
COS	getal (in radialen)	de cosinus van het getal
EXP	getal	e^x
FN		FN gevolgd door een letter, roept een door de programmeur gedefinieerde functie op (cfr. DEF). De argumenten moeten tussen haakjes staan. Ook al zijn er geen argumenten, dan nog moeten er twee haakjes staan.
IN	getal	Het resultaat van een input op het niveau van de processor, vanop de poort x (0<=x<=FFFFh). Laadt het BC-registerpaar met x en voert dan de Assembler- instructie IN A,(C) uit.
INKEY$	geen argument	Leest het toetsenbord uit. Het resultaat is het karakter dat op de ingedrukte toets staat, indien er precies 1 toets werd ingedrukt, zoniet de nulstring.
INT	getal	geheel gedeelte van het getal, altijd naar beneden afgerond
LEN	string	de lengte van de string
LN	getal	natuurlijke logaritme, op basis e. Foutmelding A indien x<=0
NOT	getal	0 indien x<>0, 1 indien x=0. NOT heeft voorrang 4.
OR	binaire bewerking, twee getallen	a OR b => 1 indien b<>0 a indien b=0 OR heeft voorrang 2
PEEK	getal	de waarde van het byte in het geheugen waarvan het adres x is, afgerond tot het dichtstbije gehele getal. Foutmelding B indien x niet tussen 0 en 65535 ligt.

PI	geen argument	3.1415927...
POINT	twee argumenten, x en y, beide een getal; haakjes	1 indien de pixel op (x,y) in inkt kleur staat, 0 indien hij papierkleur is. Foutmelding B tenzij 0<=x<=255 en tegelijk 0<=y<=175
RND	geen argument	het volgende pseudo-willekeurig getal uit een reeks, die wordt gegenereerd door de machten van 75 modulo 65537 te nemen, 1 ervan af te trekken en te delen door 65536. 0<=y<1
SCREEN$	twee argumenten, x en y, beide een getal; haakjes	Het karakter dat, normaal of inverse, op het scherm staat op regel x, kolom y. De nulstring indien het karakter niet herkend wordt. Foutmelding B tenzij 0<=x<=23 en tegelijk 0<=y<=31
SGN	getal	teken van het getal. Geeft -1 voor negatief, 0 voor nul en +1 voor positief.
SIN	getal (radialen)	sinus van x
SQR	getal	vierkantswortel. Foutmelding A, indien x<0
STR$	getal	de rij karakters die op het scherm zouden komen indien x geprint werd
TAN	getal (radialen)	de tangens van x
USR	getal	roept de subroutine in machinetaal aan, met startadres x. Bij terugkeer is het resultaat, de inhoud van het BC-registerpaar.
USR	string	het adres van het bitpatroon van de UDG die overeenkomt met de opgegeven string. Foutmelding A indien x$ geen letter tussen a en u noch een UDG is.
VAL	string	Evalueert x$ (zonder de aanhalingstekens) als een numerieke uitdrukking. Foutmelding C indien x een syntax-fout bevat, of een stringwaarde oplevert. Andere foutmeldingen mogelijk, afhankelijk van de uitdrukking.
VAL$	string	Evalueert x (zonder de aanhalingstekens) als een string-uitdrukking. Foutmelding C indien x een syntax-error bevat of een numerieke waarde oplevert. Andere foutmeldingen mogelijk (zoals bij VAL).
	getal	negatie

De volgende bewerkingen zijn binaire bewerkingen :

+	optelling (getallen) of aaneenschakeling (strings)	
-	aftrekking	
*	vermenigvuldiging	
/	deling	
^	machtsverheffing. Foutmelding B indien de linkse operandus negatief is	
=	gelijkheid	beide operandi moeten van het zelfde type zijn. Het resultaat is een getal : 1 indien de vergelijking opgaat, 0 indien niet.
>	groter dan	
<	kleiner dan	
<=	kleiner dan of gelijk aan	
>=	groter dan of gelijk aan	
<>	niet gelijk aan	

Functies en bewerkingen hebben de volgende voorrangs-niveaus :

bewerking	voorrangsniveau
indexering en slicing	12 (hoogste)
functies, behalve NOT en unaire min	11
^	10
unaire min (voor negatief getal)	9
* en /	8
= en - (aftrekking)	6
= > < <= >= <>	5
NOT	4
AND	3
OR	2

Statements

In de lijst die hieronder volgt, worden de volgende symbolen gebruikt :

l	voor één letter
v	voor een variabele
x,y,z	voor numerieke uitdrukkingen
m,n	voor numerieke uitdrukkingen die afgerond worden tot het dichtstbije geheel getal
e	voor een uitdrukking
f	voor een string-uitdrukking
s	voor een reeks statements, gescheiden door dubbele punten
c	voor een reeks kleur-items, elk gevolgd door een komma of een kommapunt. Een kleur-item kan een PAPER, INK, FLASH, BRIGHT, INVERSE of OVER-statement zijn.

Merk op dat in alle gevallen willekeurige uitdrukkingen kunnen gebruikt worden, met uitzondering van het regelnummer bij het begin van een statement.

Alle statements, behalve INPUT, DEF FN en DATA, kunnen zowel als directe commando's als in een programma gebruikt worden (hoewel het een al zinniger kan zijn dan het ander). Een commando of een programmaregel kan verscheidene statements bevatten, van elkaar gescheiden door dubbele punten. Er bestaat geen beperking voor wat betreft de plaats op de regel waar een bepaald statement gebruikt mag worden. Let evenwel op het gebruik van IF en REM.

BEEP x,y — Laat een toon door de luidspreker van de tv klinken gedurende x seconden. De toonhoogte is y halve tonen boven de middenste do (of er onder indien y negatief is).

BORDER m — Bepaalt de kleur van de rand van het scherm, en de papierkleur voor het onderste deel van het scherm. Foutmelding K indien niet 0<=m<=7 d.w.z. indien m niet tussen 0 en 7 ligt.

BRIGHT n — Bepaalt de helderheid waarmee karakters worden afgedrukt : n=0 voor gewoon, 1 voor helder en 8 voor transparant. Foutmelding K indien n niet 0, 1 of 8 is.

CAT — Werkt niet zonder microdrive enz.

CAT 1 — Geeft een lijst van de bestanden die op dat ogenblik op de RAMdisk staan.

CIRCLE x,y,z — Tekent een cirkel met middelpunt (x,y) en straal z.

CLEAR — Wist alle variabelen uit, waardoor de ruimte die ze innamen, weer vrij komt. Voert tegelijk een RESTORE en CLS uit, zet de PLOT-positie terug op onderaan links, en wist de GO SUB-stack.

CLEAR n — Zoals CLEAR, maar zet, indien mogelijk, de systeemvariabele RAMTOP op n en bouwt daar dan de nieuwe GO SUB-stack.

CLOSE # — Werkt niet zonder microdrive enz.

CLS — CLear Screen, maak scherm vrij. Wist de inhoud van de display file.

CONTINUE — Gaat verder met de uitvoering van het programma, van op het punt waar dit werd gestopt, met een mededeling die niet code 0 had. Indien de foutmelding 9 of L was, wordt gestart met het volgende statement (wel rekening houdend met sprongen). Indien andere foutmelding, wordt hetzelfde statement opnieuw uitgevoerd. Indien de foutmelding werd gegeven n.a.v. een direct commando, dan zal CONTINUE proberen om de commandoregel verder uit te voeren, en zal in een lus geraken indien de fout optrad in 0:1; trad ze op in 0:2, dan wordt de mededeling 0 gegeven. Gebeurde de fout verderop in de regel, dan wordt foutmelding N gegeven.

COPY — Stuurt, in Epson-formaat, een afbeelding van de bovenste 22 regels van het scherm naar de printer, in quadruple density bit formaat (grafische print mode), indien de printer aangesloten is. Zoniet, gebeurt er niets. Indien BREAK wordt gedrukt, foutmelding D.

DATA e1,e2,e3... — Deel van een DATA-lijst. Moet in een programma staan. Doet verder niets.

DEF FN l(l1,...,lk)=e
Definitie van een functie. Moet in een

programma staan. Heeft verder geen effect. L en l1 tot lk bestaan uit één letter, gevolgd door $ indien het om een string-argument of een string-resultaat gaat. Indien er geen argumenten zijn, is de vorm : DEF FN l()=e.

DIM l (n1,...,nk) Wist een bestaand array met de naam l uit; en initialiseert een numeriek array l met k dimensies (n1 tot nk). Initialiseert alle waarden op 0.

DIM l$(n1,...,nk) Wist een bestaand array met de naam l$ uit, en initialiseert een string-array l$ met k dimensies (n1 tot nk). Initialiseert alle waarden op " ". Kan beschouwd worden als een array van strings met een vaste lengte nk, met k-1 dimensies. Een array is onbestaand indien het niet door DIM werd gedimensioneerd. Foutmelding 4 indien er niet voldoende geheugen vrij is om het array in onder te brengen.

DRAW x,y = DRAW x,y,0

DRAW x,y,z Tekent een lijn vanaf de courante PLOT-positie, naar een punt op een horizontale afstand x en een vertikale afstand y van de PLOT-positie, daarbij over een hoek van z radialen draaiend. Foutmelding B indien de lijn buiten het scherm valt.

ERASE Werkt niet zonder microdrive enz.

ERASE ! f Wist een bestand van de RAMdisk.

FLASH n Bepaalt of afgedrukte karakters knipperen of niet. N=0 voor gewoon, n=1 voor knipperend, n=8 voor zoals voordien op het scherm stond.

FOR l=x TO y = FOR l=x TO y STEP 1

FOR l=x TO y STEP z
Wist een gewone variabele met de naam l uit, en initialiseert een controlevariabele l met startwaarde x, grenswaarde y, ophoging z, en een adres waar de lus begint, namelijk het statement dat volgt op het FOR-statement. Controleert of de startwaarde groter (stap >=0) of kleiner (stap <0) is dan de grenswaarde, en springt, indien dat zo is, direct naar het NEXT l statement. Indien dat niet gevonden wordt, geeft de +2 foutmelding 1. Zie ook NEXT. Foutmelding 4 indien er niet voldoende geheugen vrij is voor de controlevariabele.

FORMAT f;n Stelt de baud rate van apparaat f in op waarde n. Geldige namen zijn "p" en "P" (de RS232). Baudrates tussen 75 en 19200 zijn toegestaan.

GO SUB n Zet het regelnummer waarop GO SUB staat, op een stack, en werkt verder als een GO TO. Foutmelding 4 kan voorkomen indien er niet voldoende RETURNs gevonden worden.

GO TO n

Springt naar regel n. Indien die niet bestaat, gaat door naar de eerstvolgende.

IF x THEN s

Indien x waar is (verschilt van 0), wordt s uitgevoerd. Merk op dat s alle statements omvat, tot op het einde van de regel. De vorm 'IF x THEN regelnummer' is niet toegestaan.

INK n

Bepaalt de inktkleur (voorgrondkleur) van de karakters die nadien worden geprint. N moet liggen tussen 0 en 7 voor een kleur, is 8 voor transparant printen en 9 voor kontrast. Foutmelding K indien niet 0 <=n <=9.

INPUT ...

Met '...' wordt een reeks INPUT-items bedoeld die, zoals bij een PRINT-statement, van elkaar gescheiden worden door een komma, een kommapunt of een afkappingsteken. Een INPUT-item kan n van de volgende zijn :

1. een PRINT-item dat niet met een letter begint
2. de naam van een variabele
3. LINE gevolgd door de naam van een string-variabele.

De PRINT-items en scheidingstekens in 1. worden net zo behandeld als in PRINT, maar nu wordt alles in het onderste deel van het scherm afgedrukt. In geval 2. stopt de computer en wacht tot u iets intypt, dat dan aan de variabele wordt toegewezen. Wat u intypt wordt zoals gewoonlijk op het scherm gezet, en syntax fouten worden aangeduid met een knipperend "?". Voor string-uitdrukkingen wordt de input-buffer alvast voorzien van twee aanhalingstekens (die u kunt weghalen). Indien u als eerste karakter "STOP" intypt (SYMB SHIFT + A) dan stopt het programma met foutmelding H. Geval 3. werkt zoals 2., maar daarbij wordt wat u intypt, als een string verwerkt, zonder aanhalingstekens dit keer. Het "STOP"-mechanisme werkt hierbij niet : om te stoppen dient u op "cursor omlaag" te drukken.

INVERSE n

Bepaalt of volgende karakters al dan niet invers worden geprint. Is n=0, dan worden ze gewoon geprint; is n=1, dan worden ze invers afgedrukt, d.w.z. papierkleur op inktkleur. Foutmelding K (zie deel 28 van dit hoofdstuk) indien n niet 0 of 1 is.
In 48 BASIC geeft een druk op de INV VIDEO toets hetzelfde resultaat als INVERSE 1, en een druk op de TRUE VIDEO hetzelfde als INVERSE 0.

LET v=e

Kent de waarde van e toe aan de variabele v. LET mag niet weggelaten worden. Een gewone variabele is onbestaand, voordat hij werd ingevuld door LET, READ of INPUT. Indien v een geïndexeerde stringvariabele is, of een geslicete stringvariabele (substring), dan wordt hij op een Procrusteaanse manier ingevuld (vaste lengte) d.w.z. dat de stringwaarde van e ofwel ingekort wordt of

aangevuld met spaties aan de rechterkant, om dezelfde lengte te bereiken die in v werd gespecifieerd.

LIST = LIST 0

LIST n Geeft een listing van het programma in het bovenste deel van het scherm, te beginnen met de regel waarvan het regelnummer tenminste gelijk is aan n, en maakt van n de courante regel (met cursor).

LLIST = LLIST 0

LLIST n Zoals LIST, maar nu naar de printer.

LOAD f Laadt programma + variabelen in het geheugen.

LOAD f DATA l() Laadt een numeriek array in het geheugen.

LOAD f DATA l$() Laadt een string-array in het geheugen.

LOAD f CODE m,n Laadt (maximum) n bytes, vanaf adres m, in het geheugen.

LOAD f CODE m Laadt bytes in het geheugen, vanaf adres m.

LOAD f CODE Laadt bytes in het geheugen, vanaf het adres waarvandaan ze werden gesaved.

LOAD f SCREEN$ = LOAD f CODE 16384,6912

LOAD ! Werkt zoals LOAD (zie hoger voor alle mogelijkheden) maar van de RAMdisk.

LPRINT ... Werkt zoals PRINT, maar met de printer.

MERGE f Werkt zoals LOAD, maar wist geen programmaregels of variabelen uit het geheugen, tenzij er een regel met hetzelfde nummer of een variabele met dezelfde naam wordt ingelezen.

MERGE ! f Zoals MERGE f, maar van RAMdisk.

MOVE f1,f2 Werkt niet zonder microdrive enz.

NEW Start het BASIC systeem opnieuw op. Programma en variabelen worden gewist, plus het geheugen tot en met het byte waarvan het adres in de systeemvariabele RAMTOP staat. De systeemvariabelen UDG, P_RAMT, RASP en PIP blijven hun waarde houden. De controle wordt overgedragen aan het startmenu. De RAMdisk blijft zijn inhoud behouden.

NEXT l
1. Zoekt de controlevariabele l
2. Telt de ophoging bij de waarde ervan op
3. Indien de stap >=0 is en de waarde groter dan de grenswaarde (of de stap <0 en de waarde kleiner dan de grenswaarde), wordt naar het einde van de lus gesprongen, zoniet naar het begin ervan. Foutmelding 2 indien de variabele l niet bestaat. Foutmelding 1 indien de variabele l niet dezelfde is als de controlevariabele in het FOR-statement.

OPEN #

Werkt niet zonder microdrive enz.

OUT m,n

Stuurt, op processor-niveau, byte n naar poort m. Registerpaar BC wordt met m geladen, register A met n, en dan wordt de Assembler-instructie OUT (c),a uitgevoerd. Foutmelding B indien niet 0<=m<=65535 en tegelijk -255<=n<=255.

OVER n

Bepaalt hoe volgende karakters worden afgedrukt. Indien n=0 dan wordt wat voordien op de printpositie stond, uitgewist. Is n=1, dan wordt het nieuwe karakter gemengd met wat er stond. Op de plaats waar n van beide inktkleur heeft (maar niet beide) komt inktkleur, en waar beide inktkleur of papierkleur hebben, komt papierkleur. Foutmelding K indien n niet 0 of 1 is.

PAPER n

Werkt zoals INK, maar nu voor de papierkleur (achtergrondkleur).

PAUSE n

Onderbreekt de werking van de computer, en laat de display-file zien gedurende de tijd, nodig om n tv-beelden te tekenen (50 per seconde in Europa), of tot een toets wordt ingedrukt. Indien n=0, wordt de pauze niet afgeteld, maar blijft ze duren tot een toets wordt ingedrukt. Foutmelding B indien niet 0<=n<=65535.

PLAY f(,f,f,f,...)

Interpreteert maximum acht strings (cfr. deel 19 van dit hoofdstuk) en laat ze gelijktijdig horen. De eerste drie strings gaan naar de luidspreker van de tv en (mogelijk) naar de MIDI-poort. De volgende strings kunnen enkel via de MIDI-poort gestuurd worden.

PLOT c;m,n

Zet een inkt-puntje (afhankelijk van OVER en INVERSE) op de pixelplaats (m,n) en verplaatst de PLOT-positie daar naartoe. Tenzij de kleur-items c anders bepalen, wordt de inktkleur van de karakterpositie waarin de bedoelde pixel staat, gewijzigd in de permanente inktkleur die op dat moment geldt. De andere attributen (PAPER, FLASH, BRIGHT) worden ongemoeid gelaten. Foutmelding B tenzij 0<=m<=255 en tegelijk 0<=n<=175.

POKE m,n

Schrijft de waarde n in het byte met adres m. Foutmelding B tenzij 0<=m<=65535 en tegelijk -255<=n<=255.

PRINT ...

Met '...' wordt een reeks PRINT-items bedoeld, van elkaar gescheiden door een ",", een ";" of een "'". Die worden in de display file gezet, waarna ze op het scherm worden weergegeven. Een kommapunt tussen twee items heeft geen effect; het dient louter als scheidingsteken. Een komma stuurt het gelijknamige controle-karakter naar het scherm, en een afkappingsteken stuurt een ENTER. Dit laatste wordt ook automatisch gedaan, indien een PRINT-statement niet eindigt met een kommapunt, een komma of een afkappingsteken.

Een PRINT-item kan zijn :

1) leeg, d.w.z. niets
2) een numerieke uitdrukking

Indien de waarde negatief is, wordt eerst een minteken afgedrukt. Stel dat x de modulus is van waarde. Indien x<=10^-5 of x>=10^13, dan wordt x in wetenschappelijke notatie geprint. De mantisse kan uit acht cijfers bestaan (zonder nullen op het einde) en de decimale punt (niet aanwezig indien maar 1 cijfer) staat na het eerste cijfer ervan. De exponent wordt geschreven als E, gevolgd door + of -, gevolgd door een of twee getallen.

In het andere geval wordt x gewoon als decimaal getal geprint, met maximum acht significante cijfers, en geen nullen op het einde, na de decimale punt. Een decimale punt bij het begin, wordt altijd door een nul gevolgd. Bijvoorbeeld .03 en 0.3 worden op die manier geprint. Nul wordt als 0 geprint.

3) Een string-uitdrukking

Tokens in een string worden als keywords geprint, met eventueel een spatie ervoor of erna. Kontrolekarakters hebben het normale effect. Karakters die niet herkend worden, worden als "?" geprint.

4) AT m,n

Er wordt een controlekarakter voor AT naar het scherm gestuurd, gevolgd door een byte m (regelnummer) en een byte n (kolomnummer).

5) TAB n

Er wordt een controlekarakter voor TAB naar het scherm gestuurd, gevolgd door twee bytes voor n (minst significante eerst), waarmee de kolom wordt aangegeven.

6) een kleur-item, in de vorm van een PAPER, INK, FLASH, BRIGHT, INVERSE of OVER-statement

RANDOMIZE = RANDOMIZE 0

RANDOMIZE n Stelt de systeemvariabele SEED in, die gebruikt wordt om de volgende waarde voor RND te bepalen. Indien n<>0 dan krijgt SEED de waarde n. Is n=0, dan wordt in SEED de waarde gezet van een andere variabele, FRAMES, waarin geteld wordt hoeveel tv-beelden tot dan toe werden getekend; dat zou een vrij willekeurig getal moeten opleveren. Foutmelding B tenzij 0<=n<=65535.

READ v1,v2,...,vk Kent aan de opeenvolgende variabelen de waarde toe van de uitdrukkingen in de DATA-lijst. Foutmelding C indien de uitdrukking van een ander soort is dan de variabele. Foutmelding E indien de DATA uitgeput zijn, en er nog een variabele gelezen moet worden.

REM ...	Heeft geen effect. De drie puntjes kunnen door om het even welke reeks karakters worden vervangen, met als laatste een ENTER. Na REM wordt geen enkel commando als zodanig herkend op de hele regel. Een dubbele punt wordt niet als een scheidingsteken herkend.
RESTORE	= RESTORE 0
RESTORE n	Plaatst de DATA-pointer terug bij het eerste DATA-statement op regel n. Indien regel n niet bestaat of geen DATA-regel is, dan wordt de pointer geplaatst bij het eerste DATA-statement na regel n. Het eerstvolgende READ-commando zal van daaraf beginnen lezen.
RETURN	Haalt een verwijzing naar een bepaald statement van de GO SUB-stack, en springt naar de regel erna. Foutmelding 7 indien er geen dergelijke verwijzing op de stack wordt gevonden. Dit wil vermoedelijk zeggen dat er een of andere fout in uw programma voorkomt. Zorg ervoor dat bij elke GO SUB een RETURN hoort en omgekeerd.
RUN	= RUN 0
RUN n	CLEAR gevolgd door GO TO n.
SAVE f	Schrijft het programma en de variabelen op band, onder de naam f. Foutmelding F indien de naam de nulstring is, of langer dan tien karakters. Lees deel 20 van dit hoofdstuk.
SAVE f LINE m	Schrijft het programma en de variabelen op band, zodanig dat direct na het inladen ervan het vanzelf start op regel m.
SAVE f DATA 1()	Schrijft een numeriek array op band.
SAVE f DATA 1$()	Schrijft een string-array op band.
SAVE f CODE m,n	Schrijft n bytes op band, vanaf adres m.
SAVE f SCREEN$	= SAVE f CODE 16483,6912. Schrijft de inhoud van de display file op de band.
SAVE ! f	Zoals SAVE, maar nu maar RAMdisk. Lees deel 20 van dit hoofdstuk.
SPECTRUM	Schakelt de computer om van 128 BASIC naar 48 BASIC. Een aanwezig programma blijft in het geheugen staan. Er kan niet teruggeschakeld worden naar 128 BASIC.
STOP	Breekt het programma af met de foutmelding 9. CONTINUE doet het programma verdergaan vanaf een volgend statement.
VERIFY	Zoals LOAD, maar de informatie op band wordt niet in het geheugen geladen. Ze wordt alleen vergeleken met de inhoud van het geheugen. Foutmelding R indien er verschil blijkt te bestaan tussen de twee.

DEEL 31 : VOORBEELDEN VAN PROGRAMMA'S

Programma's :
- Dagen
- I Tsjing
- Raad een dier
- Vlaggen
- Galgje

In dit deel vindt u een aantal interessante programma's. Indien u ze meer dan één keer wilt gebruiken, vergeet dan niet ze te SAVEn op cassette (permanent) of op RAMdisk (tijdelijk).

Dagen

Het eerste programma vraagt u om een datum, en vertelt dan welke dag in de week met die datum overeenkomt.

```
    10 REM zet datum om in dag
    20 DIM d$(7,6): REM dagen in de week
    30 FOR n=1 TO 7: READ d$(n): NEXT n
    40 DIM m(12): REM lengte van maanden
    50 FOR n=1 TO 12: READ m(n): NEXT n
   100 REM geef datum op
   110 INPUT "Dag ?";dag
   120 INPUT "Maand ?";maand
   130 INPUT "Jaar (20ste eeuw) ?";jaar
   140 IF jaar<1901 THEN PRINT "De 20ste eeuw
       begint in 1901": GO TO 100
   150 IF jaar>2000 THEN PRINT "De 20ste eeuw
       eindigt met 2000": GO TO 100
   160 IF maand<1 THEN GO TO 210
   170 IF maand>12 THEN GO TO 210
   180 IF jaar/4-INT(jaar/4)=0 THEN LET m(2)=29:
       REM schrikkeljaar
   190 IF dag>m(maand) THEN PRINT "Die maand telt
       maar ";m(maand);" dagen.": GO TO 500
   200 IF dag>0 THEN GO TO 300
   210 PRINT "Dat is nonsens. Geef een echte datum"
   220 GO TO 500
   300 REM zet datum om in aantal dagen sinds start
       van deze eeuw
   310 LET j=jaar-1901
   320 LET b=365*y+INT(y/4): REM aantal dagen tot aan
       begin van jaar
   330 FOR n=1 TO maand-1: REM tel vorige maanden bij
   340 LET b=b+m(n): NEXT n
   350 LET b=b+dag
   400 REM zet om naar weekdag
   410 LET b=b-7*INT(b/7)+1
   420 PRINT dag;"/";maand;"/";jaar
   430 FOR n=6 TO 3 STEP -1: REM haal spaties weg
   440 IF d$(b,n)<>" " THEN GO TO 460
   450 NEXT n
   460 LET e$=d$(b, TO n)
   470 PRINT "is een ";e$;"dag"
   500 LET m(2)=28: REM februari terug normaal
   510 INPUT "Nog een keer?",a$
   520 IF a$="n" THEN GO TO 540
   530 IF a$<>"N" THEN GO TO 100
  1000 REM dagen
  1010 DATA "maan","dins","woens"
  1020 DATA "donder","vrij","zater","zon"
  1100 REM lengte van de maanden
  1110 DATA 31,28,31,30,31,30
  1120 DATA 31,31,30,31,30,31
```

I Tsjing

Dit programma werpt de stokjes voor I Tsjing, een Chinese vorm van toekomst-voorspelling. Hoewel de patronen ondersteboven worden getekend, kunnen de resultaten toch aanvaardbaar zijn...

```
  5 RANDOMIZE
 10 FOR m=1 TO 6: REM 6 worpen
 20 LET c=0: REM totaal van de worpen is nul
 30 FOR n=1 TO 3: REM 3 worpen
 40 LET c=c+2+INT(2*RND)
 50 NEXT n
 60 PRINT "    ";
 70 FOR n=1 TO 2: REM eerst voor het gegooide
    hexagram, dan voor de veranderingen
 80 PRINT "---";
 90 IF c=7 THEN PRINT "-";
100 IF c=8 THEN PRINT " ";
110 IF c=6 THEN PRINT "X";: LET c=7
120 IF c=9 THEN PRINT "O";: LET c=8
130 PRINT "--- ";
140 NEXT n
150 PRINT
160 INPUT a$
170 NEXT m: NEW
```

Typ dit programma in, run het en druk dan vijf keer op ENTER om de vijf hexagrammen te zien. Zoek die op in het Chinese Boek der Veranderingen. De tekst die u daarin vindt, beschrijft een situatie en wat u daarmee moet doen. Ga er eens rustig bij zitten, om de overeenkomst te ontdekken tussen wat er gezegd wordt, en uw eigen leven. Druk nog een zesde keer op ENTER, en het programma wist zichzelf uit. De bedoeling hiervan is, dat u het programma niet lichtzinnig gebruikt...

Een boel mensen vinden de teksten vaak onverwacht toepasbaar op zichzelf. Misschien is dit ook zo voor uw +2, misschien ook niet. Over het algemeen zijn computers vrij goddeloze schepsels.

Raad een dier

Dit programma raadt een dier. U denkt aan een dier, en de computer probeert te raden welk dier, door u vragen te stellen die met "ja" of met "nee" beantwoord kunnen worden. Blijkt hij het dier waaraan u dacht, niet te kennen, dan vraagt hij om een vraag in te typen die hij volgende keer kan stellen, zodat hij ook dat dier kan vinden indien iemand anders het hem voorlegt.

```
  5 REM raad een dier
 10 LET nq=100: REM aantal vragen en dieren
 15 DIM q$(nq,50): DIM a(nq,2): DIM r$(1)
 20 LET qf=8
 30 FOR n=1 TO qf/2-1
 40 READ q$(n): READ a(n,1): READ a(n,2)
 50 NEXT n
 60 FOR n=n TO qf-1
 70 READ q$(n): NEXT n
100 REM het spel begint
110 PRINT "Denk aan een dier.","Druk op een toets"
120 PAUSE 0
130LET c=1: REM begin met 1ste vraag
140 IF a(c,1)=0 THEN GO TO 300
150 LET p$=q$(c): GO SUB 910
160 PRINT "?": GO SUB 1000
170 LET i=1: IF r$="j" THEN GO TO 210
180 IF r$="J" THEN GO TO 210
190 LET i=2: IF r$="n" THEN GO TO 210
200 IF r$<>"N" THEN GO TO 150
210 LET c=a(c,i): GO TO 140
300 REM dier
310 PRINT "Dacht je aan een"
320 lp$=q$(c): GO SUB 900: PRINT "?"
330 GO SUB 1000
340 IF r$="j" THEN GO TO 400
350 IF r$="J" THEN GO TO 400
360 IF r$="n" THEN GO TO 500
370 IF r$="N" THEN GO TO 500
380 PRINT "Geef antwoord als ik u iets vraag":
    GO TO 300
400 REM geraden...
410 PRINT "Dat dacht ik wel.": GO TO 800
500 nieuw dier
510 IF qf>nq-1 THEN PRINT "Waarschijnlijk is dat
    een heel  interessant dier, maar ik heb er
    geen plaatsmeer voor...": GO TO 800
520 LET q$(qf)=q$(c): REM verhuis oud dier
530 PRINT "Wat is het dan ?": INPUT q$(qf+1)
540 PRINT "Met welke vraag kan ik het onderscheid
    maken tussen "
550 LET p$=q$(qf): GO SUB 900: PRINT " en "
560 LET p$=q$(qf+1): GO SUB 900: PRINT " "
570 INPUT s$: LET b=LEN s$
580 IF a$(b)="?" THEN LET b=b-1
590 LET q$(c)=s$(TO b): REM neem vraag op
600 PRINT "Wat is het antwoord op"
610 LET p$=q$(qf+1): GO SUB 900: PRINT "?"
620 GO SUB 1000
630 LET i=1: LET io=2: REM antwoorden voor oude
    en nieuwe dieren
640 IF r$="j" THEN GO TO 700
650 IF r$="J" THEN GO TO 700
660 LET i=2: LET io=1
670 IF r$="n" THEN GO TO 700
680 IF r$="N" THEN GO TO 700
```

```
 690 PRINT "Daar kan ik niets mee.": GO TO 600
 700 REM antwoorden aanpassen
 710 LET a(c,i)=qf+1: LET a(c,io)=qf
 720 LET qf=qf+2: REM volgende vrije ruimte
 730 PRINT "Je had me beet..."
 800 REM nog een keer ?
 810 PRINT "Wil je nog een keer ?": GO SUB 1000
 820 IF r$="j" THEN GO TO 100
 830 IF r$="J" THEN GO TO 100
 840 STOP
 900 REM druk af zonder spaties erna
 905 PRINT " ";
 910 FOR n=50 TO 1 STEP -1
 920 IF p$(n)<>" " THEN GO TO 940
 930 NEXT n
 940 PRINT p$(TO n);: RETURN
1000 REM vraag een antwoord
1010 INPUT r$: IF r$="" THEN RETURN
1020 LET r$=r$(1): RETURN
2000 REM dieren om te beginnen
2010 DATA "Leeft het in de zee",4,2
2020 DATA "Heeft het schubben",3,5
2030 DATA "Eet het mieren",6,7
2040 DATA "een walvis","een roompudding","een miereneter",
     "een mier"
```

Vlaggen

Dit programma tekent de Engelse vlag :

```
 5 REM Engelse vlag
10 LET r=2: LET w=7: LET b=1
20 BORDER 0: PAPER b: INK w: CLS
30 REM zwart onderaan het scherm
40 INVERSE 1
50 FOR n=40 TO 0 STEP -8
60 PLOT PAPER 0;7,n: DRAW PAPER 0;241,0
70 NEXT n: INVERSE 0
100 REM teken witte gedeelten
105 REM St.-Joris kruis
110 FOR n=0 TO 7
120 PLOT 104+n,175: DRAW 0,-35
130 PLOT 151-n,175: DRAW 0,-35
140 PLOT 151-n,148: DRAW 0,35
150 PLOT 104+n,48: DRAW 0,35
160 NEXT n
200 FOR n=0 TO 11
210 PLOT 0,139-n: DRAW 111,0
220 PLOT 255,139-n: DRAW -111,0
230 PLOT 255,84+n: DRAW -111,0
240 PLOT 0,84+n: DRAW 111,0
250 NEXT n
300 REM St.-Andreaskruis
310 FOR n=0 TO 35
320 PLOT 1+2*n,175-n: DRAW 32,0
330 PLOT 224-2*n,175-n: DRAW 16,0
340 PLOT 254-2*n,48+n: DRAW -32,0
350 PLOT 17+2*n,48+n: DRAW 16,0
360 NEXT n
370 FOR n=0 TO 19
380 PLOT 185+2*n,140+n: DRAW 32,0
390 PLOT 200+2*n,83-n: DRAW 16,0
```

```
400 PLOT 39-2*n,83-n: DRAW 32,0
410 PLOT 54-2*n,140+n: DRAW -16,0
420 NEXT n
425 REM vul extra stukjes in
430 FOR n=0 TO 15
440 PLOT 255,160+n: DRAW 2*n-30,0
450 PLOT 0,63-n: DRAW 31-2*n,0
460 NEXT n
470 FOR n=0 TO 7
480 PLOT 0,160+n: DRAW 14-2*n,0
485 PLOT 255,63-n: DRAW 2*n-15,0
490 NEXT n
500 REM rode strepen
510 INVERSE 1
520 REM St.-Joris
530 FOR n=96 TO 120 STEP 8
540 PLOT PAPER r;7,n: DRAW PAPER r;241,0
550 NEXT n
560 FOR n=112 TO 136 STEP 8
570 PLOT PAPER r;n,168: DRAW PAPER r;0,-113
580 NEXT n
600 REM St.-Patrick
610 PLOT PAPER r;170,140: DRAW PAPER r;70,35
620 PLOT PAPER r;179,140: DRAW PAPER r;70,35
630 PLOT PAPER r;199,83: DRAW PAPER r;56,-28
640 PLOT PAPER r;184,83: DRAW PAPER r;70,-35
650 PLOT PAPER r;86,83: DRAW PAPER r;-70,-35
660 PLOT PAPER r;72,83: DRAW PAPER r;-70,-35
670 PLOT PAPER r;56,140: DRAW PAPER r;-56,28
680 PLOT PAPER r;71,140: DRAW PAPER r;-70,35
690 INVERSE 0: PAPER 0: INK 7
```

Indien u geen Engelsman bent, probeer dan eens uw eigen vlag te tekenen. Driekleuren lukken nogal gemakkelijk, hoewel bepaalde kleuren (oranje bijvoorbeeld) problemen kunnen veroorzaken. Als u de Amerikaanse vlag wilt tekenen, kunt u wellicht de asterisk ("*") gebruiken. Hebt u een bepaalde vlag getekend, dan kunt u ze op de RAMdisk zetten door het commando SAVE ! "vlag" SCREEN$. Dan kunt u een volgende vlag tekenen, en die onder een andere naam op RAMdisk zetten. Er is geheugenruimte voor ongeveer tien vlaggen, zodat u voor een afwisselend spektakel kunt zorgen.

Galgje

Met dit programma kunt u "Galgje" spelen. Mocht u dat spel niet kennen : de ene speler moet een woord intypen, en de andere moet proberen het te raden.

```
  5 REM galgje
 10 REM maak scherm klaar
 20 INK 0: PAPER 7: CLS
 30 LET x=240: GO SUB 1000: REM teken mannetje
 40 PLOT 238,128: DRAW 4,0: REM mond
100 REM haal woord
110 INPUT w$: REM te raden woord
120 LET b=LEN w$: LET v$=" "
130 FOR n=2 TO b: LET v$=v$+" "
140 NEXT n: REM v$=tot hiertoe geraden woord
150 LET c=0: LET d=0: REM teller pogingen en fouten
160 FOR n=0 TO b-1
170 PRINT AT 20,n;"-";
180 NEXT n: REM zet - op de plaats van de letters
200 INPUT "Raad een letter: ";g$
210 IF g$="" THEN GO TO 200
```

```
220 LET g$=g$(1): REM maar 1 letter
230 PRINT AT 0,c;g$
240 LET c=c+1: LET u$=v$
250 FOR n=1 TO b: REM pas geraden woord aan
260 IF w$(n)=g$ THEN LET v$(n)=g$
270 NEXT n
280 PRINT AT 19,0;v$
290 IF v$=w$ THEN GO TO 500: REM geraden...
300 IF v$<>u$ THEN GO TO 200: REM letter juist
400 REM teken deel van galg
410 IF d=8 THEN GO TO 600: REM je hangt !
420 LET d=d+1
430 READ x0,y0,x,y
440 PLOT x0,y0: DRAW x,y
450 GO TO 200
500 REM mannetje vrij
510 OVER 1: REM wis mannetje uit
520 LET x=240: GO SUB 1000
530 PLOT 238,128: DRAW 4,0: REM mond
540 OVER 0: REM teken opnieuw
550 LET x=146: GO SUB 1000
560 PLOT 143,129: DRAW 6,0,PI/2: REM glimlach
570 GO TO 800
600 REM mannetje hangt
610 OVER 1: REM wis grond uit
620 PLOT 255,65: DRAW -48,0
630 DRAW 0,-48: REM valluik open
640 PLOT 238,128: DRAW 4,0: REM wis mond uit
650 REM herteken ledematen
655 REM armen
660 PLOT 255,117: DRAW -15,-15: DRAW -15,15
670 OVER 0
680 PLOT 236,81: DRAW 4,21: DRAW 4,-21
690 OVER 1: REM benen
700 PLOT 255,66: DRAW -15,15: DRAW -15,-15
710 OVER 0
720 PLOT 236,60: DRAW 4,21: DRAW 4,-21
730 PLOT 237,127: DRAW 6,0,-PI/2: REM frons
740 PRINT AT 19,0;w$
800 INPUT "Nog een keer ?";a$
810 IF a$="" THEN GO TO 850
820 LET a$=a$(1)
830 IF a$="n" THEN STOP
840 IF a$(1)="N" THEN STOP
850 RESTORE: GO TO 5
1000 REM teken mannetje op positie x
1010 REM hoofd
1020 CIRCLE x,132,8
1030 PLOT x+4,134: PLOT x-4,134: PLOT x,131
1040 REM lichaam
1050 PLOT x,123: DRAW 0,-20
1055 PLOT x,101: DRAW 0,-19
1060 REM benen
1070 PLOT x-15,66: DRAW 15,15: DRAW 15,-15
1080 REM armen
1090 PLOT x-15,117: DRAW 15,-15: DRAW 15,15
1100 RETURN
2000 DATA 120,65,135,0,184,65,0,91
2010 DARA 168,65,16,16,184,81,16,-16
2020 DATA 184,156,68,0,184,140,16,16
2030 DATA 204,156,-20,-20,240,156,0,-16
```

DEEL 32 : BINAIR EN HEXADECIMAAL REKENEN

Inhoud ...
   Getallen-systemen
   Bits en bytes

In dit deel wordt beschreven hoe computers hun rekenwerk uitvoeren, namelijk met het binaire systeem.

In de meeste Europese talen wordt in een min of meer regelmatig patroon van tientallen geteld. In het Nederlands is het begin wat ongeordend, maar vrij vlug wordt het een duidelijk systeem :

    twintig, eenentwintig, ... , negenentwintig
    dertig, eenendertig, ..., negenendertig
    veertig, eenenveertig, ... ,negenendertig

en zo gaat het door. Het systeem wordt nog duidelijker door de cijfers die we gebruiken. En toch is de enige reden waarom we per tiental tellen (het "decimale" systeem, van het Latijnse "decem", tien) het feit dat we toevallig tien vingers hebben.

In de plaats van het decimale systeem, dat op tien gebaseerd is, gebruiken computers een vorm van binair die "hexadecimaal" heet en vaak tot "hex" afgekort wordt. Dit systeem is gebaseerd op zestien. Aangezien wij slechts tien symbolen hebben in ons getallensysteem (0-9), moeten we de overige zes door andere symbolen voorstellen. We gebruiken daarvoor de letters A, B, C, D, E en F. En wat komt er na F ? Nu, net zoals wij met onze tien vingers, "10" schrijven om tien aan te duiden (een volle hand vingers...), "schrijven" computers ook "10" om zestien aan te duiden. Hieronder volgt een vergelijking tussen decimale en hexadecimale getallen :

decimaal	hex
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10
17	11
18	12
19	13
...	
25	19
26	1A
27	1B
...	
31	1F
32	20
33	21
...	

158	9E
159	9F
160	A0
161	A1
...	
255	FF
256	100

en zo verder ...

Wanneer u duidelijk wilt maken dat u een hexadecimaal getal schrijft, , zet u op het einde van het getal een "h", en zegt u "hex". Wilt u bijvoorbeeld het getal honderd achtenvijftig schrijven, dan is dat "9Eh", uitgesproken : "negen-ee-hex".

Misschien vraagt u zich nu af wat dat allemaal met computers te maken heeft. Wel, computers rekenen alsof ze slechts twee cijfers hadden. Die worden beschouwd als een lage spanning (uit) die bekend staat als 0, en een hogere spanning (aan), die bekend staat als 1. Deze manier van werken heet "binair rekenen". Zo een cijfer noemen we "bit" (van BInary digiT, binair cijfer). Een bit is dus ofwel 1, ofwel 0.

De vorige tabel kan dus uitgebreid worden met binaire getallen :

dec.	hex	binair
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001

enzovoort

Het is gebruikelijk om binaire getallen vooraan "op te vullen" met nullen, zodat ze altijd minstens vier bits bevatten. De getallen 0 tot 3 decimaal worden bijvoorbeeld geschreven als 0000, 0001, 0010, 0011.

Getallen omzetten van binair naar hex is erg gemakkelijk. De vorige tabel kan u daarbij helpen. Een beetje uitleg.

Om een binair getal in hex om te zetten, deelt u het binaire getal op in groepen van vier bits (waarbij u van rechts naar links werkt). Zet nu elke groep van vier bits om in het overeenkomstige hex-cijfer. Schrijf nu de hex-cijfers aan elkaar, en u hebt het volledige hex-getal. Een voorbeeld : u moet 10110100 omzetten in hex. De eerste groep van vier bits (0100) wordt 4h. De volgende groep van vier bits (1011) wordt Bh. Aan elkaar geschreven geeft dat B4h, het volledige hex-getal. Is het binaire getal langer dan acht bits, dan kunt u op dezelfde manier doorgaan. Het getal 1110101111O000 is bijvoorbeeld 3AF0h.

Om een hex-getal in een binair getal om te zetten, werkt u net andersom. Zet elk hex-cijfer om in vier bits. Daarbij werkt u opnieuw van rechts naar links. Zet uiteindelijk de groepen van vier bits naast elkaar, en u hebt het volledige binaire getal. Om bijvoorbeeld F3h in binair te schrijven, zet u eerst 3 om : dat is 0011b (denk aan de nullen om vier bits te krijgen). Dan komt de F, die overeenkomt met 1111b. Samen geeft dit het binaire getal 11110011b.

Hoewel computers zelf uitsluitend het binaire systeem gebruiken, schrijven wij mensen meestal de getallen in hexadecimaal. Het zal duidelijk zijn waarom : het getal 3AF0h bijvoorbeeld, kan stukken gemakkelijker gelezen worden dan 0011101011110000b, en er zullen minder snel fouten mee gemaakt worden.

De bits in een computer worden meestal samengenomen tot groepen van acht, tot "bytes". En byte kan een getal voorstellen tussen 0 en 255 decimaal (11111111b of FFh).

Twee bytes samengenomen heten in vaktermen een "woord". Een woord kan geschreven worden als een groep van zestien bits of vier hex-cijfers, en stelt een getal voor tussen 0 en 65535 decimaal (1111111111111111b of FFFFh).

Een byte is altijd acht bits lang. Woorden kunnen in lengte veschillen, afhankelijk van de computer.

In deel 14 van dit hoofdstuk hadden we het over het keyword BIN. Daarmee kunnen op de +2 binaire getallen ingetypt worden. Zo is bijvoorbeeld BIN 10 hetzelfde als 4, BIN 111 hetzelfde als 7, BIN 11111111 hetzelfde als 255, en zo verder.

Met BIN kunt u alleen nullen en enen gebruiken. Het getal moet dus een niet-negatief geheel getal zijn. U kunt dus niet schrijven : BIN -11 wanneer u -3 decimaal bedoelt. Dat is dan : - BIN 11. Het getal mag ook niet groter dan 65535 decimaal zijn, en kan dus niet langer dan zestien bits zijn. Indien u een binair getal met nullen opvult, zal BIN die terecht negeren. Indien u bijvoorbeeld typt : BIN 00000001, wordt dit verwerkt alsof u BIN 1 had getypt.

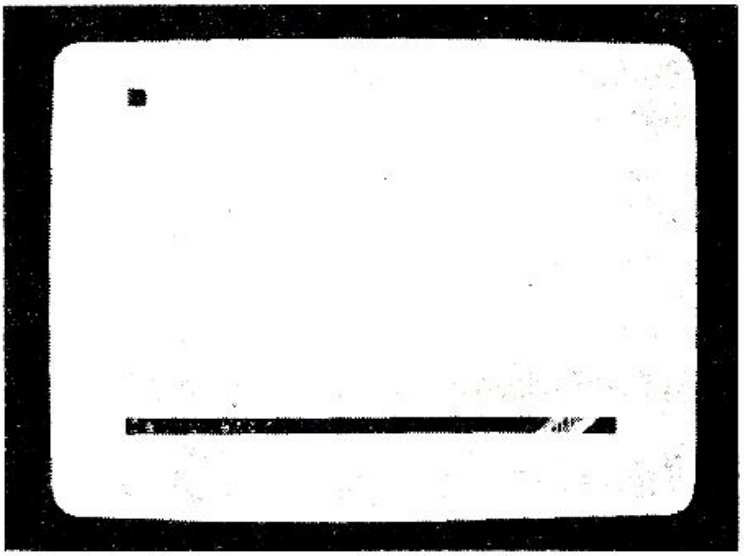
HOOFDSTUK 9 DE CALCULATOR GEBRUIKEN

Inhoud ...

De +2 kan als een volwaardige rekenmachine worden gebruikt. Om de calculator te gebruiken, roept u het startmenu op en kies de optie "Calculator". Indien u niet weet hoe dat moet, lees dan eerst hoofdstuk 2.

U kunt de calculator gebruiken van zodra u de computer inschakelt. Het is eveneens mogelijk om dit te doen, indien u met een 128 BASIC programma bezig bent. Daartoe kiest u de optie "Exit" in het edit-menu, waardoor u naar het startmenu gaat. Daaruit kunt u dan de optie "Calculator" kiezen. Het BASIC programma waarmee u bezig was, blijft behouden. Na terugkeer uit de calculator, kunt u verderwerken.

Nadat u de optie "Calculator" had gekozen, ziet het scherm er als volgt uit :



en de calculator van de +2 wacht op u. Typ in :

```
6+4
```

en zodra u dan ENTER drukt, verschijnt "10" als antwoord. Merk wel op dat u niet "=" moet intypen, zoals bij een gewone rekenmachine.

U ziet dat de cursor rechts van het antwoord staat. Dat is een tussentotaal, net zoals op een gewone rekenmachine. Dat wil zeggen dat u gewoon de volgende bewerking kunt intypen, die u op dat tussentotaal wilt uitvoeren, zonder dat u dit opnieuw moet intypen. Met de cursor rechts van de 10, typt u nu :

```
/5
```

en u krijgt het antwoord : 2. Typ dan :

```
*PI
```

wat als resultaat : 6.2831853 oplevert. De +2 heeft gebruik gemaakt van de ingebouwde PI-functie. U hoefde enkel de letters PI in te typen. Dit geldt voor alle wiskundige functies van de +2. Probeer nu eens :

```
*ATN 60
```

en u ziet : 9.7648943 verschijnen. U kunt ook wijzigingen aanbrengen aan wat er op het scherm staat. Beweeg bijvoorbeeld de cursor met de cursortoets naar het begin van de regel, en typ daar : INT. De regel wordt nu :

```
INT 9.7648943
```

en zodra u ENTER drukt, krijgt u het antwoord : 9. Dit toont ook aan dat de +2 niet per se een bewerking hoeft uit te voeren, om de waarde van een uitdrukking te printen. Druk nu eerst een keer ENTER en typ dan :

```
1E6
```

en de +2 geeft de waarde van die uitdrukking. U drukte een keer op ENTER, voordat u "1E6" intypte, om de +2 duidelijk te maken dat u met een nieuwe berekening wilde beginnen.

Een bijzonder handige functie van de calculator van de +2 is, dat u bepaalde waarden in een variabele kunt opslaan, die u daarna in berekeningen kunt gebruiken. Daartoe gebruikt u LET, net zoals in BASIC. Typ bijvoorbeeld :

```
LET x=10
```

en druk daarna twéé keer op ENTER, zodat de +2 de opgegeven waarde toekent aan de variabele x. Controleer nu dat wel degelijk de variabele wordt gebruikt, door te typen :

```
x+90
```

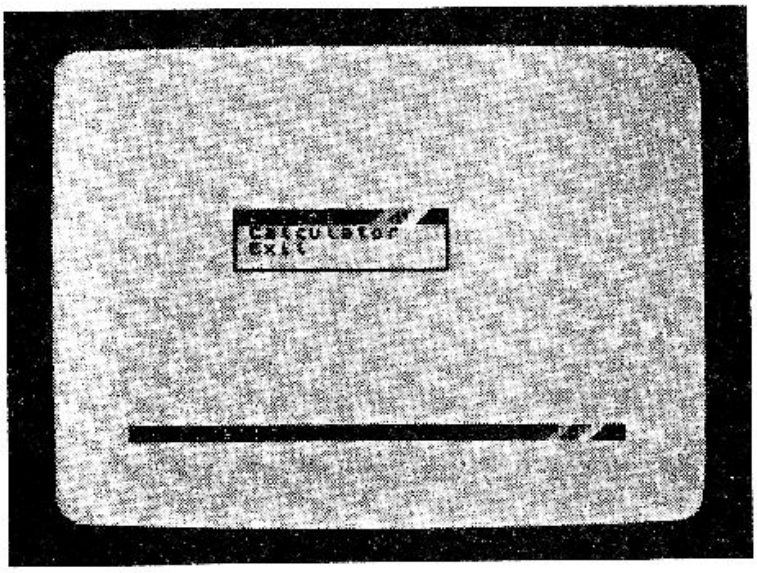
en daarna

```
+x*x
```

Wanneer u de calculator gebruikt terwijl u een BASIC programma aan het schrijven bent, dan dient u voor de calculator variabelen te gebruiken die niet door het programma worden gebruikt. BASIC keywords kunnen niet als namen van variabelen gebruikt worden.

Indien u klaar bent met de calculator, druk dan op de EDIT-toets en het scherm ziet er zo uit : (zie volgende bladzijde)

Kies de optie "Exit", die u terug naar het startmenu brengt. Wanneer u met een 128 BASIC programma bezig was, kunt u daar verder mee gaan door de optie "128 BASIC" te kiezen. Wilt u verderwerken met de calculator, kies dan de optie "Calculator".



HOOFDSTUK 10 RANDAPPARATUUR AANSLUITEN

Inhoud ...

De +2 kan werken met een breed gamma van randapparatuur, zoals een joystick, een videomonitor, een versterker, enzovoort. In dit deel vindt u alle nodige informatie om die apparaten aan te sluiten.

Joystick(s)

Het is aan te raden om enkel de SJS1 joystick van Sinclair op de +2 aan te sluiten. Andere types (Atari bv.) zullen niet werken, omdat de aansluitingen anders zijn.

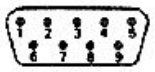
Links op de +2 zitten twee aansluitingen voor een joystick. Over het algemeen maken spelletjes gebruik van de aansluiting, gemerkt met "JOYSTICK 1".

Indien u in een programma een bepaald type van joystick kunt kiezen, kies dan de "Interface Two" of "Sinclair" : de aansluitingen van de joysticks bij de +2 zijn exact dezelfde als bij de Interface Two.

Het is niet gevaarlijk om een joystick aan te sluiten (of uit te trekken) terwijl de +2 ingeschakeld is.

Pen	Functie
1	niet gebruikt
2	aarding
3	niet gebruikt
4	schietknop
5	omhoog
6	rechts
7	links
8	aarding
9	omlaag

Aansluiting JOYSTICK 1 en 2



Video-monitor

Op de +2 kunt u een monochrome of een kleuren-monitor aansluiten in plaats van een tv-toestel. Indien bij de monitor die u wilt aanschaffen, niet wordt vermeld dat hij geschikt is voor de Spectrum +2 (of Spectrum 128), dan is er veel kans dat u een speciale aansluitkabel zult moeten halen. Uw Sinclair dealer kan u daarover voorlichten.

Indien uw monitor geen BRIGHT-signaal accepteert, kan hij slechts acht van de zestien beschikbare kleuren weergeven.

RGB-stekkerbus :

Pen	signaal	niveau
1	composiet PAL	1.2 V piek/piek, 75 Ohm
2	0 volt	-
3	helderheid	TTL
4	composiet sync	TTL
5	vertikaal sync	TTL
6	groen	TTL
7	rood	TTL
8	blauw	TTL

Indien u een monitor gebruikt, kan het nodig zijn dat u bepaalde maatregelen treft indien u ook geluid wilt horen. Indien de monitor een audio-ingang heeft, moet die verbonden worden met de SOUND-uitgang, achterop de +2. Kan uw monitor géén geluid voortbrengen, dan dient u een aparte versterker aan te sluiten. In het volgende stukje staat hoe u dit kunt doen.

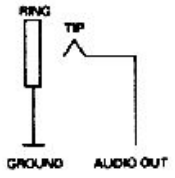
Versterker

Gewoonlijk hoort u het geluid van de +2 door de luidspreker van het tv-toestel. Indien u evenwel een monitor gebruikt, of indien u het geluid wilt opnemen of versterken, dan kunt u het geluids-signaal afnemen bij de SOUND-uitgang achterop de +2. De uitgang is een 3.5 mm mono stekkerbus, die 200 mV piek/piek levert over een impedantie van ongeveer 5kOhm. Indien u een versterker gebruikt, denk er dan aan dat de signalen bij LOAD en SAVE ook op de SOUND-uitgang staan. U kunt het beste het volume lager stellen bij het LOADen en SAVEn...

Een ander punt is, dat het geluidsniveau van BEEP, hetzelfde is als dat van alle drie de kanalen van PLAY tegelijk. In de praktijk wil dit zeggen dat BEEP behoorlijk wat harder klinkt dan PLAY. Dat kan problemen geven, indien het geluidsniveau aan bepaalde grenzen gebonden is.

Het is niet gevaarlijk om een versterker aan te sluiten of af te koppelen terwijl de +2 ingeschakeld staat.

SOUND-uitgang :



Printer (en andere seriële apparaten)

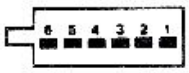
U kunt de +2 gebruiken met de meeste seriële printers die aan de RS232-standaard voldoen. Onervaren gebruikers raden we sterk af om te experimenteren met de aansluitingen van de interface. Uw Sinclair dealer kan u een geschikte kabel bezorgen, waarmee u uw printer met uw computer kunt verbinden. Volg ook steeds de instructies in het handboek van uw printer, wat de aansluiting en het in werking stellen aangaat.

U sluit de printer aan op de RS232/MIDI-uitgang achterop de +2.

Om een of ander serieel apparaat aan de +2 aan te sluiten, hebt u een seriële kabel voor de +2 nodig. Die kunt u krijgen bij uw Sinclair dealer. Wilt u er zelf een maken, dit zijn de aansluitingen :

Pen	Functie
1	aarding
2	TXD
3	RXD
4	DTR
5	CTS
6	+12 V

RS232 stekker :



MIDI-apparaat

Alhoewel de MIDI-poort van de +2 (Musical Instrument Digital Interface) in dezelfde stekkerbus uitmondt als de RS232, hebt u er toch een andere kabel voor nodig. Uw Sinclair dealer kan u die bezorgen. Die kabel moet u aansluiten op de "MIDI IN"-stekkerbus van uw synthesizer, drum machine enzovoort. De +2 kan géén MIDI-data ontvangen; hij kan enkel als bron fungeren. Om de MIDI te gebruiken moet u geen extra commando's geven, behalve degene die we bij PLAY hebben besproken.

Het gebruik van de MIDI laat de instelling van de baud rate voor de RS232 ongemoeid.

MIDI-stekker :

Pin	functie
1	RETURN
2	niet gebruikt
3	niet gebruikt
4	niet gebruikt
5	DATA UIT
6	niet gebruikt

Los toetsenbord

Het los toetsenbord biedt een groot aantal mogelijkheden bij het aanpassen van programma's, zoals "beweeg per bladzijde", "wis per woord", "wis tot einde van de regel". U kunt het ook gebruiken als toetsenbord voor de calculator. Uw Sinclair dealer kan u zegen of het verkrijgbaar is.

Het toetsenbord dient aangesloten op de KEYPAD-stekkerbus achterop de +2.

Interface I en microdrives

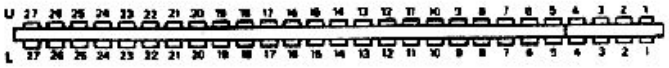
De +2 kan werken met Interface I en de microdrives. Bij aankoop ervan krijgt u een volledige handleiding. Uw Sinclair dealer kan u er meer over vertellen.

Andere uitbreidingen

U kunt de +2 op een breed gamma van apparaten aansluiten, via de EXPANSION I/O-bus achterop de machine. Hoewel die bus vrijwel op dezelfde manier aangesloten is als die van de oude 48K Spectrum, is het niet absoluut zeker dat een apparaat dat perfect werkte met de Spectrum 48K, dat ook zal doen met een +2. Voordat u dus een uitbreiding aanschaft, is het dus veilig om eerst te controleren of het wel werkt met de +2 en niet alleen met de Spectrum 48K.

WAARSCHUWING : Het is zeer gevaarlijk om een apparaat op de EXPANSION I/O-bus aan te sluiten of af te koppelen, terwijl de +2 aan staat. Zeer waarschijnlijk beschadigt u zowel de +2 als het randapparaat zelf, indien u dit doet.

EXPANSION I/O socket:



PIN	UPPER ROW (U)	LOWER ROW (L)
1	A15	A14
2	A13	A12
3	D7	+5V
4	not used	+9V
5	D0	0V
6	D1	0V
7	D2	CK
8	D6	A0
9	D5	A1
10	D3	A2
11	D4	A3
12	$\overline{\text{INT}}$	$\overline{\text{IORQGE}}$
13	$\overline{\text{NMI}}$	0V
14	$\overline{\text{HALT}}$	not used
15	$\overline{\text{MREQ}}$	not used
16	$\overline{\text{IORQ}}$	not used
17	$\overline{\text{RD}}$	not used
18	$\overline{\text{WR}}$	$\overline{\text{BUSRQ}}$
19	-5V	$\overline{\text{RESET}}$
20	$\overline{\text{WAIT}}$	A7
21	+12V	A6
22	-12V	A5
23	$\overline{\text{M1}}$	A4
24	$\overline{\text{RFSH}}$	$\overline{\text{ROMCS}}$
25	A8	BUSACK
26	A10	A9
27	not used	A11

INDEX

TERMINAL

HANDBOEK VOOR SINCLAIR ZX SPECTRUM 128K PLUS 2

Een ieder die werkt met deze computer zal behoefte hebben aan een duidelijke handleiding voor het gebruik van deze computer.
Dit handboek geeft een ieder de informatie, die hij nodig heeft om optimaal met deze unieke computer te kunnen werken.

De Spectrum 128K is in wezen een twee-in-een computer; een Spectrum 48K en een Spectrum 126K.
De beide systemen die in deze computer zijn ondergebracht, worden dan ook uitgebreid behandeld. Zowel de BASIC 48 als de BASIC 128 en het werken de beide BASICs wordt uitgebreid uit de doeken gedaan.

Daarnaast bevat het boek veel technische informatie zoals de Systeemvariabelen, de geheugenstructuur, informatie over het aansluiten van randapparatuur enz., enz.

Het boek wordt afgesloten met een aantal kant-en-klare programma's.

ISBN 90-6883-029-5 NUGI 851

TERMINAL SOFTWARE PUBLICATIES

HANDBOEK ZX SPECTRUM 128 PLUS 2

Terminal

SINCLAIR SERIE

HANDBOEK voor ZX SPECTRUM 128 + 2

I. Spital
R. Goodwins

TERMINAL SOFTWARE PUBLICATIES