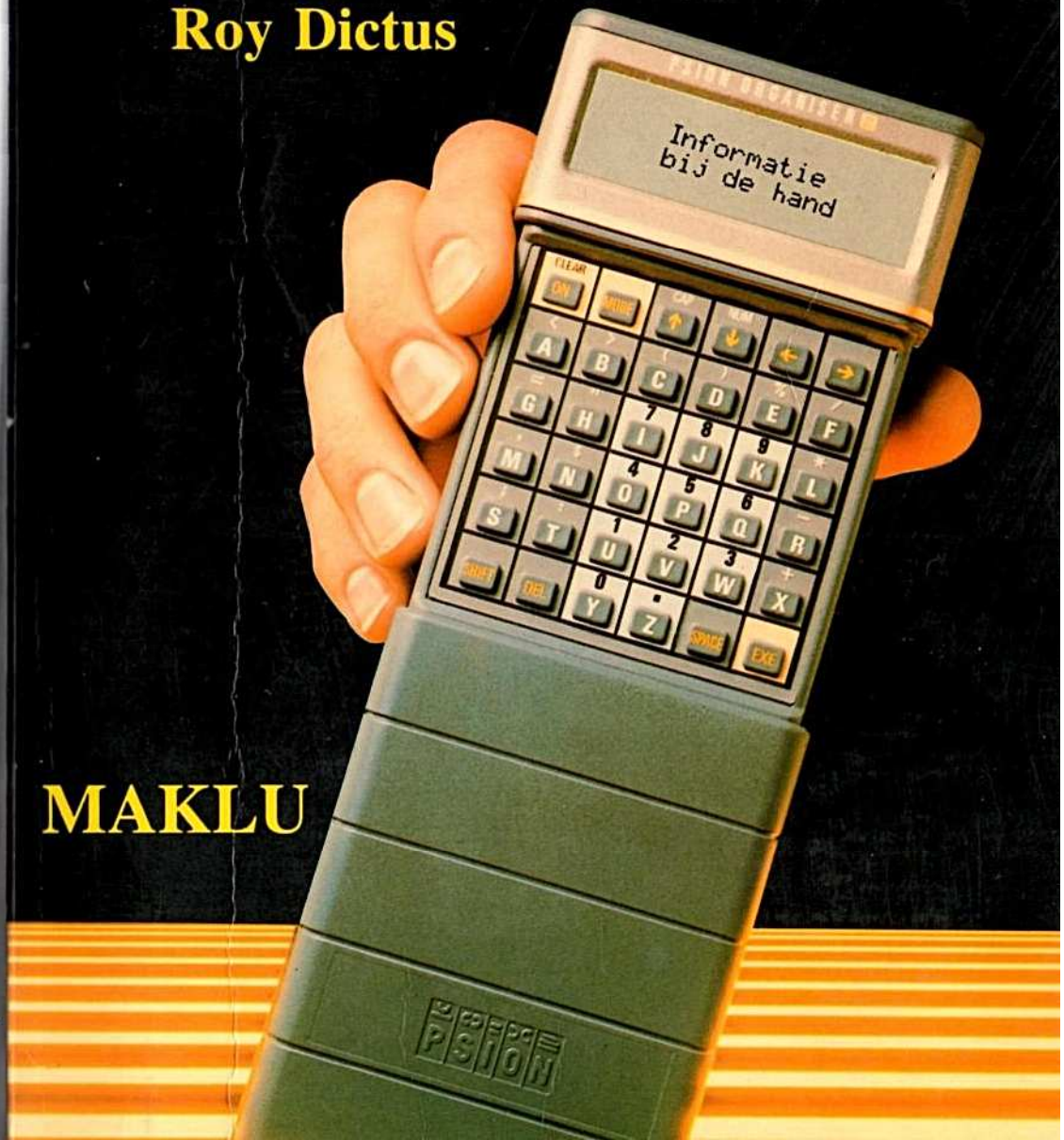
WERKEN MET DE ORGANISER II

Tweede, herziene druk

Roy Dictus



MAKLU

WERKEN MET DE ORGANISER II

Roy Dictus

WERKEN MET DE ORGANISER II

MAKLU Uitgevers
Antwerpen – Apeldoorn

Dit boek is opgedragen aan al mijn vrienden en vriendinnen.

Dankwoord
Ik wil mijn oprechte dank uitspreken aan alle personen die hebben bijgedragen tot de uiteindelijke totstandkoming van dit boek. Speciaal wens ik te danken : Johan Monsieur, Pierre Dictus, iedereen van Micro-Connection, Psion, Bart van Zwol, Wouter de Backer, Mieke Wattele-Laureyssens en Carla Boeykens-Orens.

Tussen de publicatie van de eerste en deze tweede druk vervulde ik mijn legerdienst op de Cie HK van het Eerste Belgische Legerkorps te Weiden, nabij Keulen. Daar werkte ik als programmeur op de dienst Sec AG, waar ik een echt goede tijd heb gehad en waar de prettige werksfeer altijd een stimulans is geweest. Ik bedank iedereen op de Sec AG en vooral Majoor Sys, Commandant Provost, 1e Sergeant-Majoor Huysegoms en mijn collega-dienstplichtige Robby Matthijs. Een speciale vermelding verdienen ook Dirk Van den Sanden, Rainer Fickers, Kristian Vanderwaeren, Bart Van Rooy en Jean-Michel Serpe. Hartelijk dank.

WERKEN met de ORGANISER II
ISBN 90 6215 227 9
D/1989/1997/6
NUGI 851

1ste druk: 1988
2de druk: 1989 (vermeerderd)

© Roy Dictus
MAKLU Uitgevers
Antwerpen – Apeldoorn

Behoudens uitzondering door de Wet gesteld, mag zonder schriftelijke toestemming van de rechthebbende(n) op het auteursrecht, c.q. de uitgever van deze uitgave, door de rechthebbende(n) gemachtigd namens hem (hen) op te treden, niets uit deze uitgave worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm of anderszins, hetgeen ook van toepassing is op de gehele of gedeeltelijke bewerking.
De uitgever is met uitsluiting van ieder ander onherroepelijk door de auteur gemachtigd de door derden verschuldigde vergoeding voor kopiëren, als bedoeld in artikel 17 lid 2 der Auteursweg 1912, te doen innen door en overeenkomstig de reglementen van de Stichting Reprorecht te Amsterdam.

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

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

INHOUD

Deel IV PROGRAMMEREN MET CM, XP EN ZL

Inleiding bij de tweede druk

Op 18 april 1988 verscheen de eerste uitgave van dit boek : de eerste Nederlandstalige handleiding voor de Psion Organiser II. Het werd meteen enthousiast onthaald. Vele mensen ondernamen hun eerste stappen in de wereld van de Organiser II met dit boek. Hun spontane reacties waren een inspiratie en hebben zeker bijgedragen tot de realisatie van deze tweede, herziene druk, waarvoor mijn dank.

De Organiser II is een klein maar handig toestel. Met een waaier van mogelijkheden en een ruim geheugen biedt deze machine een oplossing voor iedereen die zijn zaken onder controle wil houden. Of men nu zakenman (of -vrouw) is, havenarbeider of student, de Organiser II houdt de hoofdzaken in de hand.

Ook deze handleiding is geschreven voor iedereen. Dit handboek wil een gids zijn voor elke gebruiker, of die nu door beroep of hobby iets met informatica te maken heeft of niet. Voor de duidelijkheid is het boek ingedeeld in een aantal delen:

Deel I beschrijft algemeenheden omtrent alle modellen van de Organiser II (CM, XP en LZ), en geeft een beknopte inleiding informatica voor wie daar behoefte aan heeft.
Deel II dan is de algemene handleiding voor de modellen CM en XP.
Deel III is de algemene handleiding voor het model LZ.
Deel IV behandelt het programmeren van de drie Organiser II's.
Deel V behandelt de verkrijgbare randapparatuur.
De Appendices bevatten o.a. een verklarende jargonlijst.

Nu is het tijd uw Organiser II ter hand te nemen en de bladzijde om te slaan. Laat de wereld van de Psion Organiser II voor u opengaan!

Roy Dictus, april 1989

Deel I

ALGEMEENHEDEN

1. De machine

De Organiser II ziet eruit als een (grote) rekenmachine met veel knopjes. De afmetingen zijn : 142 mm lang, 78 mm breed en 29 mm hoog. Het gewicht (zonder batterij) bedraagt 250 gr.

1.1 Het scherm

De schermen van de CM en XP hebben elk 2 lijnen van 16 tekens; het scherm van model ZL heeft 4 lijnen van 20 tekens. Het zijn LCD-schermen (Liquid Crystal Display: de letters worden weergegeven door middel van vloeibare kristallen, net zoals vele digitale horloges). De tekens (letters, cijfers e.d.) worden weergegeven in een zgn. 'puntmatrix'. Het scherm van de ZL is net iets groter dan dat van de andere modellen.

1.2 Het toetsenbord

Het toetsenbord van de Organiser II is een hele verbetering ten opzichte van de eerste Organiser. Het zijn echte toetsen, die een goede respons geven, repeterend zijn en een bevestigend klikgeluid geven bij het indrukken (als de Organiser II aan staat). Er zijn 36 toetsen. Bijna iedere toets heeft meer dan één functie.

Het toetsenbord is gecodeerd met kleuren. Dat betekent dat alle numerieke toetsen (0 - 9) in het *lichtblauw* zijn aangeduid, alle speciale toetsen in het *geel* en alle verdere toetsen een *grijze* achtergrond hebben. Dit vergemakkelijkt het werken omdat iedere soort toets direct identificeerbaar is.

1.3 Het interne geheugen

Zoals alle computers, grote en kleine, heeft de Organiser II een *intern geheugen*. Dit betekent dat er vanbinnen een chipje zit, dat van alles kan onthouden. Om precies te zijn, is het dit chipje dat de *agenda* van de Organiser II onthoudt, de *ingebouwde alarmen*, de eventuele telefoonnummers en programma's.

In de Organiser II model CM bevat dit chipje 8192 tekens (men spreekt van een geheugen van 8K, d.i. 8 × 1024 tekens = 8192 tekens). De Organiser II modellen XP en LZ hebben een geheugen van 32K (32768 tekens).

Maar dit intern geheugen heeft één nadeel : wanneer de batterij uit de machine wordt gehaald, of is opgebruikt... dan wordt het gehele interne ge-

heugen *uitgewist*. Dit risico moeten we zoveel mogelijk opvangen. Daarom heeft Psion ook gezorgd voor *externe* gegevensopslag : de zgn. 'datapaks'.

1.4 De datapaks

De datapaks zijn een beetje kleiner dan lucifersdoosjes. Ze dienen voor de externe opslag van gegevens en programma's en kunnen uit de computer worden gehaald *zonder dat het geheugen wordt uitgewist*. Men kan ook gerust een datapak van de ene Organiser II in de andere doen... We zouden dit kunnen vergelijken met muziekcassettes. Men kan gerust een cassette opnemen op één recorder, de cassette eruit halen en weer afspelen op een andere. Dit afspelen kan onbeperkt worden herhaald (in de praktijk tot er iets ongezonds met de datapak of cassette gebeurt). De capaciteit kan per datapak of cassette verschillen.

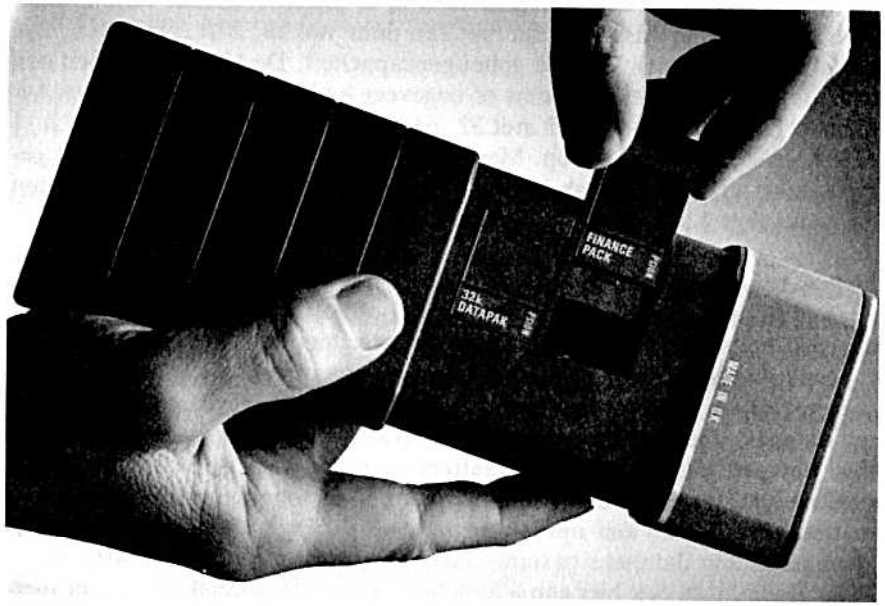
Aan de onderkant van de Organiser II is er plaats voorzien voor 2 datapaks. Net zoals er cassettes zijn met een duur van 15, 20, ... minuten, zijn er datapaks met verschillende geheugencapaciteit. De kleinste hebben een capaciteit van 16K (16384 tekens of ongeveer 8 bladzijden getypte tekst A4-formaat). Vervolgens zijn er met 32, 64 en 128K.
Datapaks zijn niet goedkoop. Maar een 128K-datapak kan 64 pagina's getypte tekst opslaan... Toch heel wat! Helaas voor CM-gebruikers kunnen datapaks met 128K alleen in XP- en LZ-Organisers gebruikt worden.

Helaas hebben ook de datapaks nadelen. Allereerst zijn ze moeilijk *uitwisbaar* (dat kunt u laten doen in de winkel of zelf, thuis, met een hiervoor speciaal ontwikkeld apparaat, waarover later meer). Bovendien, wanneer u bepaalde gegevens uitwist (gegevens over een offerte die verlopen is, bijvoorbeeld), zijn ze niet meer leesbaar maar blijven wel plaats innemen op uw datapak. De enige manier om ze werkelijk weg te krijgen is de volledige datapak uit te wissen ('formatteren') zodat weer de volle 100 % capaciteit kan benut worden. Een datapak kan tot ongeveer 100 keer geformatteerd worden en kan zijn informatie naar raming tot 50 jaar vasthouden. Bovendien zijn datapaks bestand tegen temperaturen van max. 100 °C.
Maar Psion biedt ook hier een alternatief! Buiten de datapaks verkoopt men voor de Organiser II ook nog de 'RAMpaks'.

1.5 De RAMpaks

RAM betekent 'Random Access Memory', d.w.z. geheugen dat vrij toegankelijk is. RAMgeheugen kan worden uitgewist en de vrijgekomen plaats kan weer worden benut. Het interne geheugen van uw Organiser II is trouwens RAMgeheugen.

Psion maakt een RAMpak met een capaciteit van 32K. Ook deze paks kunnen uit de Organiser II worden gehaald zonder inhoudsverlies. Ikzelf gebruik een RAMpak omdat ik het veel handiger vind dan een datapak, bovendien is de gemiddelde toegangstijd veel korter omdat de gegevens niet 'gebrand' hoeven te worden zoals in de datapaks. Maak er echter een gewoonte van steeds copieën te hebben van alle gegevens die op RAMpak staan, want deze vorm van opslag is zeer kwetsbaar. Een voorbeeld uit het leven gegrepen : iedere Organiser II heeft een 'versienummer'. Dit nummer verandert telkens als men bij Psion iets wijzigt in de structuur van de Organiser II of bijvoorbeeld in de ROM (zie volgend punt). Normaal is dit geen probleem, behalve bij de RAMpaks. *Een RAMpak die gebruikt wordt met een Organiser II versie x zal al zijn inhoud verliezen wanneer men probeert hem te gebruiken met een Organiser II versie y*. Ik wou eens een programmaatje van mij, dat ik op een RAMpak had gezet, copiëren naar het interne geheugen van de Organiser II van een vriend. Die vriend had echter een vroegere versie van de Organiser II dan ik, met als gevolg dat



ik alle gegevens op die RAMpak kwijt was. Gelukkig had ik van alle gegevens en programma's voor de veiligheid copieën gemaakt !

1.6 De ROM

De ROM (Read-Only Memory) is een chip die binnenin het toestelletje zit. Hij bevat een lang programma dat door de specialisten van Psion geschreven is. Dit programma 'stuurt' de Organiser II : alle functies worden door dit programma in de ROM gecontroleerd. De Organiser II heeft een ROM van 32K.

1.7 De microprocessor

Maar het echte hart van iedere computer is zijn *microprocessor*. Een microprocessor is een electronisch toestel – wederom een chip – die opdrachten kan uitvoeren en aldus de machine besturen. De ROM is, zoals gezegd, een programma. Dit programma vertelt de microprocessor wat hij moet doen.

Er zijn verschillende soorten microprocessors met verschillende eigenschappen. De PC's van de Amerikaanse computergigant IBM bijvoorbeeld gebruiken de i8088 en i80286 microprocessors. De Organiser II gebruikt de HD6303X, een krachtige processor die weinig stroom verbruikt.

1.8 De 'interface' voor communicatie met de buitenwereld

Aan de bovenkant van de Organiser II bevindt zich een gleufje dat opengeschoven kan worden. Hieronder bevindt zich de 'interface' : een verbinding die dient om de Organiser II aan te sluiten op andere machines, zoals PC's (personal computers), printers (afdrukeenheden), barcode-readers (streepjescodelezers) enzovoort. Via deze aansluitmogelijkheden kunnen we dus alle gegevens die in de Organiser II zijn opgeslagen afdrukken, we kunnen gegevens overseinen van/naar micro- en minicomputers, ... Dit vergroot de kracht en de toepassingsmogelijkheden van de Organiser II aanzienlijk.

1.9 De batterij

Psion raadt aan een batterij van het type 'PP3 Long- Life Alkaline' te gebruiken voor uw Organiser II. Met een dergelijke batterij kan uw Organiser II ongeveer 4 tot 6 maanden (naargelang de mate van gebruik) werken.

Wanneer u de batterij vervangt, raad ik u aan snel te zijn. Na 10 tot 30 seconden (Psion beweert 300 seconden, maar de ervaring leert dat dit te lang is) zonder elektrische stroom verliest het interne geheugen zijn inhoud. De interne klok en datum, die anders automatisch worden bijgehouden, worden op nul gezet, uw gegevens worden uitgewist. U kunt natuurlijk ook eerst uw interne geheugen naar een datapak copiëren.

2. Beknopte inleiding informatica

2.1 Wat is informatica ?

Informatica is de wetenschap die zich bezighoudt met informatieverwerking. Met de opkomst van de computer is in feite een nieuwe wetenschap geboren, die gespecialiseerd is in het verwerken van grote hoeveelheden gegevens : numerieke gegevens (boekhouding bijv.), teksten (boeken, artikelen), woordenlijsten, klantenlijsten e.d. Dank zij de informatica kunnen binnen al deze gegevens bepaalde bewerkingen plaatsvinden zoals sorteren, zoeken, lijsten afdrukken, ... Deze organisatie van gegevens is waarschijnlijk ook de reden dat u uw Organiser II gekocht heeft !

2.2 Gegevens ⟷ Informatie

De termen "gegevens" en "informatie" hebben een verschillende betekenis. Ze worden helaas maar al te vaak verkeerd gebruikt.

Een *gegeven* is een losstaand iets. *Informatie* bestaat uit één of meer gegevens die een betekenis hebben gekregen.

Bijvoorbeeld : als ik tegen iemand zeg : "Het is kwart voor vijf", dan is dat voor die persoon een gegeven. Hij weet nu dat het kwart voor vijf is, maar het uur heeft geen specifieke betekenis voor hem. Als er iemand aan mij vraagt : "Hoe laat is het ?" en ik antwoord : "Het is kwart voor vijf", dan is het uur voor die bewuste persoon informatie : het heeft betekenis voor hem.

Bij computers spreken we in 99 % van de gevallen van gegevens : de term "informatica" is dus enigszins misplaatst.

2.3 De computer

Iedere computer bestaat uit twee vaste componenten en andere, optionele componenten. De vaste componenten zijn de microprocessor en het interne geheugen. De microprocessor is een chip die de computer bestuurt; hij is de eigenlijke computer. Alle programma's worden door deze microprocessor uitgevoerd. Het interne geheugen kan opgesplitst worden in vast geheugen dat niet te wijzigen is (ROM : Read-Only Memory) en veranderlijk geheugen (RAM : Random Access Memory). De ROM bevat gewoonlijk het programma dat de verbinding gebruiker- microprocessor vormt (ook in de Organiser II). Het RAMgeheugen wordt door de computer ge-

bruikt voor tijdelijke opslag van gegevens. Wanneer de stroom wordt uitgeschakeld, is de inhoud van de RAM verloren maar de ROM blijft intact.

De optionele componenten zijn : apparaten voor externe opslag van gegevens, die niet worden uitgewist bij het uitschakelen van elektrische stroom. Bij grotere computers spreken we van *magnetische schijven* die de gegevens bijhouden. De Organiser II heeft daar zijn datapaks voor. Verder zijn er *communicatielijnen* voor het overseinen van gegevens via een 'link' (= schakel), telefoonlijn, ... en printers. Er zijn ook nog een heleboel mogelijkheden met randapparatuur, waarover we het later zullen hebben.

2.4 Opslag van gegevens

Hoe werkt de gegevensopslag? Hoe kan een computer namen, adressen, getallen onthouden ?

In de eerste plaats : een computer onthoudt *geen* alfanumerieke gegevens (namen, adressen, enz.). Hij onthoudt alleen *getallen*. We kunnen het zelfs sterker stellen : de computer kan slechts twee verschillende cijfers onthouden : 0 en 1. Het computergeheugen zit dus vol nullen en enen (ook programma's zijn uiteindelijk uit nullen en enen opgebouwd).

De truuk zit hem hierin : door die nullen en enen te groeperen kan de computer een getal als bijv. 2559 onthouden. Deze groeperingen noemen we *bytes* : één byte is een groep van 8 enen en/of nullen. Voorbeelden van bytes zijn :

```
00100111    11111001    00000001    00000000
```

(dit zijn vier bytes).

Een byte stelt dus een getal voor, dat is opgebouwd uit 0 en 1 (0 en 1 zijn zgn. 'bits' : bit staat voor BInary digiT of binair cijfer). Zo'n getal noemen we *binair* (d.w.z. 'tweetallig'). Wij zijn gewend met het tientallig stelsel te werken (decimaal) : het heeft tien cijfers (0-9) terwijl het binair stelsel er slechts 2 heeft (0-1). Dit is het enige verschil. Het decimaal getal 0 wordt in binair weergegeven als `00000000`, 1 als `00000001`, 2 als `00000010`, 200 als `11001000`, ... Het hoogste getal dat men in één byte kan opslaan is `11111111` = 255 decimaal.

Door een speciale codering toe te passen (ASCII-code : American Standard Code for Information Interchange) kan men ook letters en leestekens opslaan in het computergeheugen. Die worden dan eenvoudigweg opgeslagen als... getallen. De hoofdletter 'A' heeft als code bijvoorbeeld 65, 'B' heeft 66, 'a' heeft 97, een spatie heeft code 32, ... Door lange strengen getallen aan elkaar te rijgen (strings) krijgen we gegevens als namen en adressen.

Het RAMgeheugen is, wanneer nog ongebruikt, gevuld met allemaal nullen. Door een gegeven in de Organiser II in te tikken, worden enkele plaatsen in het geheugen opgevuld. Zo een geheugenplaats noemen we *adres*. Een geheugen van 8 K (= 8192 tekens) heeft dus 8192 adressen : genummerd van 0 tot 8191.

2.5 Talstelsels

Informatica is geen gemakkelijke wetenschap. In de informatica kent en gebruikt men verschillende *talstelsels* en jammer genoeg komt het meestgebruikte talstelsel, het decimale dat wij allemaal dagelijks gebruiken, in de echte informatica (vanuit het standpunt van de programmeur) weinig aan bod.

In de eerste plaats hebben we natuurlijk het *binaire* stelsel. Verder hebben we het *octale* (8-tallig) en het *hexadecimale* (16-tallig). Het octale stelsel wordt steeds minder gebruikt. Het meestgebruikt is het hexadecimale stelsel.

2.5.1 Hexadecimaal tellen

In 'hex', zoals dit stelsel vaak wordt genoemd, hebben we 16 cijfers : 0-9 en A-F. We tellen dus :

```
0 1 2 3 4 5 6 7 8 9 A B C D E F
```

Het decimale getal '10' wordt in hex dus voorgesteld als 'A', 11 als 'B', enz. Het decimale getal 16 wordt in hex geschreven als 10, 200 als C8, 255 als FF.

2.5.2 Lijst talstelsels

Hieronder vindt u een klein vergelijkend lijstje van getallen in verschillende talstelsels.

Dec	Bin	Hex	Dec	Bin	Hex
0	00000000	00	11	00001011	0B
1	00000001	01	12	00001100	0C
2	00000010	02	13	00001101	0D

3	00000011	03	14	00001110	0E
4	00000100	04	15	00001111	0F
5	00000101	05	16	00010000	10
6	00000110	06	17	00010001	11
7	00000111	07	18	00010010	12
8	00001000	08	19	00010011	13
9	00001001	09	20	00010100	14
10	00001010	0A	21	00010101	15

. . .

. . .

2.5.3 Omzetten van binaire getallen

In de informatica bestaat een byte dus uit 8 bits. Die 8 bits kunnen een getal tussen 0 en 255 voorstellen... Hierboven staan de binaire voorstellingen van 0 tot 21. Maar hoe *berekenen* we een binair getal ?

Iedere bit heeft zijn plaats in de byte. Die plaatsen zijn genummerd van 0 tot 7 (van rechts naar links). Wanneer we over de zesde bit spreken, meestal aangeduid als D6 (en niet B6), hebben we het dus over de zevende bit van rechts.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

Een bit kan een waarde 0 of 1 hebben. Wanneer bit *x* de waarde 0 heeft, zeggen we dat hij 'uit' is; wanneer hij de waarde 1 heeft, is hij 'aan'. Iedere bit kan dus individueel 'aan' of 'uit' gezet worden.

Nu gaan we een binair getal omzetten naar decimaal. We nemen het getal

01101101

Dit kunnen we dus voorstellen als :

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	1	1	0	1

De inhoud van iedere bit wordt vermenigvuldigd met zijn macht van 2.

$D7 = 2^7 = 128$
$D6 = 2^6 = 64$
$D5 = 2^5 = 32$
$D4 = 2^4 = 16$

$D3 = 2^3 = 8$
$D2 = 2^2 = 4$
$D1 = 2^1 = 2$
$D0 = 2^0 = 1$

Overal waar de bit 1 is, tellen we de corresponderende waarde op. Daarmee wordt ons getal 01101101 :

64 + 32 + 8 + 4 + 1 = 109.

2.5.4 Omzetten van hexadecimale getallen

Een hexadecimaal getal bestaat steeds uit een even aantal hexadecimale cijfers (meestal 2 of 4). Wanneer het hexadecimale getal uit 2 cijfers bestaat, berekenen we de waarde heel eenvoudig :

(16 × het linkse cijfer) + het rechtse cijfer

Het getal F9 wordt dus :

(16 × F) + 9 = 16 × 15 + 9 = 240 + 9 = 249.

Wanneer we met een getal van 4 cijfers te maken hebben, splitsen we dit eerst op in een koppel van elk twee cijfers, die we individueel berekenen (bv. FF4E wordt FF en 4E). We nemen nu het eerste getal (FF), vermenigvuldigen dit met 256 en tellen er het tweede getal (4E) bij op. We krijgen :

FF4E = 256 × (FF) + 4E
= 256 × 255 + 78
= 65280 + 78
= 65358.

2.6 Notatie van hex

Hexadecimale getallen worden bijna steeds weergegeven met een herkenningsteken; vaak krijgen zij de suffix 'h' of 'H'. FF4E wordt dus :

FF4Eh of F4EH.

Meestal krijgen zij echter de prefix '#' of '$'. De Organiser II gebruikt de populairste prefix om een hex getal weer te geven : het dollarteken ($).

Tot hier de beknopte inleiding informatische begrippen. Genoemde begrippen vindt men in de informatica overal terug; of het nu met kleinere

toestelletjes is (zoals bij de Organiser II) of bij kamervullende 'number crunchers' (= supercomputers). Deze inleiding illustreert ook dat de informatica geen gemakkelijk iets is dat snel wordt aangeleerd : er is tijd voor nodig om overal inzicht in te krijgen. Na het lezen van dit boek zal dat, ten aanzien van de Organiser II althans, geen probleem meer zijn.

Deel II

WERKEN MET CM EN XP

1. Het hoofdmenu

We zetten nu de Organiser II aan, door de toets linksboven (gemarkeerd ON) in te drukken.

We zien nu een gedeelte van het hoofdmenu van de Psion Organiser II :

```
FIND SAVE DIARY
CALC PROG ERASE
```

Dit hoofdmenu presenteert een aantal *opties* (FIND, SAVE, enz...). Telkens als we iets willen doen met de Organiser II (een gegeven intikken bijvoorbeeld) zullen we een optie moeten *kiezen*.

Over de letter 'F' van FIND zien we een blokje aan en uit knipperen : dit blokje noemen we de *cursor*. We zeggen dat de cursor de optie FIND aanduidt. Door de cursor te verplaatsen kunnen we hem ook alle andere opties laten aanduiden. Laten we het eens proberen.

Eerst bekijken we de bovenste rij toetsen op ons toetsenbord. De eerste toets, ON, gebruiken we om de Organiser II aan te zetten. De tweede, MODE, zullen we later nog gebruiken. Zeer interessant zijn de volgende vier toetsen waarop pijltjes staan. In dit boek zullen we naar die pijltjes verwijzen met OP, NEER, LINKS en RECHTS. De aanduidingen CAP en NUM zullen we later nog gebruiken.

Als we onze Organiser II een tijdje ongemoeid laten, zullen we zien dat hij zich na enkele minuten automatisch uitschakelt. Dit is een handigheidje dat Psion heeft ingebouwd om het vroeger meestal dramatische effect van nalatigheid van de gebruiker tot een minimum te reduceren. Het spaart met andere woorden onze batterij. Als we weer op ON drukken, zijn we terug waar we het laatst gebleven waren.

Druk nu voorzichtig op RECHTS. De cursor zal nu op de S van SAVE staan. Als u een foutje hebt gemaakt, druk dan op ON en probeer het nog eens.

Druk op NEER. De cursor staat op CALC.

Druk nogmaals op NEER. Nu staat er op het scherm :

```
CALC PROG ERASE
TIME INFO ALARM
```

Er staan dus drie nieuwe opties op het scherm. De cursor duidt TIME aan.

Druk weer op NEER. Het scherm geeft weer :

```
TIME INFO ALARM
COPY RESET OFF
```

Nogmaals NEER geeft :

```
FIND SAVE DIARY
CALC PROG ERASE
```

We hebben het hele menu doorlopen en zijn opnieuw bij de eerste regel aanbeland.

1.1 Opslaan/opzoeken van gegevens

De cursor staat op FIND. FIND betekent : zoeken naar een gegeven (bijvoorbeeld het telefoonnummer van een winkel). Stel dat we het telefoonnummer van de winkel MICRO-CONNECTION willen weten (dit staat echter nog niet in het geheugen). We gaan nu de optie FIND *uitvoeren*. We drukken op de toets EXE (rechtsbeneden). EXE staat voor *execute* (uitvoeren).

Men kan een optie ook selecteren door eenvoudigweg de eerste letter van die optie in te typen. We kunnen de Organiser II dus afzetten door op de 'O' (de letter O, niet het cijfer 0) te drukken wanneer we in het hoofdmenu zijn. Als er in het menu twee of meer opties zijn met dezelfde beginletter, laat de Organiser II ons kiezen door de cursor beurtelings naar iedere optie te sturen.

Het scherm geeft weer :

```
FIND A:_
```

FIND duidt aan dat we de FIND-optie geselecteerd hebben. A: betekent dat we gaan zoeken naar gegevens op de pak die met 'A:' wordt aangeduid; iedere datapak heeft immers een naam (letteraanduiding). 'A:' betekent 'het interne geheugen'; 'B:' staat voor de eerste datapak (bovenste); 'C:' tenslotte duidt op de tweede datapak (onderste). We gaan dus gegevens zoeken in het interne geheugen ('A:').

Om de gegevens van MICRO-CONNECTION op te zoeken, volstaat het de Organiser II een gedeelte van de naam op te geven, bijvoorbeeld 'MICRO' of 'CONNEC'. We tikken in :


```
FIND A:MICRO
```

(N.B. : FIND A: hoeven we niet in te tikken !)

We drukken nu op EXE.

FIND gaat nu in het interne geheugen van de Organiser II op zoek naar het woord MICRO. Dat zal niet gevonden worden, want er staan nog geen gegevens in het geheugen! Resultaat :

```
***************
**END OF PACK**
```

(wat zoveel betekent als : ik heb het hele 'pack' A: doorzocht, ik heb het woord MICRO niet gevonden). Druk nu op ON.

We gaan de gegevens van MICRO-CONNECTION invoeren. We sturen met de cursor naar SAVE (RECHTS) en drukken op EXE.

```
SAVE A:
```

We hebben de volgende gegevens van MICRO-CONNECTION :

MICRO-CONNECTION is de exclusieve invoerder van de
Psion Organiser II en bijbehoren voor de Benelux.
Adres : St. Katelijnevest 18, B-2000 Antwerpen
Telefoon : 03/231 01 03

We korten dit in tot :

MICRO-CONNECTION
03/231 01 03
St. Katelijnevest 18
B-2000 Antwerpen
Excl. invoerder Benelux Psion Organiser II

We tikken in :

```
SAVE A:MICRO-CONNECTION
```

(het '-'-teken krijgen we door de toets linksbeneden, SHIFT, in te drukken en ingedrukt te houden; vervolgens R in te drukken (rechtse rij, derde toets van onderen); nu komt de '-' op het scherm. Laat eerst de R los, daarna SHIFT).

Als we een tikfout maken, geen nood. Die kunnen we immers ongedaan

maken. De tweede toets op de onderste rij, DEL, (afkorting voor 'delete': uitwissen) dient om fouten te verwijderen. Druk voorzichtig op DEL; de fout is weg en de cursor is een plaats naar links opgeschoven.

We zijn nu klaar om de tweede lijn in te tikken.

Druk op NEER.

```
SAVE A: MICRO-CON
```

(maak u geen zorgen : de naam 'MICRO-CONNECTION' is niet afgekort, maar omdat hij niet volledig op de eerste lijn kan, laat de Organiser II enkel de eerste 9 tekens zien).

De cursor staat nu op de volgende regel. Tik de andere regels in.
Druk nu op EXE.

Onmiddellijk springt de Organiser II terug naar het hoofdmenu, met de cursor knipperend op FIND.

Druk op EXE : we gaan weer gegevens opzoeken. Herhaal de opzoekprocedure van daarnet, die niets opleverde omdat het geheugen nog leeg was. U zou nu de eerste twee regels die u heeft ingetikt moeten terugzien op het scherm. Om de andere regels te zien, drukt u wanneer nodig op NEER. Om terug naar boven te gaan, drukt u op OP.

Als u alles gelezen hebt, druk dan op EXE. Dit geeft de Organiser II de opdracht *verder te zoeken* in zijn geheugen naar de naam MICRO. Die is er echter niet meer (hij komt maar één keer voor) en dus krijgen we :

```
****************
**END OF PACK***
```

Druk op ON. En daar zijn we terug bij het hoofdmenu.

1.2 Andere paks

We volgen dezelfde procedure om gegevens op te slaan/op te zoeken in andere paks (data- of RAMpaks). Stop een pak in de eerste opening van de Organiser II (de 'datapak drive'). We kiezen SAVE.

```
SAVE A:
```

Druk nu op MODE. Als er inderdaad een pak in drive B: zit (de eerste opening), staat er op het scherm :


```
SAVE B:
```

Als u de pak in de tweede opening (drive C:) gestopt hebt, staat er :

```
SAVE C:
```

Tik nu wat gegevens in ... en druk op EXE.

Als de geïnserteerde pak een datapak is, zal de Organiser II opmerken dat deze nieuw en dus nog ongebruikt is. Nu is het zo, dat nieuwe datapaks door de Organiser II altijd moeten worden geïnitialiseerd (geformatteerd) voordat ze kunnen worden gebruikt om gegevens op te slaan. Wanneer men met bijvoorbeeld personal computers werkt, moet met magnetische schijfjes (floppy disks) hetzelfde gebeuren.

Zoals gezegd, ziet de Organiser II zelf dat het een nieuwe pak betreft; men hoeft het hem niet te vertellen. Automatisch wordt de datapak geïnitialiseerd :

```
SIZING PACK B:
PLEASE WAIT
```

De capaciteit van de pak wordt nagegaan en de Organiser II kijkt meteen even na of alles in orde is en de datapak voor de volle 100 % kan worden benut (het kan immers zijn dat bijvoorbeeld door een fabricagefout de pak defect is). Na enkele seconden is dit proces beëindigd en is men weer terug waar men was vóór het initialiseren van de pak.

2. Gegevens ordenen

2.1 Welke gegevens ?

We zouden de FIND/SAVE-functies van de Organiser II kunnen zien als een automatische kaartenbak. In een kaartenbak hebben we een hoeveelheid kaarten (= een bestand van kaarten) en op iedere kaart hebben we een aantal vaste gegevens. Bijvoorbeeld :

NAAM
ADRES
GEMEENTE
TELEFOONNR

Op onze kaarten kan dan staan :

Kaart 1	Kaart 2	Kaart 3
Magnum Thomas	Mr President	Charlie Chaplin
Masters Estate	White House	United Artists
Honolulu, HA	Washington, DC	Hollywood, CA
07/007 007	666 666	031 299

Op iedere kaart staat dus een aantal gegevens op verschillende regels. Dat kunnen we met de Organiser II ook, door de gegevens die we hebben op precies dezelfde manier in te typen.

2.2 Indeling

Iedere regel, die een bepaald gegeven bevat (bijv. naam), noemen we een *field* (= veld). Kaarten bestaan dus uit een aantal fields. Zo'n kaart noemen we dan niet langer kaart : het is een *record.* Ieder record bestaat dus uit een aantal bij elkaar horende velden in een zekere volgorde. De kaartenbak (verzameling bij elkaar horende velden) noemen we *file* of *bestand.* De Organiser II slaat dus bestanden op in zijn intern geheugen en op de datapaks. Alle gegevens (velden) die wij intikken vormen een record zodra we op EXE drukken en dit record aan het bestand wordt toegevoegd.

In de Organiser II hoeven we deze vaste recordindeling (opgenomen velden, volgorde van de velden) niet te volgen daar Psion geen vaste indeling

opgelegd heeft. We kunnen dus onbekommerd alles intikken wat we maar willen (max. 255 tekens per record), op zoveel regels als we willen (max. 16), in de volgorde die we willen. Ikzelf gebruik voor namen/adressen/telefoonnummers altijd de naam op de eerste lijn en het telefoonnummer op de tweede, omdat zo het terugvinden van een telefoonnummer het eenvoudigste is : je ziet meteen de naam en het bijbehorend nummer dat je nodig hebt op het scherm verschijnen.

3. De tijd instellen : TIME

In elke Organiser II zit een doorlopende klok ingebouwd (voor de technisch geïnteresseerden : een CMOS real- time klok met 36864 Hz kristalfrequentie). Ook wanneer de Organiser II niet aan staat, loopt deze klok door.

Neem uw Organiser II en zet hem aan (ON, linksboven). Verplaats de cursor naar de optie TIME en druk op EXE (of druk gewoon op T, de eerste letter van 'TIME'). Als u uw Organiser II pas heeft, ziet u bijvoorbeeld:

```
WED 1 JAN 1986
  01:10:02
```

In ieder geval is de datum onjuist en het uur zeer waarschijnlijk ook. Daar gaan we iets aan doen.

Daartoe introduceren we de toets MODE – die rechts van ON geplaatst is. Druk één keer op MODE.

De cursor staat nu op de dag van de maand (in ons bovenstaande voorbeeld op de 1). Met de pijltjes OP en NEER veranderen we de dag. Merk op dat niet alleen de dag van de maand, maar ook de weekdag verandert. Wanneer we met OP voorbij dag 31 gaan, springt de Organiser II automatisch terug naar dag 1. Omgekeerd, wanneer we met NEER lager dan 1 gaan, komen we bij 31 terecht – tenminste als de op het scherm aangeduide maand 31 dagen heeft, wat correct is in ons geval (januari). Bij februari verspringt de datum naar 28 of 29, alnaargelang het een schrikkeljaar is of niet (een slim ding zo'n Organiser II !).

Wanneer de juiste dag van de maand bereikt is, druk op RECHTS. De cursor staat dan op de maand (in ons geval JAN). Door OP of NEER te sturen, veranderen we de maand. Merk op dat ook nu de weekdagen automatisch worden aangepast. Wanneer we voorbij december gaan, komen we automatisch terug bij januari.

Nu is de maand ook juist ingesteld. Druk op RECHTS. De cursor staat nu bij het jaar. Stuur nu met OP naar het juiste jaar. (Merk op dat men het jaar ook kan verzetten met NEER – zelfs beneden 1980 – maar dat kan uiteraard nooit een juiste datum opleveren. Het gebruik van deze mogelijkheid bespreken we later).

Zo ... de datum is nu goed gezet. Nu nog het uur. Stuur wederom met RECHTS naar het uur.

De Organiser II houdt een 24-urenklok bij en maakt dus verschil tussen 11:00 en 23:00 uur. Let daarop wanneer u de klok gaat instellen. Gebruik hiervoor dezelfde methode als u voor de datum hebt gebruikt. Druk daarna op ON (om terug te keren naar het hoofdmenu).

De tijdfunctie is achter de rug. Steeds wanneer u het preciese uur en de datum nodig hebt, kunt u uw Organiser II raadplegen. Automatisch houdt die ook de dag van de week bij. Merk op dat de Organiser II, net zoals uw horloge, niet automatisch rekening kan houden met zomer- en wintertijd.

Men kan de tijdfuncie ook nog voor iets anders gebruiken. Stel dat u wil weten welke weekdag 25 augustus 1989 is. Verzet dan de datum tijdelijk naar 25/08/1989 en u weet direct het antwoord (vrijdag)!

4. Het wissen van gegevens : ERASE

Na dit korte intermezzo verder naar andere zaken. Het wissen van gegevens is immers bijna net zo belangrijk als het invoeren ervan (om plaats te sparen in RAMpaks of pak A:, om bepaalde verouderde gegevens kwijt te raken, om zeker te stellen dat vertrouwelijke gegevens niet uitlekken, enz...).

De Organiser II heeft hiervoor vanzelfsprekend een ingebouwde functie : ERASE.

ERASE ziet u op het schermpje uiterst rechts op de onderste regel als u de Organiser II aanzet.

Het werkt bijna identiek aan FIND : u tikt een gedeelte van het record in dat u wil opzoeken – bijvoorbeeld REST voor RESTAURANT – (u kunt net zo goed RANT of AURA gebruiken, of gelijk welk ander onderdeel van het record) en drukt op EXE. ERASE zoekt dan naar een record waar dat gedeelte in te vinden is en brengt dat record op het scherm (net als FIND). Als u dit record inderdaad wil wissen, drukt u op DEL (van DE-Lete, wat precies hetzelfde betekent als ERASE). De Organiser II vraagt dan :

DELETE Y/N

(wilt u dit record wissen, ja of nee ?)

Aangezien alles in verband met deze computer (en met de meeste andere computers, groot en klein) Engelstalig is, wordt van u verlangd 'Y' of 'y' in te drukken voor 'ja' en 'N' of 'n' voor 'nee' (overigens is er ook een Franse versie van de Organiser II verkrijgbaar).

Als u 'ja' kiest, wordt het bewuste record uit het bestand verwijderd. Kiest u voor 'nee', dan springt de Organiser II terug naar het punt vlak voordat u op DEL drukte. Druk op EXE voor het volgende record, en zo verder.

Het feit dat ERASE zo op FIND lijkt, beschouw ik als een positieve inbreng van Psion om programma's – de werking van dit kleine draagbare mirakel berust immers op een programma op ROM – zo gebruikersvriendelijk mogelijk te maken. Onder gebruikersvriendelijkheid dient te worden verstaan : een zo eenvoudig mogelijke operatie van de machine en de software, met zo weinig mogelijk menselijke tussenkomst en zoveel mogelijk zorg besteed aan het zo aangenaam mogelijk maken van het werken met de machine/het programma voor de gebruiker. Vele programma's

(vooral oudere, van voor 1980) beantwoorden niet aan de wens (of, tegenwoordig, de eis) zo gebruikersvriendelijk mogelijk te zijn. Dit is jammer, maar het is iets dat de informatica met zich meebrengt, immers, functionaliteit gaat voor aantrekkelijkheid. De gebruikersvriendelijkheid heeft in feite pas zijn intrede gedaan nadat computers meer intern geheugen kregen zodat er *plaats* voor was. De Organiser II heeft echter slechts 32K ROM!

5. Copiëren van gegevens : COPY

De Organiser II heeft ook een speciale, ingebouwde functie die het mogelijk maakt gegevens (records) van de ene datapak te copiëren naar de andere. Zo kunt u dus veiligheidscopieën maken van uw gegevens in het interne geheugen (pak A:), u kunt gegevens copiëren voor vrienden en kennissen, maar ook programma's (daar hebben we het later over).

Hoe gaat dat in zijn werk ? De eenvoud zelf : kies in het hoofdmenu de optie COPY (op de laatste regel).

```
TIME INFO ALARM
COPY RESET OFF
```

Het scherm toont vervolgens :

```
FROM
```

Nu moeten we de naam intoetsen van de *brondatapak* (dus waar de gegevens vandaan komen). Stel dat we gaan copiëren van A: naar B:. Tik in: A: (gevolgd door EXE).

```
FROM A:
TO
```

Tik in : B: (eveneens gevolgd door EXE).

De Organiser II zal nu beginnen te copiëren :

```
Copying . . .
```

Afhankelijk van de hoeveelheid gegevens die er op uw datapak staan, zal dit copiëerproces weinig of meer tijd in beslag nemen. Copiëren naar een RAMpak zal minder tijd in beslag nemen dan copiëren naar een datapak. Alle gegevens die op de *doelpak* stonden (in ons geval B:), zullen blijven staan. Als er niet genoeg plaats meer is op die pak, zal de Organiser II correct een fout melden :

```
PACK FULL
press space key
```

Door nu op SPACE te drukken (onderste rij toetsen, tweede van rechts), zal de Organiser II de optie COPY opnieuw oproepen, alsof we ze van het

hoofdmenu selecteerden.

Om tijdens het opvragen van bron- en doelpak (FROM, TO) de optie COPY te verlaten, drukken we eenvoudigweg op ON.

6. Leegmaken van het intern geheugen : RESET

De optie RESET dient om alle gegevens in het interne geheugen (pak A:) uit te wissen. Alle records, de tijd en datum, en ook de gegevens van de ingebouwde agenda en de alarmen (waarover later meer), zullen uit de Organiser II verdwijnen. Ik raad u dus aan voorzichtig te zijn en te allen tijde te weten waar u mee bezig bent.

Indien u de optie RESET wenst te gebruiken, kiest u deze vanuit het hoofdmenu. Voor de zekerheid presenteert de Organiser II het volgende scherm :

```
ALL DATA WILL BE
LOST - PRESS DEL
```

De vastbesloten gebruiker drukt nu op DEL (voor delete: wissen) en het geheugen wordt inderdaad uitgewist. Alle anderen drukken op ON.

De Organiser II keert dan spontaan terug naar het hoofdmenu.

7. ALARM

7.1 De 8 alarmen

De Organiser II heeft de mogelijkheid om, op door de gebruiker ingevoerde tijdstippen, een alarmsignaal te laten horen. Dit kan een éénmalig gebeuren zijn, of worden herhaald met regelmatige intervallen (bijvoorbeeld iedere dag om 19:45 uur, iedere donderdag om 21:00 uur, enz). Het interval van herhaling kan zijn ieder uur, iedere dag, iedere week. Men kan maximaal één week op van tevoren een alarmsignaal 'programmeren'.

Voor het gemak is deze functie niet beperkt tot één alarm, maar kan men met maar liefst 8 verschillende alarmen tegelijkertijd werken. Deze alarmen kunnen los van elkaar gebruikt worden en hoeven niet noodzakelijk in chronologische volgorde te worden gebruikt.

7.2 Het programmeren van een alarm

Kies de optie 'ALARM' in het hoofdmenu. U dient nu het volgende scherm te zien :

```
1) FREE
press EXE to set
```

Dit betekent dat alarm 1 (ze zijn genummerd 1 tot 8) nog niet geprogrammeerd is. We doen wat de Organiser II zegt : we drukken op EXE.

```
1) FRI 14.39
```

of iets dergelijks staat er nu op uw scherm : de weekdag, het uur in 24-urenformaat en de minuten in het uur. De cursor staat op de dag.

We gaan er voor zorgen dat het alarm over twee minuten afgaat. U zult opmerken dat het door de Organiser II afgebeelde uur correct is (als u tenminste het uur juist hebt ingevoerd bij de TIME-functie).

Stuur de cursor naar de minuten. Druk dan twee maal op OP. Het getal van de minuten zal dan met twee verhoogd worden. Druk vervolgens op EXE.

Op het hierboven gespecifieerde uur klinkt er vervolgens een alarmsignaal (in het bovenstaande geval dus om 14:41). Dit alarm blijft normaal een mi-

nuut doorbiepen, maar kan worden uitgeschakeld door op ON te drukken. Als we nu opnieuw de functie ALARM kiezen, is de programmatie verdwenen en staat er weer :

```
1) FREE
press EXE to set
```

Dit keer gaan we een repeterend alarm invoeren.

Druk op EXE en kies een bepaalde minuut waarop de Organiser II zijn alarmsignaal ten beste moet geven. Wanneer u bijvoorbeeld gekozen hebt: ieder uur om kwart over, (dus minuut = 15):

```
1) FRI 14:15
```

en de cursor staat nog steeds op de minuut, druk dan op MODE.

```
1) FRI 14:15
          R
```

Er verschijnt nu een *R* (van Repeat : herhaal) op de onderste regel. Dit betekent dat de Organiser II ieder uur op het eerste kwartier het alarm zal laten klinken. Druk nu een paar keer op LINKS of RECHTS en u zult merken dat de *R* met u meereist. Zet de *R* terug onder de 15 en druk op EXE. De Organiser II geeft nu ieder uur alarm op het eerste kwartier.

Op deze manier kunt u het alarm ieder uur, iedere dag of iedere week op een vast tijdstip laten klinken.

Om het tweede en volgende alarmen in te voeren, drukt u op NEER.

7.3 Het verwijderen van een alarm

Om een ingevoerde alarmfunctie weer te verwijderen, kiest u het te verwijderen alarm uit (sturen met OP of NEER) en drukt op DEL. Het alarm is dan verwijderd.

8. Geheugenverbruik : INFO

Het kan uiteraard steeds nuttig zijn, te weten hoeveel geheugen er al in gebruik is in de Organiser II. Daarvoor is de optie INFO bestemd.

```
CALC PROG ERASE
TIME INFO ALARM
```

INFO staat op de derde regel van het hoofdmenu. De optie kan uiteraard ook worden opgeroepen door op I te drukken.

U krijgt dan een display die op de eerste regel weergeeft hoeveel bytes (tekens) opslagruimte het interne geheugen heeft. Voor het model CM is dit 7171 bytes (8K - 1021 bytes werkgeheugen voor de ROM, de tijd- en alarmfuncties). Het oude model XP heeft 15363 bytes vrij, en de nieuwe XP-modellen hebben 23523 bytes vrij werkgeheugen.

De tweede regel, zult u zien, is een doordraaiende lijn. De Organiser II moet in feite een lange lijn laten zien, waar hij geen plaats voor heeft. Die lijn wordt dus 'doorgedraaid' (in het informaticajargon zegt men dat het scherm 'scrollt' : 'scroll' is een afkorting van 'screen roll' ofwel het rollen, doordraaien van het scherm). Een typische INFO kan zijn :

```
MEMORY 15363
DIARY 1% PACK A: 20% PACK B: 40% PACK C: 99% FREE 79%
```

(in dit geval betreft het een Organiser II, oud model XP, met twee datapaks waarvan C: vol is (de Organiser II geeft steeds 99% weer voor een volle datapak)).

DIARY 1% betekent dat de agenda, waarover we het in het volgende hoofdstukje gaan hebben, 1% van het interne geheugen opgebruikt (pak A:). Daarnaast staat; PACK A:20% en dat betekent dat de gegevens die we met SAVE in de computer hebben opgeslagen, 20% van pak A: in beslag nemen. Pak A: is dus 1% + 20% = 21% gevuld, en is nog 79% vrij (vandaar : FREE 79%).

Let er op, dat u in *datapaks* geen ruimte kunt vrijmaken door gegevens uit te wissen. Ze blijven immers hun fysieke ruimte innemen totdat de gehele datapak wordt uitgewist met een zgn. Psion Formatter. Deze maakt de gehele pak leeg d.m.v. ultraviolet licht. In *RAMpaks* (en ook in het interne geheugen) kunt u echter wel ruimte vrijmaken door gegevens uit te wissen.

Om vanuit INFO weer naar het hoofdmenu te komen, drukt u op ON.

9. De rekenmachine: CALC

De Organiser II heeft een uitstekende wetenschappelijke calculator aan boord. Deze kan worden gebruikt door de optie CALC te kiezen in het hoofdmenu.

```
FIND SAVE DIARY
CALC PROG ERASE
```

Een wetenschappelijke calculator is een calculator die is uitgerust met wiskundige functies die bijvoorbeeld in de trigonometrie gebruikt worden. De Organiser II is dus uitgerust om wiskunde aan te kunnen.

(Opmerking : in dit boek staan methoden vermeld waarmee men de kracht van de calculator aanzienlijk kan vergroten. Later zullen we zelf extra wiskundige functies bijvoegen).

9.1 Algemeen gebruik

Kies de optie CALC.

```
CALC:
```

Zoals u aan de cursor kunt zien – hij knippert niet meer – heeft de Organiser II zijn klavier automatisch op 'numeriek' gezet. U hoeft dus niet meer de SHIFT- toets te gebruiken om een getal in te typen.

We gaan nu 5+5 berekenen. Toets in :

```
CALC: 5+5
```

(de '+' kan worden verkregen door de toets X in te drukken, waarboven het plusteken staat).

Zoals we zien, is de werking enigszins anders dan bij andere rekenmachines. Men kan te allen tijde zijn berekening op het scherm zien, en d.m.v. LINKS en RECHTS zelfs eventuele fouten corrigeren.

Anders is ook, dat we niet = dienen te gebruiken om de oplossing te verkrijgen. Druk op EXE en de Organiser II toont de oplossing :

```
5+5
=10.000000000000
```


Hier zien we ook de hoge precisie van de Organiser II : maar liefst 12 cijfers achter de (vlottende) komma !

Druk 1 keer op ON. We gaan nu een andere berekening invoeren.

```
CALC: 5*5
```

5*5 ? Het sterretje betekent 'vermenigvuldigen' (× in de meeste rekenmachines). Druk weer op EXE, en op de onderste regel geeft de Organiser II de oplossing.

We kunnen ook met haakjes werken. Om een complexere berekening als

```
(5*5) - (18+3) +20
```

te kunnen laten oplossen, roepen we de haakjes van de Organiser II te hulp. Zij kunnen worden verkregen door op het klavier C en D in te drukken.

```
(5*5) - (18+3) +20
=24.000000000000
```

Zoals in Engelstalige gebieden gebruikelijk is – de Organiser II is immers een Engels produkt – wordt voor het decimale teken een punt gebruikt in plaats van een komma. Als wij dus decimale getallen willen invoeren – bijvoorbeeld 3,14 – dan zullen wij ook een punt moeten gebruiken. Van een komma begrijpt de Organiser II geen snars !

Net als de * voor vermenigvuldiging, gebruikt de Organiser II een 'speciaal' teken voor deling, nl. / (uitgesproken : 'slash'). 5/3 betekent dus : 5 gedeeld door 3. Deze tekens (*, /) zijn universeel in de informatica : zij worden toegepast in iedere programmeertaal, op ieder systeem. Voor machtsverheffing gebruikt men twee sterretjes : **. Om 5 tot de 3e macht te verheffen dient dus 5**3 te worden ingetoetst.
De precisie van de berekeningen kan worden aangepast. Standaard is deze 12 cijfers achter de komma, maar we kunnen er bijvoorbeeld 8 van maken (zoals bij de meeste rekenmachines). Tik dan :

```
CALC: FIX=8
```

(om de letters FIX te krijgen zullen we wel SHIFT moeten gebruiken, want de Organiser II verwacht zonder dat numerieke invoer).

9.2 Wetenschappelijke notering

Wetenschappelijke notering is een manier van getallen neerschrijven, zoals dat in de wetenschap gebeurt. Ook de calculator van de Organiser II, die

we daarnet trouwens 'wetenschappelijk' hebben genoemd, gebruikt deze vorm van notering.

Bereken : 2**64 (2 tot de 64e macht). Dit is uiteraard een enorm getal.

```
2**64
=1.844674407E+19
```

Op het eerste gezicht lijkt de oplossing iets meer dan 1.8 te zijn. Dat kan natuurlijk niet. Maar aan de rechterzijde van het getal zien we de notering

```
E+19
```

hetgeen betekent dat we het decimaalteken 19 plaatsen naar rechts moeten opschuiven. De oplossing zal dus ongeveer 184467440000000000000 bedragen. Had er gestaan:

```
E-19
```

dan zou dat hebben betekend dat het decimaalteken 19 plaatsen naar links moet worden verschoven, dus 0.000000000000000000018446744.

We kunnen deze notering ook zelf gebruiken in onze berekeningen. Reken bijvoorbeeld eens uit :

```
CALC: 1E3
```

(hetzelfde als 1E+3). De oplossing is 1000 : de 'komma' is drie plaatsen naar rechts verschoven.

```
CALC: 1E-3
```

Dit geeft 0.001.

```
CALC: 1/33E+3
```

Dit is hetzelfde als :

```
CALC: 1/33000
```

Voor berekeningen met hoge getallen wordt de wetenschappelijke notering zeer vaak gebruikt, omdat deze kort is.

9.3 De werkgeheugens

Een uitgebreide rekenmachine beschikt uiteraard over een geheugen voor tijdelijke opslag van getallen en uitkomsten. De rekenmachine van de Organiser II heeft er maar liefst 10 ! Deze 10 geheugens zijn genummerd 0-9 en kunnen worden geactiveerd door op de toets MODE te drukken.

```
M: PRESS 0-9
```

De Organiser II vraagt nu welk van de tien geheugens we willen gaan gebruiken. We kiezen bijvoorbeeld 0.

```
M0: +, -, EXE, DEL
```

Nu stelt de Organiser II vier keuzes voor :

\+ : door op + te drukken wordt de huidige uitkomst (zie onderste regel) bij geheugen M0 *opgeteld*. Als M0 bijvoorbeeld 18 bevatte, en de huidige uitkomst op het scherm is 24, dan is M0 nu 18 + 24 = 42.

\- : door op - te drukken wordt de huidige uitkomst van M0 *afgetrokken*.

EXE : wanneer we op EXE drukken, wordt de inhoud van het geheugen (in dit geval M0) *vervangen* door de huidige uitkomst. De oude inhoud wordt dan dus vergeten.

DEL : de inhoud van het gekozen geheugen wordt uitgewist (= krijgt de waarde 0).

De tien geheugens, M0 tot M9, kunnen vrijelijk in expressies (= berekeningen) worden gebruikt. De berekening

```
24+(M0/9)
```

bijvoorbeeld is dus 100 % geldig.

9.4 OPL-functies

Het is mogelijk om met de rekenmachine wiskundige functies aan te roepen, die gebruikt worden in de programmeertaal OPL, die in de Organiser II is ingebouwd. Dit zijn functies als SIN (sinus), COS (cosinus), INT (integer : geheel deel), ROUND (afronding), enzovoort.

Probeer :

```
CALC: SIN(1.2*PI)
=-0.587785252294
```

Het werkt. De Organiser II kent inderdaad de SIN- functie, en het getal PI. Het is ook mogelijk met de Organiser II de *waarheid* van iets te laten berekenen. Bijvoorbeeld : is de sinus van 9 gelijk aan 1 ?

```
CALC: (SIN(9)=1)
=0.000000000000
```

Het antwoord is nul; het is dus *niet waar*. Bereken nu:

```
CALC: (SIN(9)=0.412118485241)
=-1.000000000000
```

Dit keer is het antwoord -1 : een waarde die dus *niet nul* is en bijgevolg *waar* betekent. Men kan de waarheid van bepaalde stellingen dus testen met enkele zgn. '*relationele operatoren*' : = (is gelijk aan), < (is kleiner dan), > (is groter dan), <= (kleiner of gelijk aan), >= (groter of gelijk aan), <> (niet gelijk aan). Over dit soort 'algebra' (genaamd Boole-algebra of logica) zullen we het later nog hebben.

De OPL- functies, die we met de calculator kunnen gebruiken zijn de volgende :

ABS : ABS(expressie) geeft de absolute waarde van die expressie, dus de oplossing zonder teken. ABS(-5) bijvoorbeeld geeft 5.

ATAN : ATAN(expressie) geeft de boogtangens (arctangent) van de expressie in radialen.

COS : COS(expressie) geeft de cosinus van de expressie, eveneens in radialen.

DAY : DAY geeft de dag van de maand (tussen 1 en 31).

DEG : DEG(expressie) zet de waarde van de expressie, die in radialen is gegeven, om in graden. Bijvoorbeeld; DEG(2.5) geeft weer hoeveel graden 2,5 radialen zijn (oplossing is 1.432394488E+02 of 143.2394488).

EXP : EXP(expressie) geeft de waarde van de rekenkundige constante e (= 2.718281828460...) tot de macht van de expressie verheven. Om het getal *e* te krijgen volstaat het dus EXP(1) te berekenen.

HOUR : HOUR geeft het uur van de dag (tussen 0 en 23).

IABS : IABS(expressie) geeft de absolute waarde van een integer getal (een geheel getal tussen -32768 en +32767).

INT : INT(expressie) geeft het gehele deel van de waarde van de expressie. INT(PI) geeft dus 3. INT staat trouwens voor integer en kan dus enkel integere oplossingen weergeven (zie functie IABS).

INTF : INTF(expressie) is hetzelfde als INT behalve dat de oplossing een vlottende-komma getal blijft. INTF dient te worden gebruikt als de uitkomst niet binnen de integere getallen valt (zie wederom : IABS).

LN : LN(expressie) geeft de natuurlijke logaritme van de expressie.

LOG : LOG(expressie) geeft de logaritme met basis 10 (Briggse logaritme) van de expressie. Opmerking : later zullen we een manier zien om logaritmen te berekenen met een veranderlijke basis).

MINUTE : MINUTE geeft de minuut van het uur (tussen 0 en 59).

MONTH : MONTH geeft de huidige maand (tussen 1 en 12).

PI : PI geeft de natuurlijke constante pi (= ca. 3.14159265359)

RAD : RAD(expressie) zet de graden in de expressie om in radialen. Zie ook : DEG.

RND : RND geeft een willekeurig getal tussen 0 (inclusief) en 1 (exclusief). RND kan dus wel 0 opleveren maar geen 1 (max 0.999999999999). RND staat voor het Engelse woord 'random' : willekeurig).

SECOND : SECOND geeft de seconde van de minuut (tussen 0 en 59).

SIN : SIN(expressie) geeft de sinus van de expressie in radialen.

SQR : SQR(expressie) geeft de vierkantswortel van de expressie. SQR staat voor het Engelse 'square root' of vierkantswortel. OPGELET programmeurs PASCAL, ALGOL en aanverwante talen : de Organiser II gebruikt voor de vierkantswortel SQR, niet SQRT ! Om een kwadraat te krijgen dient (expressie)**2 of (expressie)*(expressie) te worden gebruikt !

TAN : TAN(expressie) geeft de tangens van de expressie.

YEAR : YEAR geeft het huidige jaar (tussen 1900 en 1999).

9.5 Uitbreidingen op OPL-functies

Later, wanneer we de programmeertaal OPL besproken hebben en enkele kleine programma's hebben geschreven, zullen we zien hoe de taal OPL en dus ook de calculator kunnen worden verrijkt met extra functies, zoals bijvoorbeeld cotangens, afgeleide van een sinus, secans, cosecans, e.d.

10. De agenda : DIARY

10.1 Inleiding

De Organiser II bezit een interne agenda, waarin we onze afspraken kunnen noteren, met optioneel alarm tot 59 minuten voor de afspraak. Zo hebben we tegelijk met een adressenboekje en rekenmachine ook steeds een agenda bij de hand !

De agenda heeft voor iedere dag – t.e.m. 31 december 1999 – één bladzijde met 48 regels (één per half uur). Op iedere regel kunnen 64 tekens worden geschreven, en iedere regel kan een alarm bevatten.

Iedere lijn van de agenda wordt afzonderlijk beschreven. Het is ook mogelijk iedere lijn afzonderlijk te bekijken – ook lijnen die nog onbeschreven zijn –, een lijst op te vragen van alle afspraken, of de agenda open te slaan op een bepaalde bladzijde (lees : datum), bijvoorbeeld "over 15 dagen".

Alle gegevens van de agenda worden opgeslagen in het interne geheugen. Geen paniek echter, de ontwerpers bij Psion hebben ervoor gezorgd dat de consumptie van geheugen tot een minimum beperkt blijft. Maar toch betekent dit, dat de agenda kwetsbaar is : wanneer de batterij in de Organiser II leeg of verwijderd wordt, is ook de agenda verloren ... Ook voor dit probleem heeft men bij Psion een oplossing bedacht : de gehele agenda kan naar een pak worden weggeschreven om later te worden teruggelezen.

Zoals bij een gewone agenda met papier, blijven de reeds verlopen afspraken – bijvoorbeeld een afspraak van gisteren of vorige week – nog genoteerd. Men kan desgewenst dus ook terugkijken naar vorige afspraken. Heeft men daaraan geen behoefte, dan kunnen die oude afspraken uit de agenda worden verwijderd (zonder scheuren), zodat geheugenruimte vrijkomt.

10.2 Het gebruik

Laten we het allemaal eens van dichterbij bekijken. Kies de functie DIARY op het hoofdmenu. De Organiser II geeft nu de "eerstkomende regel" op zijn scherm. De datum en het uur van de regel die met het halfuur in kwestie overeenkomt, wordt op de bovenste regel getoond, bijvoorbeeld :

```
AUG: 25: WED: 19. 00
```

(25 augustus, woensdag, 19.00 uur). Deze 'huidige' regel is echter wat te nabij om een afspraak te noteren. We gaan daarom schrijven op de volgende regel (in ons voorbeeld die van 19.30 uur).

Om naar de volgende regel te gaan, drukt u op NEER. Het uur is nu veranderd. We gaan invoeren :

```
Bellen naar kantoor
```

Tik deze regel in en druk op EXE. De Organiser II zal na het typen van de eerste letter het woord EDIT op het scherm brengen; tik rustig door, we komen hier later nog op terug. Als we op EXE hebben gedrukt, vraagt de Organiser II :

```
ALARM (Y/N)
```

Ja, we wensen wel degelijk een alarm, dus we drukken op Y.

Dan wil de Organiser II weten hoeveel minuten van tevoren (in ons voorbeeld dus vóór 19.30 uur) hij zijn klokken moet luiden.

```
MINUTES: 15
```

Door nu op OP of NEER te drukken, kunnen we de waarschuwingstijd respectievelijk verhogen of verlagen. We kiezen 10 en drukken op EXE.

```
AUG: 25: WED: 19.30
(A) Bellen naar kantoor
```

(De (A) staat voor 'alarm').

Ook als we de Organiser II nu uitschakelen, zal het alarm op het juiste uur klinken, net als bij de ALARM- optie van het hoofdmenu.

10.3 De agendafuncties

Eerder al hebben we even gesproken over de speciale mogelijkheden waarmee de agenda is uitgerust. We gaan deze speciale opties of functies wat naderbij bekijken. Deze functies staan in het zgn. 'DIARY-menu', dat ter beschikking komt door eerst DIARY te selecteren vanuit het hoofdmenu en vervolgens op MODE te drukken. Men werkt met dit menu op precies dezelfde wijze als met het hoofdmenu.

De opties zijn PAGE, LIST, FIND, GOTO, SAVE, TIDY, RESTORE, DIR en ERASE. Om van een gekozen optie terug te keren naar het

DIARY-menu, drukt men één keer de MODE-toets in.

PAGE ('Bladzijde') : deze functie maakt het mogelijk iedere regel van de agenda afzonderlijk te bekijken. We 'sturen' door de agenda met de toetsen OP en NEER (vorige, volgende regel) en LINKS en RECHTS (vorige, volgende bladzijde). We voeren vanuit PAGE ook afspraken in. Dit wordt eenvoudigweg gedaan door ze in te tikken (zoals we net hebben gedaan). De Organiser II toont dan op de onderste lijn :

```
EDIT:
```

gevolgd door wat dan wordt ingetikt.

Om een afspraak te verwijderen, volstaat het op de DEL- toets te drukken. In dat geval informeert de Organiser II :

DELETE (Y/N)

We drukken op Y om te bevestigen, op N om te ontkennen.
PAGE is waarschijnlijk de meestgebruikte functie van de agenda.

LIST ('Lijst') : toont één voor één, de regels die beschreven zijn (uiteraard met bijhorende datum en uur). Door EXE in te drukken wordt steeds de volgende beschreven regel gekozen.

FIND ('Vind') : voor het opzoeken van een bepaalde afspraak, op bepaalde meerdere afspraken. Om bijvoorbeeld te weten wanneer de afspraak met ene mijnheer Zahler al weer was, tikken we de naam 'Zahler' in. De agenda gaat dan kijken naar iedere beschreven regel en brengt op het scherm die regels waarin de naam Zahler voorkomt. Net als bij LIST dient EXE om naar de volgende afspraak te gaan.

GOTO ('Ga naar') : ga naar een bepaalde datum. Die datum kan worden ingesteld door gebruik te maken van de cursortoetsen en bevestigd door EXE. Dan gaat de agenda in 'PAGE'-modus.

SAVE ('Bewaar') : opslaan van een gehele agenda naar pak. Deze agenda dient een (unieke) naam te krijgen. Ikzelf gebruik voor mijn agenda altijd de naam *D*, wat kort en dus gemakkelijk in het gebruik is. Ik gebruik deze functie regelmatig, omdat ik dan in geval van nood (een lege batterij) al mijn afspraken en nodige herinneringen steeds kan recupereren.
Het is vanzelfsprekend weinig zinvol wanneer we onze gegevens naar pak A: zouden schrijven, daar blijven ze immers kwetsbaar. Ikzelf zet ze altijd op een RAMpak, omdat ik ze dan later weer door recentere agenda's kan vervangen zonder dat het gebruikte geheugen zich opstapelt, als bij de datapaks.

TIDY ('Ruim op') : Deze functie verwijdert alle verouderde afspraken (afspraken die dus reeds verlopen zijn), en maakt zo geheugenruimte vrij. Om veiligheidsredenen vraagt de Organiser II om een bevestiging van de opdracht :

DELETE UPTO (Y/N)

(Tot hier wissen?) Wederom dienen we op Y of N te drukken om respectievelijk te bevestigen of te ontkennen. Wanneer ontkend wordt, blijft de DIARY onveranderd.

RESTORE ('Herstel') : opnieuw inlezen van een eerder bewaarde agenda.

Die agendagegevens zijn eerder met de SAVE-optie naar pak geschreven onder een bepaalde naam. Om ze weer in te lezen volstaat het onder deze optie die gebruikte naam weer in te tikken.

DIR (afk. voor DIRECTORY, 'Inhoudsopgave') : geeft een lijst van alle namen van naar pak geschreven agenda's.

ERASE ('Wis') : wist eerder bewaarde agendagegevens van pak. Men moet de naam van de te verwijderen agenda opgeven.

Merk op, dat dit menu geen 'OFF'-optie heeft. Om de Organiser II uit te schakelen moet men altijd naar het hoofdmenu (druk twee keer op ON) of wachten tot de Organiser II zichzelf uitschakelt.

10.4 Meer over de agenda-alarmen

Wanneer een alarm klinkt presenteert de Organiser II op de bovenste regel de datum en het uur, en op de onderste regel de ingevoerde tekst. Als die tekst langer is dan 16 tekens (en dus niet op het scherm kan), wordt deze op het scherm horizontaal doorgerold. Om het alarm te beëindigen drukken we één keer op ON.

Ook wanneer men met zijn Organiser II aan het werk is, bijvoorbeeld met FIND of SAVE in het hoofdmenu, onderbreekt de agenda dit werk. Wanneer vervolgens het alarm beëindigd is (zij het met ON of door een minuut te wachten tot het vanzelf stopt) keert de Organiser II terug naar de toestand in welke hij zich bevond vóór het alarm.

Wanneer twee alarmen overlappen, wat best mogelijk is, zal de Organiser II ze in chronologische volgorde behandelen.

Wanneer een agenda-alarm overlapt met een ALARM-alarm (!) zal de computer de voorkeur geven aan het agenda-alarm en vervolgens overspringen naar de ALARM-optie van het hoofdmenu.

10.5 Een alarm verwijderen

Om een alarm in de agenda te verwijderen, gaan we naar de betreffende regel (met PAGE) en drukken op EXE. We krijgen dan de mogelijkheid de tekst aan te passen (EDIT). Nadat we dit eventueel gedaan hebben, drukken we op EXE en de Organiser II vuurt opnieuw de 'ALARM (Y/N)'-vraag op ons af. Dit keer drukken we op N : we willen dus géén alarm.

11. Het menu aanpassen

Het hoofdmenu, dat we telkens als we de Organiser II aanzetten zien en waar we constant mee werken, kan naar onze behoeften worden aangepast. Zodoende kunnen we de Organiser II voor 100 % benutten.

We kunnen opties verwijderen of van plaats veranderen (verwijderde opties kunnen we later weer inbrengen). De enige optie die niet kan worden verwijderd of van plaats veranderd is OFF. OFF is steeds het laatste item van het hoofdmenu.

We gaan, bij wijze van voorbeeld, de optie RESET – een gevaarlijke optie – uit het menu verwijderen. We sturen de cursor naar deze optie (zonder op R te drukken), en drukken de DEL-toets in. Voor de zekerheid informeert de Organiser II of we RESET wel degelijk willen verwijderen :

```
RESET
DELETE (Y/N)
```

(Dit type vraag noemt men trouwens een *validatievraag*).
We drukken op Y. RESET is nu uit het hoofdmenu verwijderd en kan dus niet meer worden gekozen (en dus ook niet per ongeluk).
We gaan vervolgens de functie terugroepen. We drukken op MODE. Nu wil de Organiser II weten welke optie we willen invoeren.

```
INSERT ITEM:
```

We tikken nu het woord R E S E T in en drukken op EXE om te beëindigen. De optie is nu terug in het hoofdmenu, op de plaats waar de cursor stond. Door de cursor te verplaatsen alvorens een optie opnieuw in te voeren, kunnen we iedere optie in het menu verplaatsen.

Het is echter níet mogelijk een bepaalde optie twee keer in het menu te laten voorkomen. Een tweede FIND bijvoorbeeld wordt resoluut geweigerd.

Zoals we later zullen zien bij het programmeren, kunnen we het menu verrijken met zelfgeprogrammeerde functies! Dit maakt de Organiser II tot een nóg flexibeler werkinstrument.

Deel III

WERKEN MET LZ

1. Het hoofdmenu

1.1 Inleiding

Het is aan de Organiser II model LZ niet meteen te zien, dat hij zo verschilt van de andere modellen. Dat verschil wordt pas duidelijk bij het aanzetten (door op ON te drukken, linksboven). Dan verschijnt namelijk het *hoofdmenu.*

Alle Organisers werken met menu's. Het bijzondere aan de LZ is dat hij niet met twee regels van 16 tekens werkt, maar met *vier* regels van 20 tekens. Zijn mogelijkheden zijn daarenboven gevoelig uitgebreid.

We drukken op ON. Omdat het de eerste keer is dat we de LZ aanzetten, verschijnt er even een Copyright-boodschap, waarna we gevraagd worden onze *taal* te kiezen (tussen Engels, Frans en Duits).

```
   SELECT LANGUAGE
──────────────────────
English   Français
Deutsch
```

Dit is het *taalmenu.* We kiezen nu de taal (in dit boek : Engels) door op EXE te drukken (rechts onderaan het klavier). Om Frans te kiezen drukken we op F, wie liever met een Duitstalige LZ werkt drukt op D. De taal kan later nog worden gewijzigd indien gewenst.

We zien nu het volgende :

```
■                    0.02a
Find    Save    Diary
Calc    Time    Notes
World   Alarm   Month
```

Dit is een gedeelte van het hoofdmenu.

Op de bovenste regel zien we links een ikoontje (klein Organisertje dat weergeeft dat we met het hoofdmenu bezig zijn, hierboven weergegeven als ■) en rechts de juiste tijd. Als de klok niet juist staat, geen probleem : dat lossen we later wel op. Op de overige drie regels staan een aantal *opties*, zoals bijvoorbeeld *Find, Save* en *Diary*. Telkens wanneer we met de LZ iets willen doen (iets opschrijven of opzoeken, bijvoorbeeld), zullen we een optie moeten *kiezen*.

Over de letter 'F' van Find zien we een blokje aan en uit knipperen : dit blokje noemen we de *cursor* — we zagen het daarnet bij het taalmenu ook al. We zeggen dat de cursor de optie Find aanduidt. Door de cursor te verplaatsen kunnen we hem ook alle andere opties laten aanduiden. Laten we het eens proberen.

Eerst bekijken we de bovenste rij toetsen op het toetsenbord. De eerste toets, ON, hebben we al gebruikt om de Organiser II aan te zetten. De tweede, MODE, zullen we later nog gebruiken. Zeer interessant zijn de volgende vier toetsen, waar de pijltjes op staan. In dit boek zullen we naar deze toetsen verwijzen met OP, NEER, LINKS en RECHTS. De aanduidingen CAP en NUM zullen we later nog gebruiken.

Als we onze LZ een tijdje ongemoeid laten, zullen we merken dat hij zich na enkele minuten automatisch uitschakelt. Dit is een handigheidje dat Psion heeft ingebouwd om het vroeger meestal dramatische effect van nalatigheid van de gebruiker tot een minimum te reduceren. Het spaart met andere woorden onze batterij. Als we weer op ON drukken, zijn we terug waar we laatst gebleven waren.

Druk nu voorzichtig op RECHTS. De cursor zal nu op de S van Save staan. Als u een foutje hebt gemaakt, druk dan op ON en probeer het nog eens.

Druk op NEER. De cursor staat op Calc (en niet op Time!).

Druk nogmaals op NEER. Nu staat er op het scherm :

```
■                    0.02a
Calc   Time   Notes
World  Alarm  Month
Prog   Xfiles Utils
```

Er staan dus drie nieuwe opties op het scherm. De cursor duidt World aan.

Druk weer op NEER. Het scherm geeft weer :

```
■                    0.02a
World  Time   Notes
Prog   Xfiles Utils
Off
```


Nogmaals NEER geeft :

```
■                    0.02a
Find   Save   Diary
Calc   Time   Notes
World  Alarm  Month
```

We hebben nu het hele menu doorlopen en zijn opnieuw bij de eerste regel beland.

1.2 Selecties

Opties kunnen worden geselecteerd door :

– met de pijltjestoetsen de cursor naar de desbetreffende optie te sturen en op EXE te drukken (staat voor execute; voer uit);
– de eerste letter van de optie in te drukken (bijvoorbeeld W voor World).

De Organiser II kan dus worden uitgeschakeld door op O te drukken (de eerste letter van de laatste optie : Off). Hij is dan uitgeschakeld tot weer op ON wordt gedrukt. We zijn dan weer terug in het oude, vertrouwde hoofdmenu

2. Find en Save

2.1 Inleiding

De twee opties die in de titel worden opgesomd, zijn misschien wel de meestgebruikte van de Organiser II.

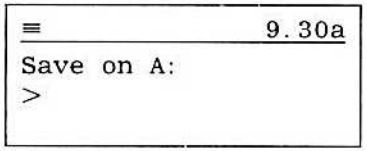
Find dient om eerder opgeslagen gegevens (bijvoorbeeld adressen en telefoonnummers) weer op te zoeken. Daar Find meer wordt gebruikt dan Save, staat deze optie vooraan.

Save dient om gegevens in het geheugen of op een data- of RAMpak op te slaan. De Find/Save-combinatie is dus vergelijkbaar met een kaartenbak: Save om de kaart te schrijven en bij te houden, en Find om ze weer op te zoeken.

2.2 Save

We gaan wat gegevens opschrijven met Save. Dat gaat als volgt :

Druk op S (voor Save). Het ikoontje linksboven wordt verondersteld een 'gegevensbank' voor te stellen...



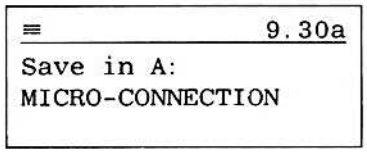
Op de eerste regel staat nu links het ikoontje dat aangeeft dat er met Save gewerkt wordt. Als dat nog niet duidelijk genoeg is, geeft de tweede regel aan dat er naar pak A: wordt geSaved. Pak A: is namelijk het interne geheugen. De twee paks die ingeschoven kunnen worden heten B: en C:. Let op de dubbele punt, die steeds aanduidt dat het om een pak gaat. Het is mogelijk (met de MODE-toets), de pakletter te veranderen, indien er een data- of RAMpak in de Organiser II zit. De gegevens worden dan naar die andere pak gestuurd i.p.v. naar A:. Op de derde regel dan vinden we een 'groter-dan'-teken (>). Dit teken noemt men de prompt. Achter deze prompt staat de cursor. De cursor geeft aan waar de letters die we gaan intikken op het scherm zullen verschijnen.

Als we nu per vergissing Save hadden geselecteerd, zouden we nu op ON drukken. We zouden dan terug bij het hoofdmenu belanden, alsof er niets was gebeurd.

Tik de eerste regel in :

MICRO-CONNECTION

(het streepje wordt verkregen door de toets SHIFT (onderste rij) ingedrukt te houden en op R te drukken). Bij iedere letter schuift de cursor een plaats op.



Druk vervolgens op NEER en tik de tweede regel in :

St. Katelijnevest 16-18 2000 Antwerpen

Kleine letters worden verkregen door op SHIFT CAP te drukken (SHIFT ingedrukt houden en op CAP drukken). Het zgn. 'hoofdletterslot' wordt dan uitgeschakeld. Om een hoofdletter te verkrijgen kunnen we ofwel het slot terug inzetten (wederom SHIFT CAP), of gewoon SHIFT samen met de benodigde letter indrukken, net als bij een schrijfmachine.

Wie een foutje tikt moet niet wanhopen. Daarvoor is de toets DEL voorzien (onderste rij toetsen). DEL staat voor delete (verwijderen).

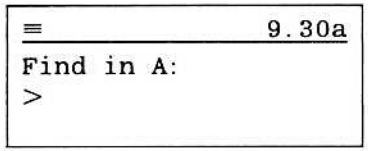
Nu alles (hopelijk) in orde is, druk op NEER. Tik nu de laatste twee regels in :

03/231 01 03
Exclusief invoerder Psion voor Benelux

Druk nu niet op NEER, maar op EXE. Deze toets beveelt de Organiser II de net ingetikte 'kaart' in zijn geheugen op te nemen. We keren dan onmiddellijk terug naar het hoofdmenu. De cursor duidt terug Find aan.

2.3 Find

We gaan die 'kaart' nu terug opzoeken, en wel met Find.



Find werkt bijna net als Save. Op de eerste regel staat nog steeds links een ikoontje. Rechts wordt de tijd aangegeven. Op de tweede regel wordt aangeduid dat we gaan zoeken met Find, en waar we gaan zoeken (pak A:, het interne geheugen). Op de derde regel staat de prompt, gevolgd door een ongeduldig knipperende cursor.

Hoe gaan we de net ingetikte gegevens weer opzoeken? Daarvoor volstaat het *iets* in te tikken, wat ook op de kaart voorkomt. We weten bijvoorbeeld dat we het adres van de firma Micro-Connection willen opzoeken. We tikken dan bijvoorbeeld in :

```
≡                    9.30a
Find in A:
>MICRO
```

Of, we weten enkel dat deze firma in Antwerpen gevestigd is. In dat geval tikken we :

```
≡                    9.30a
Find in A:
>Antw
```

of zelfs :

```
≡                    9.30a
Find in A:
>ANTWERPEN
```

Het maakt niet uit of we hoofd- of kleine letters gebruiken, de Organiser II vindt het allemaal.

We drukken nu op EXE. Dit geeft de opdracht in pak A: te gaan zoeken. We hadden ook SHIFT EXE kunnen indrukken; in dat geval zou de LZ *achteraan* in zijn geheugen gaan zoeken. Waarom is dat nuttig? Eenvoudig omdat we op die manier recent ingetikte gegevens sneller kunnen terugvinden, als we al een grote hoeveelheid 'kaarten' in ons geheugen hebben zitten.

Onmiddellijk is de 'kaart' (in het jargon : record) teruggevonden;

```
MICRO-CONNECTION
St. Katelijnevest 16
03/231 01 03
Exclusief invoerder
```

De cursor staat op de eerste regel, onder de M.

De tweede regel is niet helemaal zichtbaar, hij is immers langer dan 20 tekens. Druk éénmaal op NEER en de regel zal horizontaal op de tweede schermregel doordraaien (in het jargon zegt men : de regel scrollt naar links. Scroll is afkorting voor screen roll; het doorrollen van het scherm). Druk op LINKS en het scrollen zal ophouden. Druk nogmaals op LINKS en de regel zal in de andere richting doordraaien.

Nu de record gevonden is, zijn er de volgende mogelijkheden :

2.3.1 Gevonden!

We hebben de record gevonden die we nodig hebben. In dat geval drukken we op ON om terug naar het hoofdmenu te gaan.

2.3.2 Niet gevonden...

Het is nog niet de juiste record. We hebben bijvoorbeeld ANTWERPEN ingegeven, en vonden het adres en het telefoonnummer van een andere firma in de Scheldestad. In dat geval drukken we op EXE, wat de Organiser II opdraagt naar de volgende record te zoeken waarin ANTWERPEN voorkomt.

Als een dergelijke record niet meer voorkomt (zoals in ons geval, we hebben immers nog maar één record ingegeven), krijgen we het volgende bericht :

```
≡                    9.30a
Find in 2A:
>antw
   NO MORE ENTRIES
```

NO MORE ENTRIES : de specificatie *antw* komt niet meer voor. Druk op ON en de LZ keert terug naar het hoofdmenu.

Als een dergelijke record wel voorkomt, kunnen we hem terug doorlopen met de pijltjestoetsen, zoals hierboven in punt 2.3 beschreven is.

2.3.3 Gevonden?

We weten niet zeker of de gevonden record wel diegene is waarnaar we zoeken. Dat kan gebeuren ingeval twee firmanamen bijna identiek zijn, bijvoorbeeld. We kunnen dan, zoals in 2.3.2, verder zoeken met EXE.

Indien de volgende record die gevonden wordt toch niet de goede is, en we de vorige record terug willen opzoeken, moeten we gewoon op SHIFT EXE drukken. De vorige record wordt dan terug opgezocht. Eventueel kunnen we ook nog op MODE drukken, om op een ander pak (B: of C:) te gaan zoeken.

2.4 Find met wildcards

2.4.1 Inleiding

Wildcards? Voor wie met besturingssystemen als MS-DOS of CP/M werkt of gewerkt heeft, zullen wildcards niet onbekend zijn. Wildcards bieden de oplossing voor menig hoofdbreken.

Ze zijn, kort gezegd, *kortere wegen.* Met een wildcard kunnen verschillende zoekspecificaties worden opgegeven, of kan een lange specificatie korter worden gemaakt.

Hiervoor worden twee speciale tekens gebruikt : de *ster* (*) en het *plusteken* (+).

2.4.2 *

Om een dokter in Amsterdam op te zoeken, zouden we bijvoorbeeld opgeven :

```
≡                    9.30a
Find in A:
>Dr*Amsterdam
```

Dit duidt erop dat we zoeken naar een record die zowel de specificatie 'Dr' als de specificatie 'Amsterdam' bevat. We zoeken dus met twee parameters tegelijk! Het kan ook met drie :


```
≡                    9.30a
Find in A:
>Dr*Ka*Amsterdam
```

Op die manier zoeken we naar een Amsterdamse dokter die Karel heet, of in de Kalverstraat woont bijvoorbeeld.

2.4.3 +

Het plusteken is een vervangingsteken (in MS-DOS en CP/M aangeduid als vraagteken). '+' betekent dus : *gelijk welk teken.*

Een voorbeeld : We gaan iemand opzoeken. Maar heet hij nu Van der Graaf of Van der Graef? Met een wildcard is het snel opgelost :

```
≡                    9.30a
Find in :
>Van der Gra+f
```

De beide wildcards, * en +, kunnen vrij door elkaar gebruikt worden.

2.5 Gegevens wissen met Find

Het gebeurt wel eens, dat we bepaalde records niet meer nodig hebben. Het zou dan al te gek zijn om die gegevens in het geheugen te bewaren. Dus is het ook mogelijk deze gegevens te wissen.

Restaurant 'De Gulden LZ' sluit na één succesvol jaar haar deuren. Het heeft bijgevolg geen nut het adres van dit vijfsterrenrestaurant nog te bewaren. We zoeken het op met Find.

```
≡                    9.30a
Find in A:
>gulden lz
```

Het record is gevonden en we drukken op DEL om het te verwijderen. Voorzichtig als de Organiser II is, vraagt hij om bevestiging :

```
Restaurant De Gulden
Suikerrui 20
_____________________
Delete? Y/N/All
```

(Wilt u dit record inderdaad wissen?)

Er zijn dan drie mogelijkheden : wissen (Y), niet wissen (N of ON), alle records met de opgegeven specificatie wissen (A). Dit laatste is echter een vrij gevaarlijke aangelegenheid. Er kunnen immers records worden gewist zonder dat dat eigenlijk de bedoeling is. Daarom vraagt de LZ in een dergelijk geval om bevestiging :

```
Delete in A:
>gulden lz
_____________________
Confirm? Y/N
```

(Bevestig? Ja of Nee).

Hier zijn weer twee mogelijkheden : inderdaad wissen (Y) of niet wissen (N of ON). Indien Y wordt ingedrukt worden alle records met de opgegeven specificatie onherroepelijk gewist.

2.6 Gegevens aanpassen met Find

Gegevens hebben de eigenschap dat ze kunnen veranderen. Een vriend verhuist bijvoorbeeld, een bepaalde prijsopgave is verouderd, ... Sommige gegevens kunnen zelfs bijna dagelijks veranderen. Het is daarom mogelijk gegevens in de Organiser II aan te passen, zonder dat het gehele record opnieuw moet worden ingetikt.

De heer Van der Graaf is zonet verhuisd. We zoeken zijn oude adres op :

```
Van der Graaf Johan
Somersstraat 18
2018 Antwerpen
03/231 29 00
```

Om zijn adres en telefoonnummer te kunnen wijzigen drukken we op MODE.


```
≡                    9.30a
Save in A:
>Van der Graaf Johan
Katwilgweg 29
```

De wijzigingen kunnen worden aangebracht waar nodig, waarna op EXE wordt gedrukt. Het oorspronkelijke record is niet gewist : er zijn nu dus twee records met gegevens over Van der Graaf Johan (dit is een veiligheidsmaatregel).

3. Het schrijfblok : Notes

3.1 Inleiding

Notes is een optie die niet op de vroegere modellen (CM en XP) te vinden is. Het is, zoals de naam suggereert, een schrijfblok. Maar wat is dan het verschil tussen Notes en de Find/Save-combinatie?

Dit schrijfblok kan worden ingedeeld in een aantal onderwerpen naar keuze van de gebruiker. Het is dus in feite een *aantal* schrijfblokken. We kunnen zoveel schrijfblokken maken als we willen.

Ieder schrijfblok :

- is rechtstreeks toegankelijk vanuit het hoofdmenu (zie later);
- mag zo lang zijn als het geheugen toelaat : elke regel mag echter niet meer dan 254 tekens bevatten;
- mag op pak A:, B: of C: staan;
- kan beschermd worden met een wachtwoord.

3.2 Aanpassen van het hoofdmenu

Zoals gezegd, is ieder schrijfblok rechtstreeks toegankelijk vanuit het hoofdmenu. Als we een schrijfblok "Tedoen" opmaken, kunnen we dus een optie "Tedoen" in het hoofdmenu bijvoegen. Wanneer we deze optie dan selecteren, komen we automatisch in het betreffende schrijfblok terecht.

Dit houdt bijgevolg de noodzaak in het hoofdmenu te kunnen *aanpassen*. De optie "Tedoen" bestaat immers nog niet! Dat gaat als volgt in zijn werk: ga naar het hoofdmenu en druk op MODE.

```
INSERT MENU ITEM

Name: TEDOEN
```

(Met ON kan terug naar het hoofdmenu worden gegaan indien wenselijk).

Dit menu-item (optie) kan uiteraard een bestaande optie zijn, bijvoorbeeld Find of Save. Dat mag vanzelfsprekend niet, er mogen immers geen twee opties met dezelfde naam voorkomen!

Het volgende menu komt op het scherm :

```
INSERT MENU ITEM
    "Tedoen"

Opl Notes Files
```

Hier zijn weer drie mogelijkheden : *Opl*, *Notes* of *Files*. Enkel *Notes* is op het ogenblik belangrijk voor ons; over Opl en Files zullen we het later nog hebben. We selecteren dus optie Notes.

Nu komen we in de zgn. 'Notes Editor' terecht. Dit is waar het eigenlijke schrijfblok zich bevindt.

```
/                    9.30a
TEDOEN:
```

Als het schrijfblok beschermd zou zijn met een wachtwoord (in punt 3.3.10 zien we hoe dat moet), moeten we het juiste wachtwoord intikken.

We hebben nu de naam van ons schrijfblok (TEDOEN) op het scherm. Nu gaan we wat notities opmaken :

Opmaken veiligheidsrapport
Laten typen
Opsturen naar Keulen

Op het einde van elke regel, die 254 tekens lang mag zijn, wordt niet op NEER gedrukt maar op EXE. Met de pijltjestoetsen kan de cursor doorheen het hele schrijfblok gestuurd worden.

3.3 Het Notes-menu

Met MODE krijgen we (op de bovenste regel van het scherm) een menu te zien :

```
Find Save Load New→
TEDOEN: Opmaken veili
Laten typen
Opsturen naar Keulen
```

Dit is géén menu met slechts vier, maar met vijftien opties! Uiterst rechts op de bovenste regel staat immers een soort pijltje (→). Dit pijltje geeft aan dat er nog opties zijn. De volledige lijst is :

Find	Opzoeken woord in schrijfblok
Save	Bewaren huidig schrijfblok
Load	Laden (gebruiken) ander schrijfblok
New	Opmaken nieuw schrijfblok
Home	Breng de cursor naar het begin van het blok
End	Breng de cursor naar het einde van het blok
Calc	Maak berekeningen in het blok
Sort	Sorteer het blok
Number	Nummer de regels in het blok
Password	Geef een wachtwoord op (bescherming blok)
Print	Druk het schrijfblok af op papier
Dir	Overzichtslijst (*directory*) van alle schrijfblokken
Copy	Copiëer een schrijfblok
Delete	Wis een schrijfblok
Zap	Wis het complete huidige schrijfblok

Er zijn dus heel wat mogelijkheden! We zullen ze één voor één gedetailleerd bespreken.

3.3.1 Find

Deze Find lijkt op de Find uit het hoofdmenu, maar is toch net even anders. Deze Find zoekt namelijk alleen in het huidige schrijfblok, dus het schrijfblok waar mee gewerkt wordt.

```
Find:keu

Laten typen
Opsturen naar Keulen
```

Om de opdracht (Find) uit te voeren wordt op EXE gedrukt. Indien SHIFT EXE wordt gedrukt, begint de LZ *achteraan* in het schrijfblok te zoeken.
Op MODE drukken geeft de LZ de opdracht terug te keren naar het Notesmenu.
Op ON drukken resulteert in het terugkeren naar de editor (om verder te schrijven).

Als de opgegeven specificatie (*string* : een reeks tekens) gevonden is, wordt de cursor op het begin van het gezochte woord gezet (op de derde regel):


```
Find:keu

Opsturen naar Keulen
```

Indien nu op EXE wordt gedrukt, zoekt de LZ verder in het schrijfblok (of SHIFT EXE om hem het zoeken achteraan te laten aanvangen). Om terug te keren naar het schrijfblok (met de cursor op "Keulen") dient op ON gedrukt te worden. Met MODE keren we terug naar het hoofdmenu.

Als de opdracht "verder zoeken" was gegeven en de string komt niet meer voor, zal de Organiser II ons hierop attent maken :

```
Find:keu

  NO MORE ENTRIES
```

Met ON of EXE keren we terug naar het schrijfblok, met MODE keren we terug naar het menu.

3.3.2 Save

Met Save wordt het gehele schrijfblok op een datapak (of naar het interne geheugen) weggeschreven.

```
Save A:TEDOEN

Laten typen
Opsturen naar Keulen
```

De naam kan eventueel nog aangepast worden indien nodig. De nieuwe naam verschijnt echter niet automatisch in het hoofdmenu, dus opgepast!

Wie eigenlijk niet wil wegschrijven kan op ON drukken en keert dan weer terug naar het schrijfblok. Wie niet op pak A: wil schrijven, kan op MODE drukken en zo een ander pak kiezen.

Als de keuze gemaakt is, wordt op EXE gedrukt zodat het schrijfblok veilig wordt weggeschreven.

```
Saving...

Laten typen
Opsturen naar Keulen
```

3.3.3 Load

Load (*laden*) is net het omgekeerde van Save : er wordt geen schrijfblok opgeslagen, maar één *ingeladen*. Uiteraard moet dit een bestaand schrijfblok zijn. Het proberen in te laden van een niet-bestaand schrijfblok levert een fout op.

Indien we een schrijfblok inladen terwijl we met een ander schrijfblok bezig zijn, zal het huidige schrijfblok uit het geheugen verwijderd worden. Let er dus op, een gewijzigd schrijfblok altijd eerst weg te schrijven alvorens een ander in te laden (de LZ maakt u hierop attent) :

```
      SAVING
    A: TEDOEN

Confirm? Y/N
```

Wilt u het gewijzigde schrijfblok opslaan? Zoja, druk op Y, anders op N. Als Y ingedrukt wordt, zal de computer het schrijfblok wegschrijven :

```
Saving...

Laten typen
Opsturen naar Keulen
```

Dan wordt de gebruiker gevraagd de naam van het te gebruiken schrijfblok in te typen :

```
  LOADING NOTEPAD
Load A: TEDOEN
```

De naam van het zonet gebruikte schrijfblok staat achter "Load" genoteerd, ingeval we het oude schrijfblok terug zouden willen gebruiken. Met de pijltjestoetsen en DEL kan de naam worden aangepast. Zoals vanouds kunnen we op MODE drukken op de paknaam (hier *A:*) te wijzigen. Het inladen gebeurt pas wanneer we op EXE gedrukt hebben.

Als we niet precies weten hoe het schrijfblok dat we willen gebruiken ook weer heet, kunnen we wildcards opgeven in de naam, of de naam gewoon weglaten. In dat geval zal de LZ een lijstje opmaken van alle schrijfblokken die aan die naam beantwoorden.

```
  LOADING NOTEPAD
Load A: T*
        TEDOEN
```

(T* betekent : een schrijfblok waarvan de naam begint met T).

Om TEDOEN te kiezen drukken we dan op NEER.

```
  LOADING NOTEPAD
Load A: TEDOEN
```

Nu op EXE drukken geeft de opdracht TEDOEN in te laden. Maar wat als we een ander schrijfblok willen inladen, waarvan de naam ook met T begint? Dan drukken we nogmaals op NEER :

```
   LOADING NOTEPAD
Load A: TEDOEN

  NO MORE ENTRIES
```

(De onderste regel verschijnt uiteraard alleen wanneer er geen schrijfblokken meer zijn, waarvan de naam met T begint).

Om een nieuwe naam op te geven (bijvoorbeeld TED*), dient op ON gedrukt te worden (of MODE indien we de paknaam willen wijzigen).

Wanneer de goede naam opgegeven is en op EXE is gedrukt, zal de Organiser II het opgegeven schrijfblok inladen en terugkeren naar de editor.

3.3.4 New

New dient, zoals de naam suggereert, om een nieuw schrijfblok op te maken. Merk op dat, net als bij Load, het huidige schrijfblok vergeten wordt als het niet wordt weggeschreven.

Als we het actieve schrijfblok hebben aangepast en het niet hebben ge-Saved alvorens New op te roepen, vraagt de LZ ons of we wel zeker zijn:

```
DATA HAS CHANGED
NOTEPAD

New? Y/N
```

Als we eerst willen Saven drukken we op N en vervolgens op MODE S om het schrijfblok toch nog weg te schrijven. Daarna kunnen we het nieuwe blok creëeren.

De naam moet nieuw zijn. Er mogen dus geen twee schrijfblokken met dezelfde naam voorkomen, wat logisch is (behalve als deze schrijfblokken op verschillende paks staan... Het is perfect mogelijk een blok *A:TEDOEN* en een blok *B:TEDOEN* te hebben. Als "Tedoen" in het hoofdmenu staat en deze optie wordt geselecteerd, zal de LZ echter altijd A:TEDOEN gebruiken, omdat hij eerst gaat zoeken op pak A:, daarna op B:, en pas dan op C:). In het volgende voorbeeld noemen we het nieuwe schrijfblok BOODSCH.

```
/                    9.30a
BOODSCH:
```

Nu is het nieuwe schrijfblok klaar om gebruikt te worden.

3.3.5 Home

Met Home wordt de cursor steeds naar het begin van het gebruikte schrijfblok gestuurd. Dat gaat meestal immers sneller dan steeds maar OP drukken!

3.3.6 End

End stuurt de cursor dan weer naar het einde van het gebruikte schrijfblok. Find, Home en End maken het mogelijk snel door een schrijfblok te "manoeuvreren".

3.3.7 Calc

Een handige optie. Met Notes is het namelijk mogelijk *berekeningen* te maken. Met Notes kan aldus een rekeningetje worden opgemaakt, of iets dergelijks — net als bij een echt schrijfblok.

Een getal waarmee moet worden gerekend dient voorafgegaan te worden met een =. Het is mogelijk verschillende getallen op één regel te noteren, zolang ze maar door een = voorafgaan, bijvoorbeeld :

```
=17.8  =200  =18
```

Bij deze getallen mogen vanzelfsprekend beschrijvingen staan, bijvoorbeeld :

```
/                    9.30a
=300 benzine
=50 motorolie
tijdschriften =200
```

Zoals in de laatste regel duidelijk wordt, kan de beschrijving ook vooraan staan.

3.3.7.1 SUM

Daarna kan er bijvoorbeeld een totaal gemaakt worden, en wel met SUM=.

```
/                    9.30a
=50 motorolie
tijdschriften =200
SUM=
```

Kies dan uit het Notes-menu de optie Calc.

```
Calculating...

tijdschriften =200
SUM=
```

```
/                    9.30a
=50 motorolie
tijdschriften =200
SUM=550
```

Maar de som is niet de enig mogelijke bewerking.

3.3.7.2 ITEMS

De functie ITEMS geeft weer hoeveel getallen er zijn om mee te rekenen. In het bovenstaande voorbeeld zijn dat er 3 (*benzine*, *motorolie* en *tijdschriften*).

3.3.7.3 MEAN

MEAN berekent het gemiddelde van de getallen in het schrijfblok. In het bovenstaande voorbeeld levert dat dus 183.333... op, want de som (550) gedeeld door het aantal getallen (3) geeft het gemiddelde : 183.333.

3.3.7.4 VAR

VAR geeft de *variantie* van de getallen in het schrijfblok. In het bovenstaande voorbeeld geeft VAR de oplossing 31666.6666666 (de variantie van een reeks getallen is de som van de kwadraten van ieder getal verminderd met het gemiddelde). In de praktijk (zonder de functie VAR te gebruiken) wordt dat als volgt berekend;

$$var = (50-183.3)^2 + (200-183.3)^2 + (300-183.3)^2$$

3.7.7.5 STD

STD (Engels : *standard deviation*) geeft de standaardafwijking van de getallen in het schrijfblok. In ons voorbeeld geeft dit het resultaat 177.951304201 (de standaardafwijking is de vierkantswortel van de variantie).

3.7.7.6 MAX

MAX geeft eenvoudig het hoogste getal in de reeks, in ons geval dus 300.

3.7.7.7 MIN

Als tegenpool van MAX geeft MIN het laagste getal in de reeks, in bovenstaand voorbeeld dus 50.

3.3.8 Sort

Vervolgens de mogelijkheid de regels de *sorteren*. Elke regel van het schrijfblok wordt in de sortering opgenomen. De sorteervolgorde wordt bepaald door de ASCII-waarden (zie Appendix A). Dat houdt concreet in dat cijfers voor letters komen, leestekens voor cijfers. Laten we ons eerste voorbeeld eens sorteren.

```
Sort? Y/N

Laten typen
Opsturen naar Keulen
```

We drukken nu Y om te sorteren (N indien er niet hoeft gesorteerd te worden)


```
Sorting...

Laten typen
Opsturen naar Keulen
```

Even later is het schrijfblok gesorteerd :

```
Laten typen
Opmaken veiligheidsr
Opsturen naar Keulen
```

De volgorde klopt nu helaas niet meer, maar het illustreert toch hoe de sort-optie werkt!

3.3.9 Number

Deze functie (we hebben ze nu bijna allemaal gehad) dient om de regels van het schrijfblok te *nummeren*. Dat hoeven we dus niet zelf te doen! De regelnummers worden niet mee weggeschreven wanneer we een Save doen; in de plaats daarvan geeft de LZ een code mee die aangeeft dat de regels eigenlijk genummerd zijn. Dat heeft twee voordelen :

– het schrijfblok is dan korter (neemt minder geheugenruimte in beslag);
– bij een *sortering* wordt er geen rekening gehouden met de regelnummers. De regels worden gesorteerd en vanzelf hernummerd.

```
Number? Y/N

Laten typen
Opsturen naar Keulen
```

Welja, we willen de regels van "Tedoen" wel nummeren. In dat geval drukken we op Y, anders op N. Als nummering een vroegere keer is gekozen, en alle regels dus al genummerd waren, kan de nummering nu ongedaan gemaakt worden door op N te drukken.

Het resultaat is als volgt :

```
1) Opmaken veiligheid
2) Laten typen
3) Opsturen naar Keu
```

3.3.10 Password

De schrijfblokken zijn prima om vertrouwelijke gegevens in op te slaan. Waarom? Eenvoudig omdat deze gegevens zo gescheiden blijven van de alledaagse gegevens (niet te vinden met Find) en vooral omdat elk schrijfblok kan beschermd worden met een *wachtwoord*. Vooraan in dit hoofdstuk hebben we gezien dat het wachtwoord moet worden ingetikt wanneer iemand probeert het beschermde schrijfblok te gebruiken. Met deze optie, Password, wordt het mogelijk het wachtwoord *in te brengen* of te *veranderen*. Op die manier kan het wachtwoord worden veranderd wanneer iemand het ontdekt heeft. Wie de veiligheid van zijn gegevens wil verhogen kan uiteraard zijn wachtwoord periodiek veranderen, bijvoorbeeld eens per week of per twee weken.

Er is echter wel een nadeeltje aan : wie zijn wachtwoord vergeet kan zelf ook geen toegang tot zijn gegevens meer krijgen!

Hoe maken we de LZ het wachtwoord duidelijk? Eenvoudig door Password te kiezen :

Als er al een wachtwoord bestond, moet dat nu nogmaals worden ingetikt (uit veiligheidsredenen) :

```
Old password:

2) Laten typen
3) Opsturen naar Keu
```

Het wachtwoord wordt dan ingetikt (bijvoorbeeld : *Roger Rabbit*). Het wordt niet zichtbaar op het scherm; we zien slechts een reeks #'en verschijnen. Dit om te voorkomen dat eventuele gluurders en pottekijkers over de schouder van hun slachtoffer al te vlug het Mysterie van het Wachtwoord zouden ontdekken.

Als er fouten worden gemaakt werkt DEL. De pijltjestoetsen echter niet!

Daarna wordt, op dezelfde wijze, het nieuwe wachtwoord ingetikt.

```
New password:

2) Laten typen
3) Opsturen naar Keu
```

Om géén wachtwoord in te geven drukken we zondermeer op EXE. Het eventuele oude wachtwoord wordt dan vergeten.

Als het wachtwoord is ingetikt, vraagt de LZ het nog eens in te tikken, dit om veiligheidsredenen. Ja, daar het wachtwoord niet zichtbaar wordt gemaakt, ligt een fout in een klein hoekje. Als het wachtwoord twee keer wordt ingetikt is de kans op fouten immers veel kleiner!

```
Again:

Laten typen
Opsturen naar Keulen
```

Wanneer vanaf nu het schrijfblok wordt opgevraagd moet het wachtwoord worden ingetikt. Wie het niet kent, heeft dus geen toegang tot de geheime gegevens!

Nog een tip rondom de keuze van wachtwoorden : gebruik geen voordehandliggende wachtwoorden zoals *geheim*, uw naam, geboortedatum, firma of iets dergelijks. Gebruik eerder een onverwachte (maar best eenvoudig te onthouden) letter- en/of cijfercombinatie.

3.3.11 Print

Als er een printer is aangesloten (zie verder in dit boek), kan een schrijfblok worden afgedrukt op papier. Daarvoor dient de optie Print. Zoals altijd vraagt de Organiser II voor bevestiging van de opdracht :

```
Print?  Y/N

Laten typen
Opsturen naar Keulen
```

Als Y wordt ingetikt stuurt de Organiser II het gehele schrijfblok door naar de printer (of de communicatielink, zie verder in dit boek).

```
Printing...

Laten typen
Opsturen naar Keulen
```

Als er geen printer is aangesloten, levert deze keuze de foutmelding DEVICE MISSING op (*geen toestel aangesloten*). Als het drukken dient te worden afgebroken drukken we op ON :

```
Stop printing? Y/N

Laten typen
Opsturen naar Keulen
```

Indien Y wordt gekozen wordt het drukken definitief afgebroken.

3.3.12 Dir

De optie Dir dient om een soort inhoudstabel (*directory*) op te maken. De directory is een lijst van alle schrijfblokken.

```
     DIR of NOTES
Dir A:
```

Door op MODE te drukken kan de paknaam veranderd worden. Druk nu op EXE :

```
     DIR of NOTES
Dir A:*
      TEDOEN          60
      PROJECT       4750
```

(Het sterretje verschijnt vanzelf). Met de pijltjestoetsen OP en NEER kan de lijst doorlopen worden. Met ON gaan we terug naar het Notes-menu.

In plaats van EXE hadden we ook een wildcard kunnen opgeven, bijvoorbeeld P*T.

Op het midden van elke regel staat de naam van het schrijfblok, rechts staat de lengte in bytes.

3.3.13 Copy

De Copy-optie dient om schrijfblokken te copiëren. Het is altijd handig een veiligheidscopie (Engels : *backup*) bij de hand te hebben.

Verschillende schrijfblokken kunnen tegelijkertijd worden gecopiëerd, als één of meer wildcards worden opgegeven als naam.


```
     COPY NOTES
From A:
```

Nu moet dus de naam worden ingetikt. Hier kunnen, zoals hierboven vermeld, eventueel wildcards worden opgegeven. Om alle schrijfblokken te copiëren volstaat het op EXE te drukken. Om te ontsnappen naar het Notes-menu kan ON worden ingetikt, maar dat is nu wel duidelijk.

```
     COPY NOTES
From A:*
     TEDOEN
     PROJECT
```

Er verschijnt dan, net als bij Dir, een lijstje van de beschikbare schrijfblokken. Druk nu op EXE.

```
     COPY NOTES
From A:*
To   B:
```

Het spreekt voor zich dat hier geen wildcards of namen kunnen worden opgegeven (*behalve* als er slechts één schrijfblok te copiëren is). Druk daarom gewoon op EXE (met MODE kan de doelpak worden veranderd, maar ook dat is nu wel duidelijk).

Vervolgens verschijnt er op de onderste regel van het scherm steeds de naam van het blok dat gecopiëerd wordt.

```
     COPY NOTES
From A:*
To   B:
Copying A:TEDOEN
```

De Copy-functie kan afgebroken worden door op ON te drukken. De Organiser II vraagt in dit geval echter wel om bevestiging.

```
     COPY NOTES
STOP A:TEDOEN

Confirm? Y/N
```

Als nu Y wordt ingedrukt wordt het copiëren afgebroken. Het blok dat werd gecopieerd tijdens de onderbreking wordt op de doelpak (in bovenstaand voorbeeld op B:) gewist.

Wordt er echter op N of ON gedrukt, gaat het copiëren gewoon verder.

3.3.14 Delete

Delete (wissen schrijfblok(ken)) werkt haast net hetzelfde als Copy.

```
    DELETE NOTES
Delete A:
```

Tik hier de naam van het te wissen schrijfblok in (of een wildcard indien er meer dan één moet worden gewist). Weer vraagt de bezorgde computer om bevestiging :

```
    DELETE NOTES
Delete A:*

Confirm? Y/N
```

Y bevestigt en start het wissen. N en ON ontkennen het. Gewiste schrijfblokken kunnen niet worden gerecupereerd, dus wees uiterst voorzichtig!

3.3.15 Zap

Zap is de laatste optie van het Notes-menu. Zap wist het *huidige* schrijfblok uit het werkgeheugen. Eventuele copieën van het blok (zelfs in het intern geheugen, pak A:) blijven gewoon bestaan.

```
Zap? Y/N

2) Laten typen
3) Opsturen naar Keu
```

Wat de toetsen Y, N en ON doen, zal ik nu zeker niet meer moeten vertellen.

4. Xfiles

4.1 Inleiding

Find en Save zijn zowat de meestgebruikte items van het hoofdmenu. Met deze twee opties schrijven we namen, adressen, e.d.m. naar het geheugen (pak A:), een data- of RAMpak. Wanneer de LZ een nieuwe data- of RAMpak wil gebruiken, wordt die immers geïnitialiseerd, d.w.z. *gebruiksklaar gemaakt*. Eén van de dingen die de LZ doet wanneer hij een pak initialiseert, is het speciale bestand *MAIN* opzetten. Find en Save werken namelijk altijd met dit bestand, wat dus op elke pak te vinden dient te zijn. Wanneer het wordt opgezet, is het uiteraard nog leeg — totdat we er met Save iets in opschrijven.

Het zou uitermate prettig zijn om met *verschillende* bestanden op één pak te kunnen werken, bijvoorbeeld één met adressen van vrienden, een ander met zakelijke gegevens, e.d.m. Daarvoor dient de optie Xfiles, wat staat voor 'eXtra Files', de naam zegt het zelf.

4.2 Het Xfiles-menu

Zodra Xfiles vanuit het hoofdmenu wordt gekozen (met X gaat dat het snelste), verschijnt het Xfiles-menu.

```
≡                    9.30a
Find    Save    New
Open    Print   Sort
Dir     Copy    Delete
```

Hier volgt een kort overzicht van dit menu :

Find	Opzoeken gegevens in bestand
Save	Wegschrijven gegevens naar bestand
New	Creëren nieuw bestand
Open	Openen bestand
Print	Afdrukken record(s)
Sort	Sorteren bestand
Dir	Inhoudsopgave (directory)
Copy	Copiëren bestand(en)
Delete	Wissen bestand(en)

De meeste van deze opties zijn praktisch identiek aan de Notesopties.

4.2.1 Find

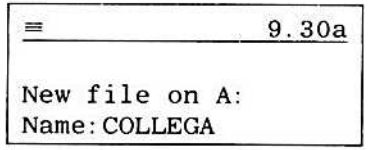
Dit behoeft weinig verdere uitleg. Het werkt immers net hetzelfde als de Find uit het hoofdmenu, behalve dat het ook met andere bestanden dan MAIN kan werken. Deze Find werkt altijd met het bestand dat actief werd gemaakt (zie verder).

4.2.2 Save

Ook Save is identiek aan de Save in het hoofdmenu.

4.2.3 New

Met New kunnen we een nieuw bestand aanmaken. Dit nieuwe bestand wordt dan onmiddellijk gebruikt (*actief gemaakt*).

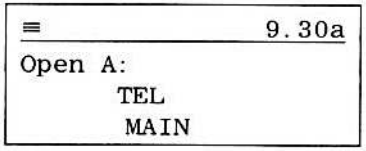


De bestandsnaam (in bovenstaand voorbeeld : COLLEGA) moet uiteraard uniek zijn. Met MODE kan de pakletter worden aangepast indien nodig.

4.2.4 Open

Met Open wordt een reeds bestaand bestand geopend voor gebruik. Dit is wat we bedoelen met een bestand *actief maken.* Men had deze optie dus net zo goed *Use* of *Load* kunnen noemen (dat zouden zelfs betere benamingen zijn geweest).

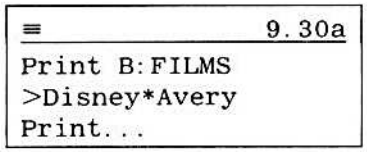
Open werkt met wildcards (net als *Load* in het Notes-menu). Om een overzicht van alle bestanden te krijgen (ingeval de naam van een bepaald bestand u ontschoten is) drukt u gewoon op NEER wanneer de LZ de bestandsnaam opvraagt (zie hieronder).



Wanneer de ingetikte bestandsnaam niet bestaat, zal de LZ protesteren.

4.2.5 Print

Als er een printer is aangesloten (zie verder in dit boek), kan een bestand volledig worden afgedrukt op papier. Daarvoor dient de optie Print die net hetzelfde werkt als bij Notes. De LZ vraagt om een *specificatie*, die wildcards mag bevatten. Om bijvoorbeeld in het bestand FILMS een lijst van tekenfilms van Walt Disney of Tex Avery af te drukken, zouden we gebruiken :



De boodschap "Print..." verschijnt onderaan het scherm. Als er geen printer is aangesloten, zorgt deze keuze voor een gepaste foutmelding (DEVICE MISSING : *toestel niet aangesloten*). Als het drukken dient te worden afgebroken drukken we op ON.

4.2.6 Sort

Sort sorteert niet een record, zoals bij Notes het geval is, maar het gehele bestand. Het bestand wordt ASCII-gesorteerd, net als bij Notes. Leestekens komen dus voor cijfers, cijfers voor hoofdletters, hoofdletters voor kleine letters, enz. Voor een overzicht van de ASCII-tekens, zie Appendix A.

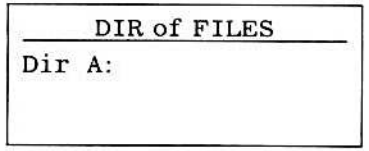
Het opgeven van een bestandsnaam gebeurt net hetzelfde als bij Notes.

Tijdens het sorteren verschijnt de boodschap "Sorting..." onderaan het scherm. Dit sorteren kan een tijdje duren, alnaargelang de lengte van het bestand.

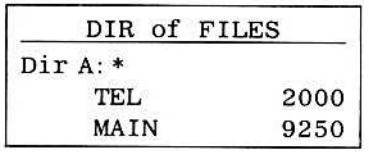
Het sorteren kan worden afgebroken door op ON te drukken. Het kan dan zijn, alnaargelang wanneer op ON gedrukt wordt, dat het bestand gedeeltelijk al gesorteerd is.

4.2.7 Dir

Dir werkt net hetzelfde als bij Notes. De optie Dir dient om een soort inhoudstabel (*directory*) op te maken. De directory is een lijst van alle files.



Door op MODE te drukken kan de paknaam veranderd worden. Druk nu op EXE of NEER :



(Het sterretje verschijnt vanzelf). Met de pijltjestoetsen OP en NEER kan de lijst doorlopen worden. Met ON gaan we terug naar het Xfiles-menu.

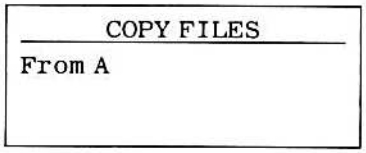
In plaats van EXE hadden we ook een wildcard kunnen opgeven, bijvoorbeeld M*.

Op het midden van elke regel staat de naam van de file, rechts staat de lengte in bytes.

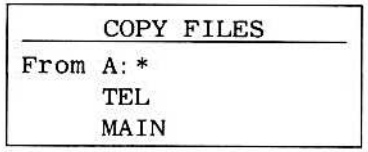
4.2.8 Copy

De Copy-optie werkt zo goed als identiek aan de Copy-optie uit het Notes-menu en dient om bestanden (files) te copiëren.

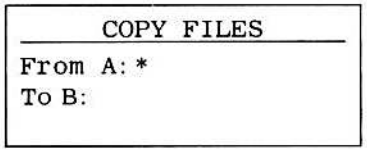
Verschillende bestanden kunnen tegelijkertijd worden gecopiëerd, als één of meer wildcards worden opgegeven als naam.



Nu moet dus de naam worden ingetikt. Hier kunnen, zoals hierboven vermeld, eventueel wildcards worden opgegeven. Om alle bestanden te copiëren volstaat het op EXE te drukken. Om te ontsnappen naar het Xfiles-menu kan ON worden ingetikt.

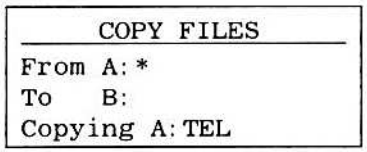


Er verschijnt dan, net als bij Dir, een lijstje van de beschikbare bestanden. Druk nu op EXE.



Het spreekt voor zich dat hier geen wildcards of namen kunnen worden opgegeven (*behalve* als er slechts één bestand te copiëren is). Druk daarom gewoon op EXE (met MODE kan, zoals overal, de doelpak worden veranderd).

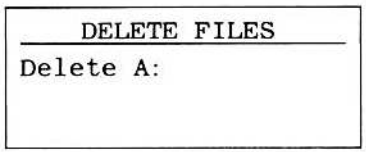
Vervolgens verschijnt er op de onderste regel van het scherm steeds de naam van het blok dat gecopiëerd wordt.



De Copy-functie kan afgebroken worden door op ON te drukken.

4.2.9 Delete

Delete (wissen bestand(en)) werkt haast net hetzelfde als Copy.



Tik hier de naam van het te wissen bestand in (of een wildcard indien er meer dan één moet worden gewist).

Dit beëindigt de bespreking van Xfiles, op naar Time.

5. De tijd instellen : Time

5.1 Inleiding

Alle modellen van de Organiser II beschikken over een intern uurwerk. Dit uurwerk wordt onder andere gebruikt voor de 8 ingebouwde wekkers (zie verder in dit boek) en de agenda (zie wederom verder in dit boek).

De Time-optie uit het hoofdmenu doet dienst als horloge. Ze laat toe de tijd in te stellen. De Organisers komen immers niet uit de fabriek met vasteland-tijd! Daarbovenop kan de LZ als stopwatch of als... eierklokje worden gebruikt!

5.2 Showtime!

Wanneer de Time-optie wordt geselecteerd, komt de juiste tijd op het scherm als volgt :

```
Thu 16 Nov 1989 Wk46
   9:30:05 am
      London
   United Kingdom
```

Op de eerste regel zien we de dag van de week (Thu : Thursday, donderdag dus), gevolgd door de dag in de maand, de maand, het jaar en de week in het jaar. Donderdag 16 november 1989 (mijn verjaardag, als het u interesseert), valt dus in de 46ste week van het jaar.

De tweede regel dan, toont ons het juiste uur, of het in de voormiddag of de namiddag is (am/pm).

De volgende twee regels geven aan waar de gebruiker zich bevindt (of zich zou moeten bevinden). Hoe weet de Organiser II dat? Eenvoudig omdat we het moeten *instellen*.

5.3 Instellen

We zouden nu op ON kunnen drukken om terug naar het hoofdmenu te gaan. In de plaats daarvan gaan we de klok instellen. We drukken daarvoor, jawel, op MODE. Dat had u natuurlijk al lang begrepen.


```
Thu 16 Nov 1989 Wk46
9:30:20 am

Stopwatch Timer Set→
```

Het Time-menu komt dan op de onderste regel van het scherm, met de vier opties *Stopwatch*, *Timer*, *Set* en *Daylight-saving*, een optie die niet op het scherm staat (het pijltje rechts onderaan geeft immers aan dat er nog een optie is!).

5.3.1 Stopwatch

De stopwatch is, zoals de naam suggereert, een chronometer. Deze chronometer kan meten tot op 5 honderdsten van een seconde precies.

```
      STOPWATCH
   00:00:00.00
```

ON : ga terug naar het Time-menu
EXE : stop stopwatch; druk nu op DEL om ze terug op nul te zetten.
DEL : bevriezen van de display; het tellen gaat door. Druk nogmaals op DEL om te 'ontdooien'.

Telkens op DEL wordt gedrukt, bevriest de display. Het flitsend puntje echter duidt aan dat het tellen doorgaat. Deze mogelijkheid dient om zgn. '*lap times*' (rondetijden) te meten. De tijd wordt weer correct aangegeven door terug op DEL te drukken.

5.3.2 Timer

De Timer dient om een ingestelde tijd te laten aftellen. Ideaal dus voor wie een ruimteveer wil lanceren of een eitje wil koken.

```
        TIMER

Mins:00    Secs:00
```

ON	: terugkeren naar Time-menu
OP, NEER, LINKS, RECHTS	: instellen tijd
EXE	: starten aftelling.

Wanneer de aftelling beëindigd is, laat de LZ gedurende één minuut een alarm horen. Dit alarm kunnen we afbreken door op ON te drukken. Maar het aftellen zelf kan ook afgebroken worden (uiteraard met ON). Om naar het Time-menu terug te keren moet ON twee keer ingedrukt worden.

5.3.3 Set

Set dient om de klok in te stellen. Dat gaat als volgt :

```
     SET THE TIME
Thu 16 Nov 1989 Wk46
      9:35:05 am
     Mode:24 Hour
```

Op de tweede schermregel stellen we de maand, dag in de maand en het jaar in (met de pijltjestoetsen). De LZ past zelf de week en de weekdag aan. Op de derde schermregel kunnen we het uur instellen, en op de onderste schermregel de *mode*. Deze mode geeft aan of we de tijd in 12-uur of 24-uur-formaat voorgeschoteld krijgen. In de regel gebruikt een Europeaan en een militair 24 uur, terwijl een Amerikaan 12 uur gebruikt.

5.3.4 Daylight-saving

Deze mogelijkheid wordt gebruikt om aan te geven dat de *zomertijd* van kracht is. De LZ weet niet wanneer de zomer- of wintertijd precies van kracht is : dit moet dus manueel worden aangegeven (om later de juiste tijd in vreemde landen te kunnen berekenen). Kies dus *Daylight-saving* om de zomertijd in of uit te schakelen (On : zomertijd, Off : wintertijd).

6. De wereld op zak : World

6.1 Inleiding

Met de Organiser II model LZ hebben we de wereld — letterlijk -op zak, zo'n 650 steden in totaal. Met deze optie heeft men gedacht aan die gebruikers die vaak reizen of internationaal bellen. Een hele reeks landen en steden zijn in het geheugen opgeslagen, tesamen met hun internationale kiescode (bijvoorbeeld 31 voor Nederland, 32 voor België, 44 voor Engeland). Zoals we in het vorige hoofdstuk al hebben kunnen zien, kunnen we ook instellen waar we wonen. Daar zal de LZ later rekening mee houden.

6.2 Bellen

We selecteren World en krijgen de laatstgezochte stad op ons scherm (in dit voorbeeld dus Johannesburg) :

```
   Johannesburg
   South Africa
Thu 16 Nov 9:00a
Dial: 010 27 11
```

Op de onderste regel zien we hoe we Johannesburg (vanuit Engeland) kunnen bereiken. Op de eerste en tweede regel staan respectievelijk de stad en het land.

Deze onderste regel kan dus voor elke stad in het geheugen aangeven hoe ze te bereiken. Nu zijn er twee mogelijkheden :

- de stad is in het binnenland;
- de stad is in het buitenland.

6.2.2 In het binnenland

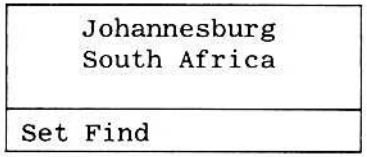
In dit geval staat er enkel een intern zonenummer op het scherm.

6.2.3 In het buitenland

In dit geval, zoals in het voorbeeld, staat de 'internationaalcode' vooraan (vanuit Engeland 010, vanuit Nederland en België 00) gevolgd door het landnummer (voor Zuid-Afrika 27) en het zonenummer (Johannesburg heeft dus zonenummer 11).

6.3 Instellen

We kunnen de klok ook instellen. Wanneer het bovenstaande scherm zichtbaar is, drukken we niet op ON (dan zouden we naar het hoofdmenu gaan) of op de pijltjestoetsen (dan zouden we een lijst van de steden in het geheugen krijgen), maar op MODE. Dan komt het volgende menu op het scherm :



6.3.1 Set

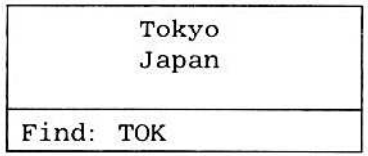
Met Set kunnen we onze woonplaats instellen.

Net als bij Find heeft de Organiser II genoeg aan enkele letters om de stad terug te vinden. Ook met OP en NEER kan de stad opgezocht worden. Druk EXE wanneer de correcte stad gevonden is. De kiescodes zullen nu altijd aangepast zijn aan het thuisland.

Merk hier op dat de internationaalcode van 010 (vanuit Engeland) veranderd wordt in 00 (vanuit België).

6.3.2 Find

Met Find kunnen we snel een stad opzoeken in het geheugen.



Meestal heeft de Organiser II slechts enkele letters nodig om de stad te vinden die we zoeken. Als de volledige stadsnaam is ingetikt en het land

verschijnt niet onderaan het scherm, dan is de stad niet in het geheugen opgenomen (jammer genoeg is de lijst niet uitbreidbaar). De lijst kan ook doorlopen worden met OP en NEER.

Wanneer de correcte stad gevonden is kunnen we op EXE drukken om onze keuze te bevestigen. Dan verschijnt op het scherm de tijd en de kiescode van de zonet opgezochte stad.

7. Calc

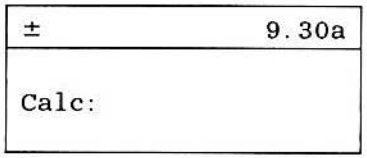
De Organiser II model LZ heeft een uitstekende wetenschappelijke calculator aan boord, welke nog uitgebreider is dan die van zijn voorgangers. Deze kan worden gebruikt door de optie Calc te kiezen in het hoofdmenu.

Een wetenschappelijke calculator is een calculator die is uitgerust met wiskundige functies die bijvoorbeeld in de trigonometrie gebruikt worden. De Organiser II is dus uitgerust om wiskunde aan te kunnen.

(Opmerking : in dit boek staan methoden vermeld waarmee men de kracht van de calculator aanzienlijk kan vergroten. Later zullen we zelf extra wiskundige functies bijvoegen).

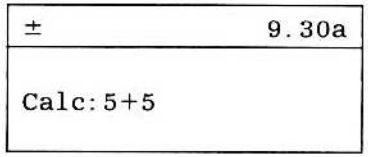
7.1 Algemeen gebruik

Kies de optie Calc. Het ikoontje linksboven lijkt op een ± teken.



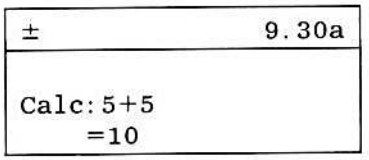
Zoals u aan de cursor kunt zien — hij knippert niet meer — heeft de LZ zijn klavier automatisch op 'numeriek' gezet. U hoeft dus niet meer de SHIFT-toets te gebruiken om een getal in te typen.

We beginnen met een eenvoudig voorbeeld : 5 + 5. Toets in :



(de '+' kan worden verkregen door de toets X in te drukken, waarboven het plusteken staat).

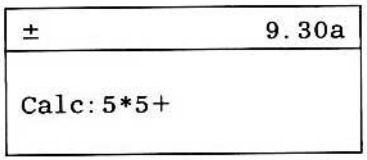
Zoals we zien, is de werking enigszins anders dan bij andere rekenmachines. Men kan te allen tijde zijn berekening op het scherm zien, en d.m.v. LINKS en RECHTS zelfs eventuele fouten corrigeren. Anders is ook, dat we niet = dienen te gebruiken om de oplossing te verkrijgen. EXE dient hier namelijk voor.



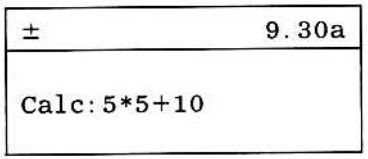
Briljant als de LZ is, toont hij meteen de correcte uitkomst op de onderste regel van het scherm.

De berekening — 5 + 5 — kan nu worden geëditeerd indien nodig. We zouden er dus 5 * 5 (vijf maal vijf) van kunnen maken zonder het geheel opnieuw in te tikken. Toegegeven, bij dit voorbeeld heeft dit weinig zin, maar om één of twee cijfers in de berekening **TAN(SIN(2*PI)+COS(1.181920))** te wijzigen, is deze mogelijkheid toch wel handig. Daarvoor gebruiken we de cursortoetsen, DEL (wissen cijfer links van de cursor), SHIFT DEL (wissen cijfer op de cursor). EXE geeft zoals gewoonlijk de uitkomst.

Maar er zijn meer mogelijkheden. NEER namelijk zorgt ervoor dat de vorige uitkomst (in dit voorbeeld dus 10) automatisch "ingetikt" wordt. Stel dat we de volgende berekening maken :

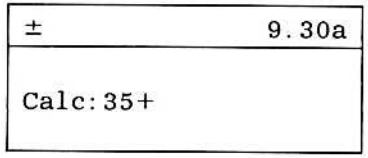


Nu drukken we op NEER en de vorige uitkomst verschijnt op de plaats van de cursor :



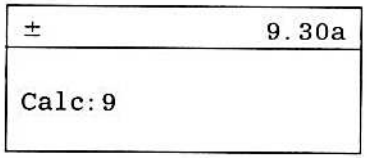
Een handigheidje om snel te kunnen optellen! De LZ onthoudt immers steeds de vorige uitkomst, zodat die op het gewenste ogenblik kan gebruikt worden. Dat kan niet alleen door in de berekening op NEER te drukken, maar ook op de volgende manier :

De berekening is uitgevoerd, de uitkomst (35) staat op de onderste regel van het scherm. Druk nu de toets A in :



De oude uitkomst is nu het eerste item in een nieuwe berekening, en het plusteken staat er al. In de plaats van A kan elke nietnumerieke toets of elke niet-manipulatieve toets (dus niet ON, MODE, DEL, SHIFT DEL, EXE, of de pijltjestoetsen) gebruikt worden.

Wanneer de uitkomst op de onderste regel staat en een nieuwe berekening moet worden gemaakt, die met de vorige geen uitstaans heeft, dient eenvoudig een cijfer ingetikt te worden. Dit cijfer vormt dan het eerste teken van de nieuwe berekening. Stel dat we, wanneer de uitkomst van de net gemaakte berekening op de onderste regel staat, op 9 drukken :



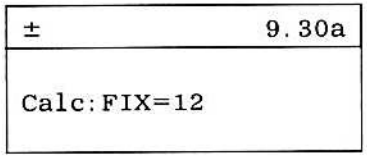
De vorige berekeningen zijn weg, en de cursor staat achter de net ingetikte 9.

Merk op dat het gebruik van de LZ-calculator tamelijk verschillend is dan het gebruik van zeg maar een HP. Ikzelf prefereer de LZ-calculator. Met deze is het werken veel aangenamer en overzichtelijker. Het is immers ook mogelijk te werken met haakjes en speciale functies, zoals tangens en sinus. Het is zelfs mogelijk, zoals we later zullen zien, om het gamma van functies zelf uit te breiden, of met een andere numerieke precisie te werken. Wat voor sommige mensen misschien een aanpassing kan vragen, is de notering voor vermenigvuldiging en deling. De Organiser II gebruikt namelijk een sterretje (*) voor vermenigvuldiging, een slash (/) voor deling en twee sterretjes (**) voor machtsverheffing, net zoals de meeste andere computers. De berekening

```
(5/3)*7
```

betekent dus : vijf delen door drie, en deze uitkomst vermenigvuldigen met zeven.

De LZ kan rekenen met een precisie tot 12 cijfers na de komma. De precisie kan worden ingesteld. Daarvoor dient de opdracht FIX (letters worden bij Calc verkregen door SHIFT ingedrukt te houden met de bijhorende lettertoets).

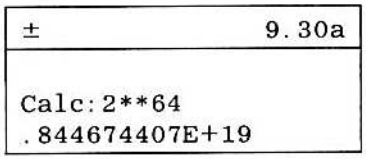


Dit brengt de precisie op haar maximum : 12 cijfers. Merk het gelijkheidsteken (=) op!

7.2 Wetenschappelijke notering

Wetenschappelijke notering is een manier van getallen neerschrijven, zoals dat in de wetenschap gebeurt. Ook de calculator van de LZ, die we daarnet trouwens 'wetenschappelijk' hebben genoemd, gebruikt deze vorm van notering.

Bereken : 2**64 (2 tot de 64e macht). Dit is uiteraard een enorm getal.



Op het eerste gezicht lijkt de oplossing iets meer dan 1.8 te zijn. Dat kan natuurlijk niet. Maar aan de rechterzijde van het getal zien we de notering

```
E+19
```

hetgeen betekent dat we het decimaalteken 19 plaatsen naar rechts moeten opschuiven. De oplossing zal dus ongeveer 18446744070000000000 bedragen. Had er gestaan :

```
E-19
```

dan zou dat hebben betekend dat het decimaalteken 19 plaatsen naar links moet worden verschoven, dus 0.000000000000000000001844674407.

We kunnen deze notering ook zelf gebruiken in onze berekeningen. 1E3 is hetzelfde als 1000 (een 1 met 3 nullen, zeg maar). 1E-3 geeft dan 0.001. Voor berekeningen met hoge getallen wordt de wetenschappelijke notering zeer vaak gebruikt, omdat deze kort is.

7.3 De werkgeheugens

Een uitgebreide rekenmachine beschikt uiteraard over een geheugen voor tijdelijke opslag van getallen en uitkomsten. Hierboven hebben we al gezien dat de uitkomst van de vorige berekening kan worden opgeroepen door tijdens het intikken van de berekening op NEER te drukken. Maar de Organiser II heeft nog een troef in de mouw. De calculator van deze handcomputer heeft maar liefts 10 werkgeheugens! Ze zijn genummerd 0-9 en kunnen door een druk op MODE geactiveerd worden.

De uitkomst staat onderaan op het scherm en we drukken op MODE.

```
±                    9.30a
          <M0=0>

Calc:25*25
```

Op de onderste regel wordt aangegeven dat de uitkomst (625) in een werkgeheugen kan worden opgeslagen. We kunnen zelf kiezen welk geheugen: van 0 tot en met 9. Druk eenvoudig op MODE tot het gewenste geheugen staat aangegeven (het kan ook met MODE gevolgd door herhaald OP en NEER of door MODE cijfer, bijvoorbeeld MODE 0).

We kunnen de uitkomst in het werkgeheugen opslaan. We kunnen deze uitkomst bijvoorbeeld ook bij het geheugen bijvoegen. Als geheugencel 0 (genoteerd als M0) het getal 200 bevat en de bovenstaande uitkomst wordt erbijgevoegd, bevat M0 nadien dus 825. Behalve opslaan in en bijvoegen bij een geheugen, kunnen we de uitkomst ook van een geheugen aftrekken. We gaan nu bepalen wat we precies gaan doen :

EXE : het getal in het geheugen wordt vergeten en vervangen door de zojuist bekomen uitkomst;
+ : de uitkomst wordt in het geheugen bijgevoegd;
− : de uitkomst wordt in het geheugen afgetrokken.

De geheugens M0 tot M9 kunnen uiteraard ook in berekeningen worden gebruikt. Wanneer op MODE wordt gedrukt, verschijnt de *geheugenselector* op de plaats van de cursor :

```
±                    9.30a

Calc:9+M0
```


Het geheugencijfer kan worden verhoogd door weer op MODE te drukken. Na M9 komt M0 terug, zoals kon worden verwacht.

```
±                    9.30a

Calc:9+M0
=634
```

7.4 OPL-functies

Het is mogelijk om met de rekenmachine wiskundige functies aan te roepen, die gebruikt worden in de programmeertaal OPL, die in de Organiser II is ingebouwd. Dit zijn functies als SIN (sinus), COS (cosinus), INT (integer : geheel deel), ROUND (afronding), enzovoort.

Probeer :

```
+                    9.30a

Calc:SIN(1.2*PI)
=-0.587785252294
```

Het werkt. De Organiser II kent inderdaad de SIN-functie, en het getal PI. Het is ook mogelijk met de Organiser II de *waarheid* van iets te laten berekenen. Bijvoorbeeld : is de sinus van 9 gelijk aan 1?

```
±                    9.30a

Calc: (SIN(9))
=0.000000000000
```

Het antwoord is nul : het is dus niet waar. Bereken nu :

(SIN(9).412118485241)

```
±                    9.30a
(9)=0.412118485241)

=-1.000000000000
```

Dit keer is het antwoord -1 : een waarde die dus niet nul is en bijgevolg waar betekent. Men kan de waarheid van bepaalde stellingen dus testen

Dit keer is het antwoord -1 : een waarde die dus niet nul is en bijgevolg waar betekent. Men kan de waarheid van bepaalde stellingen dus testen met enkele zgn. 'relationele operatoren' : = (is gelijk aan), < (is kleiner dan), > (is groter dan), <=(kleiner of gelijk aan), >= (groter of gelijk aan), <> (niet gelijk aan). Over dit soort algebra (genaamd Boole-algebra of logica) zullen we het later nog hebben.

De OPL-functies, die we met de calculator kunnen gebruiken zijn de volgende :

ABS: ABS(expressie) geeft de absolute waarde van die expressie, dus de uitkomst zonder teken. ABS(-5) bijvoorbeeld geeft 5.

ACOS (hoekcosinus): ACOS(expressie), welke is uitgedrukt in radialen, bevat een cosinus waarvan ACOS de hoek berekent.

ASIN (hoeksinus): ASIN(expressie), welke is uitgedrukt in radialen, bevat een sinus waarvan ASIN de hoek berekent.

ATAN: ATAN(expressie) geeft de boogtangens (arc tangent) van de expressie in radialen.

COS: COS(expressie) geeft de cosinus van de expressie, eveneens in radialen.

DAY: DAY geeft de dag van de maand (tussen 1 en 31).

DAYS (aantal dagen tussen opgegeven datum en 1 januari 1900): DAYS(<dag%>, <maand%> en <jaar%> zijn integere expressies). Deze functie berekent het aantal dagen tussen 1 januari 1900 en de datum die in de argumenten is weergegeven. Deze functie kan dus ook worden gebruikt om het verschil in dagen tussen twee data te berekenen, bijvoorbeeld:

```
DAYS(1,1,2000)-DAYS(16,11,1986)
```

DEG: DEG(expressie) zet de waarde van de expressie, die in radialen is gegeven, om in graden. Bijvoorbeeld : DEG(2.5) geeft weer hoeveel graden 2,5 radialen zijn (uitkomst is 1.432394488E+02 of 143.2394488).

DOW (dag in de week): DOW(<dag%>, <maand%>, <jaar%>) zijn integere expressies. Deze functie berekent de dag in de week (DOW staat voor day of week), waarbij 1 staat voor maandag en 7 staat voor zondag.

EXP: EXP(expressie) geeft de waarde van de rekenkundige constante e (=2.718281828460...) tot de macht van de expressie verheven. Om het getal e te krijgen volstaat het dus EXP(1) te berekenen.

HOUR: HOUR geeft het uur in de dag (tussen 0 en 23).

IABS: IABS(expressie) geeft de absolute waarde van een integer getal (een geheel getal tussen −32768 en +32767).

INT: INT(expressie) geeft het gehele deel van de waarde van de expressie. INT(PI) geeft dus 3. INT staat trouwens voor integer en kan dus alleen integere uitkomsten opleveren (zie functie IABS).

INTF: INTF(expressie) is hetzelfde als INT behalve dat de oplossing een vlottende-komma-getal blijft. INTF dient te worden gebruikt als de uitkomst niet binnen de integere getallen valt (zie wederom : IABS).

LN: LN(expressie) geeft de natuurlijke logaritme van de expressie.

LOG: LOG(expressie) geeft de logaritme met basis 10 (Briggse logaritme) van de expressie. Opmerking : later zullen we een manier zien om logaritmen te berekenen met een veranderlijke basis.

MAX: MAX(expressie, expressie, expressie, ...) geeft de uitkomst van de hoogste expressie. MAX(1,2,3,0) levert dus 3 op.

MEAN: MEAN(expressie, expressie, expressie, ...) geeft het gemiddelde van alle expressies in de lijst.

MIN: MIN(expressie, expressie, expressie, ...) geeft de uitkomst van de laagste expressie. MIN(1,2,3,0) levert dus 0 op.

MINUTE: MINUTE feeft de minuut in het uur (tussen 0 en 59).

MONTH: MONTH geeft de huidige maand (tussen 1 en 12).

PI: PI geeft de constante pi (= ca. 3.14159265359).

RAD: RAD(expressie) zet de graden in de expressie om in radialen. Zie ook : DEG.

RND: RND geeft een willekeurig getal tussen 0 (inclusief) en 1 (exclusief). RND kan dus wel 0 opleveren maar niet 1. RND staat voor het Engelse woord '*random*' : willekeurig.

SECOND: SECOND geeft de seconde in de minuut (tussen 0 en 59).

SIN: SIN(expressie) geeft de sinus van de expressie in radialen.

SQR: SQR(expressie) geeft de vierkantswortel van de expressie. SQR staat voor het Engelse 'square root' of vierkantswortel. OPGELET program-

meurs PASCAL, ALGOL en aanverwante talen : de Organiser II gebruikt voor de vierkantswortel SQR, niet SQRT! Om een kwadraat te krijgen dient (expressie)**2 of (expressie)*(expressie) te worden gebruikt!

STD: STD(expressie, expressie, expressie, ...) geeft de standaardafwijking van alle expressies in de lijst.

SUM: SUM(expressie, expressie, expressie, ...) geeft de som van alle expressies in de lijst. SUM(2,4,6) geeft dus 12, SUM(1,2,3,4,5,6,7,8,9,10) geeft 55.

TAN: TAN(expressie) geeft de tangens van de expressie in radialen.

VAR: VAR(expressie, expressie, expressie, ...) geeft de variantie van alle expressies in de lijst.

WEEK (week in het jaar): WEEK(<dag%>, <maand%>, <jaar%>) zijn integere expressies. WEEK berekent de week in het jaar van de opgegeven datum. De LZ neemt als eerste dag van de eerste week de eerste maandag in het jaar.

YEAR: YEAR geeft het huidige jaar (tussen 1900 en 2155).

%: percentageberekeningen. 100 + 5% levert dus 105 op, 115<15% geeft 15, 115>15% geeft 100, 100 * 5% levert 5 op.

7.5 Uitbreidingen op OPL-functies

Later, wanner we de programmeertaal OPL besproken hebben en enkele kleine programma's hebben geschreven, zullen we zien hoe de taal OPL en dus ook de calculator kunnen worden verrijkt met extra functies, zoals bijvoorbeeld cotangens, afgeleide van een sinus, secans, cosecans e.d.m.

8. Alarm

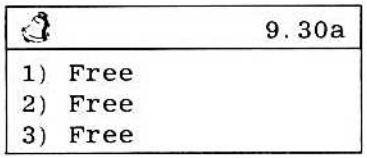
8.1 De 8 alarmen

De LZ heeft de mogelijkheid om, op door de gebruiker ingevoerde tijdstippen, een alarmsignaal te laten horen. Dit kan een éénmalig gebeuren zijn, of het kan worden herhaald met regelmatige intervallen (ieder uur, iedere dag, iedere werkdag, iedere week). Men kan maximaal één week van tevoren een alarmsignaal 'programmeren'.

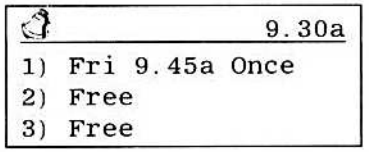
Voor het gemak is deze functie niet beperkt tot één alarm, maar kan men met maar liefst 8 verschillende alarmen tegelijkertijd werken. Deze alarmen kunnen los van elkaar en in nietchronologische volgorde worden gebruikt.

8.2 Het programmeren van een alarm

Kies de optie 'Alarm' in het hoofdmenu. Het volgende scherm, met passend klokje linksboven, verschijnt :



Met de pijltjestoetsen OP en NEER kan de cursor door de lijst bewogen worden. Achter elk nummer (1 - 8) zal Free staan, wat betekent dat nog geen enkel alarm is ingesteld. Om een alarm in te stellen of te wijzigen, bijvoorbeeld het eerste, drukken we op EXE



Met LINKS en RECHTS kan de cursor naar de verschillende 'velden' (weekdag, uur, minuut, herhalingsritme) worden bewogen. De waarden van de velden kunnen worden gewijzigd met OP en NEER.

8.2.1 Herhalingsritme

Het herhalingsritme kan zijn :
Once: Het alarm gaat slechts éénmaal af, nl. op het ingestelde moment,

zoals in bovenstaand voorbeeld. Het zal hier afgaan eerstkomende vrijdag om kwart voor tien in de ochtend.

Hourly: Het alarm gaat elk uur af op de ingestelde minuut.

Daily: Het alarm gaat dagelijks af op de ingestelde tijd.

Wrkday: Het alarm gaat enkel af op werkdagen (maandag tot vrijdag) op de ingestelde tijd.

Weekly: Het alarm gaat wekelijks af op de ingestelde dag en tijd.

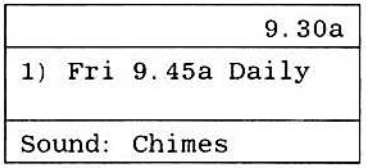
Als het herhalingsritme wordt gewijzigd in *Hourly* (herhaling elk uur), dan zullen de velden *weekdag* en *uur* worden opgevuld met streepjes (--). Bij *Daily* (dagelijks) en *Wrkday* (werkdagen) zal enkel de weekdag worden 'overstreept'.

8.2.3 Geluid

Het alarm wordt bevestigd met EXE of ontkend met ON. In het laatste geval wordt het terug als 'Free' gemarkeerd.

Indien het wordt bevestigd zal de LZ vragen welk geluid u bij het alarm wenst. Er zijn de volgende mogelijkheden :

Normal : Normaal alarm;
Siren : Een sirene van jewelste;
Chimes : Klokkenspel.

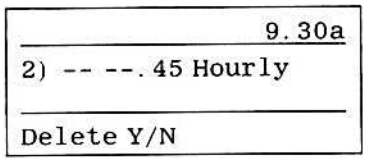


(In het oorspronkelijke ontwerp van de LZ — toen nog ZL geheten — waren eerst vier, dan zes verschillende alarmen gepland, maar in de uiteindelijke versie bleven er nog slechts drie over... Waarschijnlijk omdat deze geluidjes teveel geheugen kostten).

Kies het benodigde geluid met de cursortoetsen en druk op EXE. Het alarm is nu in het geheugen opgenomen en indien gewenst kan de optie Alarm nu verlaten worden met ON. Wanneer het alarm afgaat, wijst de LZ ons erop dat met ON dit lawaai kan worden afgezet. Wie het graag hoort, kan evenwel een minuut wachten, dan stopt het alarm vanzelf.

8.3 Het verwijderen van een alarm

Om een ingevoerde alarmfunctie weer te verwijderen, dienen we de cursor op het respectievelijke alarm te plaatsen en op DEL te drukken. Uiteraard moet deze keuze worden bevestigd :



Y om te bevestigen, N om te ontkennen.

9. Diary

9.1 Inleiding

De LZ bezit een interne agenda, waarin we onze afspraken kunnen noteren met optioneel alarm tot 59 minuten voor de afspraak. Zo hebben we tegelijk met een adressenboekje en rekenmachine ook steeds een agenda bij de hand!

De agenda heeft voor iedere dag — t.e.m. 31 december 2155 (!) — één bladzijde met een 'onbeperkt' aantal regels waarop afspraken kunnen worden genoteerd. Op iedere regel kunnen 64 tekens worden geschreven, en iedere regel kan een alarm bevatten.

Alle gegevens van de agenda worden opgeslagen in het interne geheugen van de LZ. Deze heeft daar een speciale ruimte voor voorzien, nl. zo'n 23K (LZ64)! Toch worden alle afspraken in een zo klein mogelijke ruimte opgeslagen. Dit betekent echter, dat de agenda kwetsbaar is : wanneer de batterij leeg is of verwijderd wordt, gaat ook de agenda verloren... Hij kan echter naar een pak worden weggeschreven om later te worden teruggelezen.

Zoals bij een gewone agenda met papier, blijven de reeds verlopen afspraken — bijvoorbeeld een afspraak van gisteren of vorige week — nog genoteerd. Men kan desgewenst dus ook terugkijken naar vorige afspraken. Heeft men daaraan geen behoefte, dan kunnen die oude afspraken uit de agenda worden verwijderd (zonder scheuren), zodat geheugenruimte vrijkomt.

9.2 Het weekpaneel

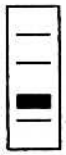
Het eerste dat we zien wanneer we Diary kiezen, is het *weekpaneel*.

```
Mo Tu We Th Fr Oct89
02 03 04 05 06 07 08
 =  =  =  =  =  =  =
 =  =  =  =  =  =  =
```

Dit weekpaneel biedt een globaal overzicht van de huidige week. De blokjes onderaan bevatten vier aanduidingen; *één aanduiding per tijdzone*. De LZ deelt de dag immers in in vier tijdzones, nl. :

Ochtend middernacht tot middag;
Lunchtijd middag (12.00u) tot 2 uur in de namiddag (14.00u);
Namiddag 14.00u tot 18.00u;
Avond 18.00 tot middernacht.

Wanneer een aanduiding vet staat aangegeven (zie onderstaand voorbeeld), bevat die tijdzone een afspraak. Het pijltje duidt de huidige tijdzone aan.



Ochtend
Lunchtijd
Namiddag
Avond

In het voorbeeld links is er een afspraak gepland in de namiddag, dus tussen 14.00u en 18.00u.

Dit overzicht is vanzelfsprekend niet gedetailleerd genoeg om als agenda te dienen; het is dan ook maar een *overzicht*.

9.3 Een afspraak noteren

Om een afspraak te noteren, moet het pijltje naar de desbetreffende tijdzone worden gestuurd OP, NEER, LINKS en RECHTS. Het is mogelijk 'over de rand van het scherm' te gaan om naar een andere week te 'sturen'. Als de juiste *tijdgleuf* gevonden is, drukken we op EXE. Eventuele afspraken van die dag komen dan op het scherm. Als er nog geen zijn, komt het volgende scherm tevoorschijn :

```
Thu 16 Nov 1989 Wk46

      <Free Day>
```

De LZ heeft gelijk; het is een vrije dag. Dat gaat echter veranderen... Want we hebben een afspraak met G. De Vader tussen 14.00u en 14.30u. We tikken in :

Ontmoeting G. De Vader

Het woordje *Edit* verschijnt vanzelf voor de tekst terwijl deze wordt ingetikt. Als de regel is ingetikt, drukken we op EXE. Dan is het tijd om te zeggen wanneer we de afspraak precies hebben. We passen de tijd aan met de pijltjestoetsen (net als bij *Time*) en beëindigen de instelling met EXE. De LZ stelt altijd een ontmoetingstijd van 15 minuten voor.

Daarna is het tijd om aan te geven of we willen dat de LZ ons op tijd waarschuwt voor de afspraak. We drukken Y voor waarschuwing of N voor

geen waarschuwing. Dan vraagt hij hoeveel minuten voor de afspraak hij de waarschuwing moet geven. Standaard is dit 15 minuten, maar met de pijltjestoetsen OP en NEER kan dit gewijzigd worden. Om te beëindigen drukken we vanzelfsprekend op EXE.

```
Thu 16 Nov 1989 Wk46
12:00p <Free>
 2:00p Ontmoeting G.
 2:30p <Free>
```

Inderdaad, vanaf 2.30p zijn we weer 'vrij'...

Om terug naar het weekpaneel te gaan, drukken we op ON. Om de volgende of de vorige afspraakdag te bekijken drukken we op RECHTS respectievelijk op LINKS.

9.4 Als een alarm afgaat

Als het agenda-alarm afgaat, wordt de ingetikte boodschap onderaan het scherm zichtbaar gemaakt. Als ze langer dan 20 tekens is (maximumlengte is 64 tekens), wordt ze *gescrolld*. Het alarm kan worden afgezet met ON.

Wanneer het alarm wordt afgezet — en we dus eigenlijk hebben gezegd dat we de boodschap gelezen hebben — zal de LZ de afspraak markeren als "gezien". Wanneer het alarm één minuut lang is afgegaan, neemt de LZ aan dat we het *niet* hebben gezien en markeert de ingave als "niet gezien". De volgende keer dat we de Organiser II aanschakelen, zal hij nakijken of er ongeziene ingaven in het geheugen zitten en zoja, dan zal hij ons er attent op maken.

```
   YOU HAVE MISSED
   66 DIARY ALARMS

Review? Y/N
```

Om de desbetreffende afspraken toch nog te bekijken, drukken we op Y (of N als we dat niet willen). Als er meer dan één afspraak te bekijken valt, zoals in het bovenstaande voorbeeld, dan drukken we ON om iedere andere boodschap op haar beurt te lezen.

9.5 Een ingave wijzigen

Om een ingave te wijzigen, sturen we gewoon het pijltje naar die ingave en drukken op EXE. We kunnen dan met de pijltjestoetsen door de tekst sturen en wijzigen waar nodig (uiteraard kunnen we wissen met DEL).

9.6 Een ingave wissen

Om een ingave gewoon te wissen, hoeven we alleen het pijltje naar de bewuste ingave te sturen en op DEL te drukken. De LZ vraagt om bevestiging. We drukken op Y om te wissen of op N om niet te wissen.

9.7 Het Diary-menu

Ja, ook *Diary* heeft een sub-menu. Het verschijnt wanneer we op MODE drukken (wat had u anders verwacht?) en bevat de volgende items :

Find	Zoek een string op in de agenda;
Goto	Ga naar een bepaalde datum;
Copy	
Cut	Deze drie items dienen om tekst te verplaatsen;
Paste	
Alarm	Stel alarmen in of wis ze;
Tidy	Vergeet 'oude' afspraken;
Print	Druk de agenda af op papier;
Save	Bewaar de agenda op data- of RAMpak;
Restore	Lees de agenda in van data- of RAMpak;
Xrestore	Lees een CM- of XP-agenda in van data- of RAMpak;
Setup	Pas de tijdgleuven aan naar behoeven.

9.7.1 Find

Find dient inderdaad om in de agenda naar een bepaalde tekst (string) te zoeken. Wanneer hadden we ook al weer een afspraak met G. de Vader? Om hoe laat moesten we naar Detectivebureau Goudvinger bellen? Zoek het op met Find.

```
Find:GOUD

Fri 17 Nov 1989 Wk46
 8:00a Bellen GOUDVI
```

Indien er meer ingaven zouden zijn die aan de specificatie beantwoorden kunnen we op EXE drukken om de volgende te zien, tot de boodschap NO MORE ENTRIES op het scherm komt. Eventueel zouden we tijdens deze zoektocht ook op DEL kunnen drukken om de huidig gepresenteerde ingave te wissen.

Als de juiste ingave bereikt is, drukken we op ON. Het pijltje duidt de ingave aan.

9.7.2 Goto

Goto dient om het pijltje snel naar een bepaalde dag te sturen. We drukken op Goto en krijgen de huidige datum te zien. Deze datum kan worden aangepast met de pijltjestoetsen, om die bepaalde interessante datum te bekomen. Zodra die op het scherm prijkt, drukken we op EXE.

9.7.3 Cut, Copy & Paste

Nee, dit is niét de naam van een advokatenkantoor of een fabrikant van waspoeder. Het zijn drie opties van het Diary-menu die ons in staat stellen bepaalde agenda-ingaven snel te verplaatsen en/of te herhalen. Stel dat we een afspraak hadden, zoals hierboven, met G. de Vader om 2 uur op donderdag 16 november 1989. De heer de Vader wenst ons ook de volgende dag te zien. We moeten dus dezelfde afspraak twee keer ingeven. Dat gaat als volgt in zijn werk :

1. Stuur het pijltje naar de afspraak.
2. Wanneer het pijltje de afspraak aanwijst (met uur en tekst), druk op MODE om het Diary-menu bovenaan het scherm te krijgen. Kies de optie *Cut*.
3. De afspraak is uit de agenda verwijderd maar wordt in een hiervoor speciaal voorzien geheugen onthouden (we noemen dit het *CCP*-geheugen : Cut, Copy & Paste).
4. Druk op MODE en kies *Copy*. De afspraak wordt nu terug op haar oude plaats neergeschreven, maar blijft ook in het CCP-geheugen.
4. Druk op RECHTS. We krijgen nu de agenda van de volgende dag te zien.
5. Druk op MODE en kies de optie *Paste*. Het CCP-geheugen wordt nu naar de huidige dag gecopiëerd.

Stappen 4 en 5 kunnen nu worden herhaald dat het een lieve lust is, tot het geheugen van de LZ nokvol afspraken met G. de Vader zit ...

9.7.4 Save

Save dient uiteraard om de (toch kwetsbare) gegevens van de agenda veilig weg te schrijven. Ikzelf gebruik altijd een korte naam, bijvoorbeeld *D*. Gebruik deze voorziening regelmatig : het is veiliger.

Er is de mogelijkheid om bijvoorbeeld een agenda-bestand D op te zetten dat regelmatig bijgewerkt wordt. Ik zeg *bijgewerkt* en niet *vervangen*, omdat we het zo kunnen regelen dat de oude ingaven van de agenda-copie bewaard worden en worden aangevuld met de meest recente ingaven. Anderszijds kunnen we opteren om de oude versie van D toch maar te vervangen door de nieuwe. Daarvoor dient het *Diary Save-menu*.

9.7.4.1 Het Diary Save-menu

biedt twee opties : *Append* en *Delete*. Append dient om de nieuwe ingaven bij de 'oude' copie te voegen, Delete zorgt ervoor dat alleen de huidige Diary onthouden wordt. Aan iedere gebruiker de keuze.

9.7.5 Restore

Restore is de tegenhanger van Save en dient om een vroeger geSavede Diary weer in te laden, bijvoorbeeld wanneer het interne geheugen is leeggemaakt (als de lege batterij lange tijd niet vervangen geweest is, bijvoorbeeld). Ook hier krijgen we met een sub-menu te maken :

9.7.5.1 Het Diary Restore-menu

biedt eveneens twee opties : *All* of *Partial*. Bij All laden we de héle geSavede Diary terug in, bij Partial alleen een te specifiëren gedeelte.

9.7.5.2 Het Diary Delete/Merge-menu

Nu verschijnt dit menu weer op het scherm. Wanneer *Delete* wordt gekozen wordt de eventuele inhoud van de Diary 100% vervangen door de copie die nu nog op data- of RAMpak staat. Wordt echter *Merge* gekozen dan zal de LZ de Diary samensmelten met de geSavede Diary. De Diary zal dan de afspraken die zij al bevatte behouden, alsook de afspraken die in de geSavede Diary genoteerd waren.

9.7.6 Xrestore

Xrestore is een Restore die dient om geSavede Diaries van de anderemodellen van de Organiser II, CM en XP, in te laden in de LZ. Het is echter niet mogelijk een *Diary Merge* uit te voeren (zie punt 9.7.5.2).

9.7.7 Tidy

Tidy ('ruim op') dient om de 'oude', dus vervallen, ingaven uit de agenda te verwijderen. Zodoende komt er geheugenruimte vrij die voor nieuwe afspraken kan gebruikt worden.

9.7.7 Setup

In punt 9.2 zagen we hoe de tijd in vier tijdzones verdeeld is. Het is echter mogelijk deze tijdzondes aan onze eigen individuele behoeften aan te passen. Het is eveneens mogelijk de steeds terugkomende vraag "Alarm? Y/N" te onderdrukken. Daarvoor dient Setup.

```
Slots change at:
12:00p 2:00p 6:00p

Alarm prompts: OFF
```

Met de pijltjestoetsen LINKS en RECHTS kunnen we iedere individuele tijdgleuf bereiken. Om ze te wijzigen drukken we op OP en NEER.

Om de alarm-vraag te onderdrukken wordt *Alarm prompts* op de onderste regel van het scherm in *OFF* veranderd met OP. De vraag Alarm? Y/N zal dan niet meer verschijnen : er worden geen alarmen voorzien (tenzij ze uitdrukkelijk met de optie Alarm, zie volgend punt, worden ingesteld).

9.7.8 Alarm

Deze optie lijkt erg op de Alarm van het hoofdmenu en dat is niet toevallig. Beide opties werken namelijk zo goed als identiek. Om een alarm in te stellen (stel dat, zoals in bovenstaand punt beschreven is, de mogelijkheid rechtstreeks een alarm in te stellen ongedaan gemaakt is), kiezen we een alarm uit, drukken op EXE en stellen we de tijd in. Daarna drukken we wederom op EXE en kiezen het passende geluid... Net zoals we gewend zijn!

9.7.9 Print

Print doet net wat je zou verwachten : het drukt de agenda af. De werking is zo voordehandliggend dat het echt onnodig is nog alle details te vermelden...

10. Month

Month is een ingebouwde maandkalender tussen 1900 en 2155. Met Month kunnen we te allen tijde een overzicht van de huidige maand (of andere maanden) op het scherm krijgen.

```
Mo Tu We Th Fr Nov89
-- -- 01 02 03 04 05
06 07 08 09 10 11 12
13 14 15 16 17 18 19
```

De cursor duidt de huidige dag aan. Met de pijltjestoetsen kan de kalender doorlopen worden. Door de cursor links of rechts 'buiten het scherm' te sturen kunnen andere maandkalenders bekeken worden.

Met ON gaan we terug naar het hoofdmenu.

Met EXE komen we in de Diary terecht op de datum aan de cursor.

11. Handigheidjes : Utils

11.1 Inleiding

Utils bevat een reeks handigheidjes, in het vakjargon *utilities* genaamd. Wat kunnen we met deze utilities doen? Het geluid aan-of uitschakelen, bijvoorbeeld, of nakijken hoeveel werkgeheugen er nog vrij is.

11.2 Het Utils-menu

Als we de Utils-optie uit het hoofdmenu kiezen, krijgen we het Utils-menu op het scherm.

```
+                    9.30a
Search  Info    Sound
Dir     Copy    Delete
Passw   Lang    Reset
```

Dit menu bevat zo'n 10 items, te weten :

Search	'Find' in alle mogelijke bestanden, inclusief Diary-, Notes- en OPL-bestanden (meer over OPL verder in dit boek), volgens specificatie in één keer;
Info	Geeft details over het geheugenverbruik;
Sound	Zet geluid aan/uit;
Dir	Directory van alle mogelijke bestanden, inclusief Diary-, Notes- en OPL-bestanden, volgens specificatie in één keer;
Copy	Copiëert alle mogelijke bestanden volgens specificatie in één keer;
Delete	Wist alle mogelijke bestaden volgens specificatie in één keer;
Passw	Beschermt de LZ met een wachtwoord;
Lang	Selecteert de taal van de LZ (Engels, Frans of Duits);
Reset	Wist het interne geheugen;
Format	Formatteert (wist) een RAMpak zodat de gehele capaciteit weer kan benut worden.

11.2.1 Search

Search werkt net als Find in de andere menu's. Men heeft het echter Search genoemd en niet Find om duidelijk aan te geven dat er een belangrijk

verschil is. Search zoekt namelijk in alle bestanden, dus niet alleen in MAIN. Er wordt gezocht in data files, Notes-, Diary-, en OPL-files (over dat laatste verder in dit boek meer).

Search zoekt naar een op te geven string (bijvoorbeeld "DR") in deze volgorde :

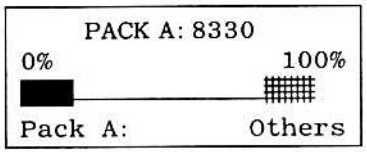
1. De huidige agenda;
2. Het huidige schrijfblok, tenzij dit met een wachtwoord is beschermd;
3. Alle andere schrijfblokken (in chronologische volg orde, dus het laatst aangepaste schrijfblok wordt het laatst doorzocht), behalve de eventuele schrijfblokken die met een wachtwoord zijn beschermd;
4. Gewone bestanden (*data files*) op pak A:, dan pak B:, en tenslotte pak C:;
5. OPL-bestanden op pak A:, B:, en C:.

De specificatie die wordt opgegeven, mag net zoals bij Find wildcards bevatten (* en +).
Wanneer Search een opgegeven string gevonden heeft, geeft de LZ de string niet alleen op het scherm, maar vertelt bovendien *waar* de string gevonden werd.

Wordt er dan op MODE gedrukt, dan gaat de LZ over in de zgn. 'edit'-staat. Dit betekent dat, wanneer de opgegeven string (bijvoorbeeld "DR") gevonden was in een bepaald schrijfblok, de LZ naar de Notes-optie springt, het desbetreffende schrijfblok actief maakt en zelfs in de edit-modus gaat. Dit schrijfblok kan dan meteen gelezen en/of aangepast worden. Dit is het geval met schrijfblokken (Notes), agendabestanden (Diary) en OPL-bestanden (Prog), maar *niet* bij databestanden (Xfiles). Bij deze laatste wordt er naar het Xfiles-menu gesprongen en wordt het databestand waarin de string werd gevonden actief gemaakt.

11.2.2 Info

Info is een zeer belangrijke utility. Het geeft namelijk aan hoeveel geheugen er verbruikt is, en hoeveel er nog overblijft. Dat gaat als volgt :



Op de bovenste regel zien we dat het hier gegevens over pak A: betreft, het interne geheugen dus. Er zijn 8330 bytes verbruikt in databestanden

(inclusief MAIN), OPL-bestanden, geSavede agendabestanden (Diary) en schrijfblokken (Notes).

Ieder zwart blokje (links, op de derde regel) geeft 5% geheugenverbruik aan. Deze 8330 bytes nemen dus zo'n 15% van het bereikbare interne geheugen in (op een LZ64).

Rechts, nog steeds op de derde regel, zien we 5 grijze blokjes. Dit betekent dat op pak A: zo'n 25% van het geheugen (5 blokjes die elk 5% van het geheugen voorstellen) wordt gebruikt voor andere zaken, zoals de Diary, de 8 alarmen, e.d.m.

Door op OP of NEER te drukken kan de pakletter veranderd worden, of kan specifiek geheugenverbruik van de Diary en de Notes opgevraagd worden.

11.2.3 Sound

Met Sound kan het geluid worden uit- of aangezet. Wanneer het geluid uit staat (kies *Off* in het Sound-menu) is de LZ zelfs stil wanneer er een alarm afgaat of wanneer we een toets indrukken.

11.2.4 Dir

Dir dient ook hier om een overzicht van alle bestanden te krijgen. Hier echter krijgen we een overzicht van verschillende soorten bestanden op aanvraag. Dit zijn de mogelijke bestandssoorten :

All	Alle soorten;
Files	Databestanden (Xfiles) of geSavede agenda's;
Notes	GeSavede schrijfblokken;
Opl	OPL-bestanden (programma's);
Comms	Protocolbestanden voor de communicatielink (zie verder in dit boek);
Plan	Bestanden van de Pocket Spreadsheet, een programma voor de Organiser II dat door Psion op de markt wordt gebracht;
Pager	Specificatiebestanden van de Pager, een nieuw randapparaat voor de Organiser II (waarover tijdens het te perse gaan nog geen gegevens beschikbaar waren);
Xdiry	Agendabestanden (*diary files*) gecreëerd met de CM of XP.

We kiezen de soort die ons interesseert (of All als ze ons allemaal interesseren). Daarna kunnen we (met MODE) opgeven op welke datapak we moeten zoeken (het is helaas niet mogelijk in één keer op te geven dat *alle* datapaks doorzocht moeten worden).

Wanneer de lijst dan zichtbaar is (druk eerst op NEER), kan de lijst met de pijltjestoetsen OP en NEER doorlopen worden.

11.2.5 Copy

Copy dient om bestanden te copiëren, dat spreekt voor zich. Ook hier vinden we de mogelijkheid om de bestandssoort te specifiëren terug.

```
      COPY FILES
From A:
```

Eerst moet worden opgegeven welk bestand gecopiëerd moet worden. We noemen dit bestand het *bronbestand*. De LZ geeft vanzelf de paknaam aan (*bronpak*), welke uiteraard kan gewijzigd worden met ON.

Om een bestand te copiëren waarvan we de naam weten, volstaat het de naam van het bestand *volledig* in te tikken en op EXE te drukken (wildcards zijn mogelijk, bijvoorbeeld * voor alle bestanden of C* voor alle bestanden waarvan de naam met C begint).

Als we niet weten hoe het bestand heet, drukken we op EXE en vervolgens op NEER. Dan verschijnt er een lijst op het scherm die doorlopen kan worden met OP en NEER, net als bij Dir. Om een bepaald bestand dan te copiëren moeten we eenvoudig op EXE drukken.

Daarna moet worden opgegeven waarnaar het (de) bestand(en) moet(en) worden gecopiëerd.

```
      COPY FILES
From A:C*
To   C:
```

Indien het (de) bestand(en) moet(en) worden gecopiëerd *onder dezelfde naam*, moet enkel de doelpak worden gespecifiëerd met MODE.

Indien er slechts één bestand moet worden gecopiëerd, en zulks onder een andere naam, kan die naam achter de *To*-prompt worden ingetikt.

Druk EXE om het copiëren te beginnen.

OPMERKINGEN :

1) Het is niet mogelijk een reeks bestanden te copiëren en voor elk bestand

een andere naam op te geven, dat moet dus in stappen gebeuren.
2) Bestanden moeten altijd naar een andere pak worden gecopiëerd. Het is dus niet mogelijk bijvoorbeeld van pak A: naar pak A: te copiëren, zelfs indien een andere naam wordt opgegeven.
3) Als een doelbestand al bestaat, wordt het niet overschreven (tenzij het een OPL-bestand is). De records van het bronbestand worden achteraan het doelbestand *bijgevoegd*. Om A:MAIN dus naar B:MAIN te copiëren, moet B:MAIN eerst gewist worden. Pas dan kunnen we er zeker van zijn dat B:MAIN een exacte copie is van A:MAIN (in het jargon noemen we dit *image copy*).

11.2.6 Delete

Delete dient om bestanden te wissen. Ook hier krijgen we te maken met de lijst van bestandssoorten. De selectie van de te wissen bestanden gebeurt net hetzelfde als bij Copy.

11.2.7 Passw

Een zeer interessante optie is wel Passw, ofwel *Password* (wachtwoord). Met deze optie kunnen we vermijden dat de één of andere pottekijker gegevens in de LZ gaat bekijken en/of wijzigen. Merk wel op dat alléén de LZ zèlf beschermd wordt, en niet de data- of RAMpaks...

De procedure is als volgt. Eerst bedenken we een niet-triviaal wachtwoord (dus bijvoorbeeld niet *geheim*, *Organiser* of *wachtwoord*). Dan geven we het wachtwoord in via Utils Passw. Het wachtwoord, dat niet langer mag zijn dan 8 tekens, moet twee keer worden ingetikt, om typefouten zoveel mogelijk te voorkomen. Het wachtwoord verschijnt niet op het scherm terwijl het wordt ingetikt (te gevaarlijk!!), in de plaats verschijnen *railway crossings* (#).

Dan verschijnt er het Password-Menu (On Off). Kies On, om de LZ te vertellen dat het wachtwoord *actief* is. Schakel nu de LZ uit en vervolgens weer aan... Onmiddellijk vraagt deze het juiste wachtwoord in te tikken, of we krijgen geen toegang tot het hoofdmenu.

Om het wachtwoord te veranderen, eist de LZ dat ook het oude wachtwoord wordt ingetikt, voordat het nieuwe wordt opgegeven.

OPMERKINGEN

1) Pas op met deze optie; wanneer het wachtwoord vergeten wordt, moet de batterij verwijderd worden om weer toegang de krijgen tot de LZ. Dat houdt echter in dat het interne geheugen gewist wordt.
2) Gebruik geen voordehandliggende wachtwoorden. Uw naam, geboortedatum, de naam van kind, echtgeno(o)t(e), stad of streek zijn dus strikt uit den boze. Dat zijn namelijk de eerste wachtwoorden die een indringer zal uitproberen.
3) Gebruik voor de Notes-wachtwoorden niet hetzelfde wachtwoord als hier.
4) Wijzig het wachtwoord regelmatig.

11.2.8 Lang

Hier kunnen we de taal van de LZ kiezen (Engels, Frans of Duits), net alsof we hem voor het eerst aanschakelen.

11.2.9 Reset

Deze gevaarlijke optie wist het gehele interne geheugen, inclusief alarmen, Diary, Notes, ... Gelukkig vraagt de Organiser II voor bevestiging :

```
DO YOU REALLY WANT
TO DELETE ALL DATA
──────────────────
Confirm? Y/N
```

11.2.10 Format

Deze optie dient om een RAMpak te formatteren (wissen), zodat de volledige capaciteit van de pak voor nieuwe zaken kan gebruikt worden. Merk op dat alleen RAMpaks kunnen worden geformatteerd met deze optie. Voor datapaks is er de Psion Formatter, een speciaal randapparaat dat de datapaks bestraalt met ultraviolet licht en zo de geheugenruimte weer vrij maakt.

Deel IV

PROGRAMMEREN MET CM, XP EN LZ

1. Programmeren : een inleiding

PROG – afkomstig van 'programming' (= programmeren) – is de optie die ons toestaat programma's te schrijven, veranderen, bekijken en uit te voeren.

Maar wat is nu precies een programma ? Een programma is in feite de computervorm van een zekere 'opdracht', die is onderverdeeld in een reeks kleinere opdrachten (instructies). Hoe dat precies in zijn werk gaat, zien we later.

Het gaat erom een bepaalde taak aan de Organiser II op te leggen. Zo'n taak kan bijvoorbeeld het bijhouden van een klantenlijst zijn of het afdrukken op printer van alle gegevens in pak B:, of een eenvoudig spelletje bijvoorbeeld. Maar er zijn ook heel ingewikkelde taken te bedenken : het bijhouden en bewerken van een elektronisch werkblad (= een soort programmeerbare tabel), tekstverwerking met inbegrip van spellingscontrole en eventueel -correctie, of het oplossen van wiskundige vraagstukken. Het kan allemaal.

Om de Organiser II – of welke andere computer dan ook – een taak op te leggen, moet deze taak eerst grondig worden geanalyseerd : wat moet er precies gebeuren? De taak wordt, eerst in gewone 'mensentaal', opgesplitst in kleinere stappen, tot men een homogeen geheel bekomt dat zo duidelijk beschreven is en waarvan iedere stap zo elementair is, dat het naar computertaal kan worden omgezet. Deze gedetailleerde omschrijving noemt men het *algoritme* – de weg die moet worden gevolgd voor het behalen van het juiste resultaat.

Als dit alles u wat moeilijk voorkomt, geen nood, een voorbeeld maakt alles duidelijker.

Stel, we hebben een eenvoudige opdracht : de computer moet tellen van 1 tot en met 10.

Hier stuiten we onmiddellijk op een probleem, dat we telkens wanneer we een algoritme willen ontwerpen zullen tegenkomen : hoe gaan we dit precies formuleren? We beginnen bij de eerste, eenvoudigste stap :

```
TEL VAN 1 TOT 10.
(Stap 1)
```

Dit is niet precies, niet elementair genoeg. Er is geen enkele manier om deze wens in één opdracht aan de computer duidelijk te maken. Dus we

gaan over tot stap 2 : de eerste 'verfijning' wordt ingevoerd.

```
BEGIN BIJ 1
TEL STEEDS 1 BIJ, TOT JE BIJ 10 KOMT.
(Stap 2)
```

Dit is al veel preciezer geformuleerd, bijna alsof we een kind zouden willen leren tellen. Daar komt het in feite ook op neer : we moeten de computer ieder klein detail van zijn opdracht vertellen, telkens opnieuw, wil hij deze uitvoeren. Een computer is dus heel dom en niet intelligent, zoals vele mensen denken : zonder door de mens gemaakte programma's kan hij *niets*.

Maar deze verfijning is nog niet genoeg. 'BEGIN BIJ 1' en 'TEL 1 BIJ' is niet te begrijpen voor een machine. Het kan ook niet geformuleerd worden in computertaal. Computers 'denken' nu eenmaal anders dan wij gewend zijn te doen. De oplossing vinden we in de moderne wiskunde.

In de moderne wiskunde worstelt men vaak met onbekende of veranderlijke getallen. Deze getallen krijgen dan een symbolische voorstelling (bijvoorbeeld x of y).

Om een vierkantswortel van een onbekend getal te formuleren, zal men dan bijvoorbeeld de notatie

$$\sqrt{x}$$

gebruiken.

Deze getallen, x, y, enzovoort, noemt men *variabelen*. Het zijn kunstmatige of algebraïsche voorstellingen. We passen ze ook toe in computertaal en in het ontwerpen van onze algoritmen :

```
x = 1
ZOLANG x niet gelijk is aan 10
       x = x + 1
(Stap 3)
```

Het valt op, dat de leesbaarheid bij iedere stap achteruitgaat. We gaan steeds een stap verder weg van de mens en een stap dichter naar de computer.

In stap 3 zijn bijna alle problemen opgelost. We hebben een variabele die het tellen bijhoudt (zo'n variabele wordt overigens 'teller' genoemd), en de verhoging van het getal is ook reeds goed geformuleerd (x = x + 1). Merk op dat we niet schrijven :


```
x = + 1
```

zoals we in onze alledaagse taal zouden doen. In dat geval zou x namelijk gewoon de waarde 1 krijgen. We zeggen in plaats daarvan : neem de waarde van x, voeg daar 1 bij, en plaats die waarde terug in x. Later zullen we ondervinden dat deze regel van exacte formulering opgaat bij *alle* handelingen met de computer. Precisie is een absolute vereiste. Heeft men hier eenmaal wat inzicht in verworven, dan wordt het vanzelfsprekend en merkt men het verschil met de gewone omgangstaal bijna niet eens meer op.

Nu gaan we over tot stap 4 : we gaan ons algoritme omzetten naar een computertaal. Er zijn heel veel verschillende computer- of programmeertalen, waarvan de meesten zijn ontwikkeld voor mini- en mainframe- (heel grote) computers : talen als BASIC (een algemene taal voor beginners), COBOL (zakelijke toepassingen), PASCAL (academisch gebruik), FORTRAN (wiskunde), dBASE (gegevensbestanden en -verwerking), C (besturingssystemen). De taal die wij gaan gebruiken is OPL, wat staat voor *Organiser II Programming Language* (Progammeertaal voor de Organiser II). OPL is dus speciaal voor deze zakcomputer ontwikkeld en is een hybride van BASIC, PASCAL en dBASE. OPL is krachtig en gemakkelijk te leren. Laten we ons algoritme eens bekijken zoals het in OPL wordt geformuleerd :

```
TELOP:
LOCAL X%
X% = 1
WHILE X% <> 10
     X% = X% + 1
ENDWH
(Stap 4)
```

Stap 4 ziet er als Chinees uit. Niet erg, het wordt snel duidelijk.

Zoals we zien komen er een aantal Engelse woorden in voor. Engels is immers de voertaal van de informatica (zo is de term 'computer' het Engelse woord voor 'berekenaar').

De eerste regel, `TELOP:` , is de *naam* van het programma (in de informatica krijgt alles een naam). Deze naam is later van belang om het programma te kunnen identificeren. De keuze van naam is vrij (het hoeft dus geen Engels zijn), maar hij moet beginnen met een letter en mag maximaal 8 tekens lang zijn. De naam wordt door de Organiser II altijd beëindigd met een dubbele punt (:).

Vervolgens : `LOCAL X%`. Dit heeft wat toelichting nodig. LOCAL betekent 'lokaal' of 'plaatselijk' : het duidt erop dat de variabele X% (over de %

dadelijk meer) in het volgende stukje programma wordt gebruikt. Alle gebruikte variabelen (de keuze van naam is vrij) moeten we immers steeds vooraan in onze programma's 'aangeven' (dus vóór gebruik).

Het feit dat we de variabele X% noemen en niet X komt doordat deze variabele enkel *gehele waarden* heeft (dus géén decimale, zoals bijvoorbeeld 3.1415). We zeggen dat X% een *integere* (= gehele) variabele is. Zonder het percentageteken zou het een reële (*real*) variabele zijn en mag wèl een punt bevatten. Om technische redenen neemt een reëel getal echter meer geheugenruimte in beslag en kan de computer er minder snel mee rekenen. Later komen we nog terug op gehele/reële getallen.

Dan : `X% = 1`. Dit heet *initialisatie* : de variabele X% krijgt zijn eerste (= initiële) waarde toegekend, in dit geval 1. Later wordt X% nog verhoogd (dit is dus géén initialisatie meer).

De volgende regel is `WHILE X% <> 10`. (<> is de OPL- notering voor *niet gelijk aan*). Dit is een zogenaamde *conditie* : het volgende stukje programma mag enkel worden uitgevoerd zolang aan de conditie is voldaan, dus ZOLANG (WHILE) X% niet gelijk aan 10.

Op de volgende regel wordt X% verhoogd : `X% = X% + 1`. Dit spreekt voor zich. Als we X% namelijk nergens zouden verhogen, zou het programma eeuwig blijven lopen omdat aan bovenstaande conditie steeds wordt voldaan en dit stukje programma dus nooit wordt verlaten. Hier nog een opmerking : deze regel is ingesprongen. Dit heeft echter enkel te maken met de leesbaarheid van het programma; het is dus niet *nodig* dit te doen. Maar omdat deze regel enkel wordt uitgevoerd als aan de WHILE-conditie voldaan is, gaat de leesbaarheid er met het inspringen op vooruit.

Op de volgende regel staat weer iets nieuws : `ENDWH`. Dit is een samentrekking van de woorden END en WHILE, wat zoveel betekent als 'einde van de WHILE-conditie'. Dit betekent dat de Organiser II 'terugspringt' naar de vorige WHILE-conditie en deze opnieuw test. Als aan die conditie niet meer wordt voldaan (dus in dit geval wanneer X% = 10) 'springt' de Organiser II naar de regel *die op de ENDWH volgt* (er zijn in ons voorbeeld geen regels meer, zodat het programma dan wordt beëindigd). Ons programma voert de lijn 'X% = X% + 1' dus 9 keer uit (tot de 10 is bereikt) en dan stopt het programma. Het werkt !

Wanneer we dit korte programma door de Organiser II zouden laten uitvoeren, zouden we niet veel merken van alles wat er binnenin plaatsvindt. Dit komt doordat het tellen bijzonder snel gaat, veel sneller dan een mens kan tellen, en omdat de machine op geen enkele wijze laat zien dat hij aan het tellen is. We zouden in ons programma bijvoorbeeld kunnen invoegen dat ieder getal op het scherm moet worden weergegeven. De lijn


```
PRINT X%
```

(= druk X% af op het scherm) wordt dan ingevoegd en het programma ziet er als volgt uit :

```
TELOP:
LOCAL X%
X% = 1
WHILE X% <> 10
      X% = X% + 1
      PRINT X%
ENDWH
```

(Stap 5)

Bekijk het programma goed en probeer de werking zo goed mogelijk te begrijpen. Wie goed kijkt en begrijpt, merkt een foutje op !

Om de fout op te sporen doen we alsof we zelf de computer zijn en proberen het programma regel voor regel uit te voeren... Waar zit de fout?

Alleen de getallen 2 tot en met 10, zonder de 1, worden op het scherm gebracht omdat X% reeds wordt verhoogd nog vóór er wordt gePRINT. Hoe kunnen we deze denkfout oplossen ? Door de PRINT-instructie te verwisselen met de verhoging ?

Helaas niet, want in dat geval wordt er, zo schijnt het, slechts tot 9 geteld (waarom ?).

Het eenvoudigste dat we kunnen doen is X% initialiseren op 0 in plaats van op 1. Het programma werkt dan feilloos (hoewel er éigenlijk van 0 tot 10 wordt geteld).

Zoals we zien moeten we, tijdens het programmeren, oppassen voor de valkuilen van onze eigen denkwijze. Als een programma niet werkt, is dat niet de fout van de computer, maar van onszelf, omdat we ons algoritme niet foutloos hebben geformuleerd. Het helpt niet om kwaad te zijn op de computer : die voert enkel zijn orders uit en wanneer die orders niet juist worden geformuleerd, kunnen ze ook niet juist worden uitgevoerd. Dit verschijnsel noemt men GIGO (garbage in, garbage out : als er rommel in gaat komt er weer rommel uit).

In dit hoofdstuk hebben we de beginselen van het programmeren uitgelegd. Wanneer iets niet duidelijk is, aarzel dan niet de moeilijke stof nog eens door te nemen.

2. Over OPL

OPL, de Organiser II-programmeertaal, is wat men noemt een *procedure-gerichte* taal : ieder programma kan worden onderverdeeld in kleine, aparte mini-programmaatjes, die elk een eigen functie vervullen, een eigen naam hebben en afzonderlijk worden geprogrammeerd en op pak gezet. Zo'n programmaatje heet een *procedure* (meestal afgekort tot *proc*). Men kan bijvoorbeeld een procedure TEL schrijven die van 1 tot x telt. Maar, deze programmaatjes kunnen ook *functies* zijn, die een bepaalde waarde berekenen; we kunnen bijvoorbeeld een functie WORTEL3 schrijven die de derdemachtswortel van een bekend getal X berekent. OPL is dus proceduraal en vanwege de mogelijkheid tot apart opslaan van de procedures, *modulair* : de programma's kunnen worden opgebouwd uit procedures die ergens op een pak staan, los van de andere procedures (of *modules*, zoals men ze noemt). Het is bijvoorbeeld mogelijk een bepaalde procedure in verschillende programma's te gebruiken. Vaak bestaat een programma uit één hoofdprocedure die meerdere kleinere procedures oproept. Dit geeft een grotere flexibiliteit bij het programmeren en het is misschien leuk om te weten dat de Zwitserse wiskundige en programmeur *Niklaus Wirth* in de jaren '70 de door hem ontwikkelde academische programmeertaal PASCAL herontwierp (hij voegde een modulaire structuur bij de taal). Zijn nieuwe geesteskind heet MODULA-2.

Er zijn in OPL 3 verschillende gegevenstypes :

a) *integer* : gehele getallen tussen -32768 en +32767.
Voorbeelden : 25, 3, -19000

b) *reëel* : reële getallen tussen -1e100 en (1e100)-1
met een precisie van 12 cijfers.
Voorbeelden : 3.14159265, -200000, .5

c) *tekst* : (= 'string') een streng van letters, cijfers en leestekens omgeven door aanhalingstekens.
Voorbeelden : "Organiser II !!", "250".

OPL kent, naast numerieke functies als sinus, cosinus, en dergelijke, dus ook functies om tekst te bewerken. Deze functies kunnen bijvoorbeeld zijn: de eerste/laatste 3 letters van een zin, de plaats van de eerste spatie in een zin, e.d. Bovendien kan men in OPL met bestanden werken. Voor wie al eerder met bestanden heeft geprogrammeerd in andere talen : de organisatie van bestanden in OPL is een slim bedachte mengelmoes van directe en sequentiële toegang. De twee vormen kunnen zelfs in hetzelfde programma op hetzelfde bestand gemengd worden gebruikt.

3. PROG : (CM, XP)

3.1 Inleiding

PROG is de optie die ons de mogelijkheid geeft de Organiser II te programmeren. Dit programmeren gebeurt in een zgn. '*programmeeromgeving*' : een geheel van bij elkaar horende functies die tot doel hebben programmeren zo efficiënt mogelijk te maken. Programmeeromgevingen komen vooral voor bij grotere machines zoals PC's, mini- en mainframecomputers.

3.2 Het PROG-menu

Net zoals DIARY, heeft PROG een eigen menu (submenu). Dit menu bevat alle benodigdheden (Eng: tools) om programma's te kunnen schrijven, aanpassen (editeren), verwijderen, afdrukken, copiëren en natuurlijk uitvoeren. Deze opties zijn EDIT, LIST, DIR, NEW, RUN, ERASE en COPY.

EDIT ('pas aan') : dient om een reeds bestaande procedure te bekijken en/of aan te passen (veranderen). Dit aanpassen behelst programmacorrecties, uitbreidingen, verkortingen of algemene verbeteringen. Het wordt *niet* gebruikt om nieuwe procedures te creëren.

LIST ('lijsten') : dient om procedures op de aangesloten printer (of ander randapparaat) af te drukken.

DIR ('inhoudsopgave') : dient om de namen van de reeds bestaande procedures op het scherm te brengen. Deze worden één voor één getoond. Door een druk op EXE wordt steeds de volgende procedurenaam op het scherm gebracht. Stoppen gebeurt met ON.

NEW ('nieuw') : dient om nieuwe procedures te creëren. Er dient een naam te worden opgegeven. Deze mag maximaal 8 tekens bevatten en moet beginnen met een letter. Geldige namen zijn bijvoorbeeld TELOP, MACHINE, BOEKHOUD. Ongeldige namen zijn bijvoorbeeld 123, *STER*, BOEKHOUDING. Vervolgens wordt de procedure ingetikt, wat precies hetzelfde werkt als EDIT.

RUN ('voer uit') : voert een al bestaande procedure uit. Ook hier moet uiteraard de naam van de betreffende procedure opgegeven worden (en op welk pak deze staat).

ERASE ('wis') : wist een procedure van pak; de naam moet worden opgegeven.

COPY ('copiëer') : copiëert procedure(s) van één pak naar een ander.

De precieze werking van iedere optie wordt uiteraard nader beschreven.

3.3 Een nieuwe procedure creëren

We gaan eens kijken hoe we een procedure kunnen creëren, editeren en uitvoeren. We gaan uit van het programma dat we zonet schreven. De eerste versie zag er als volgt uit :

```
TELOP:
LOCAL X%
X% = 1
WHILE X% <> 10
    X% = X% + 1
ENDWH
```

De eerste regel, zo was gezegd, bevatte slechts de naam van de procedure gevolgd door een dubbele punt (:). We gaan de procedure in deze vorm intikken.

We kiezen PROG, en NEW in het submenu.

```
NEW A:
```

Door nu één of twee keer op MODE te drukken kan men de pakaanduiding veranderen naar B:, respectievelijk C:.

We tikken vervolgens de naam in : T E L O P EXE (de dubbele punt is niet nodig). Het scherm klaart op, op de bovenste regel verschijnt de naam die we zojuist hebben ingetikt :

```
TELOP:
```

Nu verwacht de Organiser II het programma. Toets de eerste regel in :

```
LOCAL X%
```

(de % wordt verkregen met SHIFT-E). Druk op EXE en de cursor is naar de volgende regel verhuisd, wachtend op meer invoer van het klavier. Wanneer een foutje wordt ingetikt kan dit met DEL worden verwijderd. Door de cursor LINKS of RECHTS te sturen kan de cursor in die richtingen

worden bewogen. Het is eveneens mogelijk naar vorige lijnen te sturen met OP en weer terug te gaan met NEER.

Tik nu de rest van het programma in, en druk dan op MODE.

Op zijn beurt komt de optie NEW nu in een submenu terecht. Dit bevat de opties TRAN, SAVE en QUIT.

3.3.1 TRAN

TRAN is de afkorting voor *translate* (vertaal). Het is namelijk zo, dat alle programma's die geschreven zijn in OPL eerst worden vertaald naar een andere taal, de zogenaamde *tussencode*. Die tussencode is heel compact en gemakkelijker uit te voeren dan rechtstreekse OPL. Wat de Organiser II hier dus doet is de procedures geschreven in OPL omzetten naar tussencode, die later wordt uitgevoerd i.p.v. de eigenlijke OPL-code. De oorspronkelijke procedures in OPL blijven echter gespaard op pak, zodat men ze later kan aanpassen als dat nodig mocht zijn.

Wanneer er tijdens het intikken van de procedure toevallig een foutje is ingeslopen (X" in plaats van X% bijvoorbeeld) zal TRAN protesteren:

```
SYNTAX ERR
press space key
```

De Organiser II rapporteert een *syntaxisfout* (de X" is onwettig in de formele definitie van OPL en wordt dus niet herkend of geaccepteerd). De correcte Engelse term is trouwens *syntax error* en niet *syntax err*, zoals de heren van Psion het hebben geprogrammeerd (er is in de informatica een sterke tendens om alles af te korten).

Wanneer in dit geval SPACE wordt ingedrukt, springt de Organiser II naar de EDIT-optie (de *editor*, zoals men zegt) en wijst de fout aan zodat die onmiddellijk kan worden verwijderd.

TRAN neemt niet veel tijd in beslag (afhankelijk van de lengte van de betreffende procedures uiteraard) en springt dan automatisch naar :

3.3.2 SAVE

SAVE dient natuurlijk om procedures naar pak weg te schrijven. Ook hier kan met MODE de doelpak worden gewijzigd. Zowel de oorspronkelijke *listing* (= programma) geschreven in OPL (dit heet de zgn. '*source code*' of broncode) als de vertaalde tussencode ('*object code*') worden weggeschreven. Afhankelijk van het soort pak dat men gebruikt (datapak, RAMpak, intern geheugen) en de reeds ingenomen ruimte (weinig opslag,

bijna vol) neemt SAVE veel of weinig tijd in beslag. Wie géén SAVE wil (druk op ON) verliest zijn procedure – zowel de broncode als de object code.

3.3.3 QUIT

Opgeven : wordt gebruikt wanneer men de ingetikte procedure niét wil bewaren. Om veiligheidsredenen informeert de Organiser II nog eens of de gebruiker wel zeker is. Als dit inderdaad het geval is dan keert de Organiser II onmiddellijk terug naar het PROGmenu.

Het is gebruikelijk om na het creëren van een procedure direct TRAN te gebruiken en vervolgens SAVE. De procedure staat dan op pak, klaar om indien nodig geëditeerd te worden of om uitgevoerd te worden. We voeren eerst uit.

3.4 RUN : een procedure uitvoeren

We selecteren de optie RUN en warempel, de naam TELOP staat reeds ingevuld :

```
RUN A: TELOP
```

Dit komt doordat de Organiser II steeds de laatstgebruikte programmanaam onthoudt, of die naam nu is gebruikt bij het editeren, creëren of uitvoeren. We hoeven daarom dus enkel op EXE te drukken om TELOP uit te voeren.

Als er een andere naam staat dan die van de procedure die moet worden uitgevoerd, ga dan als volgt te werk : druk eerst op NEER. De cursor staat dan achteraan de procedurenaam. Druk vervolgens op DEL tot de naam volledig verwijderd is, tik de correcte naam in en druk op EXE.

3.5 EDIT : een procedure editeren

Zoals in hoofdstuk 14 reeds voorspeld, is de uitvoering van procedure TELOP bijna onmerkbaar voorbijgegaan. Nu gaan we de tweede, juiste versie intikken. We hoeven echter de volledige listing niet meer opnieuw over te tikken : we kunnen de reeds bestaande broncode gaan editeren.


```
TELOP:
LOCAL X%
X% = 0
WHILE X% <> 10
     X% = X% + 1
     PRINT X%
ENDWH
```

Zó moet het er gaan uitzien. Welke regels moeten worden veranderd ? In de eerste plaats : in de derde regel moet X% = 1 veranderd worden in X% = 0.

We kiezen EDIT in het menu (en de naam staat er al). We gaan met de cursor naar de *vierde* regel i.p.v. de derde. De cursor staat dan op de W van WHILE.

Nu drukken we op LINKS. Omdat er links helemaal niets staat (dat kan ook niet, want we staan uiterst links op het scherm) gaat de cursor naar het einde van de vorige regel, dus naar de 1. Het is nu een koud kunstje om de 1 in een 0 te veranderen.

Vervolgens moet de PRINT X%-instructie worden ingevoerd. We sturen de cursor 3 keer naar beneden, tot hij bij ENDWH staat. We drukken een paar keer op SPACE (ENDWH verhuist mee) tot we precies onder de vorige instructie staan, en tikken dan de nieuwe regel in, gevolgd door EXE. Alles is nu geregeld.

De procedure is nu zoals ze moet zijn, en we drukken op MODE. Opnieuw komen we in het TRAN/SAVE/QUIT-menu terecht. De procedure moet immers terug vertaald worden naar de tussencode.

Nu voeren we TELOP weer uit. Als alles goed verlopen is (geen regels of woorden vergeten) dan zien we op het scherm inderdaad tellen van 1 tot 10. We hebben nu ook een idee van de snelheid van OPL (of de tussencode, beter gezegd). Het uitvoeren van een correct programma is altijd het leukste punt van programmeren : de voldoening dat het werkt is onbeschrijfbaar.

3.6 LIST : een procedure afdrukken

Om een procedure te kunnen afdrukken, is natuurlijk een printer vereist. Die printer wordt aangesloten aan de Organiser II met de RS-232C- communicatielink (zie verderop in dit boek).

De naam van de procedure wordt opgegeven (het pak kan ook hier wor-

den veranderd met MODE). Als er geen printer is aangesloten zal de Organiser II ons hierop attent maken :

```
DEVICE MISSING
press space key
```

('Geen apparaat. Druk op SPACE').

3.7 ERASE : een procedure wissen

Een procedure die we niet meer nodig hebben kunnen we van pak wissen. Dit gebeurt met de PROG-optie ERASE. Zoals steeds moet eerst de procedurenaam worden opgegeven en vraagt de Organiser II voor de zekerheid of de gebruiker deze procedure inderdaad kwijt wil. Het gebruik van de toetsen Y en N wordt ook hier toegepast.

3.8 COPY : procedure(s) copiëren

Met COPY is het mogelijk één procedure, of alle procedures die op het pak staan ineens, te copiëren naar een andere pak. Men kan er ook voor kiezen alléén de object code (tussencode) te copiëren. Dit kan nuttig zijn in een geval dat men bijvoorbeeld een zelfgemaakte procedure op de pak van een vriend copiëert maar de broncode niet wenst prijs te geven, of ook wanneer men plaats wil besparen. Wanneer enkel de object code wordt gecopiëerd kan de copie niet worden geëditeerd.
De Organiser II zal vragen :

```
COPY
OBJECT ONLY (Y/N)
```

('Moet enkel de object code worden gecopiëerd ?')

Is dit het geval, druk dan op Y, zoniet, druk dan N.

Dan volgt de vraag :

```
FROM
```

De Organiser II wil nu weten wat de *bron* is : vanaf welke pak moet worden gecopiëerd en welke procedure(s). Als men alleen het pak opgeeft (bijvoorbeeld A:), worden alle procedures van het pak in één keer meegecopiëerd.

```
TO
```


Nu moet worden opgegeven wat de *bestemming* is : naar welke pak moet worden geschreven ? Men kan ook een procedurenaam opgeven : stel dat we op pak B: een procedure TELOP hebben die we willen copiëren naar pak C: onder de naam NIEUW. We geven dan bij FROM de naam 'B:TELOP' op en bij TO de naam 'C:NIEUW'. De procedure wordt dan naar C: gecopiëerd onder de nieuwe naam.
Wanneer men een naam opgeeft bij TO en die procedure bestaat al, dan wordt de oude versie verwijderd en vervangen door de nieuwe.

4. Prog (LZ)

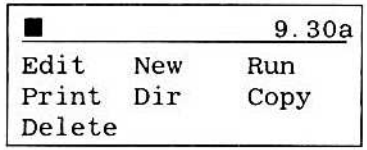
4.1 Inleiding

Ook bij de LZ is er het menu-item Prog dat dient om programma's geschreven in de taal OPL (Organiser Programming Language) te maken, te wijzigen, uit te voeren, te copiëren, enzovoort. De taal OPL is speciaal voor de Organiser II ontworpen en voor de LZ nog uitgebreid t.o.v. de modellen CM en XP. Dat kon ook niet anders, omdat de LZ o.a. een groter scherm heeft.

Bij het selecteren van Prog krijgen we... het Prog-menu.

4.2 Het Prog-menu

Het Prog-menu ziet er als volgt uit :



De zeven opties van dit menu zijn :

Edit	Aanpassen van vroeger aangemaakte OPL- programma's (*procedures*);
New	Creëren nieuw programma;
Run	Uitvoeren programma;
Print	Afdrukken programma (*listing*) op printer;
Dir	Directory van bestaande programma's;
Copy	Copiëren programma('s);
Delete	Wissen programma('s).

4.2.1 Edit

Edit is het eerste menu-item. Dat lijkt gek, want het dient om *reeds bestaande* programma's *aan te passen*. De optie om nieuwe programma's te maken is New (zie punt 4.2.2). Om de volgorde van het menu te respecteren, wordt Edit eerst besproken. Voel u echter vrij om eerst over New te lezen.

Bij Edit hoort een zgn. *editor*, dat is een soort afdeling van de Organiser II die dient om de programma's, die in de Engels-achtige taal OPL geschreven zijn, te bekijken en aan te passen. Met de editor zullen we alle programma's intikken (met New) en waar nodig wijzigen (met Edit). Beide opties gebruiken dus de Editor. We lezen daarom bij New hoe de editor precies werkt.
De prompt "Edit A:" komt op het scherm. We moeten dus een programmanaam intikken. Ieder programma heeft immers een naam, net zoals een schrijfblok of een datafile. Die naam mag 8 tekens bevatten en moet beginnen met een letter. Er mogen geen spaties of leestekens in voorkomen.

Als die naam nog niet bestaat, zal de LZ protesteren, dat is begrijpelijk. Als het programma echter wel bestaat, zal de LZ het in het werkgeheugen overbrengen (*inladen*) en komen we vanzelf in de editor terecht.

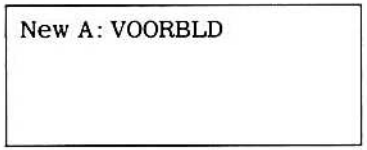
4.2.2 New

New is de optie die we gebruiken om nieuwe programma's aan te maken, zoals het onderstaande voorbeeld :

```
VOORBLD:
LOCAL A,B
CLS
PRINT "GETAL A : ",
INPUT A
PRINT "GETAL B : ",
INPUT B
PRINT "A+B =",A+B
GET
```

Hoe gaat dat nu in zijn werk? We selecteren New en zoeken het uit.

Wanneer de LZ erom vraagt, moeten we de naam van het programma intikken. De naam van het bovenstaande programma is VOORBLD (zie de eerste regel). Die naam geven we dus in en drukken op EXE.



Dan wordt het scherm gewist en komt de naam van het programma op de bovenste regel.

```
VOORBLD:
```

De cursor staat achter VOORBLD:. We drukken op EXE en de cursor gaat naar de volgende regel.

We tikken nu alle regels in, en drukken op het einde van elke regel op EXE (niet op NEER). Met DEL kunnen fouten gewist worden. Met de pijltjes kan vrij door de *listing* gelopen worden.
Op ON drukken resulteert in het wissen van de regel waar de cursor op staat.
Door op MODE te drukken komt het Editor-menu op het scherm, waarvan de volgende vier opties momenteel belangrijk zijn :

Find	Zoekt een string op in de listing. Deze optie wordt gebruikt om snel naar een bepaalde regel of een bepaalde *instructie* te springen;
Home	Stuurt de cursor helemaal naar boven, naar het begin van het programma;
End	Stuurt de cursor naar het einde van het programma;
Zap	Wist het hele programma uit de editor. De editor wordt echter niet verlaten! Gebruik deze optie dus alleen om helemaal opnieuw te beginnen.

Als alle regels zijn ingetikt, drukken we op de MODE-toets. Dit keer zijn het de andere opties van het Editor-menu die ons interesseren, namelijk :

Tran	Vertaal de listing naar *Q-Code*. Dit is een speciale, verkorte vorm van OPL die de programmeur echter nooit te zien krijgt. De LZ kan namelijk geen programma's uitvoeren die in OPL geschreven zijn. Q-Code-programma's daarentegen leest hij zo zo vlot als u en ik OPL lezen. Elk programma dient dus eerst naar die mysterieuze Q-Code vertaald te worden, en dat doet Tran (afkorting voor *translate* : vertaal);
Save	Bewaar het programma en verlaat de editor;
Quit	Bewaar het programma niet en verlaat de editor.
Xtran	Een speciale versie van Tran die vertaalt naar Q-Code die ook door de modellen CM en XP kan worden uitgevoerd.

4.2.2.1 Tran
We hebben het eigenlijk al gezegd : Tran vertaalt de listing in OPL naar een verkorte code. Als we een programma later met Run laten uitvoeren

(punt 4.2.3), dan zal de LZ dus steeds de Q-Code-versie uitvoeren. We moeten daar niet teveel over nadenken, want dat doet de LZ zelf (daar hebben we hem tenslotte voor).

Kies Tran en druk op EXE. Het programma wordt vertaald en indien er schrijffouten zijn (*syntax errors*), dan zal de LZ ze uitwijzen en de mogelijkheid voorzien ze te corrigeren. Druk in dit geval op SPACE en de LZ gaat vanzelf terug in de editor.

```
ERROR
SYNTAX ERR

press SPACE key
```

Als zich geen fouten voordoen, gaat de LZ na het vertalen vanzelf over naar Save.

4.2.2.2 Save
De naam wordt nu getoond en kan gewijzigd worden (alsook de pakletter, met MODE natuurlijk). Het Saven wordt gestart met EXE.

Indien eerst *Tran* werd gekozen, zal Save automatisch zowel de Q-Code-versie als de listing (OPL-versie) Saven. Dat gaat in één keer. Anders zal alleen de OPL-versie, geSaved worden *en kan het programma niet uitgevoerd worden*. Dat is normaal, daar de LZ alléén Q-Code kan uitvoeren. Bij het Saven wordt de listing *niet* op syntax errors gecontroleerd.

4.2.2.3 Quit
Quit wordt gekozen als het zonet ingetikte programma of de zonet ingetikte aanpassingen *niet moeten worden geSaved*. Alles verdwijnt dan *en kan niet meer teruggevonden worden*. Quit is dus definitief. Na Quit komen we terug bij het Prog-menu terecht.

4.2.2.4 Xtran
Xtran vertaalt naar een Q-Code die zich richt op een scherm van 2 regels met 16 tekens per regel. Programma's gemaakt met Xtran kunnen dus door alle modellen van de Organiser II worden uitgevoerd : CM, XP en LZ. Op de LZ wordt het CM/XP-scherm weergegeven door het fictieve scherm te omkaderen :

```
Dit is een CM-
of XP-programma.
```

Na dit menu komen we terug in het Prog-menu terecht, en kunnen we iets anders doen, een procedure uitvoeren bijvoorbeeld. Dat doen we met Run.

4.2.3 Run

Run, de naam zegt het zelf, dient om een procedure (lees : *programma*) uit te voeren. We tikken daarvoor de naam in van de procedure (of drukken op MODE om de pakletter te veranderen) en drukken op EXE. Als een procedure zonet is ingetikt of geëditeerd, hoeven we gewoon op EXE te drukken : de LZ specifiëert de naam zelf. We kunnen de naam echter ook wissen (met ON of DEL) en een andere opgeven, dat spreekt.

Als we een naam intikken die niet bestaat, zal de LZ reageren met een gepaste foutmelding (*FILE NOT FOUND*).

4.2.4 Print

Een optie die net zo voor de hand ligt als Run, is Print. Deze optie dient uiteraard om een afdruk te maken van de *listing*, dus van het programma. Alléén de OPL-listing wordt afgedrukt, niét de Q-Code-versie. Met deze laatste zouden we trouwens niets kunnen doen, die dient alleen voor de LZ zelf.

Let er wel op dat de printer of de Comms Link is aangesloten (zie verder in dit boek), of de LZ protesteert met een *DEVICE MISSING*-foutmelding.

4.2.5 Dir

Het begint langzamerhand voorspelbaar te worden. Dir werkt net hetzelfde bij Prog als bij Notes en Xfiles.

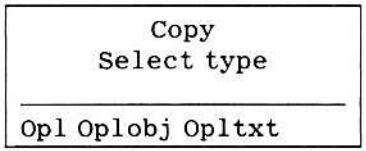
4.2.6 Copy

Dat geldt niet 100‰ voor Copy. Hier moeten we immers opgeven welke *versie(s)* van de procedure(s) we willen copiëren. Er zijn drie mogelijkheden :

Opl Beide onderstaande mogelijkheden gecombineerd;

Oplobj Alleen de vertaalde (in jargon : *gecompileerde*) versie, de Q-Code dus (ook wel *object code* genoemd). Q-Code kan niét worden geëditeerd.
Programma's waarvan geen listing in OPL beschikbaar is zijn dus *onwijzigbaar*. Deze optie wordt best gekozen wanneer we een programma aan iemand willen doorgeven maar die persoon niet de mogelijkheid willen geven de OPL-instructies te zien of aan te passen;

Opltxt Alleen de listing in OPL (met de instructies in leesbare vorm, zoals PRINT of INPUT). Deze vorm kan nadien nog met Edit geëditeerd worden. Deze optie wordt het best gekozen wanneer het programma vanop de datapak (de *doelpak*) aanpasbaar moet zijn, maar (nog) niet uitvoerbaar, of wanneer er te weinig geheugenruimte op de pak over is om ook de uitvoerbare versie te stockeren. Het vertalen met Tran kan dan immers nog later gebeuren.



4.2.7 Delete

Net als Dir werkt ook Delete als overal.

5. Variabelen

5.1 Inleiding

In de vorige voorbeelden hebben we de variabele X% gebruikt. Het percentageteken betekende dat de betrokken variabele van het type *integer* was. Nu gaan we alle types variabelen, en hun gebruik, nader bekijken.

Een *integere* variabele is, zoals al eerder gezegd, een variabele die alleen *gehele* waarden kan bevatten, en dan nog alleen tussen -32768 en +32767. Een geheel getal als 50000 kunnen we dus niet meer integer noemen, het is *reëel* geworden. Reële getallen zijn getallen die wel cijfers na de punt mogen bevatten (π bijvoorbeeld is een reëel getal), en die tevens buiten de integere begrenzing vallen. 'Reals', zoals de Engelse term luidt, liggen steeds binnen de zone van -1e100 (-1 met 100 nullen erachter) en +1e100 exclusief. Dit is ruim voldoende wanneer we weten dat er in de gehele Melkweg niet eens +1e100 moleculen zijn (!). Reële variabelen zijn uiteraard variabelen die reële getallen bevatten. In tegenstelling tot de integere variabelen wordt de naam van de reële variabele *niet* gevolgd door een herkenningsteken (% bij integer). Dus variabelen met namen als *a, x, prijs* en *id3* zijn reële variabelen.
Kleine opmerking : omdat de reële getallen de integere getallen overlappen, kan een reële variabele ook integere waarden bevatten.

Men heeft echter ook nog behoefte aan het onthouden en bewerken van *tekst*. Hiervoor kunnen we noch bij de integere variabelen terecht, noch bij de reële. Dus werd de zgn. *'string'* (Engels voor *touwtje*) ingevoerd. Een string is een verzameling (of *streng*) van één of meer tekens, waarvan begin en einde worden aangegeven met aanhalingstekens. Men kan een enkele letter of een enkel cijfer in een string zetten, maar ook een hele regel tekst. De *inhoud* van de string is dus in feite *vrij*. Er kunnen dus ook leestekens in. Een string wordt aangeduid met een dollarteken achter de naam, bv. *computer$*, *a$*, *naam$*.

We gaan nu eerst bekijken hoe we die verschillende variabelen gebruiken, en daarna kijken we naar bepaalde *verwerkingen*.

5.2 Variabelen aangeven

Voordat variabelen in een procedure kunnen worden gebruikt, moeten ze eerst worden 'aangegeven'. Dat hebben we in een vorig voorbeeld gedaan met de instructie LOCAL. Dit aangeven ('declareren') is nodig omdat variabelen geheugenruimte in beslag nemen en de Organiser II die ruimte moet reserveren voordat het eigenlijke programma begint. Wanneer we met strings gaan werken moeten we dus ook de preciese lengte van die strings opgeven. Hier zijn enkele voorbeelden :

```
LOCAL A$(20), B%, X%, UITKOMST
```

A$(20) betekent dat de string *A$* een lengte van 20 heeft. Deze lengte mag in het gebruik ook minder zijn; het gaat er hier alleen om, dat er plaats is gereserveerd voor 20 tekens en dat A$ die lengte niet mag overschrijden.

Als een variabele niet is aangegeven maar in de procedure toch wordt gebruikt, veroorzaakt dit uiteraard een foutmelding van de Organiser II tijdens TRAN.

Behalve de *lokale* variabelen, die worden aangegeven met *LOCAL*, is er ook nog een ander 'type', namelijk de *globale* variabelen. Het verschil is miniem maar van belang : de lokale variabelen kunnen slechts worden gebruikt in één procedure.

Voorbeeld van declaraties in het begin van een procedure :

```
GLOBAL X%, X$(200), PERCENTAGE
LOCAL A%, UITKMST, PAK$(1), NAAM$(20)
```

De namen van de variabelen worden steeds van elkaar gescheiden door een komma, en bij strings wordt steeds de lengte gemeld.

5.3 Eenvoudige bewerkingen

Variabelen krijgen een waarde door een eenvoudige *toekenningsinstructie*. Deze kan er als volgt uitzien :

```
X% = 19
NAAM$ = "Jansen"
UITKMST = X% * 25
```

Het eerste voorbeeld is vrij simpel : de waarde 19 wordt toegekend aan variabele X%. Het tweede is al even eenvoudig, hoewel het dit keer een string is die een waarde krijgt toegekend (tussen aanhalingstekens). In dit geval heeft de string NAAM$ (namen van variabelen mogen met kleine- of met hoofdletters geschreven worden, dat maakt geen verschil) een lengte van 6, de aanhalingstekens maken van de eigenlijke string dus geen deel uit. In het derde voorbeeld zit er een *berekening* in de toekenning : dat mag, dat is volkomen normaal en wordt heel vaak gedaan. De reële variabele UITKMST krijgt dan de waarde van de variabele X% toegekend, wat die ook moge zijn, vermenigvuldigd met 25.

Strings kan men, net zoals getallen, optellen. Men kan ze echter niet vermenigvuldigen of mengen met getallen. De volgende toekenningen zijn dus *niet geldig* :

```
X$ = "Jansen" * 3
ADRES$ = "Driekoningenstraat "+19
X% = "Organiser II"-27
```

Wel geldig zijn :

```
X$ = "Jansen"+"Jansen Jansen"
ADRES$ = "Driekoningenstraat" + "19"
```

Wanneer een integere variabele een reële waarde toegekend krijgt, zoals hier bijvoorbeeld :

```
X% = 24.3
```

dan wordt dit getal bij de punt 'afgehakt' en X% krijgt de waarde 24. Wanneer men probeert een waarde buiten de integere grenzen toe te kennen aan een integere variabele, zoals bijvoorbeeld

```
X% = 50000
```

dan reageert de Organiser II correct met een foutmelding, nl. 'INTEGER OVERFLOW' (letterlijk : 'integer loopt over'). Over deze en andere foutmeldingen gaan we het later nog uitgebreid hebben.

Met deze numerieke toekenningen kan men even complexe berekeningen invoeren als bij de rekenmachine (CALC). In het volgende hoofdstuk zullen we iedere numerieke functie van OPL nader bekijken.

5.4 Tien extra variabelen

Er zijn tien reële variabelen in de Organiser II, die te allen tijde kunnen worden gebruikt en nooit hun waarde verliezen (behalve natuurlijk wanneer de stroom wegvalt). Dit zijn de *geheugenvariabelen* van de rekenmachine : M0 tot M9. Deze variabelen kunnen in iedere procedure worden gebruikt (steeds als *GLOBAL*s). We hoeven ze niet aan te geven in het begin van de procedure. We kunnen ook waarden toekennen :

```
M0 = (M1 + X%) / 10
```

bijvoorbeeld is volkomen legaal.

6. Instructies

6.1 Inleiding

Computerprogramma's, in welke taal dan ook geschreven, bestaan steeds uit een opeenvolging van instructies. Een instructie is een eenvoudige opdracht aan de computer. Elk programma is niets anders dan een aaneenschakeling van bepaalde instructies in een bepaalde volgorde.

Eén van die instructies is *PRINT*, die we al eerder zijn tegengekomen. PRINT is een instructie die in zo goed als iedere taal voorkomt, zij het soms onder een andere naam. PRINT dient om tekst of cijfers op het scherm te brengen.

Al die instructies hebben natuurlijk een vaste gebruiksnorm, waarvan niet mag worden afgeweken. Men zegt dat de instructies een vaste *syntax* hebben. Die syntax is vastgelegd door de ontwerpers van de taal (in het geval van OPL, de heren van Psion). We gaan nu het begrip 'syntax' van naderbij bestuderen.

6.2 De syntax

De syntax wordt vastgelegd in een bepaalde vorm. Vaak wordt de zgn. BNF (Backus-Naur Form) gebruikt. BNF is echter een vrij ingewikkelde bedoening. Een voorbeeld:

In sommige instructies mogen *numerieke expressies* voorkomen. Dit zijn berekeningen (het volgende hoofdstuk is hieraan gewijd). Zo'n expressie wordt bijvoorbeeld voorgesteld door het symbool

```
<expr>
```

De instructie PRINT kan worden gebruikt om numerieke expressies (bijvoorbeeld : `18 + 3 * (22.1 * 1.1)`) op het scherm af te drukken. De syntax van PRINT is dan :

```
PRINT <expr>
```

Behalve numerieke expressies zijn er ook *stringexpressies* (tekst). Een stringexpressie kan zijn : `"Koning, " + "Keizer, " + "Admiraal"`. PRINT kan eveneens worden gebruikt om stringexpressies af te beelden. Dan wordt bij de syntax van PRINT toegevoegd :

```
PRINT <expr$>
```

(het symbool <expr$> wordt gebruikt om een stringexpressie voor te stellen).

6.3 Syntaxsymbolen

Buiten <exp> en <exp$> zijn er nog een aantal symbolen, die bij de definitie van de syntax worden gebruikt :

<exp%> : integere expressie; expressie waarvan de uitkomst een integer getal is.

<var> : reële variabele, bijvoorbeeld `x, prijs`.

<var%> : integere variabele, bijvoorbeeld `x%, btw%`.

<var$> : stringvariabele, bijvoorbeeld `a$, naam$`.

[] : wat tussen de rechte haakjes staat is optioneel; wanneer bij een bepaalde instructie de punt-komma mag worden gebruikt, staat in de definitie : [;].

{} : wat tussen de accolades staat is repetitief; het mag in één instructie dus meerdere keren worden gebruikt.

| : 'of'; wat links van dit symbool staat mag worden gebruikt, of wat rechts staat.

6.4 PRINT

Als voorbeeld gebruiken we de PRINT-syntax.

```
PRINT { [ <exp> | <exp%> | <exp$> ] [ ; | , ] }
```

Hieruit leiden wij af dat de instructie PRINT mag worden gevolgd door een expressie, of deze nu reëel, integer of een string is, en indien er meerdere expressies achter elkaar worden afgedrukt, moeten zij verbonden zijn door een puntkomma of een komma. Wanneer zij verbonden zijn door een komma dan zullen de afzonderlijke parameters op het scherm gescheiden zijn door een spatie. Voorbeelden van geldige PRINTs zijn :

```
PRINT 229 * 19
PRINT "Organisers doen het met datapaks"
PRINT A%; "keer"; B%; "is"; A% * B%
```



```
PRINT A%, "gedeeld door", B%, "is", A% / B%
```

De expressies (<exp>, ...) kunnen immers ook bestaan uit variabelen of getallen, het hoeven niet steeds berekeningen te zijn.

Verderop gaan we andere instructies definiëren en hun gebruik uitleggen met talrijke voorbeelden. Eerst moeten we wat meer te weten komen over expressies, zowel numeriek als alfanumeriek (string).

7. Numerieke bewerkingen

7.1 Inleiding

OPL kent een aantal speciale, numerieke functies. Deze functies zijn grotendeels ook toepasbaar in de optie CALC van het hoofdmenu en zijn in het desbetreffende hoofdstuk al beknopt besproken.

We zullen van iedere functie enkele voorbeelden te zien krijgen. Iedere functie wordt voorgesteld met zijn volledige *syntax* (zij het dan in een iets meer leesbare vorm dan de BNF). Incorrect gebruik van de functies (dus niet in overeenstemming met de syntax) levert onvermijdelijk foutmeldingen op bij het gebruik van 'TRAN'.

7.2 De functies en hun gebruik

ABS (absolute waarde)
Gebruik : `x = ABS(<exp>)`
ABS berekent de absolute waarde van de expressie tussen haakjes. De absolute waarde is steeds positief : bijvoorbeeld `ABS(-18) = 18 = ABS(18)`.

ACOS (hoekcosinus, alleen LZ)
Gebruik : `x = ACOS(<exp>)`
De expressie, welke is uitgedrukt in radialen, bevat een cosinus waarvan ACOS de hoek berekent.

ADDR (adres)
Gebruik : `x% = ADDR(<var>)`
Geeft het adres (= de geheugenplaats) van de variabele tussen haakjes. Het is evident dat een expressie niet is toegelaten : het gaat immers expliciet over het adres van een bepaalde variabele. Voorbeelden : `x% = ADDR(x%)`, `x% = ADDR(A$)`. Niet toegestaan : `x% = ADDR(x%+1)`, `x% = ADDR(19)`.

ASC (ASCII-code)
Gebruik : `x% = ASC(<exp$>)`
Dit levert de ASCII-code op van het *eerste teken* van de stringexpressie. ASCII-codes zijn, voor wie het niet precies meer weet, numerieke representaties van cijfers, letters en leestekens. Deze code ligt steeds tussen 0 (lege string) en 255. Een kort voorbeeldprogrammaatje maakt het misschien duidelijker (tik het eens in !) :


```
VOORB:
LOCAL A$(1)
A$ = CHR$(240)
PRINT "Teken 240 =", A$
GET
```

(Over de laatste instructie, GET, later meer). Probeer het ook eens met andere getallen.

ASIN (hoeksinus, alleen LZ)
Gebruik : `x = ASIN(<exp>)`
De expressie, welke is uitgedrukt in radialen, bevat een sinus waarvan ASIN de hoek berekent.

ATAN (arc tangent : boogtangens)
Gebruik : `x = ATAN(<exp>)`
Berekent de boogtangens van de expressie tussen haakjes. Voorbeelden : `x = ATAN(.555)`, `x = ATAN(y+2*(z))`. De berekende waarde is uitgedrukt in radialen.

CLOCK (klokje rechtsboven, alleen LZ)
Gebruik : `x% = CLOCK(<exp%>)`
Deze functie voert twee opdrachten uit:
- — de staat van de klok (aan/uit) wordt in x% (1/0) onthouden;
- — de klok wordt aan/uitgezet alnaargelang de <exp%> tussen haakjes (1/0).

COS (cosinus)
Gebruik : `x = COS(<exp>)`
Berekent de cosinus van de expressie, uiteraard ook in radialen. Voorbeelden : `x = COS(1)`, `x = COS(COS(PI))`.

DAY (dag)
Gebruik : `x% = DAY`
Geeft de dag van de maand. Dit is steeds een waarde tussen 1 en 31, en uiteraard steeds een gehele waarde. Het is *niet* mogelijk om bijvoorbeeld te gebruiken : `DAY = 22`.

DAYS (aantal dagen tussen opgegeven datum en 1 januari 1900, alleen LZ)
Gebruik : `x% = DAYS(<dag%>, <maand%>, <jaar%>)`
(`<dag%>`, `<maand%>` en `<jaar%>` zijn integere expressies.
Deze functie berekent het aantal dagen tussen 1 januari 1900 en de datum die in de argumenten is weergegeven. Deze functie kan dus ook worden gebruikt om het verschil in dagen tussen twee data te berekenen, bijvoorbeeld :

DAYS(1,1,2000)–DAYS(16,11,1968)

DEG (graden)
Gebruik : x = DEG(<exp>)
Zet de expressie tussen haakjes om van radialen in graden. Dus om bijvoorbeeld de cosinus van 2.5 te berekenen in graden gebruiken we : x = DEG(COS(2.5)). Ook toegestaan : x = DEG(.2), ...

DOW (dag in de week, alleen LZ)
Gebruik : x% = DOW(<dag%>, <maand%>, <jaar%>)
(<dag%>, <maand%> en <jaar%> zijn integere espressies).
Deze functie berekent de dag in de week (DOW staat voor day of week), waarbij 1 staat voor maandag en 7 staat voor zondag.

ERR (fout)
Gebruik : x% = ERR
Geeft het nummer van de laatste fout die is opgetreden. Binnenin de Organiser II zijn alle fouten, die kunnen optreden (bijvoorbeeld de INTEGER OVERFLOW van het vorige hoofdstuk) vooraf gedefiniëerd en genummerd. De functie ERR geeft het nummer van de recentste fout. Dit nummer is een getal tussen 0 en 255. Als het 0 is, is er geen fout opgetreden. Het gebruik is in dit stadium nog niet helemaal duidelijk, maar later zullen we zien dat deze functie heel belangrijk is.

EXP (exponent)
Gebruik : x = EXP(<exp>)
Berekent de constante *e* (= 2.71828182846 ...) tot de macht <exp>. Dus EXP(2) geeft *e tot de 2e macht*. Het resultaat is uiteraard een reëel getal. De <exp> moet een getal zijn kleiner dan 230.

FLT (vlottend)
Gebruik : x = FLT(<exp%>)
Deze functie zet de integere expressie om in een reëel getal (vlottende komma). Dit heeft in verband met variabelen weinig zin omdat men een integere expressie steeds mag toekennen tot een reële variabele. Ikzelf pas deze functie dan ook nooit toe.

FREE (vrij)
Gebruik : x% = FREE
Geeft het aantal nog vrije (ongebruikte) bytes werkgeheugen in de Organiser II (pak A:). Net als bij de functie DAY, mag FREE geen argumenten (parameters) hebben.

HOUR (uur)
Gebruik : x% = HOUR
Geeft het uur van de dag tussen 0 en 23. Merk op dat dit, net als DAY en soortgelijke functies, géén variabele is : er kan dus geen waarde aan worden toegekend. Een toekenning als HOUR = 9 is dus duidelijk niet toegestaan.

IABS (integere absolute waarde)
Gebruik : x% = IABS(<exp>)
Bijna hetzelfde als ABS : hier wordt echter een integere waarde berekend. Wanneer men bijvoorbeeld uitrekent : x% = IABS(29.61), dan zal x% de waarde 29 krijgen. Het gedeelte na de komma wordt dus afgehakt (er wordt niét afgerond, anders zou x% in bovenstaand voorbeeld de waarde 30 gekregen hebben).

INT (integer)
Gebruik : x% = INT(<exp>)
Hakt het gedeelte na de komma van <exp> af en maakt zo een integer getal. Dit integere getal wordt dan aan de variabele toegekend. Uiteraard moet de uitkomst van de expressie binnen de begrenzing van integere getallen liggen.
Voorbeeld : x% = INT(29*1.1) geeft 31, want 29 * 1.1 = 31.9.

INTF (integer-vlottend)
Gebruik : x = INTF(<exp>)
Wordt op dezelfde manier gebruikt als de bovenstaande functie INT, met de afwijking dat de uitkomst van de expressie niet noodzakelijk binnen de integergrenzen moet liggen. De berekende waarde is dus steeds een geheel getal, maar INTF(100000.1) wordt aangenomen terwijl INT(100000.1) niet geldig is.

LEN (lengte)
Gebruik : x% = LEN(<exp$>)
Geeft aan de variabele de lengte van <exp$> in tekens. Voorbeelden :

```
LEN("OPL") = 3
LEN("Stars "+"Stripes") = 13
```

(Dit is wel degelijk een *numerieke* en geen *string*functie, omdat het resultaat ervan een getal is).

LN (natuurlijke logaritme)
Gebruik : x = LN(<exp>)
Berekent de natuurlijke logaritme van de expressie tussen haakjes. De expressie moet een getal zijn groter dan nul. De basis van natuurlijke logaritmen is de numerieke constante *e*.

LOC (lokatie)
Gebruik : x% = LOC(<exp$1>, <exp$2>)
De syntax ziet er misschien een beetje gek uit; we hebben het immers over

numerieke functies terwijl beide parameters van deze functie strings zijn. De eerste string, <exp$1>, wordt opgezocht in de tweede string, <exp$2>. De plaats waar <exp$2> <exp$1> bevat, wordt aan de variabele x% toegekend. Voorbeelden :

```
LOC("boek","Telefoonboek") = 9
LOC("pak","pak") = 1
LOC("a","Analfabeet") = 3
LOC("TV","Televisie") = 0
```

Uit het laatste voorbeeld blijkt dat het resulterende getal 0 is wanneer de eerste string niet wordt teruggevonden in de tweede.

LOG (logaritme)
Gebruik : x = LOG(<exp>)
Geeft de Briggse logaritme (basis 10) van de expressie tussen haakjes. Net als bij de natuurlijke logaritmen moet de expressie een waarde opleveren die hoger ligt dan 0. In hoofdstuk 38, 'Wiskundige Bibliotheek', zullen we een logaritmefunctie programmeren waarbij we zelf de basis van de logaritme kunnen bepalen.

MAX (hoogte getal, alleen LZ)
Gebruik : x = MAX(<exp1>, <exp2>, . . . , <expn>)
x = MAX(tabel(), <exp%>)
Deze functie geeft het hoogste getal in de opgegeven reeks weer. In het eerste getal worden er *n* argumenten gegeven waarvan het hoogste getal wordt berekend. In het tweede geval is de getallenreeks weergegeven in een tabelvariabele en geeft de expressie <exp%> weer van hoeveel getallen het hoogste moet worden bepaald.
Enkele voorbeelden:

```
MAX(1,2,3)    = 3
MAX(x%(), 5)  = het hoogste getal van de eerste 5 getal-
                len in de array x%()
```

MEAN (gemiddelde, alleen LZ)
Gebruik : x = MEAN(<exp1>, <exp2>, . . . , <expn>)
x = MEAN(tabel(), <exp%>)
MEAN berekent het gemiddelde van de getallen in de opgegeven reeks (zie ook functie MAX).

MIN (kleinste getal, alleen LZ)
Gebruik : x = MIN(<xp1>, <exp2>, . . . , <expn>)
x = MIN(tabel(), <exp%>)
MIN geeft het kleinste getal in de opgegeven reeks weer (zie ook functie MAX).

MINUTE (minuut)
Gebruik : x% = MINUTE
Geeft de huidige minuut van het uur (een getal tussen 0 en 59). Zie ook DAY, HOUR, MONTH, SECOND en YEAR.

MONTH (maand)
Gebruik : x% = MONTH
Geeft de huidige maand van het jaar (een getal tussen 1 en 12).

PI (het getal π)
Gebruik : x = PI
Deze functie berekent de waarde van de numerieke constante π (3.14159265359). (Dit is eigenlijk niet de correcte waarde van π. Zelfs met supercomputers kan men de correcte waarde van dit getal niet berekenen; wetenschappers hebben totnutoe tot 1 miljoen cijfers achter de komma uitgerekend, zonder een regelmaat in de cijfermassa te ontdekken).

RAD (radialen)
Gebruik : x = RAD(<exp>)
Zet de expressie, die is uitgedrukt in booggraden, om naar zijn equivalent in radialen. Zie ook : DEG.

RND (willekeurig)
Gebruik : x = RND
Genereert een willekeurig (Eng. : random) getal tussen 0 en 1 (exclusief). Om bijvoorbeeld een willekeurig getal tussen 0 en 250 te nemen volstaat het (RND * 250) te gebruiken.

SECOND (seconde)
Gebruik : x% = SECOND
Geeft de seconde van de minuut (een getal tussen 0 en 59). Om uit te rekenen hoeveel seconden we reeds in een bepaald uur zijn, volstaat het (MINUTE - 60) +SECOND te berekenen.

SIN (sinus)
Gebruik : x = SIN(<exp>)
Berekent de sinus van de expressie in radialen. Om een sinus in graden te berekenen gebruiken we DEG(SIN(<exp>).

SQR (vierkantswortel)
Gebruik : x = SQR(<exp>)
Berekent de vierkantswortel (Eng: square root) van de expressie tussen haakjes. De expressie mag reëel zijn, maar negatieve getallen zijn uiteraard uitgesloten. Zonder zgn. 'complexe getallen' te gebruiken (die in de Organiser II niet standaard zijn voorzien) kan men immers geen vierkantswortel van een negatief getal berekenen.

STD (standaardafwijking van een reeks getallen, alleen LZ)
Gebruik : x = STD(<exp1>, <exp2>, . . . , <expn>)
x = STD(tabel(), <exp%>)
STD berekent de standaardafwijking van de getallen in de opgegeven reeks (zie ook MAX).

SUM (som van een reeks getallen, alleen LZ)
Gebruik : x = SUM(<exp1>, <exp2>, . . . , <expn>)
x = SUM(tabel(), <exp%>)
Deze functie berekent de som van de getallen in de opgegeven reeks (zie ook MAX).

TAN (tangens)
Gebruik : x = TAN(<exp>)
Berekent de tangens van de expressie tussen haakjes, welke is uitgedrukt in radialen.

VAL (waarde)
Gebruik : x = VAL(<exp$>)
In dit geval wordt verwacht dat <exp$> een getal bevat, bijvoorbeeld "2.00E+5". Dit getal wordt dan aan de variabele toegekend. Hoewel in het bovenstaande voorbeeld een reële variabele is gebruikt (x), mag deze ook integer zijn, als 100% zeker is dat het resultaat van VAL(<exp$>) ook integer is. Enkele voorbeelden zijn :

```
VAL("2 Organisers") = 2
VAL("18E2") = 1800
```

Wanneer <exp$> een numerieke expressie, een berekening, bevat, wordt deze *niet* uitgerekend : de waarde van het eerste getal wordt gebruikt en de rest vervalt.

```
VAL("9*28") = 9
```

Wanneer <exp$> een variabele bevat, wordt de waarde van deze variabele *niet* uitgerekend.

Dus als de variabele A = 17855, dan is

```
VAL("A") = 0
```

VAR (variantie, alleen LZ)
Gebruik : x = VAR(<exp1>, <exp2>, . . . , <expn>)
x = VAR(tabel(), <exp%>)
VAR berekent de statistische variantie van de getallen in de opgegeven reeks (zie ook MAX).

WEEK (week in het jaar, alleen LZ)
Gebruik : x% = WEEK(<dag%>, <maand%>, <jaar%>)
(<dag%>, <maand%> en <jaar%> zijn integere expressies).
Week berekent de week in het jaar van de opgegeven datum. De LZ neemt als eerste dag van de eerste week de eerste maandag in het jaar.

YEAR (jaar)
Gebruik : x% = YEAR
Geeft het jaar. Als het 1988 is, zal YEAR ook 1988 als resultaat hebben (en niet 88). Om het jaar in de eeuw te berekenen kunnen we (YEAR- 1899) gebruiken.

Tot zover de numerieke functies.

8. Stringbewerkingen

8.1 Inleiding

Bij de stringbewerkingen valt op dat Psion heel veel belang hecht aan de mogelijkheid om getallen om te zetten in strings (bijvoorbeeld 56 wordt "56"). Verschillende mogelijkheden heeft men op dit gebied naar voren gebracht, inclusief het gebruik van wetenschappelijke notatie. Dit is een initiatief dat vele talenontwerpers van vandaag zouden moeten volgen.

Verder hanteert men ook de standaard-stringbewerkingen van alle andere talen. Toch vind ik dit nog niet helemaal genoeg; verderop in dit boek geef ik dan ook een paar extra stringfuncties, die men zo kan intikken en gebruiken.

8.2 De stringfuncties

Concatenatie

Concatenatie betekent : samenvoeging. Men kan strings samenvoegen door ze bij wijze van spreken 'op te tellen'. Een voorbeeld ter illustratie :

```
"Orga" + "niser" + "II" = "Organiser II"
```

Concatenatie kan vrij worden gebruikt in stringexpressies. Geldig is dus:

```
A$ = "Een " + "geconcateneerde " + "string"
```

OPL kent de volgende stringbewerkingen :

CHR$ (karakter)
Gebruik : x$ = CHR$(<exp%>)
In de variabele x$ wordt een teken (karakter) gezet dat de ASCII-code <exp%> heeft. Deze code is een getal tussen 0 (leeg) en 255 (een blokje). De code voor hoofdletter 'A' is bijvoorbeeld 65, de code voor een spatie is 32, de code voor 'ε' is 227. Een volledige lijst van ASCII-codes is te vinden in Appendix A.

DATIM$ (Europees datumformaat)
Gebruik : x$ = DATIM$ Geeft de huidige datum en het uur van de dag in Europees formaat weer in een string, bijvoorbeeld : "FRI 02 OCT 1987 14:51:39". De lengte van de string is steeds 24 (tekens).

DAYNAME$ (naam van de weekdag, alléén bij LZ)
Gebruik : x$ = DAYNAME$(<exp%>)
De expressie moet tussen 1 en 7 liggen. Het resultaat van DAYNAME$ is de (Engelse) naam van de dag in de week. Voorbeelden zijn :

```
DAYNAME$(1) = "MONDAY"
DAYNAME$(2) = "TUESDAY"
...
DAYNAME$(7) = "SUNDAY"
```

ERR$ (fout)
Gebruik : x$ = ERR$ (<exp%>)
Omdat de Organiser II voor iedere mogelijke fout een code heeft (zie vorig hoofdstuk, functie ERR), heeft hij voor iedere code ook een bijhorende foutmelding. Voor fout nummer 195 bijvoorbeeld is de foutmelding "INTEGER OVERFLOW". In de niet-uitgebreide Organiser II (dus zonder randapparatuur) zitten die foutcodes tussen 195 en 255. Om bijvoorbeeld het foutbericht voor error-code 200 te krijgen, gebruiken we ERR$(200). Wanneer de expressie tussen haakjes niet tussen de grenzen ligt (in de niet-uitgebreide Organiser II dus tussen 195 en 255), dan heeft de string de inhoud "*** ERROR ***".

FIX$ (vast formaat)
Gebruik : x$ = FIX$(<exp>, <exp%1>, <exp%2>)
(een reële en twee integere expressies)
Dit is een tamelijk gecompliceerde functie, die een stringvoorstelling geeft van de eerste uitdrukking <exp>, met <exp%1> tekens na de punt. De totale lengte van de string is steeds <exp%2> tekens. Wanneer de eigenlijke lengte van de string kleiner is dan <exp%2> dan wordt hij aangevuld met spaties aan de rechterkant. Als <exp%2> een negatief getal is, heeft de aanvulling met spaties plaats aan de linkerkant. Wanneer de eigenlijke lengte van de string groter is dan <exp%2> dan bevat de string sterretjes. Voorbeelden :

```
FIX$(223344.5566,3,10) = "223344.557"
FIX$(200,5,12) = "200.00000"
FIX$(200,5,-12) = "200.00000"
FIX$(8E5,1,4) = "****"
```

In het eerste voorbeeld is het derde cijfer na de punt correct afgerond. Een soortgelijke functie is :

GEN$ (generatie)
Gebruik : x$ = GEN$(<exp>,<exp%>)
Ook hier wordt de uitkomst van de eerste, reële expressie weergegeven in een string, waarvan het 'formaat' afhangt van de tweede, integere expressie. <exp%> bepaalt namelijk de uiteindelijke lengte van de string die

GEN$ genereert. De regels in verband met lengte van de functie FIX$ gelden ook hier. De string die uiteindelijk wordt geproduceerd bevat <exp> als integer getal, indien mogelijk. Dus :

```
GEN$(18,2) = "18"
GEN$(18,5) = "18"
GEN$(18,-5) = "18"
```

Als de expressie echter geen integer getal bevat, zal GEN$ proberen het als een reëel getal voor de stellen.

```
GEN$(PI,13) = "3.14159265359"
```

Wanneer het echter wat moeilijk is reële notatie te gebruiken, zal GEN$ het wetenschappelijke formaat kiezen.

```
GEN$(100000,5) = "1E+05"
```

Wanneer ook dit niet lukt, en het getal uiteindelijk dus niet in de string past, wordt de string opgevuld met sterretjes.

```
GEN$(18532,4) = "****"
```

HEX$ (hexadecimaal)
Gebruik : x$ = HEX$(<exp%>)
De hexadecimale voorstelling van de integere expressie (er is immers geen manier om decimale getallen in hex te schrijven) wordt in de string geplaatst. HEX$(201) zal daarom dus de string "C9" opleveren.

LEFT$ (linker gedeelte)
Gebruik : x$ = LEFT$(<exp$>,<exp%>)
Geeft de eerste <exp%> tekens weer van <exp$>. Enkele voorbeelden ter verduidelijking :

```
LEFT$("California",4) = "Cali"
LEFT$("New"+" York ",6) = "New Yo"
```

LOWER$ (kleine letters)
Gebruik : x$ = LOWER$(<exp$>)
Deze functie geeft in x$ de string weer die men krijgt wanneer alle hoofdletters in <exp$> kleine letters zijn. Voorbeeld :
Als de string A$ de inhoud "Machine" heeft, dan is :

```
LOWER$(A$) = "machine"
```

en de string A$ zelf blijft onveranderd.

MID$ (stukje string)
Gebruik : x$ = MID$(<exp$>,<exp%1>,<exp%2>)
Geeft een string die bestaat uit een stukje van <exp$> : namelijk het stukje dat begint op positie <exp%1> in die string en <exp%2> tekens lang is. Voorbeelden :

```
MID$("Heerlijk, helder ...",11,6)
= "helder"

MID$("1234567",3,1)
= "3"
```

We zouden bijvoorbeeld de eerder genoemde functie LEFT$(<exp$>,<exp%>) kunnen schrijven als MID$(<exp$>,1,<exp%>).

MONTH$ (naam van de maand, alléén bij LZ)
Gebruik : x$ = MONTH$(<exp%>)
De expressie moet tussen 1 en 12 liggen. Het resultaat van MONTH$ is de (Engelse) naam van de maand. Voorbeelden zijn:

```
MONTH$(1)  = "JANUARI"
MONTH$(2)  = "FEBRUARY"
...
MONTH$(12) = "DECEMBER"
```

NUM$ (nummer)
Gebruik : x$ = NUM$(<exp>,<exp%>)
Dit is opnieuw een functie om een getal in een string te zetten, net als FIX$ en GEN$. NUM$ neemt een decimaal (reëel) getal <exp>, maakt dit getal geheel (het wordt afgerond) en zet het in een string met lengte <exp%>. Als de <exp%> (tweede parameter) negatief is wordt de string aan de linkerkant opgevuld met spaties wanneer dat nodig is. Als de string langer is dan <exp%> en dus niet in het opgelegde formaat past, wordt hij gevuld met sterretjes. Voorbeelden :

```
NUM$(3.14,4) = "3"
NUM$(99.99,-8) = "100"
NUM$(9999.99,4) = "****"
```

RIGHT$ (rechter gedeelte)
Gebruik : x$ = RIGHT$(<exp$>,<exp%>)
RIGHT$ werkt net hetzelfde als zijn tegenhanger, LEFT$; hij geeft namelijk de laatste <exp%> tekens weer van <exp$>.

```
RIGHT$("California",4) = "rnia"
RIGHT$("New York..",5) = "ork.."
```

REPT$ (herhaling)
Gebruik : x$ = REPT$(<exp$>, <exp%>)
Genereert een string die bestaat uit <exp%> keer <exp$>, bijvoorbeeld:

```
REPT$("Z", 5) = "ZZZZZ"
REPT$("ABC ", 3) = "ABC ABC ABC"
```

SCI$ (wetenschappelijke notatie)
Gebruik : x$ = SCI$(<exp>, <exp%1>, <exp%2>)
Dit zouden we de wetenschappelijke versie van FIX$ kunnen noemen. Deze functie genereert een string, met daarin de wetenschappelijke notering van <exp>, met <exp%1> cijfers na de punt en met een totale stringlengte van <exp%2> tekens. Door voor <exp%2> negatieve getallen te gebruiken wordt de string zonodig weer aangevuld met spaties vanaf links. Als de resulterende string niet in de voorgeschreven lengte past (<exp%2>), bevat hij slechts sterretjes. Voorbeelden :

```
SCI$(1000, 2, 8) = "1.00E+03"
SCI$(PI, 0, 16) = "3E+00"
SCI$(2**8, 2, -10) = "2.56E+02"
SCI$(16384, 2, 5) = "*****"
```

UPPER$ (hoofdletters)
Gebruik : x$ = UPPER$(<exp$>)
UPPER$ is de tegenhanger van LOWER$, en produceert een string die gelijk is aan <exp$>, waarin echter alle kleine letters hoofdletters zijn geworden. Dus :

```
UPPER$( "Organiser II") = "ORGANISER II"
UPPER$("!!??") = "!!??"
UPPER$("america") = "AMERICA"
```

Tot zover de alfanumerieke functies.

9. Tabelvariabelen

9.1 Inleiding

We hebben nu reeds afzonderlijke numerieke en alfanumerieke variabelen besproken. Deze variabelen hebben allemaal een eigen functie en een eigen inhoud : er is in feite geen relatie tussen hen. Soms is het echter nodig een reeks bij elkaar horende variabelen te hebben, nl. om *tabellen* (*rijen*) te kunnen opslaan. Deze variabelen heten heel toepasselijk '*tabelvariabelen*' (Eng.: *array variables*).

Eén tabelvariabele kan dus verschillende getallen of strings bevatten, alnaargelang de grootte. Die grootte bepalen we zelf, net als bij strings. Eigenlijk zouden we een string ook kunnen beschouwen als een tabel : het is immers een eindige rij tekens.

Tabellen hebben we zowel numeriek als alfanumeriek nodig. We kunnen dan ook integere, reële en stringtabellen opstellen en gebruiken.

9.2 Numerieke tabellen

Eerst moeten we aangeven hoe groot de tabel is, samen met de naam en het *type* (integer, reëel) :

```
LOCAL a%(20),prijs(100)
```

In dit voorbeeld hebben we twee numerieke tabellen, namelijk a%, een integere tabel met 20 elementen, en prijs, een reële tabel met 100 elementen. We kunnen bijvoorbeeld ook een tabel opstellen met daarin het aantal dagen van de maand. We kijken even naar het volgende voorbeeld:

```
LOCAL m%(12)
m%(1) = 31
m%(2) = 28
m%(3) = 31

...

m%(11) = 30
m%(12) = 31
```

In de eerste regel wordt duidelijk gemaakt dat we een lokale tabel m% gaan

gebruiken met 12 elementen. In de volgende 12 regels krijgt ieder element van die tabel een waarde : de tabel wordt dus geïnitialiseerd. Een tabelelement wordt aangeroepen door de naam van de tabel op te geven gevolgd door de *index* van dat element tussen haakjes (voor het 12e element is de index dus 12).

Verder worden tabelvariabelen op net dezelfde manier gebruikt als gewone variabelen.

Probleempje : stel dat we met een tabel prijs werken, waarin een prijslijst is ingetikt. De prijs van het 18e artikel moet worden verhoogd (met 100 eenheden). Hoe kunnen we nu de inhoud van een tabelelement verhogen?

```
prijs(18) = prijs(18) + 100
```

Merk op dat de verhoging + 100 *niet* tussen de haakjes staat! In dat geval zou `prijs(18)` de waarde van `prijs(118)` krijgen (waarom?).

Programma 9.1 : BOODSCH

In het volgende voorbeeldprogramma vragen we de gebruiker om via het klavier 25 prijzen in te tikken van boodschappen die hij/zij gedaan heeft. Iedere prijs komt in een tabel te staan en vervolgens worden de totale en de gemiddelde prijzen berekend.

```
BOODSCH:
LOCAL PRIJS(25), SOM, GEMIDD, X%
CLS
PRINT "Boodschappen"
X% = 1
WHILE X% <= 25
  PRINT X%, " =",
  INPUT PRIJS(X%)
  X% = X% + 1
ENDWH
SOM = 0
X% = 1
WHILE X% <= 25
  SOM = SOM + PRIJS(X%)
  X% = X% + 1
ENDWH
GEMIDD = SOM /25
CLS
PRINT "Som =", SOM
PRINT "Gemiddelde =", GEMIDD
GET
```


Door een tweede WHILE te gebruiken tellen we de prijzen op. Door die som te delen door 25 (er zijn 25 prijzen ingetikt) krijgen we de gemiddelde prijs.

Programma 9.2 : BOODSCH, tweede versie

Het kan uiteraard efficiënter. We kunnen de SOM namelijk verhogen wanneer de prijs net is ingetikt. Kijk naar de volgende listing en vergelijk met de vorige :

```
BOODSCH:
LOCAL PRIJS(25), SOM, X%
CLS
PRINT "Boodschappen"
SOM = 0
X% = 1
WHILE X% <= 25
  PRINT X%, " =",
  INPUT PRIJS(X%)
  SOM = SOM + PRIJS(X%)
  X% = X% + 1
ENDWH
CLS
PRINT "Som =", SOM
PRINT "Gemiddelde =", SOM/25
GET
```

Een aardig stukje korter, niet? Ook de variabele GEMIDD, die eigenlijk overbodig was, hebben we in de tweede versie geëlimineerd.

Merk op dat iedere prijs afzonderlijk in de tabel `prijs` is opgenomen en dat we, als we dat zouden wensen, de prijs van ieder artikel later in ons programma kunnen gebruiken.

9.3 Stringtabellen

Net als getallen kunnen we ook strings in een tabel stoppen. Iedere string in die tabel moet dan wel een maximumlengte hebben die gelijk is aan die van alle andere strings in die tabel. Bijvoorbeeld :

```
GLOBAL NAAM$(10, 20)
```

geeft een tabel NAAM$ aan die bestaat uit 10 strings met ieder een maximumlengte van 20 tekens.

Programma 9.3 : NAMEN

Dit programma vraagt de gebruiker om 10 namen in te tikken. Die namen worden daarna vliegensvlug op het scherm afgedrukt.

```
NAMEN:
LOCAL NAAM$(10,20),N%
CLS
PRINT "Geef 10 namen : "
N% = 1
WHILE N% <= 10
  PRINT N%, ": ",
  INPUT NAAM$(N%)
  N% = N% + 1
ENDWH
N% = 1
WHILE N% <= 10
  PRINT NAAM$(N%)
  N% = N% + 1
ENDWH
GET
```


10. Invoer van het klavier

Instructies : EDIT, GET, INPUT, KSTAT
Functies : GET$, KEY, KEY$

10.1 Inleiding

Verreweg de meeste invoer die de Organiser II krijgt komt binnen via het klavier. Men tikt zijn programma's in en alle programma's, inclusief natuurlijk het besturingssysteem van de Organiser II dat op een ROMchip binnenin de computer staat, worden 'bestuurd' via het klavier. Het is dus evident dat invoer via het klavier een belangrijk aspect van programmeren is, en dat we er uiterste zorg aan moeten besteden. Zoals uit de bovenstaande opsomming van instructies blijkt, is OPL rijkelijk voorzien van klavierfuncties.

10.2 INPUT

Waarschijnlijk de meestgebruikte instructie voor invoer is *INPUT*. Die naam is de Engelse term voor invoer. De instructie *INPUT* zoals wij die in OPL kennen, is afgeleid van de programmeertaal BASIC.

Syntax : INPUT <var> | <var%> | <var$>

INPUT kan als parameter (die niet tussen haakjes staat) dus enkel variabelen hebben en géén expressies. INPUT onderbreekt het programma en vraagt de gebruiker iets in te tikken. Wanneer men een numerieke variabele opvraagt, zoals *A%* of *BTW*, zal de Organiser II zijn klavier in *numerieke staat* zetten : men hoeft niet meer op SHIFT-NUM te drukken om cijfers in te kunnen tikken. Nadat men een getal/tekst heeft ingetikt, drukt men op EXE. Wat is ingetikt gaat dan naar de gebruikte variabele... Enkele voorbeeldprogramma's maken dit duidelijker.

Programma 10.1 : SOM

Dit eenvoudige programmaatje vraagt twee decimale getallen aan de gebruiker en telt die op. Vervolgens moet de gebruiker op EXE drukken om weer uit het programma te geraken. *Opmerking:* de regelnummers links en het commentaar rechts mogen niet worden ingetikt; ze zijn hier opgenomen ter verduidelijking.

```
1  SOM:
2  LOCAL G1, G2, X$(1)     ; twee getallen en een string
3  PRINT "Getal 1 : ",
4  INPUT G1                ; vraag getal 1 op
5  PRINT "Getal 2 : ",
6  INPUT G2                ; vraag getal 2 op
7  PRINT "Som : ", G1+G2; druk de som af
8  INPUT X$                ; nu moet de gebruiker op EXE drukken
```

De laatste lijn is misschien wat onduidelijk. We gebruiken INPUT X$ (een string met lengte 1) om het programma te onderbreken omdat de Organiser II direct na het uitvoeren van de afdruk van de som terug zou springen naar het PROGmenu, en zo snel kan natuurlijk niemand lezen. Daar *INPUT* steeds wordt beëindigd met EXE, volstaat het dat de gebruiker van het programma op deze toets drukt om het programma te beëindigen en terug te keren naar het PROGmenu.

Programma 10.2 : NAAM

Dit kleine programma vraagt de naam van de gebruiker op en voert vervolgens wat 'kunstjes' uit.

```
 1 NAAM:
 2 LOCAL NAAM$(20), L%, X$(1)
 3 PRINT "Wat is uw naam", CHR$(63)
 4 INPUT NAAM$
 5 L% = LEN(NAAM$)
 6 PRINT "Lengte : ", L%, "letters"
 7 INPUT X$
 8 PRINT "Helft : ", LEFT$(NAAM$, L%/2)
 9 PRINT "Laatste : ", RIGHT$(NAAM$, 1)
10 INPUT X$
```

Wat uitleg rond de kunstjes : omdat de variabele L% de lengte van de naam bevat, levert de expressie L%/2 de helft van de lengte op. De functie LEFT$(NAAM$, L%/2) levert dan ook de eerste helft van de naam op.

10.3 GET

GET is zowel een functie als een instructie, en heeft dan ook twee verschillende syntaxen.

Syntax : GET
x% = GET

In het eerste geval wacht de Organiser II tot er een toets wordt ingedrukt en gaat dan voort. We zouden in de bovenstaande programma's SOM en NAAM de instructie INPUT X$ dan ook eenvoudigweg kunnen vervangen door GET. In dat geval wacht het programma niet op EXE, maar een willekeurige toets. Een voorbeeldje :

```
1 HALLO:
2 GET
3 PRINT "Hallo "+CHR$(33)
4 GET
```

Het tweede geval is een beetje ingewikkelder. Wanneer GET als functie gebruikt wordt, wacht de Organiser II eveneens op een toets en geeft aan de variabele de ASCII-code van de toets die ingedrukt werd.

Programma 10.3 : ASCII

Hier wordt van de gebruiker verwacht dat hij een toets indrukt en vervolgens wordt de ASCII-code van het teken dat hij indrukte op het scherm gebracht. Probeer het programma eens uit en druk op enkele toetsen, met of zonder SHIFT.

```
1 ASCII:
2 LOCAL A%
3 PRINT "Druk een toets in"
4 A% = GET
5 PRINT "Dat was", A%
6 GET
```

Het kan misschien ook leuk zijn de voorlaatste regel te vervangen door :

```
5 PRINT "Dat was", CHR$(A%)
```

Hier is dan wel een kleine omzetting nodig (van getal naar string). Het kan ook nog een stapje korter.

10.4 GET$

GET$ heeft, in tegenstelling tot GET, slechts één syntax, namelijk als functie.

Syntax : x$ = GET$

GET$ wacht op een toets en plaatst de ASCII-code van die toets in de variabele. Probeer eens het volgende voorbeeld :

Programma 10.4 : GETTEST

Dit programmavoorbeeld laat de gebruiker 32 keer een toets indrukken en geeft dan op het scherm het ingedrukte teken weer. Waarom 32? Omdat precies dàn het scherm helemaal vol is.

```
1 GETTEST:
2 LOCAL A%
3 A% = 1
4 WHILE A% <> 32   ; 32 keer ...
5   PRINT GET$   ; wachten en afdrukken
6   A% = A% + 1
7 ENDWH
```

Het programma beëindigt zichzelf als het scherm vol is (wanneer 32 tekens zijn ingetikt).

10.5 KEY en KEY$

Deze twee functies zijn identiek aan GET en GET$, op één subtiel verschil na : zij *wachten* niet op een toets. Als er nu net een toets is ingedrukt, geven zij respectievelijk de ASCII-code en de string weer. Als er géén toets wordt ingedrukt, resulteren zij in respectievelijk 0 of de lege string ("").

Programma 10.5 : KEYTEST

Dit programma drukt tekens af als die worden ingetoetst, anders niets. Niet dat men een dergelijk programma dagelijks kan gebruiken, maar het demonstreert het gebruik van KEY$ wel leuk.

```
1 KEYTEST:
2 LOCAL T%
3 T% = 1
4 WHILE T% <> 150
5   PRINT KEY$;   ; druk teken af
6   T% = T% + 1
7 ENDWH
```

Als er geen toets wordt ingedrukt is KEY$ leeg en de lijn *PRINT KEY$* drukt dan ook niets af.

10.6 Editeren van een stringvariabele : EDIT

Het is mogelijk om, net als bij *FIND* en *SAVE*, een string te editeren. De instructie *EDIT* zorgt daar voor.

EDIT
Syntax : EDIT <var$>
De stringvariabele wordt geëditeerd door de gebruiker. Dit editeren wordt beëindigd door EXE in te drukken. Het is mogelijk <var$> op voorhand te verdelen in verschillende regels door op het einde van een regel het teken CHR$(9) in te voegen (zie Appendix A).

Programma 10.6 : EDITVB

```
EDITVB:
LOCAL E$(255)
E$ = "Een"+CHR$(9)+"Twee"+CHR$(9)+"Drie"
EDIT E$
...
```

(De rest van het programma werkt dan met de nieuwe versie van E$).

10.7 KSTAT

Op de bovenste rij toetsen van het klavier staan twee speciale toetsen (CAP en NUM). Door deze toetsen in te drukken samen met SHIFT kan de *staat van het klavier* worden gewijzigd. Er zijn vier staten mogelijk :

	CAP	NUM
1.	AAN	UIT
2.	UIT	UIT
3.	AAN	AAN
4.	UIT	AAN

Men kan de staat van het klavier op twee manieren veranderen, namelijk:

1) **fysiek** (door de toetsen in te drukken)
2) **logisch** (door de staat te programmeren)

Dit programmeren gaat via de instructie KSTAT (afkorting van de Engelse term 'keyboard state' : staat van het klavier).

Syntax : KSTAT <expr%>

De expressie moet een getal zijn tussen 1 en 4.

KSTAT 1 zet de toets CAP aan en NUM uit
KSTAT 2 zet de toetsen CAP en NUM uit
KSTAT 3 zet de toetsen NUM en CAP aan
KSTAT 4 zet de toets NUM aan en CAP uit

Zo eenvoudig gaat dat.

Programma 10.7 : KSTEST

Wie dit programma uitvoert, moet vier keer een string intikken (maximumlengte 16 tekens). Het is eigenlijk de bedoeling dat de gebruiker 4 keer dezelfde toetsen indrukt en zo verschillende resultaten te zien krijgt. Het spreekt vanzelf dat die resultaten beter tot uiting komen wanneer ook van de SHIFT-toets gebruik gemaakt wordt en dat de toetsen CAP en NUM met rust moeten worden gelaten.

```
 1 KSTEST:
 2 LOCAL K$(16)
 3 KSTAT 1
 4 INPUT K$
 5 KSTAT 2
 6 INPUT K$
 7 KSTAT 3
 8 INPUT K$
 9 KSTAT 4
10 INPUT K$
```

Wanneer de Organiser II de instructie INPUT <expr> tegenkomt, dus een numerieke invoer, voert hij automatisch KSTAT 3 uit (het feit dat de Organiser II klein is hoeft dus duidelijk geen belemmering voor kracht en flexibiliteit te zijn!).

11. Uitvoer naar het scherm

Instructies : PRINT, CLS, AT, CURSOR, MENU, VIEW

11.1 Inleiding

Net als invoer van het klavier, is uitvoer naar het scherm erg belangrijk. De meeste uitvoer van de Organiser II verloopt immers via het scherm. In de meeste, zoniet alle programma's die we ooit zullen schrijven, vindt uitvoer naar het scherm plaats.

11.2 PRINT

Deze instructie kennen we eigenlijk al. Maar wat extra uitleg is nooit weg.

PRINT drukt tekens af op het scherm. Dit scherm bestaat uit 2 rijen van elk 16 kolommen, en kan dus in totaal slechts 32 tekens bevatten. Het is dus duidelijk dat we in een PRINT-opdracht nooit meer dan 32 tekens laten afdrukken. Als we dat wel doen, dan zullen die eerste 32 wel afgedrukt worden, maar onmiddellijk daarna wordt het scherm verticaal doorgerold om plaats te maken voor de volgende tekens. Het volgende miniprogramma demonstreert dit uitstekend :

Programma 11.1 : PR32

```
1 PR32:
2 PRINT REPT$("SCROLL ",20)
3 GET
```

De string die wordt afgedrukt is 140 tekens lang (7 * 20).

Er is echter een manier om langere strings toch op dat kleine schermpje te krijgen. Later in dit hoofdstuk komen we daar nog op terug.

Nog iets dat misschien leuk is om weten : als we PRINT gebruiken zonder parameters, is het resultaat een lege regel. De volgende PRINT gebeurt dan op de volgende regel.

11.3 CLS

CLS is de afkorting voor "CLear *S*creen" ('maak het scherm schoon'). Wanneer deze instructie wordt toegepast, wordt het scherm schoongemaakt en het volgende teken dat zal worden afgedrukt komt in de linker-

bovenhoek te staan (de *cursor* wordt linksboven gezet).

Syntax : CLS

Programma 11.2 : CLSDEMO

Nu demonstreren we het gebruik van de CLS-instructie.

```
 1 CLSDEMO:
 2 CLS
 3 PRINT "Druk een   toets in . . . "   ; 2 spaties na 'toets'
 4 GET
 5 CLS
 6 PRINT "OK"
 7 GET
```

11.4 AT

Vanzelfsprekend kan het ook voorkomen dat we niet altijd in die linkerbovenhoek willen afdrukken. De instructie AT geeft ons de mogelijkheid om waar we willen op het scherm iets af te drukken.

Syntax : AT <exp%1>, <exp%2>
<exp%1> en <exp%2> zijn coördinaten op het scherm. <exp1%> duidt de kolom aan en <exp%2> de rij.

	<exp%1>	<exp%2>
CM, XP	tusen 1 en 16	tussen 1 en 2
LZ	tussen 1 en 20	tussen 1 en 4

11.5 CURSOR

Soms zien we, bij INPUT bijvoorbeeld, een knipperende cursor op het scherm. Die cursor laat zien waar het volgende teken zal worden afgedrukt.

Het kan wenselijk zijn, ook bij een gewone PRINT bijvoorbeeld, die cursor te laten zien. Hiervoor heeft men de instructie CURSOR bedacht.

Syntax : CURSOR ON | OFF

CURSOR ON zet de cursor aan, CURSOR OFF zet hem weer uit. Als standaard staat de cursor *uit*. Een programma ter demonstratie :

Programma 11.6 : CDEMO

```
 1 CDEMO:
 2 CLS
 3 PRINT "Cursor uit . . . "
 4 CURSOR ON
 5 PRINT "Cursor aan . . . "
 6 GET
 7 CURSOR OFF
 8 GET
 9 CURSOR ON
10 GET
```

Wanneer een programma beëindigd wordt, zet de Organiser II de cursor *altijd* weer aan.

11.6 MENU

Ook dit is eigenlijk geen instructie, net als bijvoorbeeld GET$, maar een functie. MENU biedt de mogelijkheid programma's te schrijven die zelf ook met menu's werken, net als het hoofdmenu en de opties PROG en DIARY.

Syntax : x% = MENU (<exp$>)

In de <exp$> staan een aantal opties opgesomd. Die worden dan in menuvorm op het scherm gebracht en de gebruiker kan met behulp van de cursortoetsen of door een eerste letter in te tikken een optie selecteren, net als bij de andere menu's. Het rangnummer van de gekozen optie wordt dan aan de variabele toegekend.

Programma 11.7 : PROG2
Dit programma imiteert het PROGmenu.

```
1 PROG2:
2 LOCAL P%
3 P% = MENU ("EDIT, LIST, DIR, NEW, RUN, ERASE, COPY")
4 PRINT "Gekozen nr", P%
5 GET
```

Als we met deze namaak-PROG bijvoorbeeld NEW kiezen, krijgt de variabele P% de waarde 4, omdat NEW de vierde optie van het menu is.

Een ander voorbeeld demonstreert het gebruik van MENU overduidelijk:

Programma 11.8 : EXBTW

Dit voorbeeld berekent de basisprijs van een produkt. Als invoer wordt de brutoprijs en het BTW-percentage (het laatste via een menu) opgevraagd.

```
 1 EXBTW:
 2 LOCAL B%, P, M%, A%(3)
 3 A%(1)=19
 4 A%(2)=25
 5 A%(3)=33
 6 M%=MENU("19%, 25%, 33%")
 7 PRINT "Inc. BTW :",
 8 INPUT P
 9 B%=A%(M%)
10 P=P/(100+B%)*100
11 PRINT "EX :", INTF(P)
12 GET
```

In de voorlaatste regel dient INTF te worden gebruikt, omdat de ingevoerde prijs (P) een reële variabele is.

11.7 VIEW

En dan zijn we nu bij de functie gekomen waarmee we strings op één lijn kunnen brengen die langer zijn dan 16 tekens (max. 32 tekens). VIEW (Engels voor 'zicht' of 'uitzicht') drukt een string op een regel af, en wanneer die string langer is dan 16 tekens, wordt hij automatisch horizontaal gescrolld naar links. Die scrollrichting kan door de gebruiker van het programma naar keuze worden veranderd door op de cursortoetsen LINKS of RECHTS te drukken. Wanneer echter een andere toets wordt ingedrukt, geeft VIEW de ASCII-code van die toets door aan een variabele.

Syntax: x% = VIEW(<exp%>, <exp$>)

<exp%> stelt de regel op het scherm voor (1 of 2) en <exp$> bevat de string.

Programma 11.9 : VDEMO

```
1 VDEMO:
2 LOCAL V$(33), V%
3 V$ = "Demonstratie van de functie VIEW"
4 CLS
5 V% = VIEW(1, V$)
6 AT 3, 2
7 PRINT "onderbroken"
8 GET
9 V% = VIEW(1, V$)
```


Programma 11.10 : NAMEN, tweede versie

Dit is een ietwat veranderde versie van het programma **NAMEN** (21.3), dat nu gebruik maakt van VIEW.

```
 1 NAMEN:
 2 LOCAL NAAM$(3,10), N$(210), V%
 3 CLS
 4 PRINT "Geef 3 namen :"
 5 V% = 1
 6 WHILE V% <= 3
 7    PRINT V%, ":",
 8    INPUT NAAM$(V%)
 9    N$ = N$ + NAAM$(V%) + " "
10    V% = V% + 1
11 ENDWH
12 V% = VIEW(1, N$)
```

De 3 namen worden aan elkaar geregen, gescheiden door een spatie, in de lange string N$. Deze string zal later met VIEW op het scherm worden afgedrukt.

11.8 Menu's bij de LZ : MENUN

MENUN is een nieuwe functie, specifiek voor de LZ. Het dient om menu's te creëren die, net zoals de overige LZ-menu's, op de eerste regel een ikoontje en een klokje laten zien.

MENUN (LZ-menu)
Gebruik : m% = MENUN(<exp%>, <exp$>)
De integere expressie bevat een getal tussen 0 en 2 :

0 : het menu werkt net als bij MENU;
1 : het is een munu van één regel;
2 : het menu werkt net als bij MENU, met de uitzondering dat het menu start op de regel waar dc cursor staat.
De regel(s) boven de cursor wordt (worden) ongemoeid gelaten.

De stringexpressie bevat een aantal menu-items, gescheiden door komma's.

Programma 11.11 : MDEMO

Dit programma demonstreert het gebruik van een menu (de instructie UDG wordt later nog uitgelegd).

```
MDEMO:
UDG 0,1,2,3,7,15,31,15,0,31
UDG 1,0,0,0,0,0,0,0,31
PRINT CHR$(0);REPT$(CHR$(1),14)
x% = CLOCK(1)
x% = MENUN(2, "Find,Save,Load,Print,Exit")
```


12. Beslissingen

12.1 Inleiding

Programma's die steeds identiek uitgevoerd worden bestaan bijna niet. Wat een programma doet zal vaak afhangen van de invoer. Bijvoorbeeld zal een programma dat berekent hoeveel belastingen iemand moet betalen andere verrichtingen doen voor iemand die kinderen ten laste heeft dan voor iemand die géén kinderen heeft. Met andere woorden : het programma zal *beslissingen* moeten nemen.

Die beslissingen hangen af van bepaalde omstandigheden (zoals hierboven kinderen of niet, en zo ja, hoeveel?). Om een beslissing te nemen moet de computer zich dus één of meerdere vragen stellen. De antwoorden op die vragen moeten eigenlijk reeds voorradig zijn nog vóór de vraag gesteld is, zodat de computer ze alleen maar moet 'opzoeken'. Dit is bij ons mensen trouwens net zo : voor we een beslissing over iets nemen, moeten we alle feiten kennen.

12.2 Beslissingen in 'mensentaal'

Deze keer nemen we een voorbeeld dat de meesten van ons goed kennen: TV-programma's. We moeten beslissen welk TV-programma we gaan zien. Voor ons is dat meestal vrij eenvoudig : we kiezen op voorkeur. De computer heeft geen voorkeur, daarvoor is hij van de mens afhankelijk.

Stel dat we moeten kiezen tussen twee programma's : een politieserie op Nederland 1 of een financieel programma op BBC 2. Wie van actie houdt, zal kiezen voor het eerste en wie financieel geïnteresseerd is kiest voor het tweede programma.

Het programma ziet er in de eerste stap als volgt uit :

```
ALS we actie willen, kiezen we
  voor de politieserie
ANDERS voor het financiële programma
(Stap 1)
```

Dit is vrij eenvoudig, maar het eigenlijke probleem wordt niet opgelost : wat willen we eigenlijk? Dit is niet door de computer te beslissen, hij heeft geen voorkeur, en bijgevolg moet de computer aan de gebruiker vragen of hij al dan niet voor actie kiest.

```
VRAAG of we actie willen
ALS het antwoord JA is,
  Politieserie
ANDERS Financieel programma
(Stap 2)
```

Dit begint er al op te lijken. De vraag wordt gesteld, de computer neemt een beslissing. Nu gaan we het geheel programmeren in OPL.

12.3 Overdracht naar de Organiser II

Programma 12.1 : TV

```
 1 TV:
 2 LOCAL ACTIE$(1)
 3 CLS
 4 PRINT "Actie (J/N)",CHR$(63)
 5 INPUT ACTIE$
 6 IF ACTIE$ = "J"
 7   PRINT "Politieserie"
 8 ELSE
 9   PRINT "Financieel       programma"
10 ENDIF
11 GET
```

Met INPUT wordt de vraag gesteld. Het antwoord op die vraag wordt onthouden in de variabele ACTIE$.

Dan wordt de inhoud van ACTIE$ vergeleken met hoofdletter J. Als de gebruiker die letter inderdaad heeft ingetikt, wordt (worden) de volgende regel(s) uitgevoerd, tot de computer een ELSE of een ENDIF tegenkomt (in ons geval dus ELSE).

ELSE betekent : *anders* (of *zo niet*). Het stukje dat na ELSE komt wordt dus enkel uitgevoerd als ACTIE$ *niet* gelijk is aan J (bijvoorbeeld N). Uiteindelijk wordt de Organiser II verteld dat de beslissing beëindigd is door ENDIF (einde-als).

Syntax :
```
IF <exp>
(stukje programma)
[ELSE[IF] <exp>
(stukje programma)]
ENDIF
```

Een kleine toelichting is hier op zijn plaats. ELSE is niet *verplicht* in de

keuze en kan dan ook worden overgeslagen. Bovendien is er nog de mogelijkheid een andere instructie te nemen : ELSEIF.

De <exp> is een *waarheidstest* : er wordt gekeken of iets waar is of niet. Als we naar programma 24.1 kijken, TV:, zien we dat er getest wordt of ACTIE$ de inhoud J heeft. Eigenlijk wordt er gevraagd :

```
Is het waar dat ACTIE$ de inhoud "J" heeft ?
```

Als het antwoord op deze vraag bevestigend luidt, zeggen we dat aan de vraag, aan de voorwaarde *voldaan* is (net als bij WHILE). In dit soort expressies vinden er dus steeds vergelijkingen plaats : is dit gelijk aan dat, is het ene groter of kleiner dan het andere, enzovoort.

Het stukje programma dat volgt op ELSE wordt dus steeds uitgevoerd wanneer aan de IF-voorwaarde niet is voldaan. Na een ELSE kunnen we trouwens direct weer een volgende vraag stellen. In dat geval gebruiken we de instructie ELSEIF.

We gaan deze nieuwvergaarde kennis toepassen in een klein programma. Dit programma, genaamd WNEMER, vraagt aan een werknemer wat zijn basissalaris is en hoeveel jaar hij reeds bij zijn huidige baas werkt. We berekenen dan de premie die deze werknemer krijgt, gebaseerd op de volgende regels :

- als hij 10 jaar of langer werkt, krijgt hij een premie van 7.5%
- als hij 20 jaar of langer werkt, een premie van 12.5%
- als hij 35 jaar of langer werkt, een premie van 15%

Het programma in OPL ziet er als volgt uit :

Programma 12.2 : WNEMER

```
1WNEMER:
2 LOCAL LOON, JAAR, PREMIE
3 CLS
4 PRINT "Basisloon : ",
5 INPUT LOON
6 PRINT "Jaren werkzaam : ",
7 INPUT JAAR
8 IF JAAR >= 35
9   PREMIE = 15
10 ELSEIF JAAR >= 20
```

```
11    PREMIE = 12.5
12 ELSEIF JAAR >= 10
13    PREMIE = 7.5
14 ENDIF
15 PREMIE = (LOON/100)*PREMIE
16 PRINT "Premie : ",PREMIE
17 PRINT "Loon : ",LOON+PREMIE
18 GET
```

12.4 Gebruik in beslissingen van logische operatoren

Een andere opdracht nu. We gaan een programma schrijven waarin wordt bepaald of iemand al dan niet wordt toegelaten tot een speciale club. Toegelaten worden :

Mannen

- ouder dan 25 jaar :
  getrouwd of lievelingskleur groen
- jonger dan 25 jaar en ouder dan 18 :
  niet getrouwd of lievelingskleur blauw

Vrouwen

- ouder dan 25 jaar :
  getrouwd en minstens 1 kind
- jonger dan 25 jaar :
  niet getrouwd

Dat ziet er al heel wat ingewikkelder uit, is het niet? Dit soort complexe regels maken een computerprogramma niet alleen moeilijk te schrijven, maar ook moeilijk te lezen. Desalniettemin gaan we het proberen.

Wie de voorwaarden wat nader bekijkt, merkt in de eerste plaats op dat deze verschillend zijn voor mannen en vrouwen, en in de tweede plaats dat de woorden 'of' en 'en' er in voorkomen. Met onze huidige kennis omtrent beslissingen zou ons programma onnodig ingewikkeld worden (men noemt dat soort onnodig ingewikkelde programma's in het Engels trouwens *spaghetti code*).

We hebben dus wat nieuwigheden nodig. Die zijn er in de vorm van de volgende *logische operatoren* : AND (= 'en') en OR (= 'of'). Om uit te maken of iemand getrouwd is en minstens 1 kind heeft gebruiken we bijvoorbeeld :


```
IF (GETROUWD% = 1) AND KINDEREN% >= 1
```

(in dit voorbeeld heeft de variabele GETROUWD% de waarde 1 voor getrouwd en 0 voor niet getrouwd).

Of iemand getrouwd is of als lievelingskleur groen heeft beslissen we als volgt :

```
IF (GETROUWD% = 1) OR LIEVKLEUR$ = "groen"
```

Om een beslissing te kunnen nemen omtrent geldigheid van lidmaatschap moeten we de volgende dingen weten :

- geslacht (m/v)	G$(1)
- leeftijd	LT%
- getrouwd/niet getrouwd	GETR%
- aantal kinderen	AK%
- lievelingskleur	LK$(6)

De lievelingskleur is alleen belangrijk bij de mannen, het aantal kinderen alleen bij de vrouwen.

Rechts van deze criteria staan de namen van de variabelen die we gaan gebruiken. Deze namen zijn kort omdat dit het programmeren vergemakkelijkt en minder geheugenruimte inneemt.

En dan volgt nu de listing.

Programma 12.3 : CLUB

```
 1 CLUB:
 2 LOCAL G$(1),LT%,GETR%,AK%,LK$(6),LID$(3)
 3 LID$ = "NEE"
 4 CLS
 5 PRINT "CLUBTOEGANG"
 6 PRINT "Geslacht : ",
 7 INPUT G$
 8 CLS
 9 PRINT "Leeftijd : ",
10 INPUT LT%
11 PRINT "Getrouwd : ",
12 INPUT GETR%
13 CLS
14 IF UPPER$(G$) = "M"
15    PRINT "Liev.kleur : ",
```

```
16    INPUT LK$
17    IF LT% > 25
18       IF (GETR% = 1) OR UPPER$(LK$) ="GROEN"
19          LID$ = "JA"
20       ENDIF
21    ELSEIF (LT% < 25) AND (LT% >18)
22       IF (GETR% = 0) OR UPPER$(LK$) = "BLAUW"
23          LID$ = "JA"
24       ENDIF
25 ELSE
26    PRINT "Kinderen : ",
27    INPUT AK%
28    IF LT% > 25
29       IF (GETR% = 1) AND (AK% > 0)
30          LID$ = "JA"
31       ENDIF
32    ELSEIF (LT% < 25) AND (GETR% = 0)
33       LID$ = "JA"
34    ENDIF
35 ENDIF
36 CLS
37 LT% = VIEW(1,"Toelating goedgekeurd "+CHR$(63)+" "
   +LID$)
```

We zullen beginnen met een toelichting van de gebruikte variabelen. Alle variabelen (regel 2) zijn bekend, behalve de laatste. LID$ is een string die in het programma 2 mogelijke inhouden heeft : "JA" of "NEE" (vanwege de NEE is de lengte 3 tekens). Het doel van deze variabele is dus duidelijk: onthouden of de persoon die met het programma werkt al dan niet toegelaten zou worden tot de club. LID$ krijgt in het begin de inhoud "NEE" (regel 3) en die inhoud wordt alleen veranderd wanneer de gebruiker aan alle eisen voldoet om lid te mogen worden.

De regels 4 tot 13 zouden geen problemen mogen opleveren : hier worden alleen maar de benodigde gegevens opgevraagd die voor zowel mannelijke als vrouwelijke kandidaten gelden. Wanneer die vragen allemaal gesteld zijn, gaat het programma over tot de specifieke vragen.

Het geslacht van de gebruiker zit in de variabele G$. In regel 14 wordt gekeken of we met een mannelijke kandidaat te maken hebben : G$ zou dan de inhoud "M" of "m" moeten hebben. Omdat we rekening moeten houden met zowel hoofdletters als kleine letters, wordt de functie UPPER$ gebruikt. Het programma vergelijkt dus de inhoud van G$ die, indien het een kleine letter betreft, voor de vergelijking tot een hoofdletter wordt gemaakt (de werkelijke inhoud van G$ blijft echter ongewijzigd).

Als de vergelijking opgaat, en het dus inderdaad een mannelijke kandidaat betreft, wordt de eerste 'mannelijke' vraag gesteld (regel 16). De leeftijd wordt nagekeken in regel 17. Die moet (voor de vraag van de lievelingskleur) *absoluut hoger* zijn dan 25. Als dat het geval is, springt de Organiser II naar regel 18, waar gekeken wordt of de kandidaat getrouwd is en *of* zijn lievelingskleur groen is (ook hier wordt gebruik gemaakt van de functie UPPER$ zodat geen onderscheid wordt gemaakt tussen "groen" en "GROEN"). Als dat inderdaad zo is wordt regel 19 uitgevoerd, waar LID$ de inhoud "JA" krijgt, wat betekent dat de kandidaat is aanvaard. Regel 20 dient om de vraag van regel 18 te beëindigen. De ELSEIF van regel 24 verwijst naar de IF van regel 17. Hier wordt gekeken of de leeftijd tussen 25 en 18 jaar is (exclusief). Vervolgens worden weer de nodige vragen gesteld.

De ELSE in regel 25 verwijst naar de IF van regel 14 : wat te doen als de kandidaat *niet* mannelijk is ? Heel eenvoudig : mits er in dit geval slechts twee mogelijkheden zijn, is de kandidaat zeker vrouwelijk (denk ik). En het vragenproces herhaalt zich.

In regel 37 wordt de 'uitslag' ten slotte op het scherm gebracht met VIEW. We hebben hiervoor de variabele LT% gebruikt : dat mag, omdat deze variabele voor de *rest* van het programma geen enkel nut meer heeft; zijn functie is vervuld. We hoeven dus geen nieuwe variabele te gebruiken.

12.5 NOT

Er zijn nog meer logische operatoren buiten AND en OR, en NOT is daar één van. NOT – de letterlijke vertaling is 'niet' – ontkent de waarheid van zijn parameter : we zouden bijvoorbeeld

```
IF (GETR% = 0)
```

even goed kunnen schrijven als :

```
IF NOT (GETR% = 1)
```

of zelfs :

```
IF NOT (GETR%)
```

Dit maakt de leesbaarheid van de programma's en procedures er stukken beter op. Bovendien wordt het programma een ietsje sneller uitgevoerd.

13. Boole-algebra

13.1 Inleiding

En dan nu wat wiskunde, voor wie daarin geïnteresseerd is. Boole-algebra is heel belangrijk in de informatica. Het wordt gebruikt in allerhande toepassingen en zelfs het ontwerp van de computer steunt op deze tak van de wiskunde.

De grondlegger van de Boole-algebra, ook wel kort *logica* geheten, was *George Boole* (1815-1864), een Engels wiskundige. Hij noteerde de resultaten van zijn onderzoek naar wiskundige logica in het boek *Investigation of the laws of thought* (1854).

In essentie is Boole-algebra niets anders dan wiskunde in het binaire stelsel. De enige cijfers die worden gebruikt zijn 0 en 1. Voor het uitvoeren van bewerkingen in Boole-algebra worden de logische operatoren (AND, NOT, OR, ...) gebruikt. De uitkomst van zo'n Boole-bewerking kan zijn 1 (waar) of 0 (vals).

13.2 AND

AND hebben we besproken in het vorige hoofdstuk, waar het werd gebruikt om voorwaarden voor beslissingen aan elkaar te koppelen. Eigenlijk wordt daar ook een soort Boole-algebra toegepast, maar daar gaan we hier niet verder op in.

De bewerking AND neemt 2 binaire cijfers en geeft als resultaat *de binaire vermenigvuldiging* van die cijfers. Dit kan zijn :

```
0 AND 0 = 0 * 0 = 0
0 AND 1 = 0 * 1 = 0
1 AND 0 = 1 * 0 = 0
1 AND 1 = 1 * 1 = 1
```

Het komt er dus op neer dat het resultaat van AND alleen 1 (waar) is wanneer zowel de linkse als de rechtse parameters 1 zijn (AND betekent trouwens '*en*' : alleen waar als de linkse *en* de rechtse waarde waar zijn).

13.3 OR

Met een beetje voorstellingsvermogen kunnen we nu wel raden wat de resultaten van OR kunnen zijn. OR betekent immers '*of*' : het resultaat zal

dan ook 1 zijn wanneer de linkse *of* de rechtse parameter 1 is. Dit resultaat is, strikt genomen, de *binaire optelling* van de parameters.

```
0 OR 0 = 0 + 0 = 0
0 OR 1 = 0 + 1 = 1
1 OR 0 = 1 + 0 = 1
1 OR 1 = 1 + 1 = 1
```

(1 + 1 geeft niet 2 als uitkomst, omdat in logica enkel de cijfers 0 en 1 zijn toegestaan. Een wet van de logica schrijft voor dat voor alle getallen x, die binaire getallen zijn, geldt : x + x = x).

Bij OR komt het er dus op neer dat het resultaat alleen 0 geeft wanneer beide parameters 0 zijn.

13.4 NOT

NOT is de ontkenning, zoals gezegd in het vorige hoofdstuk. NOT heeft maar één parameter en geeft als resultaat de ontkenning van die parameter. De parameter wordt vermenigvuldigd met -1.

```
NOT 0 = 0 * -1 = 1
NOT 1 = 1 * -1 = 0
```

NOT wordt dus steeds gebruikt om het resultaat van een logische vergelijking om te keren.

13.5 Bytegebruik van logische operatoren

Een byte bestaat uit 8 bits, dus uit een groepering van 8 binaire cijfers. Het is mogelijk logische bewerkingen op die bits uit te voeren.

Nemen we twee getallen, bijvoorbeeld 151 (binair : 10010111) en 46 (binair : 00101110).

13.5.1 AND

```
        10010111 = 151
AND     00101110 = 46
        ----------
=       00000110 = 6
```

Het spreekt voor zichzelf, nietwaar ? De bewerkingen worden per bit uitgevoerd.

13.5.2 OR

```
        10010111 = 151
OR      00101110 = 46
-----------------------
=       10111111 = 191
```

13.5.3 NOT

```
NOT     10010111 = 151
-----------------------
=       01101000 = 104
```

Opmerking : aangezien het binaire getal 11111111 = 255, is het duidelijk dat het resultaat van NOT <exp%> steeds 255-<exp%> is (zie voorbeeld). Andere voorbeelden die de stelling bevestigen :

```
NOT     00000000 = 0
-----------------------
=       11111111 = 255

NOT     10101010 = 170
-----------------------
=       01010101 = 85


NOT     11110000 = 240
-----------------------
=       00001111 = 15
```

13.6 Gebruik

Om de eerste vier bits van de variabele b% bijvoorbeeld op 1 te zetten, gebruiken we de operator OR. Omdat het binaire getal 1111 = 15, programmeren we :

```
b% = b% OR 15
```

Om die vier bits op 0 te zetten, zouden we programmeren :

```
b% = b% AND 240
```

(waarom ?).

14. Volgorde van operatoren

In OPL hebben we drie verschillende operatoren die we in numerieke expressies kunnen gebruiken :

14.1 Rekenkundige operatoren

Dit zijn :

```
+       : optelling
-       : aftrekking of negatie (bijv. : i% = - i%)
*       : vermenigvuldiging
/       : deling
**      : machtsverheffing
```

14.2 Vergelijkingsoperatoren

Dit zijn :

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

14.3 Logische operatoren

Dit zijn :

```
AND     : en
OR      : of
NOT     : niet
```

14.4 Voorrang van operatoren

Die operatoren zijn niet gelijkwaardig. We zeggen dat ze zijn onderworpen aan een voorrangsregeling die in OPL is vastgelegd. In de berekening

```
25 + 18 * 3
```

bijvoorbeeld, zal de vermenigvuldiging *(18 * 3) eerst worden uitgerekend, het resultaat wordt dan bij* 25 opgeteld. Heel anders zou bijvoorbeeld zijn:

```
(25 + 18) * 3
```

In deze expressie zijn haakjes gebruikt. Haakjes hebben steeds voorrang en zullen eerst worden uitgerekend. De twee expressies zijn dus niet gelijk. Ziehier een voorranglijst :

1. () (hoogste voorrang)
2. NOT, negatie
3. **
4. *, /
5. +, -
6. =, >, <, >=, <=, <>
7. AND, OR (laagste voorrang)

Wanneer twee operatoren naast elkaar worden gebruikt die in dezelfde voorrangsklasse zitten, wordt de uitdrukking van links naar rechts uitgerekend, zoals in de volgende voorbeelden :

```
25 + 3 - 9

88 - 12 + 77

55 * 17 / 3

6 / 2 * 3

3 ** 2 ** 0
```

Deze voorrangswetten worden toegepast in elke programmeertaal, niet alleen in OPL.

15. Opmerkingen

Instructie : REM

Opmerkingen zijn in een programma vrij belangrijk. Het is steeds gemakkelijk ergens te kunnen lezen wat een bepaalde regel nu eigenlijk doet, waarvoor een zekere variabele precies dient, enzovoort. Tot nu toe hebben we elk commentaar naast de programma's geschreven, of hebben we de regels genummerd voor later commentaar. Het is echter ook mogelijk binnenin een programma zelf commentaar te geven.

Hiervoor dient de instructie REM. REM is de afkorting van het Engelse woord *remark*, wat zoveel betekent als 'opmerking'.

Syntax : REM commentaar . . .

REMs mogen op aparte regels staan of achter een programmaregel, daarvan gescheiden door een dubbele punt.

Voorbeeld :

```
REM ————————-
REM PRIJSBEPALING
REM ————————-
IF PRIJS > 2500                 : REM 2500 is acceptabel
   PRINT "Te duur"              : REM hoger is te duur
ELSE
   PRINT "OK"                   : REM prijs aangenomen
ENDIF
```

16. Functies

Instructie : RETURN

16.1 Inleiding

In OPL zijn een aantal functies standaard opgenomen : numerieke functies (SIN, COS, ...) en stringfuncties (LEFT$, FIX$, ...). Het is mogelijk deze waaier van functies te verrijken met zelf-ontwikkelde functies.

16.2 Numerieke functies

Een numerieke functie die niet standaard aanwezig is in de Organiser II is de functie SEC (secans). Deze wordt als volgt gedefinieerd :

De *secans* van een vaste hoek α is 1/cos α (waarvoor geldt : cos α is verschillend van 0).

Het is dus duidelijk dat we met een parameter werken en dat er in de functie rekening zal moeten worden gehouden met de waarde 0 voor de cosinus van α.

We kiezen in het PROGmenu de optie NEW en tikken de naam SEC in. Daarop verschijnt zoals gewoonlijk de naam van de in te tikken procedure/functie op de eerste regel :

```
SEC:
```

In plaats van nu op EXE te drukken, zoals we gewend zijn, tikken we in :

```
SEC: (A)
```

Dit is de parameter : de variabele A. Vervolgens tikken we de rest van de functie in :

Functie 16.1 : SEC

```
1 SEC: (A)
2 LOCAL S
3 S = COS(A)
4 IF S
5   RETURN(1/S)
6 ELSE
7 PRINT "Mag niet"
8 GET
9 ENDIF
```


In de tweede regel is al meteen duidelijk dat de parameter, in dit geval A, *niet* moet worden gedeclareerd.

In regel 5 zien we een nieuwe instructie, nl. RETURN. Deze instructie wordt enkel gebruikt in functies, en dient om een bepaalde waarde aan de functie te geven.

Syntax : RETURN <exp> | <exp%> | <exp$>

Er mogen dus ook strings worden doorgegeven, maar daarover hebben we het pas onder het volgende punt.

De bovenstaande functie SEC kan nog wat korter :

Functie 16.2 : SEC, tweede versie

```
1 SEC: (A)
2 IF COS(A)
3   RETURN(1/COS(A))
4 ENDIF
5 PRINT "Mag niet"
6 GET
```

De extra variabele S was dus helemaal niet nodig.

In onze programma's (of via CALC) kunnen we de functie oproepen door:

```
SEC: (<exp>)
```

(let op de dubbele punt). Voorbeeldprogramma :

Programma 16.1 : SECVB

```
1 SECVB:
2 LOCAL G
3 CLS
4 PRINT "Secans"
5 PRINT "Getal :",
```

```
6 INPUT G
7 CLS
8 PRINT SEC: (G)
9 GET
```

16.3 Stringfuncties

Bij de stringfuncties krijgt de naam achteraan een dollarteken, net als bij LEFT$, GEN$, e.d. We schrijven bijvoorbeeld de functie HALF$ die als resultaat de linkerhelft geeft van de string die ze als parameter krijgt (dus de string tussen haakjes) :

Functie 16.3 : HALF$

```
HALF$: (A$)
RETURN (LEFT$ (A$, LEN (A$) /2) )
```

Verklaring : stel dat de functie een string krijgt die een lengte heeft van 20 tekens. De linker helft bestaat dan uit de eerste 10 tekens (LEN (A$) /2). Die lengte, 10, wordt dan aangegeven aan de functie LEFT$, die correct de eerste 10 tekens weergeeft, waarna deze via RETURN naar de aanroepende procedure worden gebracht.

Een ander voorbeeld : SP$ (gespatiëerde string). Deze functie krijgt als invoer een string en geeft als uitvoer diezelfde string met spaties ertussen. Als de invoer bijvoorbeeld is :

```
"Spaties"
```

zal de uitvoer zijn :

```
"S p a t i e s"
```

De functie ziet er als volgt uit :

Functie 16.4 : SP$

```
1 SP$: (S$)
2 LOCAL A%, X$ (255)
3 X$ = ""
4 A% = 1
5 WHILE A% <= LEN (S$)
6    X$ = X$ + MID$ (S$, A%, 1) + " "
```



```
7    A% = A% + 1
8 ENDWH
9 RETURN (X$)
```

De string S$ wordt teken voor teken naar X$, die ook de spaties krijgt, gecopiëerd. X$ wordt daarna doorgegeven.

Het is duidelijk dat de *naam* van de geschreven functie moet weergeven van welk *gegevenstype* (integer, reëel, string) het resultaat is. Als het resultaat van een percentagefunctie bijvoorbeeld steeds integer is, zal de naam PERC% luiden i.p.v. PERC.

16.4 Recursieve functies

Recursie is één van de meest intrigerende aspecten van programmeren. Het is een programmeertechniek die vooral wordt toegepast bij het oplossen van wiskundige problemen met de computer en ook voor *kunstmatige intelligentie*, een heel boeiend aspect van de informatica.

Een functie die recursie gebruikt, is een functie die *zichzelf* aanroept om een bepaald resultaat te behalen (een oud grapje zegt dat in een verklarend woordenboek informatica te lezen staat : *recursie : zie recursie*). Dat klinkt aardig ingewikkeld, maar het is vrij eenvoudig.

Een klassiek voorbeeld van recursie is de *faculteit*. De faculteit van 5 bijvoorbeeld is 5 * 4 * 3 * 2 * 1. Nu is het duidelijk dat de faculteit van 5 gelijk is aan 5 vermenigvuldigd met de faculteit van 4. De faculteit van 4 is op zijn beurt weer gelijk aan 4 keer de faculteit van 3, enzovoort. De faculteit van 0 is per definitie 1. De functie ziet er als volgt uit :

Functie 16.5 : FAC

```
1 FAC: (F)
2 IF F = 0
3    RETURN (1)
4 ENDIF
5 RETURN (F * FAC: (F-1) )
```

Deze functie gebruikt zichzelf dus zo vaak als nodig is.

Bij deze functie is het niet echt nodig recursie toe te passen. Het is dus niet de enige oplossing, maar wel de elegantste. Recursie heeft echter ook nadelen : het is niet altijd gemakkelijk fouten op te vangen die zich in recursieve functies voordoen, en sommige mensen vinden recursie moeilijk

te doorgronden. Wie echter tijd besteedt aan goede bestudering van recursie en er verantwoordelijk mee omspringt, kan er ingewikkelde vraagstukken heel eenvoudig mee oplossen.

Een klassiek voorbeeld dat wordt opgelost met recursie is het *acht-koninginnen-probleem*. De opdracht : plaats 8 koninginnen op een schaakbord en wel zo, dat geen enkele van die koninginnen een andere kan slaan (pro beer dat eens zonder computer !).

17. Bestanden

Instructies : APPEND, CLOSE, COPY, COPYW, CREATE, DELETE, DELETEW, ERASE, FIRST, NEXT, OPEN, POSITION, RENAME, UPDATE, USE

Functies : COUNT, DISP, EXIST, FIND, FINDW, RECSIZE, SPACE, DIR$, DIRW$

17.1 Inleiding

Twee van de meestgebruikte functies van de Organiser II zijn het opslaan en weer opzoeken van gegevens met FIND en SAVE. Een opsomming van bij elkaar horende gegevens (naam, telefoonnummer, adres) noemen we een *record*. We hebben onze Organiser II dus al volgepropt met records; namen en adressen van vrienden, bedrijven, bepaalde belangrijke data, ... Een record bestaat vaak zelf uit verschillende gegevens (naam, functie, ...). Zo'n enkel gegeven noemen we een *veld* (Engels: *field*). Records bestaan dus uit één of meer velden.

Al die records worden op een pak bijgehouden, en wel in een zekere volgorde en in een zekere vorm. Een verzameling bij elkaar horende records noemen we een *bestand* (Engels : *file*). Iedere pak waarop we gegevens hebben opgeslagen bevat dus een bestand van records, net als een kaartenbak een verzameling kaarten bevat. Er is echter wel een beperking : ieder record mag niet meer dan 254 tekens bevatten.

We kunnen ook nog andere bestanden creëren, die we met onze programma's beheren. We kunnen als handelsbedrijf bijvoorbeeld een programma schrijven dat een prijslijst beheert (artikelen toevoegen, verwijderen, prijzen veranderen)... Die prijslijst is dan niets anders dan een bestand : een verzameling (rij) bij elkaar horende gegevens van artikelen.

Die bestanden hebben dan ook een naam nodig (zoals een etiket op een kaartenbak). Een prijslijstenbestand kan bijvoorbeeld de naam PRIJZEN krijgen, een bestand van Franse woorden met hun Nederlandse vertaling kan FRANSNL heten, en ga zo maar door. Het spreekt vanzelf dat een bepaalde bestandsnaam per pak maar één keer kan voorkomen.

De bestanden die we reeds gebruiken, nl. die door FIND en SAVE worden beheerd, heten trouwens MAIN. De gegevens op pak A: staan dus in het bestand A:MAIN, die op pak B: in het bestand B:MAIN, en het bestand op pak C: kreeg de naam C:MAIN. Om die gegevens via het hoofdmenu te gebruiken, hoeven we dit niet te weten : de Organiser II regelt dit voor ons.

Het is vanzelfsprekend, dat OPL de nodige instructies en functies bevat die ons in staat stellen bestanden te creëren en te manipuleren. Om hun gebruik duidelijk te illustreren, creëren we een situatie waarin we een bestand gaan gebruiken.

17.2 De prijslijst als voorbeeld

De firma '*Zwart & Morbide*', een nieuw bedrijf dat hypermoderne kunststof doodskisten vervaardigt, heeft ons gevraagd een volledig programma te schrijven dat hun prijslijsten bijhoudt en dat hen in staat stelt nieuwe produkten toe te voegen en prijzen te wijzigen. Zo'n aanbod slaan wij niet af en we beginnen er dadelijk mee.

Allereerst moeten we natuurlijk weten wat dit programma allemaal moet doen. De lijst met taken is als volgt :

- een prijzenbestand creëren
- een prijs van een produkt opvragen
- artikelen toevoegen
- artikelen verwijderen
- prijzen wijzigen

Later kunnen er eventueel nog andere taken bijkomen. De eerste taak, het creëren van een prijzenbestand, zou eigenlijk het eerste moeten zijn wat ons programma doet.

17.3 CREATE

De instructie CREATE dient om nieuwe bestanden te creëren. Daarvoor moeten we natuurlijk wel iets meer weten over het bestand, nl. de *naam* en de *structuur* (uit welke velden het bestaat). Ook die velden krijgen een naam.

Recordstructuur van het bestand KISTEN :

- kistnaam
- prijs

(We houden het eenvoudig). Is de prijs integer of reëel ? Dat hangt er vanaf of we met Nederlandse guldens of met Belgische franken rekenen. Laten we maar allemaal *reëel* gebruiken, omdat dit type het voordeel heeft praktisch onbegrensd te zijn (en het zijn dure kisten).

We gaan dit bestand creëren met CREATE.

Syntax : CREATE <exp$>,<logische naam>,<veldnaam>
{[,<veldnaam>]}

<exp$> bevat de *naam* van het bestand.

<logische naam> is een afkorting bestaande uit één letter, A, B, C of D, die we later gebruiken om naar het bestand te verwijzen. Let op dat de logische naam *niet* mag worden omgeven door haakjes of aanhalingstekens!

<veldnaam> bevat uiteraard de naam van het gebruikte veld. Er kunnen verschillende velden achter elkaar worden opgesomd. In deze volgorde worden de gegevens door de Organiser II opgeslagen. Zo'n veld wordt behandeld als een variabele en kan dus bijvoorbeeld ook KIST$ of PRIJS heten. Iedere veldnaam moet in zijn bestand uniek zijn. De lengte hoeft de lengte niet opgegeven te worden.

Voorbeelden van CREATE :

```
CREATE "A:KISTEN",A,KIST$,PRIJS
CREATE "C:TELEFOON",B,NAAM$,TEL$,ADR$
CREATE "ZEGELS",D,NAAM$,NUMMER%,PRIJS
```

Voor onze opdracht zullen we uiteraard het eerste voorbeeld gebruiken.

Het spreekt voor zich dat een bepaald bestand slechts één keer kan worden gecreëerd. Is dat eenmaal gebeurd, dan is het bestand klaar voor gebruik tot het programma beëindigd wordt. Wanneer we het bestand later nog wensen te gebruiken, moeten we het niet meer creëren, maar *openen*. Het bestand wordt geopend zoals een boek wordt geopend als we daarin willen lezen.

17.4 OPEN

Om bestanden voor gebruik te openen, gebruiken we de instructie OPEN. De syntax is vrijwel identiek aan die van CREATE :

Syntax : OPEN <exp$>,<logische naam>,<veldnaam>
{[,<veldnaam>]}

Voor een volledige bespreking van deze syntax, zie punt 29.3.

Alleen reeds bestaande bestanden kunnen worden geopend. Een programma kan overigens slechts vier bestanden tegelijk open hebben (vandaar de logische namen A, B, C en D). Omdat we de bewerkingen die we

op de bestanden gaan uitvoeren niet op twee bestanden tegelijkertijd kunnen doen, gaan we tussen die mogelijke vier kiezen. We kunnen dus, om bij dat eerder genoemde boek te blijven, vier boeken open hebben, maar niet uit twee tegelijk lezen. Om te bepalen op welk bestand de volgende handelingen moeten aangrijpen, gebruiken we de volgende instructie :

17.5 USE

USE betekent 'gebruik'. De benaming komt waarschijnlijk van het bestandenpakket dBASE dat op personal computers draait.

Syntax : USE <logische naam>

Om aan te duiden welk bestand wordt gebruikt, zetten we USE A, USE B, USE C of USE D in ons programma waar dat nodig is. Vanzelfsprekend moet de opgegeven logische naam wel in gebruik zijn. USE moet eigenlijk niet worden gebruikt wanneer het programma maar één bestand tegelijk open heeft (zoals in onze opdracht het geval is). Om bijvoorbeeld het reeds bestaande bestand A: KISTEN gebruiksklaar te maken, programmeren we:

```
OPEN "A:KISTEN",A,KIST$,PRIJS
USE A
```

Wanneer het programma het bestand heeft gebruikt en niet meer nodig heeft (tot de volgende keer) wordt het bestand *gesloten*. Het wordt niet verwijderd, maar slechts tijdelijk niet meer gebruikt, net als dat boek. Het wordt gesloten en opzij gelegd om er later verder in te lezen.

17.6 CLOSE

Daarvoor dient CLOSE.

Syntax : CLOSE

Close neemt geen parameters, maar het *sluit* het bestand dat momenteel in gebruik is. Dit bestand kan dan door het programma niet meer worden gebruikt zonder dat het eerst weer wordt geopend (een zoveelste verwijzing naar het boek maakt duidelijk waarom).

17.7 Detecteren van reeds bestaande bestanden : EXIST

Soms wanneer we een bestand willen creëren, komt het voor dat dat bestand al bestaat. Het kan dan uiteraard niet meer gecreëerd worden (de Organiser II rapporteert een fout en het programma ligt stil). Het is echter mogelijk om alvorens een bestand te creëren eerst even te kijken of het al bestaat en zo ja, het te *openen* i.p.v. het te *creëren*. Dit omzeilt een eventuele fout. De functie EXIST komt ons in dergelijke gevallen te hulp.

EXIST (bestaat)
Gebruik : x% = EXIST(<exp$>)
of : IF EXIST(<exp$>)
    ...
    ENDIF

De stringexpressie bevat de naam van een bestand (bijvoorbeeld "A:MAIN"). Als dat bestand reeds bestaat, levert EXIST de waarde -1 op (waar). Als het nog niet bestaat, is het resultaat 0 (niet waar). Om een bestand te creëren zullen we voorzichtigheidshalve dus steeds de werkwijze van het nu volgende voorbeeld volgen :

```
IF NOT EXIST("A:KISTEN")
  CREATE "A:KISTEN",A,KIST$,PRIJS
ELSE
  OPEN "A:KISTEN",A,KIST$,PRIJS
ENDIF
```

17.8 Invoer en toevoeging aan het bestand van records : INPUT en APPEND

We schrijven reeds een module van ons programma voor de doodskistenfabrikanten. Deze module geeft iemand van dat bedrijf de mogelijkheid via het klavier records in te voeren.

Programma 17.1 : INKIST

```
1 INKIST:
2 LOCAL NOG$(1)
3 OPEN "A:KISTEN",A,KIST$,PRIJS
4 USE A
5 NOG$ = "J"
6 WHILE UPPER$(NOG$) = "J"
7   CLS
```

```
 8   PRINT "Kistnaam : ",
 9   INPUT A.KIST$
10   PRINT "Prijs : ",
11   INPUT A.PRIJS
12   APPEND
13   CLS
14   PRINT "Nog (J/N) "; CHR$(63)
15   INPUT NOG$
16 ENDWH
17 CLOSE
```

In regel 9 wordt het veld KIST$ van het bestand met de logische naam A ('logisch bestand A') via het klavier opgevraagd met de klassieke INPUT-instructie. Hetzelfde zien we terug in regel 11. Die twee velden hebben nu in het *geheugen* van de Organiser II een inhoud gekregen. Die inhoud is echter nog niet bij het bestand geschreven. Daarvoor dient de nieuwe instructie APPEND die we in regel 12 zijn tegengekomen.

Syntax : APPEND

Deze instructie neemt de huidige waarden van de velden van het open bestand (in dit geval A) en voegt die achteraan het bestand toe. APPEND is het Engelse woord voor 'achteraan bijvoegen'. Deze instructie maakt nog maar eens duidelijk dat Engelstalige mensen het veelal gemakkelijker hebben om hun weg te vinden in de informatica dan anderen.

17.9 Hoeveel records ?

Om sommige taken goed te kunnen uitvoeren, is het nodig te weten uit hoeveel records het bestand bestaat. OPL is dan ook uitgerust met twee functies daartoe. COUNT (Engels voor 'tel') geeft weer hoeveel records het bestand bevat en EOF (afkorting voor *end-of-file*; einde van het bestand) geeft aan wanneer het einde van het bestand bereikt is.

COUNT (tel)
Gebruik : x% = COUNT
Telt het aantal records in het bestand. Merk op dat COUNT een functie is en géén variabele!

EOF (einde van het bestand)
Gebruik : x% = EOF
Geeft weer of het einde van het bestand al dan niet bereikt is : -1 (waar) of 0 (niet waar).

Programma 17.2 : TELREC

Dit programma telt het aantal records in pak A:.

```
1 TELREC:
2 LOCAL Tpc
3 OPEN "A:MAIN",A,V$
4 USE A
5 PRINT COUNT,"records"
6 CLOSE
7 GET
```

In regel 3 wordt slechts rekening gehouden met één veld per record. Het is onnodig met meerdere velden rekening te houden omdat de functie COUNT onder alle omstandigheden het juiste aantal records weergeeft.

Een voorbeeld van EOF volgt later

17.10 Positionering in het bestand : FIRST, LAST en POSITION

Ieder record heeft een volgnummer. Het laatst bijgevoegde record heeft het hoogste volgnummer, het eerste heeft nummer 1. De instructie FIRST (Engels voor 'eerste') vertelt aan de Organiser II dat we willen werken met het eerste record.

Syntax : FIRST

Een voorbeeldprogramma :

Programma 17.3 : FIRSTVB

Dit programma neemt het bestand A:KISTEN, en brengt het eerste record op het scherm :

```
1 FIRSTVB:
2 OPEN "A:KISTEN",A,KIST$,PRIJS
3 USE A
4 FIRST
5 PRINT A.KIST$
6 PRINT A.PRIJS
7 CLOSE
8 GET
```

Regels 5 en 6 brengen de inhoud van respectievelijk het kistveld en het

prijsveld van het eerste record op het scherm. Dat het hier het eerste record betreft wordt geïnstrueerd in regel 4.

Er is ook een instructie voorradig om te werken met het *laatste* record i.p.v. het eerste. Die instructie luidt LAST.

Syntax : LAST

Het gebruik is overeenkomstig met dat van FIRST.

Programma 17.4 : LASTVB

Dit programma is bijna identiek aan het vorige. Het gebruik is echter een beetje anders : het geeft op het scherm de eerste twee regels van het laatste record van het bestand A:MAIN.

```
1 LASTVB:
2 LOCAL V%
3 OPEN "A:MAIN",A,V1$,V2$
4 USE A
5 LAST
6 V% = VIEW(1,A.V1$)
7 V% = VIEW(2,A.V2$)
8 CLOSE
```

Op deze manier kunnen dus het eerste en het laatste record aangesproken worden. Als OPL hiertoe beperkt was gebleven, zou onze opdracht vrijwel nooit kunnen worden uitgevoerd : hoe gaan we immers al die andere records gebruiken ?

Men heeft enkele oplossingen voor dit probleem bedacht en de eerste heet POSITION.

Syntax : POSITION <exp>

Deze instructie vertelt de Organiser II dat het volgende record waarmee we gaan werken record nummer <exp> is : deze expressie stelt dus eigenlijk een *volgnummer* voor (zie hierboven). Die expressie moet natuurlijk wel binnen de grenzen van het mogelijke blijven : niet- bestaande volgnummers leveren foutmeldingen op.

Hoe werkt deze positionering eigenlijk? Binnenin de Organiser II, in het interne geheugen, is een plaats gereserveerd waar een aantal gegevens over geopende bestanden wordt opgeslagen. Dit zijn onder andere de naam, de logische naam en het volgnummer van het huidige werkrecord. Er wordt hier dus eigenlijk naar een bepaald record *verwezen* : vandaar dat men dit

gegeven de *recordwijzer* (Eng. : *record pointer*) noemt. Wat FIRST, LAST en POSITION dus eigenlijk doen, is enkel die recordwijzer veranderen.

17.11 Doorbladeren van het bestand : NEXT en BACK

NEXT en BACK zijn instructies die het mogelijk maken stapsgewijs een bestand te doorlopen. Ook deze instructies wijzigen de recordwijzer. De syntax is voor beide instructies even eenvoudig :

Syntax : NEXT

'Spring' naar het volgende record.

Syntax : BACK

'Spring' naar het vorige record.

Programma 17.5 : EOFVB

Dit programma drukt van ieder record in pak A: de eerste regel af.

```
 1 EOFVB:
 2 CLS
 3 OPEN "A:MAIN",A,V$
 4 USE A
 5 FIRST
 6 PRINT A.V$
 7 WHILE NOT EOF
 8    GET
 9    NEXT
10    CLS
11    PRINT A.V$
12 ENDWH
13 CLOSE
```

In regel 3 wordt duidelijk gemaakt dat alleen het eerste veld ons interesseert (er wordt maar één veld opgegeven in de opening van het bestand). In regel 5 wordt gezegd dat we beginnen met het eerste record. De conditie EOF wordt toegepast in een WHILE : zolang EOF niet -1 oplevert, en het einde van het bestand dus niet bereikt is, gaat de verwerking voort. Wanneer dit einde wel bereikt is gaat het programma verder na de ENDWH (dus springt naar regel 13).

17.12 Muteren van records : UPDATE

Muteren is niets anders dan *veranderen*. Het kan voorkomen dat een bepaald record niet meer actueel is en moet worden gemuteerd (zoals in onze opdracht : het aanpassen van de prijzen in prijslijsten).

De inhoud van het record dat door de recordwijzer wordt aangewezen kan worden veranderd met de instructie UPDATE (Engelse term, die 'maak actueel' betekent).

Syntax : UPDATE

Neemt de huidige inhouden van de velden (bijvoorbeeld A. KIST$) en wijzigt die in het bestand.

Een voorbeeld :

Programma 17.6 : MUTEER record X%

Nou ja, programma? Het is meer een module! Het geopende bestand A: KISTEN (logische naam A) bevat de velden KIST$ en PRIJS. Het veld PRIJS in record nummer X% moet een nieuwe inhoud krijgen.

```
1 MUTEER: (X%)
2 POSITION X%
3 PRINT A.KIST$
4 PRINT CHR$(63),
5 INPUT A.PRIJS
6 UPDATE
```

De recordwijzer krijgt de waarde X% in regel 2. Dan wordt de oude inhoud van het eerste veld op het scherm gebracht. In regel 4 drukt de functie een vraagteken (CHR$(63)) en een spatie (vandaar de komma) op het scherm, om de gebruiker duidelijk te maken dat hij voor dit type kist een nieuwe prijs moet opgeven. Die prijs wordt dan opgevraagd in regel 5 en het nieuwe record wordt naar het bestand geschreven op regel 6. Dan keert de Organiser II terug naar het punt waar de procedure opgeroepen is.

17.13 Verwijderen van records : ERASE

Het hoofdmenu van de Organiser II bevat de optie ERASE. De bedoeling van ERASE is bepaalde records uit het bestand en dus van de pak te verwijderen. Deze mogelijkheid is ook in OPL ingebouwd en wel onder dezelfde naam : ERASE.

Syntax : ERASE

Wist het huidige record.

Het volgende voorbeeld verwijdert het eerste record uit het bestand A: KISTEN :

Programma 17.7 : WEG1

```
1 WEG1:
2 OPEN "A:KISTEN", A, V$
3 USE A
4 FIRST
5 ERASE
6 CLOSE
```

Dit verwijderen kan een tijdje duren, aangezien alle volgende records een plaatsje naar voren moeten worden geschoven om het plotselinge 'gat' weer dicht te maken (anders neemt het weggeveegde record nog onnodig plaats in).

17.14 De grootte van records : RECSIZE

Niet alle records zijn even lang. In ons voorbeeld kunnen er misschien kisten zijn met een lange naam en kisten met een korte naam. Soms kan het van belang zijn de grootte van een record te weten. Daarvoor heeft men de functie RECSIZE in het leven geroepen.

RECSIZE
Gebruik : x% = RECSIZE
Geeft weer uit hoeveel bytes het huidige werkrecord bestaat. Dit aantal mag uiteraard de 254 niet overschrijden.

De functie wordt bijvoorbeeld gebruikt wanneer de gebruiker records bijvoegt. Allereerst berekent RECSIZE of de grootte van het record de 254 niet overschrijdt. Is dat wel het geval, dan moeten nieuwe waarden worden opgevraagd. Normaliter kan zoiets niet voorkomen omdat men aan het begin van de creatie of de opening van het bestand rekening moet houden met de maximumlengte.
Het is echter wel nuttig om, met behulp van de volgende functie, te kijken of er nog wel voldoende *plaats* is op de pak om het record toe te voegen. Wanneer pak A: nog slechts 20 bytes vrij heeft, heeft het immers geen zin te proberen een record van 200 bytes bij te schrijven! Een praktijkvoorbeeld volgt zodadelijk.

17.15 Vrij geheugen op paks : SPACE

SPACE
Gebruik : x = SPACE
De functie SPACE geeft weer hoeveel bytes vrij geheugen er nog zijn op de huidige pak. Het resultaat is reëel en niet integer omdat er datapaks met capaciteiten van 64K en 128K verkrijgbaar zijn.

Programma 17.8 : INKIST, tweede versie

Ziehier de tweede versie van programma 29.1 (zie het begin van dit hoofdstuk). In deze versie wordt gecontroleerd of er nog wel ruimte is om het ingetikte record weg te schrijven.

```
 1 INKIST:
 2 LOCAL NOG$(1)
 3 OPEN "A:KISTEN",A,KIST$,PRIJS
 4 USE A
 5 NOG$ = "J"
 6 WHILE UPPER$(NOG$) = "J"
 7    CLS
 8    PRINT "Kistnaam : ",
 9    INPUT A.KIST$
10    PRINT "Prijs : ",
11    INPUT A.PRIJS
12    IF SPACE >= RECSIZE
13         APPEND
14    ELSE
15       PRINT "Geen ruimte"
16       GET
17    ENDIF
18    CLS
19    PRINT "Nog (J/N) "; CHR$(63)
20    INPUT NOG$
21 ENDWH
22 CLOSE
```

Die controle vindt plaats in de regels 12 tot 16. De nog vrije ruimte (SPACE) moet groter dan of gelijk aan de grootte van het record (RECSIZE) zijn.

Programma 17.9 : VRIJ

Dit voorbeeldprogramma is een variant van de optie INFO van het hoofdmenu. In plaats van percentages brengt geeft het (onderaan op het scherm)

per aangesloten pak het aantal nog vrije bytes. De functie EXIST wordt gebruikt om na te gaan of pak B: en C: al dan niet zijn aangesloten.

```
 1 VRIJ:
 2 LOCAL C$(32),V%
 3 OPEN "A:MAIN",A,R$
 4 USE A
 5 C$ = "PAK A: "+GEN$(SPACE,5)
 6 CLOSE
 7 IF EXIST("B:MAIN")
 8    OPEN "B:MAIN",A,R$
 9    USE A
10    C$=C$+" B: "+GEN$(SPACE,6)
11    CLOSE
12 ENDIF
13 IF EXIST("C:MAIN")
14    OPEN "C:MAIN",A,R$
15    USE A
16    C$=C$+" C: "+GEN$(SPACE,6)
17    CLOSE
18 ENDIF
19 CLS
20 PRINT "GEHEUGEN VRIJ : "
21 V% = VIEW(2,C$)
```

De openingen in de regels 3, 8 en 14 definiëren MAIN steeds als een bestand met slechts één veld : dit doen we omdat de velddefinitie in een procedure als deze totaal onbelangrijk is (het gaat immers om de functie SPACE, die met recorddefinitie helemaal niets te maken heeft).

We gebruiken de functie EXIST, zoals reeds gezegd, om uit te zoeken of pak B: en/of C: aanwezig is/zijn (regels 7 en 13). Dit is voor pak A: niet nodig, omdat het altijd bestaat.

De hoeveelheid vrije ruimte wordt aan de string C$ toegevoegd (regels 5, 10 en 16) met behulp van de GEN$- functie. Bij pak A: wordt slechts rekening gehouden met een grootte van vijf cijfers terwijl we bij B: en C: zes cijfers opgeven. De reden is eenvoudig : de maximumhoeveelheid bytes in pak A: is 32768 (32 K, 5 cijfers), in pak B: en C: is dat 131072 bytes (128 K, 6 cijfers).

In regel 21 wordt de resultrende string C$ op de tweede regel van het scherm gebracht. Wanneer de gebruiker dan een toets indrukt, wordt de procedure beeindigd.

17.16 Bestanden verwijderen : DELETE

Wanneer een bepaald bestand niet meer wordt gebruikt en dus overbodig aanwezig is, kan het worden uitgewist. Dit heeft bij RAMpaks en het interne geheugen het bijkomende voordeel dat er ruimte wordt vrijgemaakt. De instructie om een bestand uit te wissen is DELETE.

Syntax : DELETE <exp$>

De stringexpressie houdt de volledige naam in van het uit te wissen bestand, bijvoorbeeld :

```
DELETE "A:KISTEN"
DELETE "B:MAIN"
```

Zowel bij data- als RAMpaks (en dus ook voor het interne geheugen) is de schade die DELETE veroorzaakt onherstelbaar. Een uitgewist bestand is nooit meer te herstellen! Het is wel mogelijk dit bestand opnieuw te creëren, maar in dat geval zijn natuurlijk alle oude gegevens verdwenen. Opletten is dus de boodschap.

17.17 Copiëren van bestanden : COPY

Net als we datapaks kunnen copiëren (met de optie COPY vanuit het hoofdmenu), is het ook mogelijk aparte bestanden te copiëren. De instructie die dit voor ons doet heet eveneens COPY.

Syntax : COPY <exp$1>, <exp$2>

Beide stringexpressies bevatten een volledige bestandsnaam of een paknaam. Het bestand in <exp$1> wordt gecopiëerd en opgeslagen onder de benaming in <exp$2>. Wanneer het bestand in <exp$2> nog niet bestaat wordt het gecreëerd, wanneer het wel bestaat wordt het overschreven. Voorbeelden :

```
COPY "A:MAIN","B:GEGS"
COPY "B:KISTEN","C:PLIJST"
```

In het eerste voorbeeld wordt het bestand A:MAIN dus gecopiëerd naar pak B: onder de naam B:GEGS. In het tweede voorbeeld wordt het bestand B:KISTEN gecopiëerd naar pak C: onder de naam C:PLIJST.

Wanneer in <exp$2> enkel een paknaam wordt gegeven, zoals in het volgende voorbeeld, dan blijft de bestandsnaam ongewijzigd :


```
COPY "A:MAIN","B:"
```

is dus identiek aan :

```
COPY "A:MAIN","B:MAIN"
```

Wanneer enkel paknamen worden opgegeven, worden *alle* bestanden van de pak in <exp$1> gecopiëerd naar de pak in <exp$2> onder dezelfde naam :

```
COPY "A:","B:"
```

Let op! Een bestand moet *altijd* worden gecopiëerd naar een *andere* pak. Het volgende voorbeeld is dus niet toegestaan :

```
COPY "A:MAIN","A:BACKUP"
```

Het is echter wel mogelijk, via een achterdeurtje als het ware, om een bestand naar dezelfde pak te copiëren. Stel dat we het bestand A:KISTEN willen copiëren naar A:COPIE. De te volgen procedure is dan :

Programma 17.10 : COPIE

```
1 COPIE:
2 COPY "A:KISTEN","B:XXX"
3 COPY "B:XXX","A:COPIE"
4 DELETE "B:XXX"
```

Eerst maken we van A:KISTEN een tijdelijke copie naar pak B: onder de naam B:XXX. Dit bestand wordt dan terug gecopiëerd naar A: onder de nieuwe naam, A:COPIE. Het bestand B:XXX, dat we nu niet meer nodig hebben, wordt vervolgens (regel 4) gewist.

17.18 Bestanden hernoemen : RENAME

Het is mogelijk al bestaande bestanden een andere naam te geven. Het bestand A:KISTEN kan dan A:PRIJZEN gaan heten, of iets dergelijks.

Syntax : RENAME <exp$1>, <exp$2>

De oude naam van het bestand staat in <exp$1>; in <exp$2> wordt de nieuwe naam opgegeven. Voorbeelden :

```
RENAME "A:KISTEN","PRIJZEN"
RENAME "A:MAIN","A:GEGS"
```

Het is eigenlijk volstrekt onnodig om de paknaam in <exp$2> te noemen, zoals in het tweede voorbeeld. Een andere paknaam opgeven dan die in <exp$1> is trouwens compleet onlogisch en niet toegestaan. Nog een opmerking : de naam van MAIN veranderen (op welke pak dan ook), brengt de Organiser II niet in de war : hij zal met FIND en SAVE blijven werken als vanouds!

17.19 Inhoudsopgave van bestanden : DIR$

Wie regelmatig met bestanden werkt, zal het op prijs stellen dat er in OPL een functie aanwezig is die het mogelijk maakt de namen te zien van alle bestanden die op pak staan. Een dergelijke functie komen we ook tegen in de DIARY- en PROGmenu's, nl. DIR. Omdat het hier strings betreft, heet het OPL-equivalent DIR$.

DIR$ (inhoudsopgave)
Gebruik : x$ = DIR$(<exp$>)
De stringexpressie kan een paknaam bevatten (dus A: , B: of C:) en in dat geval is het resultaat van de functie de naam van het *eerste* bestand op die pak. Wanneer de stringexpressie *leeg* is (""), is het resultaat steeds het *volgende* bestand, tot er geen bestanden meer zijn, dan is het resultaat de lege string zelf ("").

Programma 17.11: BSTDIR

Het nu volgende, overigens zeer bruikbare, voorbeeld geeft op het scherm een lijst van alle bestanden op pak in typische Organiser II-stijl.

```
 1 BSTDIR:
 2 LOCAL D$(8),PAK$(1)
 3 CLS
 4 PRINT "PAK : ",
 5 INPUT PAK$
 6 D$ = DIR$(PAK$+":")
 7 WHILE D$ <> ""
 8    PRINT D$
 9    GET
10    D$ = DIR$("")
11 ENDWH
```

*D$* heeft als maximumlengte 8 omdat bestandsnamen nooit langer dan 8 tekens mogen zijn (regel 2). PAK$ heeft lengte 1 en mag dus 1 letter bevatten (A, B of C). Deze variabele dient om te bepalen met welke pak we willen werken en wordt opgevraagd in regel 5. Dan wordt aan de Organiser II de eerste bestandsnaam opgevraagd en in de variabele D$ gezet (regel 6). Zolang D$ niet leeg is (en het einde van de reeks bestanden dus niet bereikt is), wordt de bestandsnaam op het scherm afgedrukt (regel 8), wacht de Organiser II op een toets (regel 9) en wordt de volgende bestandsnaam opgevraagd (regel 10).

Let op : deze functie kan enkel worden gebruikt om namen van gewone bestanden op te vragen; namen van procedures of het DIARY-bestand worden *niet* gemeld en zijn via OPL *niet* beschikbaar.

17.20 FIND in OPL

Ook de veelgebruikte functie FIND vinden we in OPL terug. Deze functie dient, net als haar naamgenoot in het hoofdmenu, om een string op te zoeken in een bestand.

FIND
Gebruik : x% = FIND(<exp$>)
In het momenteel gebruikte bestand wordt *voorwaarts* naar <exp$> gezocht *vanaf het huidige record*, tot de string gevonden is of het einde van het bestand is bereikt. Indien <exp$> gevonden wordt, is het resultaat van FIND het volgnummer van het betreffende record. Bovendien voert de Organiser II dan automatisch een POSITION x% uit (de recordwijzer wordt op het volgnummer van het record gezet). Als de string echter niet gevonden wordt, is het resultaat van FIND nul en blijft de recordwijzer onveranderd.

Opmerkelijk detail : wanneer <exp$> de lege string is ("") wordt er niet gezocht maar voert de Organiser II een NEXT-instructie uit!

Programma 17.12 : ZOEKKIST

Zoals de naam impliceert, zoekt het volgende voorbeeld naar een bepaalde kistnaam en presenteert daarna kistnaam en prijs op het scherm.

```
 1 ZOEKKIST:
 2 LOCAL K$(16),RW%
 3 CLS
 4 PRINT "-Opzoeken kist-";
 5 OPEN "A:KISTEN",A,KIST$,PRIJS
 6 USE A
 7 PRINT CHR$(63),
 8 INPUT K$
 9 RW% = FIND(K$)
10 CLS
```

```
11 IF RW%
12    PRINT A.KIST$
13    PRINT A.PRIJS
14 ELSE
15    PRINT "Niet aanwezig"
16 ENDIF
17 GET
```

K$ bevat de naam van de op te zoeken kist, RW% bevat de record-wijzer na de 'zoekactie' in regel 9. Als RW% dus niet 0 is worden de naam en de prijs op het scherm gebracht (regels 11 tot 13), anders is de betreffende kist niet aanwezig (regels 14 en 15).

In regels 12 en 13 worden de velden op het scherm afgedrukt. Dit hebben we al vaker gedaan. Er is echter een veel eenvoudiger en doeltreffender manier om dit te doen.

17.21 Bekijken van records : DISP

DISP is de afkorting voor *display* (Engelse term voor 'ontvouwen' of 'uitstallen'). De Organiser II gebruikt deze functie o.a. bij FIND in het hoofdmenu.

DISP (uitstallen)
Gebruik : x% = DISP(<exp%>,<exp$>)
De vorm van DISP kent drie variaties, die worden aangegeven door <exp%>. Deze expressie mag de waarden -1, 0 of 1 bevatten.

-1 : de stringexpressie (<exp$>) wordt compleet genegeerd (en zal dan ook vaak leeg zijn) en het huidige record (het record dat door de recordwijzer wordt aangeduid) wordt op het scherm 'uitgestald' met één veld per regel. De gebruiker kan met behulp van de cursortoetsen (LINKS, RECHTS, OP en NEER) het record doorlopen (net als bij FIND in het hoofdmenu). De functie wordt verlaten wanneer er een *andere* toets wordt ingedrukt. De ASCII-code van die toets (zie appendix A) wordt dan het resultaat van de functie DISP (in het gebruiksvoorbeeld hierboven krijgt de variabele x% dus die code).

Voorbeeld :

```
d% = DISP(-1,"")
```

1 : niet het huidige record, maar de string in <exp$> wordt op het scherm

uitgestald (op precies dezelfde wijze als bij de waarde -1 beschreven is). De string, die eigenlijk uit slechts 1 regel bestaat, kan in verschillende 'velden' worden verdeeld door op het einde van ieder 'veld' een *tabulatorteken* (CHR$(9), zie Appendix A) te plaatsen.
Voorbeeld :

```
d% = DISP(1,"Een"+chr$(9)+"Twee"+chr$(9)+"Drie")
```

0 : de laatst uitgestalde string/het laatst uitgestalde record wordt herhaald. De stringexpressie wordt genegeerd (en zal dus, net zoals bij -1, meestal leeg zijn).

Voorbeeld :

```
d% = DISP(0,"")
```

Belangrijk : deze laatste vorm van DISP kan slechts worden toegepast wanneer er tussen twee DISPs geen andere uitvoer naar het scherm wordt gebracht (bijvoorbeeld met PRINT). Wanneer dit wel het geval is, zal DISP niet correct werken en is de afloop niet voorspelbaar. *Het kan funest zijn voor het interne geheugen.*

In bovenstaand programma 29.10 kunnen regels 12 en 13 vervangen worden door :

```
RW% = DISP(-1,"")
```

Omdat het gebruik van de functie DISP niet beperkt is tot het uitstallen van records, is het ook een leuk alternatief voor bijvoorbeeld VIEW of PRINT.

Kunt u nu de opdracht van '*Zwart & Morbide*' vervullen ?

17.22 Bestandsfuncties en -instructies voor de LZ

Zoals kon worden verwacht, biedt deze nieuwe machine een aantal nieuwe bestandsfuncties en -instructies. het betreft hier COPYW, DELETEW, FINDW en DIRW$. Deze nieuwe instructies en functies dienen om te kunnen werken met elk type bestand, niet alleen met gewone databestanden (zoals MAIN). Om aan te geven welk soort bestand moet worden gecopieerd, gebruikt OPL extensies, net zoals bekende besturingssystemen als MS-DOS, CP/M en UNIX. De extensies zijn de volgende:

.ODB	Organiser Data Base : een gewoon bestand;
.OPL	OPL-procedure;
.NTS	Notepad-bestand (schrijfblok);
.COM	COMMS LINK setup-bestand (zie later);
.PLN	Pocket Spreadsheet-bestand;
.PAG	Pager setup-bestand;
.DIA	Agendabestand (diary file) van CM of XP;
.OPT	OPL-procedure, alleen tekstversie;
.OPO	OPL-procedure, alleen object-versie (Q-Code).

17.22.1 COPYW

COPYW (de W staat voor *wildcard*) dient uiteraard om één of meerdere bestanden te copiëren.

Syntax : COPYW "pak1:naam1.ext", "pak2:naam2.ext"
COPYW "pak1:naam1.ext", "pak2:"
COPYW "pak1:", "pak2:"

In het eerste geval wodt het bestand pak1:naam1.ext (bijvoorbeeld A:IDEE.NTS) gecopieerd naar pak2 (bijvoorbeeld B) onder de naam 2.ext (bijvoorbeeld GOED.NTS). De instructie zou dan luiden :

```
COPYW "A:IDEE.NTS", "B.GOED.NTS"
```

In het tweede geval wordt het bestand pak1:naam1.ext naar pak2 gecopieerd. De volgende instructies doen dus hetzelfde:

```
COPYW "A:IDEE.NTS", "B:"
COPYW "A:IDEE.NTS", "B:IDEE.NTS"
```

In het derde geval worden alle bestanden op pak1 en pak2 gecopieerd. De volgende instructies doen dus hetzelfde :

```
COPYW "A:*.*", "B:"
COPYW "A:", "B:"
```

17.22.2 DIRW$

DIRW$ werkt net als DIR$, behalve dat we een bestandsnaam met extensie moeten opgeven.

Gebruik : `x$ = DIRW$("pak:naam.ext")`

17.22.3 DELETEW

DELETEW dient, u raadt het al, om een aantal bestanden te wissen.

Syntax: DELETEW "pak:naam.ext"

Om bijvoorbeeld alle OPL-procedures op pak C: te wissen (zowel de tekstvorm als de Q-Code) gebruiken we :

```
DELETEW "C:*.OPL"
```

17.22.4 FINDW

De laatste functie, FINDW, laat ons toe in het geopende bestand te gaan zoeken met wildcards.

Gebruik : `x% = FINDW(<exp$>)`

De <exp$> bevat de stringexpressie die we gaan opzoeken, bijvoorbeeld:

```
F% = FINDW("ROGER R*T")
F% = FINDW("DR. JEK+LL")
```

18. Labels, lussen en sprongen

Instructies : BREAK, CONTINUE, DO, ENDWH, ESCAPE, GOTO, STOP, UNTIL, WHILE

18.1 Inleiding

Dit is een belangrijk aspect van programmeren. Labels, lussen (en ook sprongen) maken efficiënt gebruik van geheugen en mooi programmaverloop mogelijk.

Lussen komen in *elke* taal voor : BASIC, PASCAL, C, dBASE, ALGOL, ADA, FORTH, ... en OPL. Een lus is in feite een herhaling van een aantal instructies waar een aantal voorwaarden aan verbonden zijn (zoals WHILE/ENDWH die we al kennen).

Labels zijn regels, die een naam hebben gekregen. Ze worden meestal gebruikt om bepaalde programmagedeelten te identificeren, om duidelijk te maken wat het volgende stukje programma doet. Labels worden echter ook gebruikt om sprongen mogelijk te maken.

Sprongen zijn precies wat hun naam doet vermoeden : plotselinge sprongen naar een andere, vastgestelde regel in het programma. Ze zijn bij beroepsprogrammeurs (die echter meestal met andere talen werken) niet erg populair (de bekende computergoeroe **Edsger Dijkstra** vindt het gebruik van sprongen zelfs crimineel). Dit komt omdat sprongen niet erg *structureel* zijn : programma's die ze bevatten zijn moeilijker te 'onderhouden' (aanpassen) en te lezen. Deze niet-populariteit is dus wel terecht ...

18.2 Lussen

Er zijn verschillende soorten lussen. De eerste soort zijn we al een aantal keer tegengekomen :

18.2.1 WHILE/ENDWH

Syntax : WHILE <exp>
```
         ...
ENDWH
```

Zolang aan de conditie wordt voldaan (<exp> heeft als uitkomst niet nul) worden alle instructies tussen de WHILE en de ENDWH uitgevoerd. Zodra

aan de conditie niet wordt voldaan, gaat de uitvoering van het programma verder vanaf de eerste instructie na ENDWH.

Programma 18.1 ASCWHILE

Met gebruik van WHILE en ENDWH worden de ASCII-tekens 33 tot 64 op het scherm afgedrukt.

```
1 ASCWHILE:
2 LOCAL A%
3 CLS
4 A% = 33
5 WHILE A% <= 64
6    PRINT CHR$(A%);
7    A% = A% + 1
8 ENDWH
9 GET
```

18.2.2 DO/UNTIL

Dit is precies het omgekeerde van WHILE/ENDWH. DO/UNTIL (doe/tot) voert instructies uit niet *zolang* aan een conditie wordt voldaan, maar *tot* aan die conditie wordt voldaan.

Syntax : DO
```
         ...
         UNTIL <exp>
```

Een voorbeeld illustreert het gebruik.

Programma 18.2 ASCUNTIL

Dit programma doet precies hetzelfde als programma 30.1.

```
1 ASCUNTIL:
2 LOCAL A%
3 CLS
4 A% = 33
5 DO
6    PRINT CHR$(A%);
7    A% = A% + 1
8 UNTIL A% > 64
9 GET
```

DO mag ook vlak voor een instructie worden gegeven. Lijnen 5 en 6 zouden dus mogen worden samengevat tot :

```
DO PRINT CHR$ (A%)
```

Ikzelf gebruik DO/UNTIL eigenlijk als nooit, ik prefereer WHILE/ENDWH. Het is een kwestie van smaak.

18.3 Onderbreken van lussen : BREAK

Een lus kan rechtstreeks in het programma onderbroken worden, wanneer dat nodig zou zijn. BREAK 'breekt' de lus en de uitvoering van het programma gaat verder vanaf het punt na de ENDWH of UNTIL.

Syntax : BREAK

Breekt een lus af.

Programma 18.3 : BREAKVB

geeft een voorbeeld dat BREAK illustreert. Er wordt een lus opgezet (de machine telt van 1 tot 5000). Dit duurt een tijdje en de gebruiker kan, als hij dat wil, de lus stoppen door een toets in te drukken.

```
1 BREAKVB:
2 LOCAL A%
3 A% = 1
4 WHILE A% <> 5000
5    IF KEY
6         BREAK
7    ENDIF
8    A% = A% + 1
9 ENDWH
```

In regel 5 wordt gekeken of er al dan niet een toets wordt ingedrukt (de functie KEY geeft de ASCII-code weer van de toets die wordt ingedrukt; wordt er geen toets ingedrukt dan is het resultaat 0). In regel 6 wordt de lus dan effectief gebroken.

18.4 De lus onderbreken tijdens de uitvoering

Programma's kunnen ook onderbroken worden tijdens de uitvoering van een programma door de gebruiker, zonder dat het programma daarvoor voorzieningen treft. Men spreekt dan van een *ontsnapping* (Eng: *escape*) uit het programma.

Programma 18.4 : ESCVB

Dit programma telt van 1 tot 5000 en drukt daarbij steeds een willekeurig ASCII-teken af tussen 32 en 255 (de tekens 0-31 zijn niet zichtbaar) en dient om het gebruik van een ontsnapping aan te tonen.

```
1 ESCVB:
2 LOCAL A%
3 A% = 1
4 WHILE A% <> 5000
5    PRINT CHR$ ( (RND*223) +32) ;
6    A% = A% + 1
7 ENDWH
```

We RUNnen dit programma en op een bepaald ogenblik drukken we op ON, gevolgd door Q (van het Engelse woord *quit* wat *ophouden* betekent). Het scherm meldt :

```
ESCAPE
IN A: ESCVB
```

Nogmaals ON brengt ons terug naar het PROGmenu.
Ontsnappingen zijn vooral bruikbaar bij programma's die anders *nooit* zouden eindigen, zoals het volgende voorbeeld :

Programma 18.5 : STOPNIET

Zoals de naam suggereert, stopt de uitvoering van dit programma nooit tenzij iemand ON-Q drukt.

```
1 STOPNIET:
2 LOCAL A
3 A = 100
4 WHILE A >= 20
5    A = A + 1
6 ENDWH
```

Dit programma *kan* niet stoppen, omdat A wordt geïnitialiseerd op 100 en in de lus wordt verhoogd terwijl de voorwaarde van de lus (regel 4) zegt dat A groter moet zijn dan 20 (wat *altijd* het geval zal zijn); het programma gaat dus door tot in het 'oneindige' (in de praktijk tot de batterij leeg is).

18.5 ESCAPE

Het is echter ook mogelijk, maar tevens gevaarlijk, de mogelijkheid tot ontsnappen met ON-Q uit te schakelen.

Syntax : ESCAPE ON | OFF

ESCAPE ON maakt het mogelijk te onderbreken
ESCAPE OFF maakt het onmogelijk te onderbreken

Standaard gebruikt de Organiser II ESCAPE ON. Het gebruik van ESCAPE OFF wordt stellig afgeraden; het eindigt in alle gevallen met de verwijdering van de batterij en dus verlies van het interne geheugen.

18.6 Labels

Een label wordt in een programma aangebracht door de naam op te geven, gevolgd door twee dubbele punten. Geldige labels zijn bijvoorbeeld :

```
BEGIN::
X::
A123::
```

Labels moeten beginnen met een letter en mogen maximaal 8 tekens lang zijn (de dubbele punten niet inbegrepen). Er mogen behalve letters ook cijfers in voorkomen. Ongeldige labels zijn dus :

```
1::
LABEL12345678910::
START%::
```

Elk label mag maar één keer voorkomen in eenzelfde procedure.

Programma 18.6 : LABEL

```
1 LABEL:
2 LBL1::
3 PRINT "Dit was label 1"
4 XXX::
5 PRINT "Dit was nr 2"
6 EINDE::
7 GET
```

Labels nemen dan wel plaats in in het programma, maar zij doen eigenlijk niets (net als REM). Behalve de verhoogde leesbaarheid van programma's

maken labels ook onderdeel uit van het volgende :

18.7 Sprongen : GOTO

GOTO is weer een term uit het Engels. Het betekent 'ga naar'. Het komt voor in de meeste programmeertalen, maar de enige talen waar het frequent gebruik van een dergelijke instructie gerechtvaardigd is, zijn de zgn. 'assembleertalen', die bestaan uit rechtstreekse instructies aan de microprocessor. Assembleertalen hebben geen gestructureerde lussen maar gebruiken steeds sprongen die lussen imiteren. De ROM van de Organiser II is geschreven in de assembleertaal van de microprocessor (Hitachi HD6303X). Het gebruik van sprongen wordt in 'hogere' programmeertalen, zoals OPL, zo mogelijk vermeden (ze mogen wel vrijelijk worden gebruikt, zoals we later zullen zien, bij 'lokale foutenbehandeling').

Syntax : GOTO <label>
De verwerking wordt voortgezet vanaf de eerste instructie die op het <label> volgt.

Programma 18.7 : GOTOVB

```
1 GOTOVB:
2 CLS
3 PRINT "Dit wordt afge- drukt"
4 GOTO EINDE::
5 PRINT "Dit wordt over- geslagen"
6 EINDE::
7 GET
```

De sprong hoeft niet voorwaarts te zijn; hij kan naar ieder label in de procedure verwijzen. Het is niet mogelijk naar een label in een andere procedure of functie te springen.

18.8 STOP!

Een programma kan ook midden in de listing abrupt worden gestopt (dus zonder dat de Organiser II het einde van het programma bereikt). De instructie die daarvoor wordt gebruikt is genaamd STOP.

Syntax : STOP

Beëindigt de uitvoering van een programma.

Programma 18.8 : STOPVB

Dit voorbeeld spreekt voor zichzelf.

```
 1 STOPVB:
 2 LOCAL JN$(1)
 3 CLS
 4 PRINT "Verdergaan (J/N)"; CHR$(63)
 5 INPUT JN$
 6 IF UPPER$(JN$) = "N"
 7    STOP
 8 ENDIF
 9 PRINT "OK"
10 GET
```

Net als GOTO, is STOP niet bevorderlijk voor de structuur van een programma. Het is dus beter deze instructie beperkt of helemaal niet te gebruiken.

18.9 Verdergaan : CONTINUE

CONTINUE is een instructie die, net als GOTO en STOP, alleen maar kan bijdragen tot de onleesbaarheid van een programma. Ikzelf gebruik deze instructie *nooit*. Ze is onnodig, ongestructureerd en programma's die een CONTINUE bevatten, zijn geen knip voor de neus waard. Volledigheidshalve zullen we toch éven stilstaan bij deze instructie.

Syntax : CONTINUE
Doet het programma springen naar :
- WHILE <exp> bij een WHILE/ENDWH-lus
- UNTIL <exp> bij een DO/UNTIL-lus.

De <exp> wordt geëvalueerd en het programma gaat verder alsof er niets gebeurd is.

19. Geluid

Instructie : BEEP

19.1 Inleiding

Zoals we duidelijk kunnen horen bij ALARM en DIARY, beschikt de Organiser II over een zgn. *beeper* : een klein toestelletje dat op verschillende frequenties geluid kan produceren. Beepers vinden we overal terug : in deurbellen, PC's en zelfs in synthesizers (hoewel die wel een beetje ingewikkelder en geavanceerder zijn).

19.2 BEEP !

Die beeper is uiteraard programmeerbaar. De instructie die de beeper stuurt heet heel toepasselijk BEEP.

BEEP
Syntax : BEEP <exp%1>, <exp%2>
Voor een periode van <exp%1> milliseconden (duizendsten van een seconde), piept de Organiser II op frequentie <exp%2>. Die frequentie is echter niet uitgedrukt, zoals men zou verwachten, in Hz. De omzettingsformule is als volgt :

```
Frequentie in Hz = 921600/(78 + 2 * <exp%2>)
```

Een hele berekening ! Om uit te rekenen hoeveel <exp%1> nu eigenlijk moet zijn om een frequentie in HZ% Hz te berekenen, kunnen we gebruiken :

```
<exp%2> = (921600 / HZ% / 2) - 78
```

Programma 19.1 BEEPVB

produceert een geluidseffect dat begint met een frequentie van 500 Hz en dat geleidelijk verhoogd wordt tot 1000Hz :

```
1 BEEPVB:
2 LOCAL HZ%, NOOT%
3 HZ% = 500
```

```
4 WHILE HZ% <= 1000
5    NOOT% = (921600/HZ%/2)-78
6    BEEP 2,NOOT%
7    HZ% = HZ% + 1
8 ENDWH
```

Wie dit programma uitvoert, zal merken dat het geluid steeds even kort onderbroken wordt. Dit komt door regel 5; hier wordt een heel complexe berekening uitgevoerd, die wat tijd nodig heeft. Dit probleem kan worden opgelost door een tabel op te stellen met 500 elementen, en alle berekeningen op voorhand te doen. Later hoeft het programma slechts die tabel te doorlopen om aan de correcte waarden te komen, wat de geluidskwaliteit verbetert :

Programma 19.2 : BEEPVB, tweede versie

```
 1 BEEPVB:
 2 LOCAL HZ%(500),B%
 3 B% = 1
 4 WHILE B% <= 500
 5    HZ%(B%) = (921600/(B%+499)/2)-78
 6    B% = B% + 1
 7 ENDWH
 8 B% = 1
 9 WHILE B% <= 500
10    BEEP 2,HZ%(B%)
11    B% = B% + 1
12 ENDWH
```

In de regels 3 tot 7 worden de berekeningen uitgevoerd. Dit duurt enkele seconden. Daarna wordt het geluid geproduceerd op de correcte frequenties (de regels 8 tot 12).

19.3 Geluidseffecten

Er kunnen verscheidene geluidseffecten worden gemaakt, zonder moeilijke berekeningen. Probeer bijvoorbeeld eens het volgende programma :

Programma 19.3 : ROLLETJE

```
1 ROLLETJE:
2 LOCAL A%,B%
3 A% = 16
```



```
4 WHILE A%>0
5    AT A%,1
6    PRINT "Organiser II XP "
7    B%=1
8    WHILE B%<5
9       BEEP 4,50*A%
10      BEEP 2,0
11      BEEP 4,75*A%
12      B%=B%+1
13   ENDWH
14   A% = A% - 1
15 ENDWH
16 GET
```

(Let op de laatste spatie in regel 6 !)

Een leuk effect, nietwaar? Het verder leren kennen van de beeper van de Organiser II vergt enkel ervaring. Probeer bijvoorbeeld eens de Organiser II om te toveren in een scheerapparaat :

Programma 19.4 : SCHEER

```
1 SCHEER:
2 CLS
3 PRINT REPT$(CHR$(219),32);
4 WHILE 1 : REM Oneindig
5    BEEP 1,500
6    BEEP 1,600
7 ENDWH
```

(Het programma kan onderbroken worden met ON-Q).

Programma 19.5 : GEK

Dit spreekt voor zichzelf.

```
1 GEK:
2 WHILE 1
3    BEEP 3,(RND*1000)+100
4 ENDWH
```

20. Wachtwoorden

Instructie : OFF

20.1 Inleiding

Het kan soms wenselijk zijn, onze Organiser II te beveiligen met een wachtwoord, zodat niemand anders toegang tot het toestel kan krijgen. Het is dan mogelijk een programma te schrijven dat, zodra de Organiser II met ON wordt aangezet, onmiddellijk een wachtwoord opvraagt. Als dat wachtwoord niet correct ingegeven wordt, zet de Organiser II zichzelf onmiddellijk uit.

20.2 De Organiser II uit zetten : OFF

De instructie OFF zet de Organiser II uit.

OFF
Syntax : OFF
Zet de Organiser II uit. Hij kan weer worden aangezet door ON in te drukken. Dan gaat het programma verder.

Programma 20.1 : WACHTW

```
 1 WACHTW:
 2 LOCAL WACHTW$(32)
 3 WACHTW$ = "VERKEERD"
 4 WHILE WACHTW$ <> "CORRECT WACHTWOORD"
 5    OFF
 6    REM Hier begint hij wanneer
 7    REM op ON wordt gedrukt
 8    PRINT "Wachtwoord : ",
 9    INPUT WACHTW$
10 ENDWH
11 CLS
12 PRINT "OK"
13 GET
```

Regel 4 bevat het wachtwoord. Het betreft hier een lus, die wordt uitgevoerd zolang het wachtwoord niet correct is. Daarom wordt het wachtwoord geïnitialiseerd met een opzettelijk verkeerde inhoud, omdat de lus dan minstens één keer wordt uitgevoerd. Wanneer het wachtwoord dat in regel

9 wordt ingevoerd via het klavier verkeerd is, springt de Organiser II terug naar regel 5 : hij wordt dus weer afgezet ...

Dit is een vrij veilige manier om de Organiser II tegen gebruik door pottekijkers (en tegen het per ongeluk of opzettelijk wissen van gegevens door derden). We kunnen dit programma bijvoorbeeld in het hoofdmenu gebruiken.

Eén risico zit er echter wel aan vast : we mogen het wachtwoord zelf niet vergeten. Men zou om die reden al snel geneigd zijn de eigen naam, geboortedatum, naam van het bedrijf, e.d. als wachtwoord te gebruiken. Dit ligt echter zo voor de hand dat het de veiligheid van het wachtwoordsysteem aanzienlijk aantast. We moeten dus eigenlijk steeds een niet-triviaal wachtwoord gebruiken en dat terdege onthouden.

De Organiser II LZ accepteert ook een inegere parameter achter OFF. Bijvoorbeeld om de LZ 50 seconden uit te zetten, gebruiken we:

```
OFF 50
```

21. Foutenbehandeling

Instructies : ONERR, RAISE, TRAP
Functies : ERR, ERR$

21.1 Inleiding

Met fouten hebben we al te maken gehad. Een verkeerde parameter, het 'openen' van een bestand dat niet bestaat, in een berekening proberen te delen door nul ... Allemaal dingen die niet kunnen. De Organiser II rapporteert in zo'n geval dan ook een fout (Engels : *error*).

Wanneer een fout optreedt, wordt het programma bruusk afgebroken. Het zou veel mooier zijn, wanneer we die fouten zouden onderscheppen en het programma de fout zelf zouden laten oplossen. Gelukkig is dat mogelijk.

Er zijn verschillende manieren voor. We zullen beginnen met de minst elegante.

21.2 ONERR

ONERR is een instructie die als parameter een <label> heeft.

ONERR
Syntax : ONERR <label>
Wanneer zich een fout voordoet, zal de Organiser II onmiddellijk naar de regel na het label springen.

Programma 21.1 : FOUTVB

Dit voorbeeld is, toegegeven, wat kunstmatig, maar het geeft wel enig idee van de werking van ONERR.

```
 1 FOUTVB:
 2 LOCAL A% : REM Integer
 3 ONERR FOUT: :
 4 A% = 50000 : REM Kan niet
 5 STOP
 6 FOUT: :
 7 PRINT "KAN NIET"
 8 GET
```


In regel 3 wordt gezegd dat, wanneer er een fout optreedt, de Organiser II onmiddellijk naar FOUT: : moet springen. Er treedt een fout op in regel 4 : een integer kan geen getallen bevatten die groter zijn dan 32767.

21.3 Het foutnummer : ERR

Niet alleen is dit voorbeeld kunstmatig, het is ook weinig bruikbaar. De foutenopvang (vanaf regel 6; FOUT: :) kijkt niet eens welke fout er is opgetreden. Dat kàn echter wel : het *foutnummer* wordt gerapporteerd in de functie ERR. Een volledige lijst van fouten en hun nummer vinden we terug in Appendix B (op nummer) en Appendix C (alfabetisch).

Hoewel de functie ERR heet en niet ERR%, is het resultaat steeds een integer getal tussen 0 en 255. De Organiser II zelf gebruikt enkel de nummers 195 tot 255 voor fouten. Wanneer echter randapparatuur is aangesloten, zoals de COMMS LINK, worden extra foutmeldingen en -nummers toegevoegd.

Programma 21.2 : FOUT

Dit is eigenlijk geen programma, maar een geïsoleerd stukje van een programma dat fouten opvangt. We zien hier enkele voorbeelden van fouten die kunnen optreden en hoe ze kunnen worden opgevangen.

```
 1 FOUT:
 2 ONERR OPVANG: :
 3 . . . : REM     Hier komt het programma
 4 OPVANG: :
 5 IF ERR = 195
 6    REM INTEGER OVERFLOW
 7    PRINT "Overlopend getal"
 8    GET
 9    STOP
10 ENDIF
11 IF ERR = 246
12    REM NO PACK
13    PRINT "Er is geen pak"
14    GET
15    STOP
16 ENDIF
   . . .
```

Het is duidelijk dat deze manier van werken niet erg efficiënt is en eigenlijk geen oplossing biedt voor het probleem. Een andere, wat subtielere aanpak is de volgende.

21.4 Fouten ter plaatse detecteren en opvangen : TRAP

TRAP betekent 'detecteer' en dat is precies wat deze (halve) instructie doet (halve instructie omdat ze steeds wordt gebruikt in combinatie met een andere instructie).

TRAP
Syntax : TRAP <instructie>
De <instructie> wordt uitgevoerd en als er een fout is opgetreden, bevindt het foutnummer zich in ERR. De fout kan onmiddellijk na de TRAP worden opgevangen.

Niet alle instructies kunnen met TRAP worden gebruikt. De lijst is beperkt tot :

APPEND	ERASE	OPEN
BACK	EDIT	POSITION
CLOSE	FIRST	RENAME
COPY	INPUT	UPDATE
CREATE	LAST	USE
DELETE	NEXT	

Programma 21.3 : TRAPVB

```
1 TRAPVB:
2 LOCAL I%
3 I1::
4 TRAP INPUT I%
5 IF ERR
6   PRINT "Fout nummer"
7   GOTO I1::
8 ENDIF
```

In regel 4 wordt de variabele I% van het klavier opgevraagd. Wanneer de gebruiker bijvoorbeeld 50000 intikt (wat dus niet mag), rapporteert de Organiser II de fout in de functie ERR. Wanneer er geen fout is opgetreden heeft ERR de inhoud 0.

21.5 Fouten kunstmatig opwekken : RAISE

De instructie RAISE dient om een fout kunstmatig op te wekken.

RAISE
Syntax : RAISE <exp%>
De expressie bevat een integer getal tussen 0 en 255. De corresponderende fout wordt dan door de Organiser II in de functie ERR gemeld.

Deze instructie kan worden gebruikt om eigen foutmeldingen te creëren (met een code die dus niet in het bereik 195 - 255 ligt), of om een fout kunstmatig op te wekken.

Voorbeelden :

RAISE 206	Resultaat : ESCAPE
RAISE 220	Resultaat : STRING TOO LONG

(Zie Appendix B en C).

21.6 Foutmeldingen : ERR$

ERR$ zouden we het stringequivalent van ERR kunnen noemen.

ERR$
Gebruik : x$ = ERR$ (<exp%>)
De expressie bevat een geheel getal tussen 0 en 255 (volgnummer van een foutmelding). De corresponderende foutmelding is dan het resultaat van ERR$.

Voorbeelden :

ERR$(206)	Resultaat : "ESCAPE"
ERR$(196)	Resultaat : "FILE NOT OPEN"

Wanneer de Organiser II het volgnummer niet herkent (bijvoorbeeld volgnummer 10) zal het resultaat van ERR$ zijn :

"*** ERROR ***"

Foutenbehandeling is steeds aan te raden bij bestanden. Daar kan immers een hoop misgaan (een pak proberen aan te spreken dat er niet is, bijvoorbeeld) en het is altijd prettig te werken met programma's die fouten degelijk opvangen.

Een voorbeeldtechniekje :

```
LOCAL P$(1), ...
GEEFPAK::
CLS
PRINT "PAK : ",
INPUT P$
P$ = UPPER$(P$)
IF P$ <> "A" AND P$ <> "B" AND P$ <> "C"
  GOTO GEEFPAK::
ENDIF
TRAP OPEN P$+":MAIN",A,V$,V2$,V3$,V4$
IF ERR
  PRINT "Pak niet aanwe- zig"
  GET
  GOTO GEEFPAK::
ENDIF
USE A
...
```

Het is niet helemaal gestructureerd opgelost, maar bij lokale foutbehandeling is dat door de vingers te zien. Lokale foutbehandeling is immers een lineair proces met niet te voorziene sprongen.

22. Op machineniveau

Instructies : POKEB, POKEW
Functies : PEEKB, PEEKW, USR, USR$

22.1 Het niveau van de machine

OPL is een zgn. 'hoog-niveau'-programmeertaal. Dat betekent dat de taal vrij dicht bij de mensentaal staat en dus gemakkelijk kan worden aangeleerd en gebruikt.

Er zijn ook programmeertalen met gemiddeld of laag niveau. Een voorbeeld is C. Deze complexe taal wordt gebruikt om zeer grote en ingewikkelde programma's te schrijven, zoals besturingssystemen (het besturingssysteem UNIX bijvoorbeeld, voor heel grote computers, is geschreven in C).

Deze laatste talen zijn wel moeilijk, maar ze hebben ook voordelen : ze staan dichtbij de computer en zijn daarom zeer snel en compact.

De taal met het allerlaagste niveau is de assembleertaal. Er is niet maar één assembleertaal : er zijn zoveel verschillende assembleertalen (of *assemblers*) als er verschillende microprocessors zijn. Iedere microprocessor heeft dus zijn eigen assembler. De microprocessor die in de Organiser II zit, de Hitachi HD6303X, dus ook.

Deze taal is niet eenvoudig te leren. Het is ook heel moeilijk om fouten uit programma's te halen, doordat er geen foutmeldingen zijn zoals OPL die kent. Iedere fout betekent onvermijdelijk een 'crash' : de computer loopt vast. De enige manier om weer leven in de machine te krijgen is de stroombron af- en weer aan te koppelen. Het programma is dan uiteraard verdwenen...

Hoewel er weinig mensen zullen zijn die in HD6303X assembler programmeren, heeft Psion toch de mogelijkheid voorzien om via OPL programma's uit te voeren die in die taal geschreven zijn (omgezet naar *machinetaal* : de enige taal die door de computer begrepen wordt).

22.2 USR

USR is de afkorting van '*user*' (= '*gebruiker*').

USR
Gebruik : x% = USR (<exp%>, <exp%2>)
De eerste expressie wordt doorgegeven aan het PC-register van de microprocessor, de tweede expressie gaat naar het D-register. Het machinetaalprogramma dat in het interne geheugen ligt opgeslagen vanaf adres <exp%1> wordt dan uitgevoerd (het PC-register duidt aan welke instructie wordt uitgevoerd; PC is in dit geval de afkorting van *program counter* of programmateller).

Wanneer het machinetaalprogramma is uitgevoerd, heeft de functie USR als resultaat de inhoud van het X- register van de microprocessor (niet te verwarren met variabele x of x%).

OPGELET : wie geen verstand heeft van machinetaal wordt aangeraden van deze functie af te blijven. Zoals gezegd kan ieder foutje de Organiser II doen vastlopen.

22.3 USR$

USR$
Gebruik : x$ = USR$ (<exp%1>, <exp%2>)
Deze functie is vrijwel identiek aan de vorige. Het verschil is dat bij het beëindigen van de machinetaalroutine het X-register naar een string moet wijzen. De eerste byte van die string moet de stringlengte bevatten.

22.4 POKE en PEEK

In tegenstelling tot USR en USR$, zullen we de POKE- *en* PEEK-familie waarschijnlijk geregeld gebruiken.

Het geheugen in de Organiser II is verdeeld in een aantal bytes. Iedere byte heeft zijn eigen lokatie in het geheugen : het adres. Een geheugen van 32K heeft dus 32 × 1024 = 32768 adressen, genummerd tussen 0 en 32767.

De Organiser II, model XP, heeft 64K aan boord (32K op ROM en 32K RAM). Model CM heeft 40K (32K ROM, 8K RAM). Model LZ beschikt over 96K (LZ) of 128K (LZ64) waarvan 64K ROM. We kunnen deze lokaties byte voor byte, of woord voor woord (een woord is een groepering van 2 bytes) gaan bekijken.

22.4.1 PEEKB, PEEKW

PEEKB (kijk naar 1 byte)
Gebruik : x% = PEEKB (<exp%>)
Geeft de inhoud van de byte op geheugenadres <exp%>.

Voorbeelden :

```
PEEKB(1)
PEEKB($0077)
```

De lengte van het resultaat is steeds 1 byte (een getal tussen 0 en 255).

PEEKW (kijk naar een woord)
Gebruik : x% = PEEKW(<exp%>)
Geeft de inhoud van de byte op adres <exp%> met daarbijgeteld de inhoud van de byte op adres <exp%> + 1 vermenigvuldigd met 256. Dit komt uiteindelijk neer op een woord : een groepering van 16 bits die een getal weergeven tussen -32768 en +32767.

Voorbeelden :

```
PEEKW($20CD)
PEEKW(107)
```

Zometeen volgen meer praktische voorbeelden.

22.4.2 POKEB, POKEW

POKEB (vul adres met 1 byte)
Syntax : POKEB (<exp%1>, <exp%2>)
De byte op adres <exp%1> krijgt de inhoud <exp%2>. Deze laatste expressie moet dan ook een geheel getal tussen 0 en 255 zijn.

Voorbeelden :

```
POKEB($0069),200
POKEB($007C),1
```

POKEW (vul adressen met woord)
Syntax : POKEW (<exp%1>, <exp%2>)
Het woord op adres <exp%1> en <exp%1>+1 krijgt de inhoud <exp%2>. Deze laatste expressie mag dus een volledig integer getal zijn (tussen -32768 en +32767).

Voorbeelden :

```
POKEW($20CD),500
POKEW($20CB),0
```

22.5 De geheugenkaart

De geheugenkaart van de Organiser II is een histogram dat de indeling van het geheugen voorstelt. Hierop is duidelijk te zien hoe de Organiser II zijn geheugen indeelt. De geheugenkaart is in detail te zien in Appendix D.

De ROM van beide modellen gebruikt geheugenlocaties $8000 tot $FFFF. De *systeemvariabelen*, handige dingen waarover we het dadelijk zullen hebben, zitten tussen de lokaties $0040-$0100 en $2000 omhooggroeiend naar de zgn. 'taalstack'.

22.5.1 De taalstack

Een *stack* (of *heap*) wordt in vertaling ook wel *stapel* genoemd. Het betreft een opstapeling in het geheugen van woorden, die adressen aanduiden. Bijvoorbeeld : we hebben een programma in OPL dat een label BEGIN: : heeft. Dat label heeft een eigen adres. Wanneer de machine in dat programma de instructie GOTO BEGIN: : tegenkomt, wordt het adres van BEGIN: : op de stapel 'gelegd'. Daarna wordt een speciale interne functie aangeroepen die het adres weer van de stapel afneemt en naar het adres springt. De stapel wordt dus o.m. gebruikt om tijdelijk gegevens op te slaan en door te geven aan andere functies. Allerhande gegevens over variabelen, WHILE/ENDWH-lussen, enzovoort, worden allemaal bewaard op de stapel.

Deze stapel, de zgn. *taalstapel*, groeit naarmate er meer gegevens op komen te liggen. De stapel groeit, hoe raar het ook klinkt, naar *beneden*. Tegelijkertijd is er een soort stapel in de Organiser II die naar *boven* groeit (ook een stapel voor gebruik door het besturingssysteem van de Organiser II). Op een bepaald ogenblik kunnen die stapels elkaar ontmoeten. Wanneer dat gebeurt, vindt er een zgn. *stack overflow* plaats : de stapels hebben geen geheugenruimte meer. De taalstapel wordt in dat geval leeggemaakt (het programma wordt vanzelf beëindigd of, wanneer men probeert de fout op te vangen, loopt de boel in het honderd). De stapelruimte is echter zo groot (7936 bytes) dat een stack overflow bijna nooit voorkomt.

22.5.2 De systeemvariabelen

Dit is een zeer interessant aspect van de geheugenkaart. De Organiser II

heeft ruimte in het geheugen gereserveerd waar speciale waarden worden opgeslagen. Er staat bijvoorbeeld in hoe lang de Organiser II moet wachten voordat hij zichzelf uitschakelt. Die waarden kunnen met PEEKB en PEEKW worden bekeken en met POKEB en POKEW worden veranderd. *Sommige van deze systeemvariabelen zijn in de handleiding niet gedocumenteerd.*

22.5.2.1 Cursorpositie

Deze systeemvariabele zegt waar de cursor op het scherm staat.

Adres : $0062 (cursorpositie)
De cursorpositie (tussen 0 en 32) kan gewijzigd worden door op dit adres POKEB te gebruiken.

Voorbeelden :
Thuispositie : `POKEB ($0062), 0`
Tweede regel : `POKEB ($0062), 16` (CM en XP)
`POKEB ($0062), 20` (LZ)

22.5.2.2 Cursorstatus

De status van de cursor kan twee waarden hebben : 0 (alfanumeriek) of 1 (numeriek).

Adres : $0063

Voorbeelden :

Alfanumeriek : `POKEB ($0063), 0`
Numeriek : `POKEB ($0063), 1`

22.5.2.3 Horizontale scrollsnelheid

De Organiser II heeft de gewoonte een string die niet op één regel kan te laten scrollen (zoals in de functie VIEW en de optie INFO bijvoorbeeld). De snelheid van die scroll kan worden veranderd.

Adressen : $0069, $006A (normale inhoud : 4)
Deze adressen bevatten samen een woord dat de snelheid van de horizontale scroll aangeeft. De normale waarde is 4, maar deze kan worden versneld door lagere getallen (2 om de snelheid te verdubbelen) of vertraagd door hogere getallen (8 om de snelheid te halveren).

Voorbeelden :

Sneller : `POKEW ($0069), 2`
Trager : `POKEW ($0069), 8`

22.5.2.4 Verticale scrollsnelheid

De snelheid van de verticale scroll wordt beheerst door deze systeemvariabele.

Adressen : $006B, $006C (normale inhoud : 10)
Deze adressen bevatten samen een woord dat de snelheid van de verticale scroll aangeeft. Door deze adressen een kleinere inhoud te geven wordt het scrollen versneld, een hogere inhoud geeft een tragere scroll.

Voorbeelden :

```
Sneller : POKEW ($006B) , 5
Trager : POKEW ($006B) , 20
```

22.5.2.5 Klaviersnelheid : wachttijd vóór automatische herhaling

De toetsen van de Organiser II worden, indien ze een tijdje ingedrukt worden gehouden, zeer snel automatisch herhaald (Engels : *auto repeat*). Deze geheugenplaats bevat een byte die aangeeft hoe lang de Organiser II wacht voor hij een toets laat herhalen.

Adres : $0077 (normale inhoud : 14).
Om sneller te werken kan deze wachttijd verkort worden.

Voorbeelden :

```
Korter : POKEB ($0077) , 5
Langer : POKEB ($0077) , 50
```

22.5.2.6 Wachttijd tijdens automatische herhaling

Tussen de automatische herhalingen door, is het mogelijk de Organiser II te laten wachten (deze mogelijkheid wordt standaard *niet* gebruikt). Het is enkel mogelijk deze wachttijd te *verlengen*, aangezien die normaal 0 is.

Adres : $0078 (normale inhoud : 0)
Om trager te werken kan deze wachttijd verlengd worden.

Voorbeelden :

```
Traag : POKEB ($0078) , 25
Zeer traag : POKEB ($0078) , 200
```

22.5.2.7 Klavierstaat

De huidige staat van het klavier kan op dit adres bekeken worden.

Adres : $007B (inhoud : hangt af van het klavier)

Voorbeelden :

```
KSTAT 1 : POKEB ($007B) , 0
KSTAT 2 : POKEB ($007B) , 1
KSTAT 3 : POKEB ($007B) , 64
KSTAT 4 : POKEB ($007B) , 65
```

22.5.2.8 Automatische uitschakeling

De Organiser II schakelt zich, als hij niet wordt gebruikt, na ongeveer vijf minuten automatisch uit. Er is echter een systeemvariabele die erin voorziet om dit uitschakelen onmogelijk te maken.

Adres : $007C (normale inhoud : 5)
Een waarde die niet gelijk is aan nul (zoals 5) maakt het automatisch uitschakelen van de Organiser II mogelijk. Om het uitschakelen te desactiveren moet dit adres dus de inhoud 0 krijgen.

Voorbeelden :

```
Wel uitschakelbaar : POKEB ($007C) , 5
Niet uitschakelbaar : POKEB ($007C) , 0
```

22.5.2.9 Huidig pak

Het pak waarop de laatste activiteit heeft plaatsgevonden (A, B of C) wordt hier numeriek weergegeven.

Adres : $0097 (huidig pak)
Pak A: wordt aangegeven als 0, pak B: als 1 en pak C: als 2.

Voorbeeld :

```
Maak C: huidig : POKEB ($0097) , 2
```

22.5.2.10 Klavierklik

Telkens wanneer we een toets indrukken op het klavier, wordt een klik gegenereerd (door de beeper). We kunnen die klik ongedaan maken of juist verlengen.

Adres : $20C0 (normaal : 1)
We kunnen de klik uitschakelen door de waarde 0 te gebruiken. Waarden tussen 1 en 255 geven de tijdsduur van de klik weer.

Voorbeelden :

```
Afzetten : POKEB ($20C0) , 0
Langer : POKEB ($20C0) , 5
```

22.5.2.11 Automatische teller
Deze systeemvariabele (een woord, dus 16 bits lang) heeft een inhoud die iedere 50 milliseconden met 1 verhoogd wordt. Ze wordt door het besturingssysteem van de Organiser II gebruikt om bepaalde timingintensieve acties uit te voeren (zoals het ALARM) en is voor ons niet interessant.

Adressen : $20CD, $20CC (inhoud wordt 20 keer per seconde verhoogd).

Aan te raden is deze waarden niet te veranderen.

22.5.2.12 Wachttijd vóór automatische uitschakeling
Deze systeemvariabele bevat de tijd (in seconden) die voorafgaat aan het automatisch uitschakelen van de Organiser II. Normaal is dit 5 minuten (dus 300 seconden).

Adressen : $20CD, $20CE (normale inhoud : 300)
Door een nieuwe waarde in deze adressen te brengen kan de wachttijd verlengd of verkort worden.

Voorbeelden :

Korter : POKEW($20CD), 120 (2 minuten)
Langer : POKEW($20CD), 480 (8 minuten)

22.5.2.13 Beeper aan/uit
In normale omstandigheden is de beeper van de Organiser II aan, d.w.z. wanneer het besturingssysteem hem 'beveelt' geluid te maken, dan wordt dat geluid ook inderdaad voortgebracht. Hij klikt wanneer er een toets wordt ingedrukt, biept wanneer een alarm gaat, wanneer in OPL het commando BEEP wordt gegeven, enzovoort. De beeper kan echter ook worden uitgezet.

Adres : $00A4 (normale inhoud : 0)
Wanneer deze systeemvariabele een 0 bevat, staat de beeper aan. Wanneer de inhoud niet nul is, staat de beeper uit.

Voorbeelden :

Aan : POKEB($00A4),0
Uit : POKEB($00A4),1

22.5.2.14 Werkdagen-alarmen (alleen LZ)
Deze systeemvariabele bevat een patroon van 7 bits dat aangeeft welke dagen moeten worden gezien als werkdagen voor het Werkdagen-alarm. Standaard is dit maandag tot vrijdag.

Adres : $20A7 (Standaard-inhoud : $1F)

Met het volgende programma kunnen we de individuele dagen van de week instellen als werkdagen :

```
WKALM:
LOCAL A%, C%, JN$
A% = PEEKB($20A7)
C% = 1
WHILE C% <= 7
  CLS
  PRINT "Dag", C%, "? J/N : ";
  INPUT JN$
  IF UPPER$(JN$) = "J"
    A% = A% OR (2**(C%-1))
  ELSE
    A% = A% AND 255-(2**(C%-1))
  ENDIF
  C% = C% + 1
ENDWH
POKEB($20a7),A%
```

Hiermee hebben we alle systeemvariabelen besproken. Gebruik ze met zorg en weet wat u doet. Verkeerd gebruik van de systeemvariabelen kan het gebruik van de Organiser II een heel stuk minder aangenaam maken.

23. Gebruiker-gedefinieerde tekens

23.1 Inleiding

De *tekenset* van de Organiser II (Appendix A), bestaat grotendeels uit tekens die zijn opgemaakt uit puntjes. Op het scherm van de Organiser II zijn die puntjes duidelijk te onderscheiden (duidelijker op CM en XP dan op LZ, uiteraard).

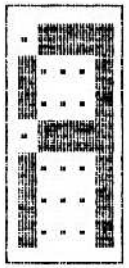
Ieder ASCII-teken tussen 33 en 255 bestaat uit 8 rijen van 5 puntjes. Die puntjes kunnen worden *aangezet* — donker — of *uitgezet* — licht. De donkere puntjes vormen uiteindelijk de letter.

De tekens tussen ASCII 8 en 255 (zie Appendix A) zijn *voorgedefiniëerd*: ze zijn door het besturingssysteem van de Organiser II 'in hun vorm gegoten'. Het is ook mogelijk de eerste acht tekens (ASCII 0 t.e.m. 7) zelf te definiëren.

23.2 UDG's

Die zelfgedefiniëerde tekens noemt men UDG's (afkorting voor de Engelse term *user-defined graphics*). UDG's worden in het geheugen onthouden op een hiervoor speciaal voorziene plaats.

Een UDG bestaat eveneens uit 8 rijen die ieder uit 5 punten (bits) bestaan. Een 1-bit is zwart, een 0-bit is wit. We maken de UDG's dus op uit 8 getallen van 5 bits. Een voorbeeld :



```
= 01111 = 15
= 10001 = 17
= 10001 = 17
= 01111 = 15
= 10001 = 17
= 10001 = 17
= 10001 = 17
= 00000 = 0
```

De bovenstaande Russische letter ('ya'), wordt dus opgebouwd uit de rij getallen 15, 17, 17, 15, 17, 17, 17, 0.

De manieren waarop UDG's op de Organiser II's moeten worden gedefiniëerd zijn verschillend voor LZ en CM/XP. De LZ heeft namelijk de ingebouwde instructie *UDG* om een graphic te definiëren, en dat hebben de andere modellen niet. Daarom vinden we hieronder een procedure, speciaal voor CM en XP-gebruikers, die net hetzelfde doet als de overeenkomstige LZ-instructie.

UDG
Gebruik : UDG <udg%>, <exp%1>, <exp%2>, . . . , <exp%8> [LZ]
UDG: (<udg%>, <exp%1>, <exp%2>, . . . , <exp%8>) [CM/XP]
<UDG%> is een getal tussen 0 en 7 : het tekennummer van de te definiëren user-defined graphic. LZ-gebruikers moeten de onderstaande procedure uiteraard niet intikken.

Programma 23.1 : UDG

```
UDG: (n%, a%, b%, c%, d%, e%, f%, g%, h%)
POKEB $180, 64+(n% * 8)
POKEB $181, a%
POKEB $181, b%
POKEB $181, c%
POKEB $181, d%
POKEB $181, e%
POKEB $181, f%
POKEB $181, g%
POKEB $181, h%
```

Het voorgaande voorbeeld van de Russische letter ya definiëren we dus als volgt :

```
LZ :      UDG 0,15,17,17,15,17,17,17,0
CM, XP:   UDG: (0,15,17,17,15,17,17,17,0)
```

Het teken zit nu in het geheugen opgeslagen en kan worden opgeroepen met PRINT CHR$(0).

Programma 23.2 : KGB
Dit voorbeeldprogramma, dat de grafische mogelijkheden demonstreert, kan van nut zijn voor aankomende geheim agenten en spionnen.

```
KGB:
LOCAL KGB$(33), V%
UDG 0,15,17,17,15,17,17,17,0
UDG 1,17,19,21,21,21,25,17,0
UDG 2,31,17,16,16,16,16,16,0
UDG 3,6,10,10,10,10,14,17,0
UDG 4,31,16,16,30,17,17,30,0
UDG 5,10,10,10,10,10,30,1,0
```

```
KGB$="KOMM"+CHR$(1)+"CAP"+CHR$(0)+"T
"+CHR$(2)+"OCY"+CHR$(3)+
(vervolg vorige regel) "A"+CHR$(5)+"BEH"+CHR$(1)+"
"+CHR$(4)+
(vervolg vorige regel) "ECO"+CHR$(247)+CHR$(5)+"HO"+CHR$(5)+
(vervolg vorige regel) CHR$(1)
CLS
PRINT "KGB ="
V% VIEW(2, KGB$)
```

LZ-gebruikers moeten hun UDG's telkens wanneer ze dienen te worden gebruikt opnieuw definiëren, omdat de LZ ze overschrijft voor zijn eigen UDG's : de eerste regel van elk menu is immers steeds opgebouwd uit UDG's.

24. Voorbeeldprogramma's

Programma 24.1 : DATEDIFF

DATEDIFF dient om het verschil in dagen tussen twee data te berekenen. De gebruiker voert via het klavier de twee data in. Het verschil tussen 1 juni 1988 en 31 december 1999 bijvoorbeeld is 4320 dagen of ± 11.5831 jaar.

Merk op dat dit programma is geschreven voor de CM en XP en voor de LZ veel korter kan gemaakt worden!

```
DATEDIFF:
LOCAL J1,MND1,D1,J2,MND2,D2,M%(12),VERSCH,DD1,DD2,X%
M%(1)=31
M%(2)=28
M%(3)=31
M%(4)=30
M%(5)=31
M%(6)=30
M%(7)=31
M%(8)=31
M%(9)=30
M%(10)=31
M%(11)=30
M%(12)=31
DI1::
CLS
PRINT "EERSTE DATUM"
PRINT "Dag : ";
INPUT D1
IF D1<1 OR D1>31
  GOTO DI1::
ENDIF
CLS
MI1::
PRINT "Maand (1-12) :"
INPUT MND1
IF MND1>12 OR MND1<1
  GOTO MI1::
ENDIF
JI1::
CLS
```

```
PRINT "Jaar : 19";
INPUT J1
IF J1<0 OR J1>99
   GOTO JI1::
ENDIF
IF ((J1/4)=INT(J1/4)) AND J1<>0
 M%(2)=9
ENDIF
IF D1>M%(MND1)
   PRINT "KAN NIET"
   GET
   GOTO DI1::
ENDIF
DI2::
M%(2)=28
CLS
PRINT "TWEEDE DATUM"
PRINT "Dag : ";
INPUT D2
IF D2=0
   D2=DAY
ENDIF IF D2<1 OR D2>31
   GOTO DI2::
ENDIF
MI2::
CLS
PRINT "Maand : ";
INPUT MND2
IF MND2=0
   MND2=MONTH
ENDIF
IF MND2<1 OR MND2>12
   GOTO MI2::
ENDIF
JI2::
PRINT "Jaar : 19";
INPUT J2
IF J2=0
   J2=YEAR
ENDIF
IF J2<0 OR J2>99
   GOTO JI2::
ENDIF
   IF ((J2/4)=INT(J2/4)) AND J2<>0
   M%(2)=29
   ENDIF
```



```
   IF D2>M%(MND2)
   CLS
   PRINT "KAN NIET"
   GET
   GOTO DI2::

ENDIF
REM ************
REM    ALLES OK
REM ************
REM EERST BEREKENEN HOEVEEL DAGEN NOG TE GAAN IN J1
DD1 = D1
X%=1
WHILE X%<MND1
   DD1 = DD1+M%(X%)
   X% = X% + 1
ENDWH
DD1 = 365.25-DD1
REM NU DE DAGEN IN J2
DD2 = D2 X%=1
WHILE X%<MND2
   DD2 = DD2 + M%(X%)
   X% = X% + 1
ENDWH
REM ************
REM NU HET AANTAL DAGEN IN DE TUSSENLIGGENDE JAREN
REM ************
DD1=DD1+((J2-J1-1)*365.25)
PRINT "Verschil : "; INT(DD1+DD2);" dagen"
GET
CLS
PRINT "= "; (DD1+DD2)/365.25;" jaar"
GET
```

Programma 24.2 : SIM

SIM is een simulatie van een legeralarm.

```
SIM:
LOCAL A%, B%
A%=1
WHILE A%<=5
   B%=2000
   WHILE B%>=100
      BEEP 2, B%
      B%=B%-2
```

```
ENDWH
WHILE B%<000
    BEEP 2,B%
    B%=B%+2
ENDWH
A%=A%+1
ENDWH
```

Programma 24.3 : FX

FX gebruikt de volgende speciale-effecten-programma's voor een verrassende introductie van de Organiser II. Hoewel deze programma's geschreven zijn voor de CM en de XP, kunnen ze eenvoudig worden aangepast om op de LZ volledig tot hun recht te komen.

```
FX:
LOCAL F$(16)
F$ = "Probeer dit ook"+CHR$(33)
FX1: (1, "Welkom bij FX"+CHR$(33))
FX2: (2, " ORGANISER II ")
GET
FX3: ("EFFECT NUMMER 3", "****************")
GET
FX4: ("****************", "EFFECT NUMMER 4")
GET
FX5: ("Dit is natuur-", "lijk effect Nr 5")
GET
FX6: (1, "FX6 op regel 1..")
FX62: (2, "FX62 op regel 2.")
FX3: (F$, F$)
WGL: (1, F$)
WGR: (1, F$)
WG: (1)
WG: (2)
```

Programma 24.4 : FX1

Bedoeld voor programma's zoals het bovenstaande is FX1 een 'speciaal effect' dat twee parameters vereist : een schermregelnummer (op de CM en XP 1 of 2, op de LZ 1 t.e.m. 4) en een string (max. 16 tekens op CM en XP, 20 tekens op LZ).

```
FX1: (X%, A$)
LOCAL A%
A%=1
WHILE A%<=LEN(A$)
```



```
 AT 1,X%
 PRINT RIGHT$(A$,A%)
 BEEP 2,A%*20
 BEEP 2,A%*15+200
 A%"%+1
ENDWH
```

Programma 24.5 : FX2

Het tweede effect, dat dezelfde parameters vereist als FX1. Voor de LZ : vervang de 17 in de vijfde regel door 21.

```
FX2: (X%, A$)
LOCAL A%
A%=1
WHILE A%<=LEN(A$)
AT 17-A%,X%
PRINT LEFT$(A$,A%)
BEEP 2,A%*30
BEEP 1,A%*10+200
A%"%+1
ENDWH
```

Programma 24.6 : FX3

FX3 neemt geen schermregelnummer. Het effect vindt plaats over de eerste en de tweede schermregel. De benodigde parameters zijn allebei strings met een maximumlengte van 16 of 20 (respectievelijk voor CM/XP en LZ). *LZ-gebruikers* : vervang het getal 16 overal door 20, 7 door 9, 8 door 10 en 9 door 11.

```
FX3: (A$,B$)
LOCAL C$(16),D$(16),I%
C$=A$+REPT$(" ",16-LEN(A$))
D$=B$+REPT$(" ",16-LEN(B$))
I%=0
WHILE I%<=7
AT 8-I%,1:PRINT CHR$(255);
AT 8-I%,2:PRINT CHR$(255);
AT 9+I%,1:PRINT CHR$(255);
BEEP 2,100*I%
BEEP 2,50*I%
AT 9+I%,2:PRINT CHR$(255);
AT 8-I%,1:PRINT MID$(C$,8-I%,1);
AT 8-I%,2:PRINT MID$(D$,8-I%,1);
AT 9+I%,1:PRINT MID$(C$,9+I%,1);
```

```
AT 9+I%,2:PRINT MID$(D$,9+I%,1);
I%=I%+1
ENDWH
```

Programma 24.7 : FX4

Ook dit effect neemt twee strings van max. 16 tekens als parameters.

```
FX4: (A,B$)
LOCAL C$(17),D$(17),T%, P%
C$=A$+REPT$(" ",16-LEN(A$))
D$=B+REPT$(" ",16-LEN(B$))
T%=1
WHILE T%<=16
  AT 17-T%,1 : PRINT CHR$(255);
  AT 17-T%,2 : PRINT CHR$(255);
  P% = 1
  WHILE P%<=50
    P% = P% + 1
  ENDWH
  AT 17-T%,1 : PRINT LEFT$(C$,T$)
  AT 17-T$,2 : PRINT LEFT$(D$,T%)
  T%=T%+1
ENDWH
T%=KEY
```

Programma 24.8 : FX5

```
FX5: (A$,B$)
LOCAL C$(16),D$(16),I%
C$=A$+REPT$(" ",16-LEN(A$))
D$=B$+REPT$(" ",16-LEN(B$))
I%=7
WHILE I%>=0
AT 8-I%,1 : PRINT CHR$(255);
AT 8-I%,2 : PRINT CHR$(255);
AT 9+I%,1 : PRINT CHR$(255);
BEEP 2,100*I%
BEEP 2,50*I%
AT 9+I%,2 : PRINT CHR$(255);
AT 8-I%,1 : PRINT MID$(C$,8-I%,1);
AT 8-I%,2 : PRINT MID$(D$,8-I%,1);
AT 9+I%,1 : PRINT MID$(C$,9+I%,1);
AT 9+I%,2 : PRINT MID$(D$,9+I%,1);
I%=I%-1
ENDWH
```


Programma 24.9 : FX6

De eerste parameter is een schermregelnummer, de tweede is een string (max. 16 tekens).

```
FX6: (L%,A$)
LOCAL T%,X%
T%=1
WHILE T%<=LEN(A$)
 X%=15
 WHILE X%>=T%
  AT X%,L%
  PRINT MID$(A$,T%,1),
  X%=X%-1
  PAUSE 1
 ENDWH
 T%=T%+1
ENDWH
AT LEN(A$),L%
PRINT RIGHT$(A$,1)
```

Programma 24.10 : FX62

Een ietsje gewijzigde versie van FX6.

```
FX62: (L%,A$)
LOCAL T%,X%
T%=1
WHILE T%<=LEN(A$)
  X%=15
 WHILE X%>=T%
  AT X%,L%
  PRINT MID$(A$,T%,1),
  X%=X%-1
  BEEP 25,X%*5
 ENDWH
 T%=T%+1
ENDWH
AT LEN(A$),L%
PRINT RIGHT$(A$,1)
```

Programma 24.11 : WGL

WGL staat voor "Weg links" en de effectendemo (FX) maakt duidelijk waarom. Wederom twee parameters : schermregelnummer en string.

```
WGL: (A%, A$)
LOCAL X%
X%=LEN(A$)
WHILE X%>=0
  AT 1, A%
  PRINT RIGHT$(A$, X%)
  X%=X%-1
  BEEP 25, X%*A%
  BEEP 25, X%*5
ENDWH
```

Programma 24.12 : WGR

Het tegengestelde van WGL.

```
WGR: (A%, A$)
LOCAL X%
X%=1
WHILE X%<=16
 AT X%, A%
 PRINT " "; LEFT$(A$, 16-X%)
 X%=X%+1
 BEEP 25, X%*A%
 BEEP 25, X%*2
ENDWH
```

Programma 24.13 : WG

Een effect om de schermregel, die als parameter wordt opgegeven, leeg te maken.

```
WG: (A%)
LOCAL B%, X% B%=1
WHILE B%<=2 X%"%
 WHILE X%<=14
  AT X%, A% : PRINT " "
  BEEP 25, X%*A%*2
  X%=X%+2
 ENDWH
 B%"%+1
ENDWH
```


25. Utilities

Programma 25.1 : Scopy (Selectief Copiëren)

Deze utility dient om bepaalde records van de ene pak te copiëren naar de andere. De optie COPY van het hoofdmenu kan alleen gehele bestanden copiëren. Met SCOPY kan men bijvoorbeeld enkel de records die het woord 'restaurant' bevatten copiëren.

De Organiser II vraagt eerst de bronpak, daarna de doelpak. Vervolgens wordt de string opgevraagd die de te copiëren records moeten bevatten. Het programma houdt rekening met records van 5 regels (dit kan aangepast worden).

```
SCOPY:
LOCAL BR$(1), BP$(1), ZOEK$(12), X%, A%, Y%, Z%
CLS
PRINT "VAN ";
INPUT BR$
PRINT "NAAR ";
INPUT BP$
CLS
PRINT "ZOEK ";
INPUT ZOEK$
OPEN BR$+":MAIN", A, BE$, BE2$, BE3$, BE4$, BE5$
OPEN BP$+":MAIN", B, BMAIN$, BM2$, BM3$, BM4$, BM5$
USE A
FIRST
X%=FIND(ZOEK$)
Y%=X%
DO
  IF X%<>0
    USE B
    B.BMAIN$=A.BE$
    B.BM2$=A.BE2$
    B.BM3$=A.BE3$
    B.BM4$=A.BE4$
    B.BM5$=A.BE5$
    APPEND
    USE A
  ENDIF
  NEXT
  X%=FIND(ZOEK$)
UNTIL EOF
```

```
CLS
PRINT "OK"
CLOSE
GET
```

Programma 25.2 : BROWSE (doorlopen bestand)

BROWSE is afgeleid van de gelijknamige functie in het PC-programma dBASE. Het dient om snel een bestand te doorlopen. Men kan bijvoorbeeld met één toets vooraan of achteraan in het bestand gaan kijken, naar het volgende of het vorige record, een bepaalde record opzoeken, ...

Dit zijn de mogelijkheden :

Toets	Functie
F	Eerste record (**first**);
L	Laatste record (**last**);
X	Volgende record (er staat een + boven);
R	Vorige record (er staat een - boven);
S	Search; record zoeken. Er wordt, net als bij Find, een string opgevraagd. Indien gewoon op EXE wordt gedrukt, herhaalt BROWSE de vorige Search;
E	Editeer eerste regel van record;
MODE	Hetzelfde als E;
C	Tel records; geeft ook aan de hoeveelste record; men op het scherm ziet (**count**);
T	Geef de tijd aan (**time**);
B	Browse op ander pak;
DEL	Verwijder record;
ON	Verlaat BROWSE.

BROWSE gebruikt de module PK$, die net na de listing wordt afgedrukt. PK$ laat toe de pak te selecteren met MODE, net als bij Find of Save.

```
BROWSE:
REM BROWSE 2.1
REM R.DICTUS JUL87-FEB89
LOCAL A%,R%,P$(1),K$(1), S$(16), O$(16)
O$=""
KSTAT 1
GPAK::
P$=PK$:("Browse")
TRAP OPEN P$+":MAIN",A,RECORD$
IF ERR<>0
  CLS
```



```
  PRINT "Foutief pak"
  GET
  GOTO GPAK::
ENDIF
TRAP USE A
IF ERR<>0
  PRINT "Bestand kan nietworden gebruikt"
  GET
  GOTO GPAK::
ENDIF
FIRST
V::
A%=DISP(-1,"")
REM    C
IF A%=67
  PRINT CHR$(15);"record",POS;"/";COUNT
  GET
ENDIF
REM    +
IF A%=88 AND POS<COUNT
  NEXT
ENDIF
REM    -
IF A%=82
  BACK
ENDIF
REM    L
IF A%=76
  LAST
ENDIF
REM    F
IF A%=70
  FIRST
ENDIF
IF A%=84
  PRINT CHR$(15);"    ";HOUR;":";MINUTE;":";SECOND;
  A% = GET
ENDIF
REM    ON
IF A%=1
  GOTO X::
ENDIF
REM    E
IF A%=69 OR A%=2
  TRAP EDIT A.RECORD$
  IF ERR<>0
```

```
      PRINT "EDITEREN NU NIETTOEGESTAAN"
      GET
      GOTO X: :
    ENDIF
    TRAP UPDATE
    IF ERR<>0
      PRINT "EDITEREN ONMOGE-LIJK"
      GET
    ENDIF
  ENDIF
  REM     DEL
  IF A%=8
    AT 1,2
    PRINT "     Delete Y/N     "
    K$=GET$
    IF K$=BY"
      ERASE
    ENDIF
    IF POS=COUNT+1
      BACK
    ENDIF
  ENDIF
  REM     B
  IF A%=66
     CLOSE
     GOTO GPAK: :
  ENDIF
  REM     S
  IF A%=83
    CLS
    PRINT CHR$(15);"<";LEFT$(O$,14);">";CHR$(14);
    PRINT "Search:";
    INPUT S$
    IF S$=""
      S$=O$
    ENDIF
     O$=S$
    IF POS<COUNT : NEXT : ENDIF
    A% = FIND (S$)
    IF A%=0
        CLS : AT 8-INT((LEN(O$)/2)),1 : PRINT O$
        PRINT "** not found ***";
        BEEP 200,800 : BEEP 200, 1000 : BEEP 200, 1400
        A%=GET : FIRST
    ENDIF
  ENDIF
```



```
GOTO V: :
X: :
```

Hier volgt de listing van de module PK$, die nodig is om Browse te kunnen gebruiken :

```
PK$: (A$)
LOCAL P$(1),E$(1),P%
CLS
CURSOR ON
P%=PEEKB($A2)
P$=CHR$(P%+65)
PRINT A$,P$;":";
E$="X"
WHILE E$<>CHR$(13)
  IF E$=CHR$(1)
    RETURN("")
  ENDIF
  IF E$=CHR$(2)
    P%=P%+1
    IF P%=3
      P%=0
    ENDIF
    P$=CHR$(P%+65)
    AT LEN(A$)+2,1 : PRINT P$;":";
  ENDIF
  E$=GET$
ENDWH
POKEB($A2),ASC(P$)-65
RETURN(P$)
```

Programma 25.3 : Sve (speciale Save)

Ik heb het altijd jammer gevonden dat het niet mogelijk is rechtstreeks via het klavier bepaalde tekens, zoals de **ampersand** (&) en het uitroepteken in te tikken. Zo gaat het gebruik van die tekens aan de SAVE-optie voorbij.

De oplossing is hier : SVE, een speciale versie van SAVE. Doorhet gebruik van coderingen worden de speciale tekens ingevoegd (we gebruiken bijvoorbeeld /A voor ampersand en /U voor uitroepteken). Het programma vertaalt de codes onmiddellijk.

Het programma houdt rekening met 4 regels tekst. Door een lege regel in te tikken (dus door gewoon op EXE te drukken), wordt SVE gestopt. NEER heeft geen effect, gebruik ook hiervoor EXE.

De codes zijn :

/A	Ampersand &	
/U	Uitroepteken	!
/V	Vraagteken	?
/1	Enkel aanhalingsteken	'
/S	Sigma	Σ

SVE gebruikt de modules IN$, WEG$ en VERT$ die na SVE afgedrukt staan, en PK$ (zie Browse).

```
SVE:
LOCAL L$(4,255),P$(1),T%
CLS
L$(1)=""
L$(2)=""
L$(3)=""
L$(4)=""
P$=PK$:("Sve")
IF P$=""
  STOP
ENDIF
OPEN P$+":MAIN",A,L1$,L2$,L3$,L4$
USE A
T%
WHILE T%<=4
INPUT L$(T%)
IF LEN(L$(T%))=0
  GOTO X::
ENDIF
L$(T%)=VERT$:(L$(T%))
T%=T%+1
ENDWH
X::
A.L1$=L$(1)
A.L2$=L$(2)
A.L3$=L$(3)
A.L4$=L$(4)
APPEND
CLOSE
```

IN$, een module geschreven voor SVE, krijgt een doelstring, een bronstring en een positie. De bronstring wordt op die positie in de doelstring ingevoegd.

```
IN$:(D$,B$,P%)
RETURN(LEFT$(D$,P%-1)+B$+RIGHT$(D$,LEN(D$)-P%+1))
```


Ook WEG$ werd speciaal voor SVE geschreven. Het neemt een string(als parameter A$) en wist daaruit L% tekens vanaf positie S%.

```
WEG$:(A$,S%,L%)
LOCAL V$(255)
V$=LEFT$(A$,S%-1)
V$=V$+RIGHT$(A$,LEN(A$)-(S%+L%)+1)
RETURN(V$)
```

VERT$ is de module die de codes omzet naar de eigenlijke tekens. De afkorting staat voor vertaal.

```
VERT$:(V$)
LOCAL L%,C$(255)
C$=V$
U::
L%=LOC(C$,"/U")
IF L%<>0
  C$=WEG$:(C$,L%,2)
  C$=IN$:(C$,CHR$(33),L%)
  GOTO U::
ENDIF
V::
L%=LOC(C$,"/V")
IF L%
  C$=WEG$:(C$,L%,2)
  C$=IN$:(C$,CHR$(63),L%)
  GOTO V::
ENDIF
O::
L%=LOC(C$,"/1")
IF L%
  C$=WEG$:(C$,L%,2)
  C$=IN$:(C$,CHR$(39),L%)
  GOTO O::
ENDIF
A::
L%=LOC(C$,"/A")
IF L%
  C$=WEG$:(C$,L%,2)
  C$=IN$:(C$,CHR$(38),L%)
  GOTO A::
ENDIF
S::
L%=LOC(C$,"/S")
IF L%
```

```
   C$=WEG$: (C$, L%, 2)
C$=IN$: (C$, CHR$(246), L%)
   GOTO S::
ENDIF
REM ———
RETURN(C$)
```

Programma 25.4 : Vind (een speciale versie van FIND)

De procedure VIND is een FIND die automatisch op iedere aanwezige pak zoekt. Dus geen FIND A:, FIND B: of FIND C: meer, maar gewoon VIND. Ikzelf heb de FIND uit mijn hoofdmenu vervangen door dit kleine maar handige programma.

```
VIND:
LOCAL V$(16), P$(1), X%, C%, F%, A%
REM Algemene FIND op alle datapaks
REM R. Dictus 17/11/87
CLS
PRINT "VIND: ";
INPUT V$
X%=1
WHILE X%<=3
 P$=CHR$(X%+64)
 IF EXIST(P$+":MAIN")
   OPEN P$+":MAIN", A, R$
   USE A
   C% = COUNT
   F% = FIND(V$)
   WHILE F%<>0
    A%=DISP(-1,"")
    IF A%=1 : STOP : ENDIF
    TRAP POSITION F%+1
    F%=FIND(V$)
   ENDWH
 ENDIF
 TRAP CLOSE
 X%=X%+1
ENDWH
CLS
PRINT "******************END OF PACKS**";
GET
```

Programma 25.5 : Sprint (Selectief Printen)

Net als SCOPY toelaat selectief records van de ene pak naar de andere te

copiëren, biedt SPRINT de mogelijkheid selectief records af te drukken (ofwel door te seinen naar de COMMS LINK).

SPRINT gebruikt de module PK$ van Browse.

```
SPRINT:
LOCAL BP$(1), ZOEK$(32), X%
CLS
BP$=PK$: ("SPRINT")
AT 1,2 : PRINT "ZOEK: ";
INPUT ZOEK$
CLS
PRINT "Ik zoek . . . ";
TRAP OPEN BP$+":MAIN", A, V1$, V2$, V3$, V4$, V5$, V6$
IF ERR = 246 : CLS : PRINT "*** GEEN PAK ***druk op space" :
   GET : STOP : ENDIF
IF ERR : CLS : PRINT "** FOUTIEF PAK *druk op space" : GET :
  STOP : ENDIF
USE A
FIRST
X%=FIND(ZOEK$)
WHILE NOT(EOF)
 IF X%
  IF A.V1$<>"" : LPRINT A.V1$ : ENDIF
  IF A.V2$<>"" : LPRINT A.V2$ : ENDIF
  IF A.V3$<>"" : LPRINT A.V3$ : ENDIF
  IF A.V4$<>"" : LPRINT A.V4$ : ENDIF
  IF A.V5$<>"" : LPRINT A.V5$ : ENDIF
  IF A.V6$<>"" : LPRINT A.V6$ : ENDIF
  LPRINT
 ENDIF
 NEXT
 X%=FIND(ZOEK$)
ENDWH
TRAP CLOSE
IF ERR : CLS : PRINT "** FOUTIEF PAK *druk op space" : GET :
ENDIF
```

Programma 25.6 : SNEL

Dit programma verhoogt schijnbaar de snelheid van de Organiser II. Voer het programma uit en zoek eens een gegeven op, bijvoorbeeld!

```
SNEL:
POKEB($0069), 0
POKEB($006A), 2
POKEB($0077), $08
```

26. Wiskundige bibliotheek

26.1 Inleiding

Dit hoofdstuk bevat een zogenaamde 'wiskundige bibliotheek' of een verzameling van wiskundige functies. Deze functies zijn allemaal geprogrammeerd in OPL. Daar waar een verkeerde parameter werd opgegeven en dus eigenlijk geen uitkomst kon worden berekend, werd -99 als uitkomst gebruikt.

Functies :

Algebra

```
38.1 : FAC (faculteit)
38.2 : BINOM (binomium)
```

Irrationele getallen

```
38.3 : WRT (wortel)
38.4 : WM (wortel-met-macht)
```

Logaritmen

```
38.5 : LOG (logaritme)
38.6 : E: (getal e)
```

Combinaties

```
38.7 : P (permutaties)
38.8 : C (combinaties)
38.9 : V (variaties)
```

Trigonometrie

```
38.10 : COT (cotangens)
38.11 : SEC (secans)
38.12 : COSEC (cosecans)
38.13 : SINH (sinus hyperbolicus)
38.14 : COSH (cosinus hyperbolicus)
38.15 : TGH (tangens hyperbolicus)
38.16 : COTH (cotangens hyperbolicus)
38.17 : SECH (secans hyperbolicus)
```



```
38.18 : COSECH (cosecans hyperbolicus)
38.19 : ISINH (inverse sinus hyperbolicus)
38.20 : ICOSH (inverse cosinus hyperbolicus)
38.21 : ITGH (inverse tangens hyperbolicus)
38.22 : ICOTH (inverse cotangens hyperbolicus)
```

Analyse

```
38.23 : DLOG (afgeleide van een logaritme)
38.24 : DLN (afgeleide van een natuurlijke logaritme)
38.25 : DLOGB (afgeleide van Briggse logaritme)
38.26 : DSIN (afgeleide van sinus)
38.27 : DCOS (afgeleide van cosinus)
38.28 : DTG (afgeleide van tangens)
38.29 : DCOT (afgeleide van cotangens)
38.30 : DISIN (afgeleide van inverse sinus)
38.31 : DICOS (afgeleide van inverse cosinus)
38.32 : DITAN (afgeleide van inverse tangens)
38.33 : DICOT (afgeleide van inverse cotangens)
38.34 : DSINH (afgeleide van sinus hyperbolicus)
38.35 : DCOSH (afgeleide van cosinus hyperbolicus)
38.36 : DTGH (afgeleide van tangens hyperbolicus)
38.37 : DCOTH (afgeleide van cotangens hyperbolicus)
38.38 : DISINH (afgeleide van inverse sinus hyperboli-
        cus)
```

26.2 Algebra

Functie 26.1 : FAC (faculteit)
Gebruik : x = FAC: (g)
Berekent de faculteit van g.

```
FAC: (G)
IF G = 0
  RETURN (1)
ENDIF
RETURN (G * FAC: (G-1))
```

Functie 26.2 : BINOM (binomium)
Gebruik : x = BINOM(n%, k%)
Berekent het binomium van Newton met basis n% en k% :

$$\binom{n\%}{k\%} = \frac{n\%!}{k\%!\ (n\%-k\%)\ !}$$

```
BINOM: (N%, K%)
RETURN ( (FAC: (N%) ) / ( (FAC: (K%) ) * (FAC: (N%-K%) ) )
```

26.3 Irrationele getallen

Functie 26.3 : WRT (wortel)
Gebruik : x = WRT: (n%, g)
Berekent de n%-de machtswortel van g. Wanneer n% = 0 is de uitkomst -99 (en dus opzettelijk fout).

```
WRT: (N%, G)
IF N% = 0
  RETURN (-99)
ENDIF
RETURN (G** (1/N%) )
```

Functie 26.4 : WM (wortel-met-macht)
Gebruik : x = WM:(n%, g%, m)
Berekent de n%-*de machtswortel van een getal* g dat tot de macht m verheven is.

```
WM: (N%, G, M)
IF N% = 0
  RETURN (-99)
ENDIF
RETURN (G** (M/N) )
```

26.4 Logaritmen

Functie 26.5 : LOG (logaritme)
Gebruik : x = LOG: (b%, a%)
Berekent de logaritme van argument a% met basis b%.

Merk op dat deze functie een andere functie LOG oproept. Ze is echter *niet* recursief; zoals gezegd betreft het hier een *andere* functie LOG, met één parameter, die in OPL is ingebouwd.

```
LOG: (B%, A%)
RETURN (LOG (A%) /LOG (B%) )
```

Functie 26.6 : E: (getal e)
Gebruik : x = E:

Berekent de numerieke constante e.

```
E:
RETURN (EXP (1) )
```

26.5 Combinaties

Functie 26.7 : P (permutaties)
Gebruik : x = P: (g)
Berekent het aantal mogelijke permutaties met g elementen.

```
P: (G)
RETURN (FAC: (P) )
```

Functie 26.8 : C (combinaties)
Gebruik : x = C: (n%, k%)
Berekent het aantal mogelijke combinaties van n% elementen aan k%.

```
C: (N%, K%)
RETURN (BINOM: (N%, K%) )
```

Functie 26.9 : V (variaties)
Gebruik : x = V: (n%, k%)
Berekent het aantal mogelijke variaties van n% elementen aan k%.

```
V: (N%, K%)
RETURN (FAC: (N%) /FAC: (N%-K%) )
```

26.6 Trigonometrie

Functie 26.10 : COT (cotangens)
Gebruik : x = COT: (a)
Berekent de cotangens van de hoek a, (a is in radialen).

```
COT: (A)
IF SIN (A) = 0
   RETURN (-99)
ENDIF
RETURN (COS (A) /SIN (A) )
```

Functie 26.11 : SEC (secans)

Gebruik : x = SEC: (a)
Berekent de secans van de hoek a, (a is in radialen).

```
SEC: (A)
IF COS(A) = 0
  RETURN(-99)
ENDIF
RETURN(1/COS(A))
```

Functie 26.12 : COSEC (cosecans)
Gebruik : x = COSEC: (a)
Berekent de cosecans van de hoek a, (a is in radialen).

```
COSEC: (A)
IF SIN(A) = 0
  RETURN(-99)
ENDIF
RETURN(1/SIN(A))
```

Functie 26.13 : SINH (sinus hyperbolicus)
Gebruik : x = SINH: (A)
Berekent de sinus hyperbolicus van de hoek a (a in radialen).

```
SINH: (A)
RETURN((E: **A) - (E: **-A))/2)
```

Functie 26.14 : COSH (cosinus hyperbolicus)
Gebruik : x = COSH: (A)
Berekent de cosinus hyperbolicus van de hoek a (a in radialen).

```
COSH: (A)
RETURN((E: **A) + (E: **-A))/2)
```

Functie 26.15 : TGH (tangens hyperbolicus)
Gebruik : x = TGH: (A)
Berekent de tangens hyperbolicus van de hoek a (a in radialen).

```
TGH: (A)
RETURN(SINH: (A)/COSH: (A))
```

Functie 26.16 : COTH (cotangens hyperbolicus)
Gebruik : x = COTH: (A)

Berekent de cotangens hyperbolicus van de hoek a (a in radialen).

```
COTH: (A)
IF TGH: (A) = 0
  RETURN(-99)
ENDIF
RETURN(1/TGH: (A))
```

Functie 26.17 : SECH (secans hyperbolicus)
Gebruik : x = SECH: (A)
Berekent de secans hyperbolicus van de hoek a (a in radialen).

```
SECH: (A)
IF COSH: (A) = 0
  RETURN(-99)
ENDIF
RETURN(1/COSH: (A))
```

Functie 26.18: COSECH (cosecans hyperbolicus)
Gebruik : x = COSECH: (A)
Berekent de cosecans hyperbolicus van de hoek a (a in radialen).

```
COSECH: (A)
IF SINH: (A) = 0
RETURN (-99)
ENDIF
RETURN(1/SINH: (A))
```

Functie 26.19 : ISINH (inverse sinus hyperbolicus)
Gebruik : x = ISINH: (A)
Berekent de inverse van de sinus hyperbolicus a.

```
ISINH: (A)
RETURN(LOG(A+SQR(A*A+1)))
```

Functie 26.20 : ICOSH (inverse cosinus hyperbolicus)
Gebruik : x = ICOSH: (A)
Berekent de inverse van de cosinus hyperbolicus a.

```
ICOSH: (A)
IF A<1
  RETURN(-99)
```

```
ENDIF
RETURN(LOG(A+SQR(A*A-1)))
```

Functie 26.21 : ITGH (inverse tangens hyperbolicus)
Gebruik : x = ITGH: (A)
Berekent de inverse van de tangens hyperbolicus a.

```
ITGH: (A)
IF NOT((A > -1) AND (A < 1))
  RETURN(-99)
ENDIF
RETURN(.5 * (LOG((1+A)/(1-A))))
```

Functie 26.22 : ICOTH (inverse cotangens hyperbolicus)
Gebruik : x = ICOTH: (A)
Berekent de inverse van de cotangens hyperbolicus a.

```
ICOTH: (A)
IF (A>-1) AND (A<1)
  RETURN(-99)
ENDIF
RETURN(.5 * LOG((1+A)/(A-1))))
```

26.7 Analyse

Functie 26.23 : DLOG (afgeleide van een logaritme)
Gebruik : x = DLOG: (b%, a%)
Berekent de afgeleide van de logaritme met basis b en argument a. Deze functie vereist de aanwezigheid op pak van functie 38.5 (LOG) en 38.6 (E).

```
DLOG: (B, A)
IF A = 0
  RETURN(-99)
ENDIF
RETURN(1/A * LOG: (B, E:))
```

Functie 26.24 : DLN (afgeleide van een natuurlijke logaritme)
Gebruik : x = DLN: (a)
Berekent de numerieke afgeleide van de natuurlijke logaritme met argument a.


```
DLN: (A)
IF A = 0
  RETURN(-99)
ENDIF
RETURN(1/A)
```

Functie 26.25 : DLOGB (afgeleide van Briggse logaritme)
Gebruik : x = DLOGB: (a)
Berekent de afgeleide van de Briggse logaritme met argument a. Deze functie vereist de aanwezigheid op pak van functie 38.6 (E).

```
DLOGB: (A)
IF A = 0
  RETURN (-99)
ENDIF
RETURN (1/A * LOG(E:))
```

Functie 26.26 : DSIN (afgeleide van sinus)
Gebruik : x = DSIN: (a)
Berekent de afgeleide van de sinus van de hoek a (a in radialen).

```
DSIN: (A)
RETURN(COS(A))
```

Functie 26.27 : DCOS (afgeleide van cosinus)
Gebruik : x = DCOS: (a)
Berekent de afgeleide van de cosinus van de hoek a (a in radialen).

```
DCOS: (A)
RETURN(-SIN(A))
```

Functie 26.28 : DTG (afgeleide van tangens)
Gebruik : x = DTG: (a)
Berekent de afgeleide van de tangens van de hoek a (a in radialen).

```
DTG: (A)
RETURN(1+ (TAN(X) *TAN(X)))
```

Functie 26.29 : DCOT (afgeleide van cotangens)
Gebruik : x = DCOT: (a)
Berekent de afgeleide van de cotangens van de hoek a (a in radialen). Deze functie vereist de aanwezigheid op pak van functie 38.10 (COT).

```
DCOT: (A)
RETURN(-(1+(COT: (A)*COT: (A))))
```

Functie 26.30 : DISIN (afgeleide van inverse sinus)
Gebruik : x = DISIN: (A)
Berekent de afgeleide van de inverse sinus a (a in radialen).

```
DISIN: (A)
IF A = 1
  RETURN(-99)
ENDIF
RETURN(1/(SQR(1-(A*A))))
```

Functie 26.31 : DICOS (afgeleide van inverse cosinus)
Gebruik : x = DICOS: (a)
Berekent de afgeleide van de inverse cosinus a (a in radialen).

```
DICOS: (A)
IF A = 1
  RETURN(-99)
ENDIF
RETURN(1/(SQR(1-(A*A))))
```

Functie 26.32 : DITAN (afgeleide van inverse tangens)
Gebruik : x = DITAN: (a)
Berekent de afgeleide van de inverse tangens a (a in radialen).

```
DITAN: (A)
RETURN(1/(1+(A*A)))
```

Functie 26.33 : DICOT (afgeleide van inverse cotangens)
Gebruik : x = DICOT: (a)
Berekent de afgeleide van de inverse cotangens a (a in radialen). Deze functie vereist de aanwezigheid op pak van functie 38.32 (DITAN).

```
DICOT: (A)
RETURN(-DITAN: (A))
```

Functie 26.34 : DSINH (afgeleide van sinus hyperbolicus)
Gebruik : x = DSINH: (a)
Berekent de afgeleide van de sinus hyperbolicus van hoek a (a in radialen). Deze functie vereist de aanwezigheid op pak van functie 38.14 (COSH).


```
DSINH: (A)
RETURN(DCOSH: (A))
```

Functie 26.35 : DCOSH (afgeleide van cosinus hyperbolicus)
Gebruik : x = DCOSH: (a)
Berekent de afgeleide van de cosinus hyperbolicus van hoek a (a in radialen). Deze functie vereist de aanwezigheid op pak van functie 38.13 (SINH).

```
DCOSH: (A)
RETURN(SINH: (A))
```

Functie 26.36 : DTGH (afgeleide van tangens hyperbolicus)
Gebruik : x = DTGH: (a)
Berekent de afgeleide van de tangens hyperbolicus van hoek a (a in radialen). Deze functie vereist de aanwezigheid op pak van functie 38.14 (COSH).

```
DTGH: (A)
IF COSH: (A) = 0
  RETURN(99)
ENDIF
RETURN(1/(COSH: (A)*COSH: (A)))
```

Functie 26.37 : DCOTH (afgeleide van cotangens hyperbolicus)
Gebruik : x = DCOTH: (a)
Berekent de afgeleide van de cotangens hyperbolicus van hoek a (a in radialen). Deze functie vereist de aanwezigheid op pak van functie 38.13 (SINH).

```
DCOTH: (A)
IF SINH: (A) = 0
  RETURN(-99)
ENDIF
RETURN(-1/(SINH: (A)*SINH: (A)))
```

Functie 26.38 : DISINH (afgeleide van inverse sinus hyperbolicus)
Gebruik : x = DISINH: (a)
Berekent de afgeleide van de inverse sinus hyperbolicus a (a in radialen).

```
DISINH: (A)
RETURN(1/(SQR(1+A*A)))
```

Deel V

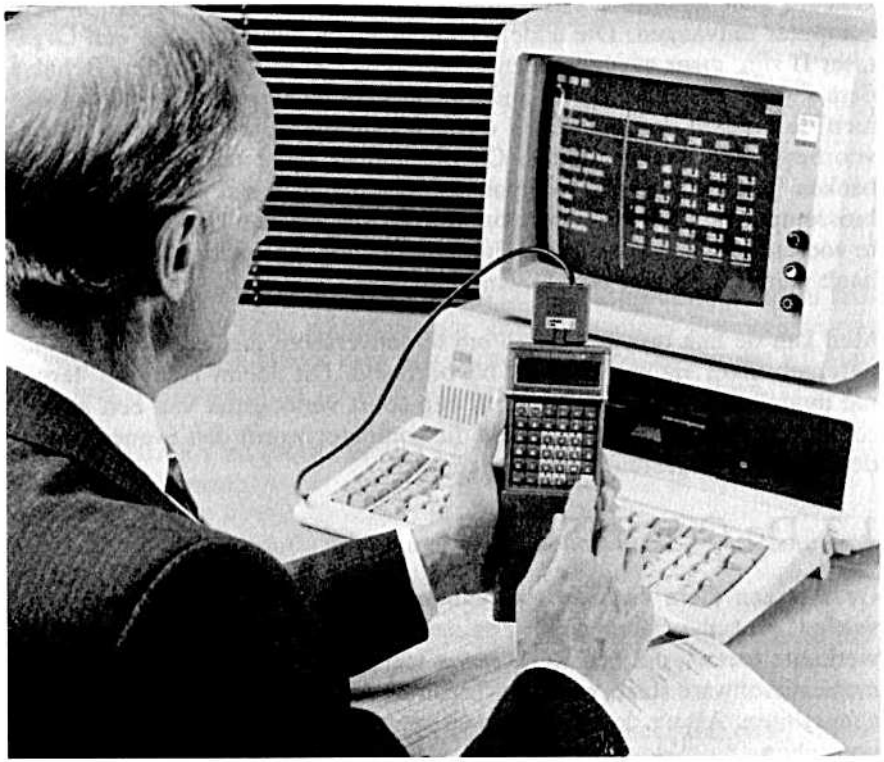
RANDAPPARATUUR

1. De communicatielink

1.1 Inleiding communicatiegebruiken

De communicatielink is het eerste van een reeks randapparaten die specifiek voor Organiser II-gebruik door Psion op de markt worden gebracht. Deze randapparaten zijn uiteraard optioneel en dienen een bepaald doel : het gebruiksspectrum van de Organiser II te verbreden.

Deze link is zonder twijfel het belangrijkste stuk randapparatuur dat men op het ogenblik voor de Organiser II kan krijgen. Hij zorgt namelijk voor de communicatie tussen de Organiser II en andere randapparaten (inclusief grote printers, computers, modems, e.d.). Het gebruik van de link is op zichzelf zeer breed en we zullen er dan ook de nodige aandacht aan schenken.



Er zijn verschillende manieren om elektronisch gegevens door te sturen : men kan dit doen door middel van *geluid*, bijvoorbeeld over een telefoonlijn (zie verderop). Een andere manier is de zgn. '*parallelle aansluiting*' (Eng: parallel interface), een aansluiting die een reeks bits tegelijk doorzendt of de meer populaire '*seriële aansluiting*' (Eng: *serial interface*), een aansluiting die bit voor bit doorzendt en waarin men een aantal parameters kan wijzigen (bijvoorbeeld de snelheid van transmissie). Psion heeft gekozen voor een seriële aansluiting in de communicatielink.

Deze link gebruikt een zgn. RS-232C -interface (ook wel -poort genoemd), die aansluiting met zowat alles mogelijk maakt. Een voorbeeld:

De Organiser II wordt verbonden met de COMMS LINK (zoals Psion de verbinding noemt), deze laatste wordt verbonden met een *modem*. Een modem (samentrekking van moduleren/demoduleren) is een toestel dat kan worden aangesloten op een gewone telefoon om gegevens en programma's (algemeen *data* genoemd) door te seinen naar andere computers. Een modem dient om inkomende gegevens (bits) om te zetten in geluiden. Deze geluiden komen dan over de telefoonlijn aan in een andere modem, die de geluiden weer omzet in bits en doorstuurt naar de verbonden computer. Resultaat : alle data is doorgeseind en zit nu in de andere computer. Omgekeerd gaat natuurlijk ook : door de telefoonlijn data van een andere computer ontvangen. Die andere computer kan natuurlijk óók een Organiser II zijn, maar evengoed een PC, of zelfs een reuzecomputer. Behalve data van de ene lijn naar de andere transfereren (eigenlijk copiëren) kan men de modem ook gebruiken om een aansluiting te realiseren met bijvoorbeeld een bankcomputer. Op die manier zijn trouwens al heel wat banken 'gekraakt'; slimme computerjongens die er in zijn geslaagd met hun huiscomputer in de bank 'in te breken' en wat willekeurige transacties uit te voeren! Het kan allemaal, en het gebeurt nog regelmatig. Deze grappen haalt men dus uit met behulp van modems.

Men kan de link ook gebruiken om te converseren met een andere computergebruiker via een netwerk bijvoorbeeld. Dit noemt men dan 'terminal mode' : de Organiser II wordt een soort verlengstuk van een andere computer. Alle schermuitvoer van die computer wordt dan eveneens naar de Organiser II gestuurd en omgekeerd.

1.2 De COMMS LINK

Deze is een klein vierkant toestelletje, waaraan een lange kabel is bevestigd, met aan het uiteinde een aansluiting met pinnetjes. Het kleine vierkante toestel, dat een ROM bevat van 32K waarop de complexe communicatiesoftware staat, wordt bovenaan de Organiser II (alleen model XP) aangesloten. Als we de afsluiting daar wegschuiven komt de aansluiting tevoorschijn.

De kabel is 1 meter lang. De 25-pinnige aansluiting is de standaard-seriële interface. Deze dient te worden aangesloten aan de randapparatuur.

De andere, kleinere aansluiting wordt zoals gezegd bovenaan de Organiser II bevestigd. Hij geeft een klik wanneer hij op zijn plaats zit. Hij moet worden aangebracht wanneer de Organiser II uit staat, anders kan er schade optreden (en verliest u uw garantie).

Wanneer de COMMS LINK succesvol is aangesloten en de Organiser II wordt aangezet (druk *twee* keer op ON) is er een extra optie verschenen op het hoofdmenu : COMMS (vlak voor 'OFF'). Deze optie wordt gebruikt voor rechtstreekse verbindingen met randapparaten.

Ook de programmeertaal OPL is nu verrijkt met extra functies. Deze functies zijn namelijk geprogrammeerd op de 32K ROM waarmee de COMMS LINK is uitgerust. Ook de optie COMMS staat op deze ROM geprogrammeerd.

Bij dit alles vinden we ook nog een floppy disk terug. Deze bevat software (programma's) voor communicatie tussen de Organiser II en de IBM PC/XT/AT en compatibelen. Deze software, waar we later dieper op in zullen gaan, is alleen geschikt voor bovenstaande machines.

Er wordt een 160-bladzijden-lange (Engelstalige) handleiding bijgeleverd. Wij zullen ons beperken tot een beknopte beschrijving van het gebruik van de COMMS LINK.

1.3 COMMS

De optie COMMS wordt bij gebruik van de link aan het hoofdmenu toegevoegd. COMMS is de afkorting voor 'communications' (communicaties of verbindingen). Wanneer men een rechtstreekse communicatie wil aangaan met randapparatuur zal men dus steeds deze optie bezigen.

Net zoals DIARY en PROG bevat COMMS een sub-menu. Dit menu heeft de volgende opties :

TRANSMIT ('zend door') : doorzenden van een bestand of procedure via de RS-232C-poort.

RECEIVE ('ontvang') : ontvangen van een bestand of procedure via de RS-232C-poort.

SETUP ('zet op') : parameters voor communicatie bekijken en/of veranderen.

TERM ('terminal') : maakt van de Organiser II een terminal.

AUTO ('automatisch') : bepaalt automatisch welke communicatieparameters nodig zijn.

CAPTURE ('vang op') : vangt binnenkomende data, editeert en stuurt weer uit.

BOOT ('start op') : voert een OPL-procedure uit die opgeslagen is op een aangesloten PC.

1.4 Communicatieparameters

Seriële communicatie werkt als volgt : de hele reeks bytes die moeten worden doorgeseind worden opgeslagen in een tijdelijke ruimte. Deze ruimte noemt men *buffer*. Daarna worden de bits één voor één doorgeseind door de lijn, met een door de gebruiker vast te stellen snelheid, gecontroleerd op wat men noemt hun *pariteit*. *Even pariteit* betekent dat het aantal 1-bits in de byte *oneven* is (bijvoorbeeld 3 of 5). Het wordt dan door een bijkomende bit even gemaakt. *Oneven pariteit* is precies het omgekeerde : als het aantal 1-bits even is wordt het oneven gemaakt. Pariteit wordt gebruikt om een goede controle te hebben van de doorgeseinde data zodat fouten tijdig kunnen worden opgespoord.

Soms worden de doorgeseinde gegevens onmiddellijk op het scherm weergegeven. Dit noemt men *echo*.

Men heeft ook speciale stuurcodes die het einde van een lijn weergeven (EOL : end-of-line) of het einde van het bestand (EOF : end-of-file). Deze stuurcodes kunnen door de gebruiker worden aangepast maar het spreekt vanzelf dat ze aan beide kanten van de lijn hetzelfde moeten zijn, anders treedt er verwarring op.

Er zijn voorts verschillende technieken om het correct doorseinen van data te bevorderen. Men laat de ontvanger bijvoorbeeld na iedere 255e byte bevestigen dat hij alles ontvangen heeft. Dit proces noemt men *handshaking* (handen schudden). Met wat fantasie kan men zich voorstellen waarom. Er zijn verschillende methoden van handshaking en de populairste zijn in de communicatiesoftware van de COMMS LINK opgenomen. Zij heten XON/XOFF en RTS/CTS. Men kan er ook voor kiezen géén handshaking toe te passen.

Men kan zelf kiezen hoe lang de bytes zijn die worden doorgestuurd : 7 of 8 bits. Het gebruik van 7 bits behelst het doorseinen van ASCII-bestanden (codes 0-127), 8 bits omvat binaire bestanden (zoals procedures). Bovendien kan men per byte één of twee zgn. '*stop-bits*' laten doorseinen of ontvangen, wederom om communicatiefouten zoveel mogelijk te elimineren.

Een heel gedoe om iets te klaren wat voor de mens eigenlijk triviaal is : communicatie. Bij elektronische communicatie komt echter heel wat te pas, dat is nu wel duidelijk.

Dit geheel van parameters (snelheid, even/oneven pariteit, ...) noemt men het *protocol*. Voor succesvolle communicatie is uiteraard vereist dat beide apparaten die aan elkaar zijn aangesloten hetzelfde protocol gebruiken. Dit protocol zal anders zijn voor een printer dan voor een bankkaart-lezer, bijvoorbeeld. We moeten dus iedere keer uitvissen hoe het benodigde protocol voor een bepaalde communicatie eruit ziet. Deze protocols kunnen gelukkig onder een bepaalde naam, bijvoorbeeld 'PRINTER', op een pak worden opgeslagen voor later hergebruik.

Er is al een protocol aanwezig in de interface. Dit wordt zichtbaar gemaakt en zo nodig veranderd met de optie SETUP.

1.5 SETUP

Met deze optie wordt het mogelijk de parameters van de communicatie te bekijken en ze desgewenst te wijzigen. Er zijn in totaal 15 parameters. Net als bij FIND in het hoofdmenu kunnen we met de cursortoetsen alle gegevens doorlopen.

De vijftien parameters hebben ieder een eigen regel bij SETUP. Hun namen staan steeds links op de regel; rechts staat hun waarde. Deze waarden gaan we waar nodig veranderen. De namen zijn onveranderlijk.

De parameters staan hieronder kort beschreven. Waar een zekere 'standaard'-inhoud door de Organiser II voorzien is, is die vet gedrukt (rechter kolom).

NAAM	BESCHRIJVING	MOGELIJKE INHOUD
BAUD	Snelheid van transmissie in bits per seconde	50, 75, 110, 150, 300, 600, 1200, 2400, 4800, **9600**
PARITY	Pariteit	**NONE**, ODD, EVEN, MARK, SPACE
BITS	Hoeveelheid bits per byte	**7**, 8
STOP	Hoeveelheid stopbits	**1**, 2
HAND	Methode van handshaking	NONE, **XON/XOFF**, RTS/CTS, XON+RTS, DTR, XON+DTR, RTS+DTR, ALL
PROTOCOL	Methode van doorseinen	**NONE**, XMODEM, PSION
ECHO	Echoën van tekens	LOCAL, **HOST**
WIDTH	Opgelegde lijnlengte	**NONE**, 1 tot 250
TIMEOUT	Verbinding verbreken na x seconden bij onsuccesvolle communicatie	**NONE**, 1 tot 255
REOL	Ontvangen einde v/d lijn	NONE, 1 of 2 tekens
REOF	Ontvangen einde v/h bestand	NONE, 1 of 2 tekens
RTRN	Vertalen van bepaalde ontvangen tekens in andere	**NONE**, 1 of 2 tekens
TEOL	Doorseinen einde v/d lijn	NONE, 1 of 2 tekens
TEOF	Doorseinen einde v/h bestand	NONE, 1 of 2 tekens
TTRN	Vertalen van bepaalde door te seinen tekens in andere	**NONE**, 1 of 2 tekens

De standaardparameters van enkele speciale functies van hierboven zijn :

NAAM	Standaard-inhoud	OP SCHERM ALS
REOL	`Industrie-standaard einde regel (tekens 13, 10)`	<CR><LF>
TEOL	"	<CR><LF>
REOF	`Industrie-standaard einde bestand (teken 26)`	<SUB>

De meeste van deze parameters worden bij elke machine toegepast. De andere parameters dienen dan te worden ingesteld zoals de documentatie van het apparaat aan de andere kant van de lijn dat voorschrijft. Kijk in ieder geval steeds de documentatie van de aangekoppelde machine na om te kijken hoe de parameters gezet moeten worden. Verkeerde parameters zullen *altijd* de communicatie in het honderd laten lopen.

1.5.1 Snelheid van datatransmissie : BAUD

De eenheid van snelheid van datatransmissie is de *baud*. Een snelheid van één baud betekent dat er één bit per seconde wordt doorgeseind. De meestgebruikte snelheden zijn 300, 1200 en 9600 baud. De Organiser II kan

echter seinen aan 50, 75, 110, 150, 300, 600, 1200, 2400, 4800 of 9600 baud. Het is alleen mogelijk aan deze snelheden te seinen of te ontvangen. Een snelheid van bijvoorbeeld 500 baud is dus onmogelijk.

De snelheid kan worden gekozen door LINKS of RECHTS in te drukken tot de gewenste snelheid rechts aangegeven staat. Voor communicatie met de PC zal meestal voor 9600 baud gekozen worden. Dit is dan ook als standaard voorzien.

Let er echter op dat het apparaat aan de andere kant van de lijn de gebruikte snelheid aankan.

1.5.2 PARITY

Hier moet worden aangegeven hoe de pariteit verloopt. Iedere byte wordt namelijk doorgeseind met 2, 3 of 4 extra bits : de startbit, die steeds het begin van de byte aangeeft en onveranderlijk is; de (optionele) pariteitsbit en de stopbit(s). De pariteitsbit dient om na te kijken of het aantal 1-bits in de byte even of oneven is. Naargelang de keuze by PARITY kan die pariteitsbit verschillende inhouden hebben. Deze keuze kan zijn *even* (de hoeveelheid bits wordt door een extra bit even gemaakt), *oneven* (ODD), *mark* (de pariteitsbit is steeds 1, of de eigenlijke pariteit nu even of oneven is), *space* (de pariteitsbit is steeds 0), of de standaard, NONE. Wanneer deze laatste gekozen is, wordt er geen pariteitsbit meegeseind. Dit heeft als voordeel dat de communicatie iets sneller verloopt, maar als nadeel dat er onopgemerkt fouten kunnen optreden.

1.5.3 BITS

Bestaat iedere byte uit 7 of 8 bits ? Normaal zijn het er acht. Maar het kan nodig zijn, namelijk bij zgn. 'ASCII-communicatie', slechts zeven bits door te sturen of te ontvangen. In dit geval kunnen de gestuurde of ontvangen tekens enkel die tekens zijn tussen ASCII- codes 0 en 127. Voor teksten en listings zijn zeven bits voldoende. Kijk toch voor de zekerheid na hoeveel bits aan de andere kant worden gebruikt!

1.5.4 STOP

Er worden bij iedere byte steeds 1 of 2 zgn. stop-bits meegeseind of -ontvangen. Deze stopbits zijn nodig om het einde van een byte aan te geven of om het aantal bits per byte, dat bij communicatie steeds 10 of 11 moet zijn, volledig te maken. Bijgevolg is de keuze van het aantal stopbits niet 100% vrij.

Wanneer het aantal bits per byte 7 bedraagt (BITS = 7) en de pariteit is

NONE (geen pariteitsbit), zal de Organiser II steeds 2 stop-bits toepassen. Bedraagt het aantal bits per byte echter 8 en is de pariteit niet NONE, dan zal de Organiser II steeds 1 stop-bit toepassen. De eigenlijke parameter bij STOP wordt in beide gevallen compleet genegeerd.

Deze regels worden overigens overal toegepast, niet alleen bij de Organiser II.

1.5.5 Handshaking : HAND

Handshaking is wederom een manier om te zien of er tijdens de verbinding geen fouten optreden. Door handshaking kan de transmissie ook tijdelijk door de machines worden stilgelegd. Dit komt voor wanneer de ontvanger de continue stroom van bytes niet kan verwerken en even een pauze 'inlast' tot hij weer verder kan.

Er zijn verschillende handshakingmethodes voorzien; XON/XOFF, RTS/CTS en DTR. Deze methodes kunnen allemaal met elkaar worden gecombineerd wanneer dat nodig is. Ze kunnen ook alledrie worden uitgeschakeld en in dat geval (NONE) vindt er geen handshaking plaats. Wanneer ALL wordt gekozen, zijn deze methodes alledrie tegelijk van toepassing.

1.5.5.1 NONE

In dit geval wordt geopteerd voor geen handshaking. Het bestand wordt dan byte voor byte doorgestuurd, aan één stuk door, zonder te pauzeren. Dit wordt gebruikt bij het doorsturen van gegevens naar machines die hoge snelheden aankunnen, zoals bijvoorbeeld PC's, of wanneer het aangekoppelde randapparaat geen van de volgende methodes kent.

1.5.5.2 XON/XOFF

Deze methode berust op het gebruik van twee speciale ASCII-tekens. Net als alle ASCII-tekens tussen 0 en 32 hebben deze tekens een naam. Ze worden in het algemeen XON of DC1 (CHR$(17)) en XOFF of DC2 (CHR$(19)) genoemd (DC staat voor *device control* : besturing van apparatuur). Wanneer de ontvanger aan de 'afzender' wil vertellen dat hij een pauze wil omdat hij de snelheid niet aankan, stuurt hij een XOFF : 'even de transmissie afzetten'. Daarna wordt XON doorgestuurd om te bevestigen dat de bytes weer kunnen worden doorgeseind.

XON/XOFF wordt zeer vaak gebruikt bij bijvoorbeeld printers.

1.5.5.3 RTS/CTS

RTS/CTS (*request-to-send, clear-to-send*; vragen om te seinen, klaar om te seinen) is een methode die het meest wordt gebruikt wanneer communicatie via *modems* geschiedt (dus over de telefoonlijn). Dit wordt rechtstreeks via bepaalde pinnen van de RS-232-interface ontvangen of doorgestuurd.

Deze methode is zeer complex. Het is handshaking die stop-bits gebruikt en een volledige beschrijving kunnen we in dit bestek niet geven.

1.5.5.4 DTR

DTR wordt slechts in één richting gebruikt : van de Organiser II naar het randapparaat. Dit in tegenstelling tot XON/XOFF en RTS/CTS die in beide richtingen werken.

DTR is de afkorting voor *data terminal ready* en wordt, net als de zojuist behandelde methode, rechtstreeks via een pin van de interface doorgestuurd. Om het niet onnodig verwarrend te maken zullen we deze ingewikkelde werking ook niet beschrijven.

1.5.6 PROTOCOL

Er zijn niet alleen verschillende snelheden van communicatie, verschillende methoden van handshaking en verschillende lengtes van bytes, er zijn ook verschillende manieren om bestanden door te seinen en te ontvangen. In verband hiermee spreekt men van een *seinprotocol*.

De COMMS LINK van de Organiser II kent slechts twee protocols : XMODEM en PSION.

1.5.6.1 NONE

Géén seinprotocol : men spreekt van ASCII-communicatie. De bestanden worden byte voor byte doorgeseind, aan één stuk door. Er wordt niet op eventuele fouten gelet.

1.5.6.2 PSION

Het PSION-protocol is uiteraard ontwikkeld door Psion zelf. Het gebruik beperkt zich tot het communicatieprogramma dat bij de interface wordt geleverd voor gebruik op de PC. Wanneer PSION wordt gebruikt, wordt de eventueel gekozen XON/XOFF-handshaking ongedaan gemaakt.

1.5.6.3 XMODEM

XMODEM (spreek uit : *cross-modem*) is een protocol dat zijn gegevens doorstuurt in *blokken* : het bestand wordt ingedeeld in een aantal blokken met een vaste lengte. Dit bestand wordt dan blok voor blok doorgeseind. Wanneer in een bepaald blok een fout optreedt, hoeft het bestand niet weer vanaf het begin doorgeseind te worden; vanaf het foutieve blok is voldoende.

1.5.7 ECHO

Deze parameter wordt alleen gebruikt wanneer de Organiser II als een *terminal* fungeert. Een terminal is een apparaat, dat de gebruiker verbindt met een computer op afstand. Wanneer de Organiser II een terminal *emuleert* (dus doet of hij een terminal is), kan hij worden gebruikt als aansluiting op een andere computer. De toetsen die dan bij de Organiser II worden ingedrukt, worden doorgeseind naar die computer. De 'reactie' van die computer wordt dan teruggeseind naar de Organiser II. Zo lijkt het alsof de Organiser II een verlengstuk is van de computer aan de andere kant van de lijn (of omgekeerd).

ECHO kan LOCAL of HOST zijn.

1.5.7.1 LOCAL

Wanneer de ECHO LOCAL is, worden de tekens die in de Organiser II worden ingetoetst ook op het scherm getoond, zodat we steeds kunnen zien wat we typen. LOCAL wordt alleen gebruikt wanneer de verbonden computer (de 'HOST') géén ECHO toepast (meestal is dat echter wel zo).

1.5.7.2 HOST

Dit is standaard. Meestal zal de *host* zelf ECHO toepassen en komen de ingetoetste tekens op het scherm van de Organiser II èn bij de host zelf. Wanneer ECHO op HOST staat en de host zelf ook nog ECHO toepast, wordt ieder ingetoetst teken twee keer op het scherm weergegeven. In dat geval moet uiteraard ECHO LOCAL worden gebruikt.

1.5.8 TIMEOUT

Het kan zijn dat de verbinding tussen de twee machines niet direct tot stand komt. Er moet dan even worden gewacht. Wanneer dit wachten echter een zekere tijdslimiet overschrijdt, bijvoorbeeld na 30 seconden, spreekt men van een *device timeout*. De verbinding is dan niet gelukt.

Bij het gebruik van sommige nieuwe OPL-instructies en bij TRIGS, waarover later meer, kan de timeout-periode worden beïnvloed. Deze kan worden *uitgeschakeld* (de link wacht oneindig lang) of op 1 tot 255 seconden (4.25 minuten) worden gezet.
Wanneer de timeout-periode overschreden wordt, genereert de Organiser II een fout. Dit is een nieuwe fout, die door de link wordt voorzien : DEVICE WRITE ERR (code 192).

1.5.9 WIDTH

WIDTH zorgt voor de regellengte van de doorgestuurde gegevens en wordt bijna uitsluitend gebruikt voor terminalemulatie. Er kan een bepaalde re-

gellengte aan de interface worden opgelegd. Wanneer die lengte bereikt is zal de interface automatisch een EOL (end-of-line) invoegen. Dit is enkel nodig wanneer het aangekoppelde apparaat zelf geen end-of-line verzorgt wanneer dat nodig is.

1.5.10 REOL

REOL staat voor *ontvang einde v/d regel*(Engels : *receive end-of-line*). Het omvat één of twee tekens die de Organiser II in staat stellen te weten wanneer het einde van een ontvangen regel bereikt is. Deze code moet uiteraard gelijk zijn aan de code die het aangekoppelde apparaat gebruikt om het einde van een regel aan te geven.

REOL mag zijn NONE (géén aanduiding voor het einde van een regel), of één of twee tekens. Meestal zijn het twee tekens : CHR$(13) + CHR$(10). Deze tekens heten respectievelijk *carriage return* en *line feed* en worden afgebeeld als <CR><LF>.

1.5.11 TEOL

TEOL staat voor *sein einde v/d regel* (Engels : *transmit end-of-line*). Het wordt gebruikt om het aangekoppelde randapparaat te laten weten wanneer het einde van de regel bereikt is. Dit apparaat moet uiteraard deze code kunnen herkennen.

Net als REOL mag TEOL NONE zijn, of één of twee tekens. Wanneer de Organiser II gebruikt wordt als een terminal, zal hij TEOL doorseinen bij het indrukken van EXE.

1.5.12 REOF en TEOF

REOF staat voor *ontvang einde v/h bestand* (Engels : *receive end-of-file*). De werking is hetzelfde als REOL.

TEOF staat voor *sein einde v/h bestand* (Engels : *transmit end-of-file*). De werking is hetzelfde als TEOL.

Meestal wordt voor EOF het teken CHR$(26) gebruikt, of het nu voor seinen (TEOF) of ontvangen (REOF) is. CHR$(26) wordt genoteerd als <SUB>.

1.5.13 RTRN en TTRN

RTRN (zet ontvangen tekens om in andere tekens; Eng. : *receive translate*) en TTRN (zet door te seinen tekens eerst om in andere tekens; Engels : *transmit translate*) doen precies wat hun benamingen aangeven.

Zij worden gebruikt wanneer het nodig is een bepaald teken dat ontvangen of doorgeseind wordt te vertalen in een ander teken. Wie deze mogelijkheden niet wenst te gebruiken maakt ze inactief met NONE. Wanneer slechts één teken wordt opgegeven, wordt dit teken vanzelf verwijderd tijdens het zenden/ontvangen. Wanneer er twee tekens worden opgegeven, wordt het eerste omgezet in het tweede tijdens het zenden/ontvangen.

1.5.14 Weg uit SETUP

Nadat alle parameters gezien en/of gewijzigd zijn, verlaten we SETUP door op MODE te drukken. We komen dan terecht in een submenu. Dit menu heeft de volgende opties :

EXIT	Weg uit SETUP; de eventuele veranderingen worden onthouden.
ABANDON	Weg uit SETUP; de eventuele veranderingen worden niet onthouden.
EDIT	Keer terug naar de parameterlijst om eventueel verdere veranderingen aan te brengen.
SAVE	Zet de huidige parameters op pak onder een naam.
LOAD	Lees vroeger geSAVEde parameters in.
DIR	Geef inhoudsopgave van parameters op pak.
ERASE	Wis vroeger geSAVEde parameters.
RESET	Gebruik de standaard parameters; de huidige worden vergeten.

De werkingen van deze opties spreken voor zichzelf. Bij SAVE, LOAD en ERASE moet een naam worden opgegeven. Deze naam moet uniek zijn en beantwoorden aan de algemene voorwaarden voor bestandsnamen : 8 tekens, enkel letters en cijfers, beginnend met een letter.

1.6 Doorseinen : TRANSMIT

TRANSMIT dient om bestanden of procedures over te seinen door de COMMS LINK. Daarvoor worden de parameters van SETUP gebruikt. Ook TRANSMIT brengt ons in een submenu :

FILE	Stuur een bestand door (bijvoorbeeld A: MAIN).
PROCEDURE	Stuur een procedure door (bijvoorbeeld A: TEL).

Wanneer de keuze gemaakt is, vraagt de Organiser II de naam van het bestand of de procedure op te geven. Als we FILE hebben gekozen, zal er staan :

```
SEND A:MAIN
```


Door MODE te drukken kan de paknaam worden gewijzigd. De naam MAIN is dus standaard en wanneer we een ander bestand willen doorseinen, drukken we op ON en tikken een nieuwe naam in. Wanneer de naam in orde is, drukken we op EXE waarop de Organiser II een connectie zal aangaan.

Met het PSION-protocol echter moet ook de naam worden opgegeven die de PC zal gebruiken om het ontvangen bestand op schijf op te slaan.

1.7 Ontvangen : RECEIVE

RECEIVE ontvangt bestanden of procedures door de COMMS LINK. Daar worden eveneens de parameters van SETUP voor gebruikt. Ontvangen procedures moeten eerst naar q-code worden omgezet met TRAN voor ze kunnen worden gebruikt.

De werkwijze van RECEIVE is identiek aan die van TRANSMIT.

1.8 Terminalemulatie : TERM

Deze optie verandert de Organiser II tijdelijk in een terminal. Alle binnenkomende tekens worden onmiddellijk op het scherm weergegeven en alle ingedrukte toetsen worden onmiddellijk doorgeseind naar de andere computer.

1.8.1 Speciale tekens

Speciale tekens, die normaal niet via het Organiser II- klavier kunnen worden ingetikt, worden tijdens terminalemulatie ingetikt via speciale codes. Deze codes worden verkregen door de toetsen LINKS en RECHTS te gebruiken in combinatie met andere toetsen.

De toets LINKS wordt gebruikt om de toets CTRL op PC's te emuleren. LINKS-A zal dus CHR$(1) produceren, LINKS-Z levert CHR$(26) op, enzovoort.

De toets RECHTS werkt als een speciale SHIFT die speciale tekens doorseint.

LINKS met levert	RECHTS met levert
A ^A	A ^[
B ^B	B ^\
C ^C	C ^]
D ^D	D ^^
E ^E	E ^_
F ^F	F !
G bel	G #
H delete	H &
I tabulator	I ‘
J line feed	J ?
K ^K	K @
L nieuw blad	L [
M carriage return	M \
N ^N	N]
O ^O	O ^
P ^P	P -
Q XON	Q ‘
R ^R	R {
S XOFF	S |
T ^T	T }
U ^U	U ~
V ^V	V delete
W ^W	W ?
X ^X	X ?
Y ^Y	Y ?
Z ^Z	Z ?

Het ASCII-teken *backslash* (\) wordt op het scherm van de Organiser II gebracht als een *Yen* (¥). Het teken ~ wordt veranderd in een →. Ontvangen *tabulatortekens* (CHR$(9)) worden op het scherm gebracht als een grafisch teken met daarin de afkorting TB.

1.8.2 Uit TERM

Om uit de optie TERM te gaan wordt gewoon ON gebruikt. Bij handshaking kan TERM weer gebruikt worden terwijl de oude gegevens nog steeds aanwezig zijn, anders zijn ze verloren.

1.9 CAPTURE

CAPTURE ('vang op') stelt in het geheugen van de Organiser II een *buffer* op : een ruimte in het geheugen waar bepaalde gegevens tijdelijk worden opgeslagen voor ze naar een randapparaat of een andere ruimte worden overgebracht. Terwijl de gegevens in de buffer zitten kunnen we desgewenst wijzigingen aanbrengen.

CAPTURE heeft een submenu. Dit heeft de volgende opties :

TERM	Emuleer een terminal, met CAPTURE-buffer.
EDIT	Editeer de inhoud van de buffer.
CLEAR	Maak de buffer leeg (door hem uit te wissen).
SAVE	Zet de inhoud van de buffer op pak (met naam).
TRANSMIT	Sein de inhoud van de buffer door.

1.9.1 TERM

De buffer is steeds zo groot als nodig is. Hij groeit mee met de hoeveelheid binnenkomende data.

De TERMinal-emulatie werkt net zoals de TERM van het COMMSmenu. Het verschil is dat alle binnenkomende data onmiddellijk naar de buffer gaat waar later kan worden ge-editeerd. Door in de emulatie op ON te drukken komen we weer in het CAPTUREmenu terecht.

1.9.2 EDIT

Het editeren van de buffer gebeurt precies zoals het editeren van een procedure bij EDIT in het PROGmenu. Echter, CHR$(9), de tabulator, wordt ook weergegeven met een speciaal symbool (afkorting TB). Wanneer MODE wordt ingedrukt, keert EDIT terug naar het CAPTUREmenu.

1.9.3 CLEAR

Deze optie maakt de gehele buffer leeg. Het is dus een gevaarlijke optie. Daarom vraagt de Organiser II voor de zekerheid of we de buffer wel leeg willen hebben.

1.9.4 SAVE

SAVE werkt, zoals we kunnen verwachten, zowat hetzelfde als RECEIVE of TRANSMIT in het COMMSmenu. Het presenteert een klein submenu met de keuze de buffer als FILE of als PROCEDURE op pak te zetten. Dit maakt het bijvoorbeeld mogelijk een procedure die op schijf staat te laden, te editeren en op pak te zetten, in één beweging.

1.9.5 TRANSMIT

TRANSMIT werkt hier bijna hetzelfde als in het COMMSmenu. Er wordt echter niet gevraagd welk bestand het betreft, maar de inhoud van de buf-

fer wordt automatisch doorgeseind.

1.10 AUTO

AUTO dient om te weten te komen welke communicatieparameters het aangesloten apparaat gebruikt, als we deze informatie niet kunnen vinden in de bijhorende handleiding.

Het werkt als volgt : AUTO gebruikt de vier parameters BAUD, PARITY, BITS en STOP en loopt voor iedere parameter alle opties door (*handshaking* wordt dus *niet* getest). Ondertussen zijn die opties van het scherm af te lezen en worden ze via de COMMS LINK naar het randapparaat gestuurd. Wanneer de Organiser II dus bijvoorbeeld aan een printer is gekoppeld, zullen de juiste parameters, wanneer bereikt, op het papier afgedrukt worden.

AUTO stopt niet wanneer een goede connectie is bereikt. Het kan gestopt worden door ON in te drukken of door te wachten tot alle opties doorlopen zijn. De parameters in SETUP zijn steeds de recentst geteste.

1.11 BOOT

BOOT dient om een procedure, geschreven in de machinetaal van de microprocessor (HD6303X), via IBM PC, XT, AT of compatible te laden (terwijl die computer het programma CL draait, dat bij de COMMS LINK op floppy disk wordt meegeleverd) en uit te voeren. Het is dus alleen interessant wanneer er Organiser II-programma's in machinetaal op schijf beschikbaar zijn.

De naam van de procedure op schijf dient te worden ingetikt en wordt dan automatisch geladen. Om uit BOOT te ontsnappen drukken we op ON.

2. Programmeren van de COMMS LINK

Instructies : LPRINT, LSET, XFCLOSE, XFEOF, XFPOS, XFPUT, XTSEND, XTRECV
Functies : LINPUT$, TRIG$, XFGET$

2.1 Inleiding

In het vorige hoofdstuk hebben we de mogelijkheden van de COMMS LINK kort besproken. We hebben het echter bijna steeds gehad over werking van de COMMS LINK via menu's. Het is ook mogelijk de COMMS LINK te programmeren, met instructies die door het aankoppelen van de link aan OPL worden toegevoegd. Het is duidelijk dat deze instructies alleen dan kunnen worden gebruikt wanneer de link aangekoppeld is, anders herkent de Organiser II ze niet.

Buiten een aantal instructies zijn op de COMMS LINK ook nog enkele speciale procedures geprogrammeerd die kunnen worden gebruikt om hele bestanden ineens naar/van een PC te sturen/ontvangen.

2.2 LSET

LSET wordt gebruikt om de parameters te wijzigen. Ze worden als een lange lijst opgegeven.

LSET
Syntax : LSET: (baud%, pariteit%, bits%, stop%, hand%, echo%, width%, reol$, reof$, rtrn$, teol$, teof$, ttrn$, timeout%, protocol%)

(Voor de duidelijkheid hebben we steeds de functie als parameter gebruikt i.p.v. <exp%>. Het betreft echter overal expressies en geen gewone variabelen. De parameter baud% bijvoorbeeld, mag dus ook een berekening bevatten).

Om een bepaalde parameter *ongewijzigd* te laten, kunnen we -1 in zijn plaats zetten (zelfs daar waar strings verwacht worden, zoals bijvoorbeeld reol$). Om de snelheid op 9600 baud te zetten, maar alle andere parameters ongewijzigd te laten, kunnen we dus gebruiken :

```
LSET: (9600,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1)
```

Het is echter ook mogelijk de parameters die volgen weg te laten als ze niet veranderd worden. De bovenstaande regel zouden we dus evengoed kunnen schrijven als :

```
LSET: (9600)
```

Alle parameters kunnen bovendien hun standaardwaarde krijgen door LSET geen parameters te geven :

```
LSET: ()
```

Zoals we zien, is het nodig getallen op te geven bij de meeste parameters. Die getallen worden gekozen uit de volgende tabel :

PARAMETER	GETAL	MOGELIJKHEDEN
baud%	50-9600	50, 75, 110, 150, 300, 600, 1200, 2420, 4800, 9600
pariteit%	0-4	0: NONE, 1: ODD, 2: EVEN, 3: MARK, 4: SPACE
bits%	7, 8	7: 7 bits/byte, 8: 8 bits/byte
stop%	1, 2	1: 1 stop-bit, 2: 2 stop-bits
hand%	0-7	0: NONE, 1: XON, 2: RTS, 3: XON+RTS, 4: DTR, 5: XON+DTR, 6: RTS+DTR, 7: ALL
echo%	0, 1	0: LOCAL, 1: HOST
protocol%	0-2	0: NONE, 1: XMODEM, 2: PSION
width%	0-250	0: NONE, 1-250: 1-250 tekens/regel
timeout%	0-255	0: NONE, 1-255: 1-255 seconden

De stringparameters bevatten een *lege* string voor NONE, of één of twee tekens waar nodig.

2.3 LPRINT

Deze instructie werkt net zoals PRINT, maar i.p.v. naar het scherm te schrijven, wordt naar de COMMS LINK geschreven. LPRINT heeft dezelfde syntax als PRINT. Het drukken kan echter onderbroken worden door ON in te drukken.

Voorbeeld :

Programma 2.1 : COMMTEST

```
COMMTEST:
LSET: () :REM Standaardparameters
LPRINT "Communicatie OK"
```

2.4 LINPUT$

LINPUT$ (Engels: *line-input*) is een functie, die als invoerapparaat de COMMS LINK gebruikt i.p.v. het klavier (zoals INPUT).

LINPUT$
Gebruik : x$ = LINPUT$: (<exp%>) | (<exp%1>, <exp%2>)
Wanneer slechts één parameter wordt opgegeven, wacht LINPUT$ tot het <exp%> tekens ontvangen heeft (of tot EOL of EOF is bereikt) en geeft die weer in de resulterende string. Als die ene parameter de waarde 0 heeft, wordt de invoerbuffer leeggemaakt (zoals met CLEAR in het CAPTURE-menu).

Wanneer twee parameters worden opgegeven, bevat de eerste de functie die hierboven beschreven is, en de tweede de *timeout* in seconden. Deze laatste mag enkel tussen 0 en 255 liggen.

Voorbeelden :
Om een string van 50 tekens te laden voor de standaard timeout :

```
L$ = LINPUT$: (50)
```

Om een string van 50 tekens te laden voor de limiet van 60 seconden :

```
L$ = LINPUT$: (50, 60)
```

2.5 TRIG$

Deze functie doet hetzelfde als LINPUT$, maar seint éérst een string door.

TRIG$
Gebruik : x$ = TRIG$: (<exp%>, <exp$>) | (<exp%1>, <exp%2>, <exp$>)

De string <exp$> wordt eerst naar de COMMS LINK gestuurd, vervolgens wordt een string van de link ingelezen zoals bij LINPUT$.

2.6 XTSEND

XTSEND is een procedure die dient om een bestand in de Organiser II naar een PC te sturen die het programma CL draait.

XTSEND
Syntax : XTSEND: (<exp$1>, <exp$2>, <exp%>)
De eerste parameter, <exp$1>, bevat de *naam*van het bestand zoals het op de PC zal heten. De *backslash*(\) wordt weergegeven met een gewone *slash* (/).

Voorbeelden van deze parameter :

```
"C: /ORG2/PROC"
  "A: /DATA"
```

De tweede parameter, <exp$2>, bevat de naam van het bestand op de Organiser II. De paknaam mag erbij worden vermeld.

Voorbeelden van deze parameter :

```
"A: MAIN"
 "MAIN"
"B: KISTEN"
```

De derde parameter, <exp%>, bepaalt het *type* bestand en de *extensie* op de PC, volgens deze tabel :

<exp%>	TYPE	EXTENSIE OP PC
0	gewoon bestand	.ODB
1	procedure (listing)	.OPL
2	agendabestand	.OB2
3	procedure (incl. q-code)	.OB3
4	SETUP-parameters	.OB4
5	werkblad	.OB5

(Het werkblad, type 5, wordt enkel gebruikt met speciale software).

Type 0 en 1 zijn tekstbestanden, dus 7-bits, die op de PC met een tekstverwerker of *editor* (een soort beperkte tekstverwerker) kunnen worden gewijzigd. Ze kunnen in MS-DOS worden afgedrukt als volgt:

```
C>COPY <bestandsnaam> PRN:
```


Type 2 tot 5 zijn binaire bestanden, dus 8-bits, en kunnen dus door de PC niet gebruikt worden. Zij worden alleen maar naar de PC overgebracht om tijdelijk op schijf opgeslagen te worden en weer ingelezen wanneer nodig (de schijf van de PC wordt dus een soort vierde pak).

Voorbeeld : om de OPL-procedure INKIST te copiëren naar schijf C: op de PC :

```
XTSEND: ("C: /INKIST", "INKIST", 1)
```

2.7 XTRECV

XTRECV is een procedure die dient om een bestand van de PC die het programma CL draait naar de Organiser II te sturen.

De gehele werking, inclusief syntax, is identiek aan die van XTSEND.

2.8 XFOPEN

XFOPEN is de eerste in een reeks procedures die dienen om bestanden die op de schijf van een PC opgeslagen zijn op te roepen alsof ze op een pak stonden. Merk op dat wanneer XON/XOFF-handshaking werd gebruikt, dit door deze functies en procedures ongedaan wordt gemaakt.

XFOPEN
Syntax : XFOPEN: (<exp$>, <exp%1>, <exp%2>)
Opent een bestand, genaamd <exp$>, opgeslagen op een schijf van de aangesloten PC, rekening houdend met een aantal voorschriften.

Er mag slechts één bestand tegelijkertijd open zijn. Het is dus niet mogelijk verschillende schijfbestanden te gebruiken, zoals bij paks.

Net als bij XTSEND en XTRECV, mag de naam van het bestand een volledige specificatie bevatten, bijvoorbeeld :

```
"B: FILES DEMO"
```

De tweede parameter, <exp%1>, bevat een zogenaamde *toegangscode* tussen 0 en 4 :

CODE	BESCHRIJVING
0	Het bestand mag enkel gelezen worden.
1	Het bestand wordt gecreëerd en, als het reeds bestaat, wordt de oude versie over schreven.
2	De oude versie van het bestand wordt over schreven.
3	Het bestand wordt gecreëerd. Indien het reeds bestaat wordt een foutmelding gegeven.
4	Het bestand wordt geopend met als doel te schrijven. Als het een binair bestand is kan er ook van gelezen worden.

De derde parameter, <exp%2>, bevat het *type* van het bestand. Dit type is een nummer tussen 0 en 2, volgens deze tabel :

CODE	BESCHRIJVING
0	Binair bestand (8-bits).
1	Tekstbestand (ASCII, 7-bits).
2	Directory.

Een directory (2) mag enkel worden geopend met toegangscode 0 (lezen).

Wanneer de toegangscode 4 is, en de typecode is 1 (tekst), dan wordt de *pointer* naar het einde van het bestand gebracht. Zo kunnen snel records aan het bestand worden toegevoegd.

Wanneer de toegangscode 1 is, maar de typecode is 0 (binair), dan blijft de pointer aan het begin van het bestand staan.

2.9 XFCLOSE

Het momenteel geopende bestand op schijf wordt gesloten.

2.10 XFEOF

XFEOF werkt voor bestanden op schijf zoals EOF voor bestanden op pak.

2.11 XFGET$

Deze functie leest een string uit het momenteel geopende bestand.

XFGET$

Gebruik : x$ = XFGET$: (<exp%>)
De parameter bevat de *lengte* van de in te lezen string in bytes. Wanneer voortijdig het einde van het bestand bereikt is, zal de string minder lang zijn dan <exp%> tekens. De lengte mag niet meer zijn dan 255 tekens.

Als het bestand een tekstbestand is, leest XFGET$ een record in en verplaatst de pointer naar het volgende record. Als geprobeerd wordt voorbij het laatste record te lezen is de resulterende string leeg.

2.12 XFPUT

De instructie XFPUT schrijft een string naar het bestand, op de plaats waar de pointer staat.

XFPUT
Syntax : XFPUT: (<exp$>)
De string wordt naar het bestand geschreven. Dit bestand moet uiteraard geopend zijn voor schrijven (dus niet met toegangscode 0).

Als het bestand een tekstbestand is, wordt de string achteraan het bestand bijgevoegd.

2.13 XFPOS

Dit is het schijfequivalent van POSITION. De pointer van het geopende, binaire bestand wordt verschoven naar een nieuwe positie. Echter, XFPOS is geen instructie maar een functie.

XFPOS
Gebruik : x = XFPOS: (<exp%>, <exp>)
De eerste parameter bevat een code die bepaalt hoe de tweede parameter, die de positie in het bestand weergeeft, moet worden geïnterpreteerd :

CODE	POSITIE
0	Absoluut; dus t.o.v. het begin van het be stand.
1	Ten opzichte van de huidige positie
2	Ten opzichte van het einde van het bestand.

De tweede parameter is een reëel getal. Wanneer de eerste parameter 1 is en de tweede is 0, zoals in het volgende voorbeeld, geeft XFPOS gewoon de huidige positie in het bestand weer:

```
A% = XFPOS: (1,0)
```

Om de pointer naar het eerste record te verplaatsen :

```
A% = XFPOS: (0,1)
```

Om de pointer naar het volgende record te verplaatsen :

```
A% = XFPOS: (1,1)
```

Het vorige record :

```
A% = XFPOS: (1,-1)
```


3. De streepjescodelezer

3.1 Inleiding

De streepjescodelezer is een toestel dat gebruikers van de Organiser II de mogelijkheid geeft met de handige streepjescode (Eng. : *bar code*) te werken. Deze code wordt toegepast voor identificatie van zaken als handelswaar, bibliotheekboeken enzovoort.

De lezer kan rechtstreeks aan de machine worden aangesloten, zonder de COMMS LINK. Hij kan verschillende soorten streepjescode lezen. Bovendien wordt een speciale functie, BAR$, aan OPL toegevoegd.

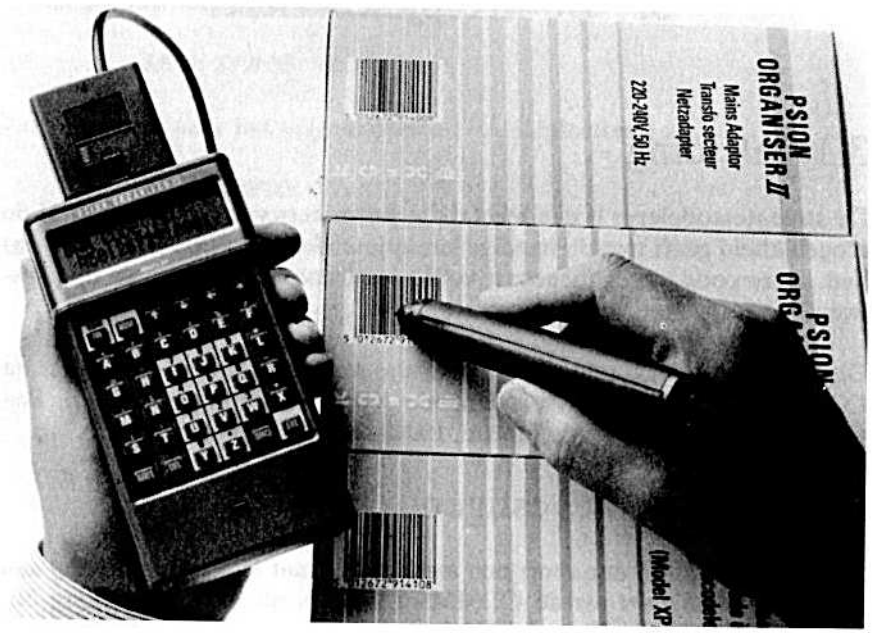
3.2 De lezer bevestigen

De lezer bestaat uit een soort pen aan de ene kant en een aansluiting aan de andere kant. Net als de COMMS LINK wordt deze aansluiting bovenaan de Organiser II model XP bevestigd. Dit dient te gebeuren wanneer de Organiser II uit staat. Daarna wordt hij met ON weer aangezet, met als resultaat dat de software die op de interface aanwezig is, in de Organiser II wordt geladen. Deze software – die 4K in omvang is – maakt het lezen van streepjescode mogelijk. Alles is dan klaar om te beginnen.

3.3 Streepjescode lezen

Streepjescode wordt gelezen door de pen er zonder druk in één beweging over te trekken, van links naar rechts of van rechts naar links. Het is steeds het beste een zo recht mogelijke lijn te trekken. De snelheid kan liggen tussen 1 tot 10 centimeter per seconde (Psion raadt 5 centimeter per seconde aan). Het lezen kan worden gehinderd als de achtergrond van de streepjescode rood is. Dit komt doordat de lezer een rood licht gebruikt om de strepen te onderscheiden.

De streepjescodelezer gebruikt stroom. De batterij van de Organiser II zal dus aanzienlijk vlugger leeg raken wanneer de lezer intensief wordt gebruikt!



3.4 Soorten streepjescode

De lezer kan 4 verschillende soorten streepjescode onderscheiden. Dit zijn:

EAN 8 tekens/13 tekens
UPC
CODE 39
ITF

Deze types kunnen allemaal door elkaar worden gebruikt. De lezer zal ze automatisch onderscheiden en de juiste gegevens opslaan.

3.5 BAR$

Dit is een nieuwe functie, die aan OPL wordt toegevoegd wanneer de lezer is ingeschakeld.

BAR$
Gebruik : x$ = BAR$(<exp%1>, <exp%2>)

De eerste expressie bevat het type streepjescode, de tweede de modus. Het type wordt als volgt gebruikt:

TYPE	AFKORTING	<exp%1>
EAN	A	1
UPC	B	2
EAN, UPC	A of B	3
CODE 39	C	4
EAN, CODE 39	A of C	5
UPC, CODE 39	B of C	6
EAN, UPC, CODE 39	A, B of C	7
ITF	D	8
EAN, ITF	A of D	9
UPC, ITF	B of D	10
EAN, UPC, ITF	A, B of D	11
CODE 39, ITF	C of D	12
EAN, CODE 39, ITF	A, C of D	13
UPC, CODE 39, ITF	B, C of D	14
Alle vier	A, B, C, of D	15

Wanneer de code gelezen is, bevat de resulterende string de afkorting van het type (A, B, C, of D) als eerste teken. De rest van die string bevat dan de gegevens van de streepjescode zelf.

De tweede parameter, <exp%2>, bepaalt de modus van de lezing. Die is als volgt ingedeeld :

0 : De lezer wacht tot de code 100% succesvol is ingelezen.

−1 : De lezer wacht tot de code 100% succesvol is ingelezen, of tot de gebruiker een toets indrukt.

>0 : De lezer wacht tot de code 100% succesvol is ingelezen, of tot <exp%2> twintigsten van een seconde.

<−1 : De lezer wacht tot de code 100% succesvol is ingelezen, de gebruiker een toets heeft ingedrukt of <exp%2> twintigsten van een seconde.

Wanneer een toets wordt ingedrukt, is het mogelijk met de functie GET de ASCII-waarde te bepalen (zie Appendix A).

Het is mogelijk dat, wanneer geen tijdsduur wordt aangegeven (modus 0), de lezer vanzelf de werking stopzet. Dan spreekt men van een *device ti-*

meout : de leespoging duurde te lang.

De lezer voegt geen nieuwe foutmeldingen aan OPL toe.

3.6 De Y-connector

De Y-connector is een toestel dat bestaat uit een streepjescodelezer en een COMMS LINK. Dit (prijzige) apparaat maakt het mogelijk streepjescode naar de PC (of andere computers) te sturen via de Organiser II. Het wordt aan de Organiser II gekoppeld net als alle andere apparaten. Opgelet : twee randapparaten verbruiken uiteraard veel elektriciteit!

Programma 3.1 : BARCODE

Het nu volgende programma leest barcodes in (en kan er enkele herkennen). In de regels 12 en 16 stuurt het de code en de naam van het produkt ook door de COMMS LINK.

```
 1 BARCODE:
 2 LOCAL B%, BC$(15), N$(10,15), P$(4,32), V%
 3 N$(1)="A5012672910001" : P$(1)="Organiser II XP"
 4 N$(2)="A5012672936452" : P$(2)="COMMS LINK"
 5 N$(3)="A5012672905106" : P$(3)="16K DATAPAK"
 6 N$(4)="A7612100002476" : P$(4)="ISOSTAR POEDER
   450g"
 7 X::
 8 CLS
 9 PRINT "Klaar"
10 BC$=BAR$:(15,-1)
11 BEEP 20,100
12 LPRINT BC$
13 B%=1
14 WHILE B%<=4
15    IF N$(B%)=BC$
16       LPRINT P$(B%)
17       V%=VIEW(2,P$(B%))
18       B%=500
19    ENDIF
20    B%=B%+1
21 ENDWH
22 IF B%<>501
23    V%=VIEW(2,"ONBEKEND PRODUKT ONBEKEND PRODUKT")
24 ENDIF
25 GOTO X::
```


4. De magneetkaartlezer

4.1 Inleiding

De magneetkaartlezer is een zwart doosje waaraan de verbinding met de Organiser II is bevestigd. Deze verbinding gaat, net als de COMMS LINK en de streepjescodelezer, rechtstreeks in de opening bovenaan de Organiser II.

De bedoeling van dit toestel is het lezen van magneetkaarten (zoals betaalkaarten) met de Organiser II mogelijk te maken. Psion is daar uitstekend in geslaagd. Waar zij niet in geslaagd zijn is het samenstellen van een handleiding voor die technologie. Dit hoofdstukje maakt alles wat duidelijker voor (potentiële) gebruikers van de lezer.

4.2 De werking van de lezer

De lezer (Eng. : *swipe reader*) kan gegevens van magneetkaarten lezen. Die magneetkaarten bevatten twee magnetische sporen waarop de gegevens zijn aangebracht. De lezer is echter beperkt tot het tweede spoor (wie wil weten waarom, belt even naar Psion).

Het tweede spoor bevat maximaal 39 tekens van 8 bits. De lezer geeft er maximaal 37 door : het startteken en het eindteken (speciale tekens die de lezer gebruikt om het begin en het einde van de gegevens te vinden) worden niet doorgegeven. Dat is ook nergens voor nodig, want die tekens hebben geen enkele invloed op de gegevens die op het spoor staan.

De inhoud van dat spoor is niet gestandaardiseerd, er kan dus van alles op staan, in vrije volgorde. Een programma dat gegevens van één type kaart kan inlezen zal dus niet noodzakelijk met andere kaarten werken. Hoe dan de structuur van de gegevens te achterhalen? Ofwel door de gegevens ergens op te zoeken of na te vragen, of door, zoals de Engelsen zeggen, '*trial and error*' : blijven zoeken tot de structuur gevonden is. Maar wees van één ding zeker : vertrouwelijke gegevens zijn op spoor 2 niet aangebracht.

4.3 Programmeren van de lezer : SWIPE$

Er wordt geen standaardsoftware bij de lezer geleverd zoals bijvoorbeeld bij de COMMS LINK. De enige manier om de lezer te gebruiken is dus zelf de software te schrijven.

Daarvoor dient de functie SWIPE$.

SWIPE$
Gebruik : x$ = SWIPE$: (<exp1%>, <exp2%>)

De eerste expressie bevat het type *pariteit* dat gebruikt wordt :

1 voor even pariteit
2 voor oneven pariteit
3 voor zowel even als oneven pariteit (altijd goed)

De tweede expressie bevat het type van de *timeout* :

0 : de functie wordt slechts verlaten bij succesvol lezen
−1 : de functie wordt verlaten bij succesvol lezen of na het indrukken van een toets
>0 : de functie wordt verlaten bij succesvol lezen of na <exp2%> twintigsten van een seconde
<−1 : combinatie van de twee voorafgaande types

Wanneer de functie verlaten wordt door het indrukken van een toets of het verlaten van de timeout-periode is het resultaat de lege string (""). Zo niet, dan kan de correcte pariteit nagekeken worden op adres $E0; 1 voor even en 2 voor oneven.

Programma 4.1 : MISTER CASH

Het volgende programma illustreert het gebruik van SWIPE$. Dit voorbeeld dient om enkele gegevens van een bankkaart type *Mister Cash* in te lezen. Het is niet zeker of het ook met andere bankkaarten werkt.

```
 1 CARD:
 2 REM BANKKAARTLEZER
 3 REM (C) 1987 R. Dictus
 4 LOCAL B%, C$(37)
 5 CLS
 6 PRINT "KAARTLEZER"
 7 C$=SWIPE$: (3, -1)
 8 B%=VIEW(2, "Rek. nr "+MID$(C$, 3, 3)+"-
   "+MID$(C$, 7, 7)+"-"+ MID$(C$, 14, 2))
 9 B%=VIEW(2, "Kaartnr. "+MID$(C$, 18, 8))
10 B%=VIEW(2, "Geldig tot eind 19"+MID$(C$, 29, 2))
```


5. De printer

5.1. Inleiding

Een printer kon aan het gamma randapparatuur natuurlijk niet ontbreken.

De Psion Printer II is speciaal ontwikkeld voor de Organiser II en kan op alle modellen gebruikt worden. Hij is zo ontworpen dat uw Organiser II er netjes in past en geruststellend vastklikt. Het ergonomische ontwerp van de printer beantwoordt volledig aan het uiterlijk van de Organiser II. Hij meet 253 x 153 x 76 mm, niet te groot dus. Daarenboven verandert de printer de computer in een krachtig en draagbaar systeem.

Deze printer, die vanuit OPL programmeerbaar is (de instructie LPRINT zit reeds in iedere Organiser II), is niet de eerste de beste. Hij kan bijvoorbeeld afdrukken verzorgen in 20, 40, 60 of zelfs 80 tekens per regel. Hij drukt op smal thermisch papier (10 cm drukwijdte), waarvan attent een rol wordt bijgeleverd. Net zoals de grotere printers is het mogelijk tekens te onderlijnen, extra groot te drukken en zelfs grafisch te werken. Hij drukt de regels tegen een snelheid van 0.8 regels per seconde. Hij is capabel om zo'n 300 000 regels af te drukken zonder mechanische storingen.

Om het systeem echt draagbaar te maken werkt deze matrixprinter met een (bijgeleverde) herlaadbare batterij (Nikkel-Cadmium, 7.2 Volt) waar nodig. Met een volle batterij kan de printer een uur aan de slag zonder ophouden, daarna moet de batterij herladen worden, wat zo'n 12 tot 14 uur duurt. Maar het kan ook zonder : met de (eveneens bijgeleverde) 9-Volt-adapter kan de printer gewoon op het elektriciteitsnet worden aangesloten. In dat geval krijgt ook de Organiser II zelf zijn stroom van het net en blijft de batterij gespaard. De batterij wordt herladen door de printer op het net aan te sluiten zonder hem te gebruiken.

Aan de linkerkant van de printer is een extra aansluitingspoort voorzien (net als aan de bovenkant van de Organiser II), om bijvoorbeeld een streepjescodelezer (zie verder in dit boek) aan te sluiten.

5.2 Algemeen gebruik

Wanneer aangesloten legt de printer beslag op zo'n 2.5K intern geheugen. Wanneer men applicaties wenst te draaien die het volledige geheugen nodig hebben is het dus nodig de printer te ontkoppelen.

Zodra de gebruiker ON intikt op het klavier van de Organiser II wordt ook de printer geactiveerd. Nogmaals ON drukken laadt de extra software, die op de interface is aangebracht, in het geheugen. Deze software neemt zo'n 2.5K in beslag. Het kan dus voorvallen dat, wanneer de printer is aangesloten, bepaalde applicaties klagen dat er te weinig werkgeheugen vrij is. In dat geval volstaat het de printer te ontkoppelen, naar het hoofdmenu te gaan en wederom op ON te drukken, zodat de ingenomen ruimte vrijgemaakt wordt.

5.3 Printerbesturing via het klavier

De printer kan rechtstreeks via het klavier van de Organiser II bestuurd worden. De volgende mogelijkheden zijn ingebouwd :

— Line feed (lege regels);
— Afdruk huidige scherminhoud/huidig record;
— Stopzetten printer.

Lege regels
Om drie lege regels te 'printen' (om verschillende afdrukken duidelijk te kunnen onderscheiden, bijvoorbeeld), volstaat het de combinatie SHIFT SPACE te drukken. De printkop beweegt naar de linkermarge (in het jargon noemt men dit een carriage return) en het papier wordt drie regels verdergeschoven (driedubbele line feed).

Afdruk huidige scherminhoud/huidig record
Om de scherminhoud op de printer te krijgen dient men de combinatie SHIFT LINKS in te drukken. Het papier wordt een regel doorgeschoven en het scherm wordt afgedrukt. In de optie FIND wordt i.p.v. een schermafdruk het huidig record volledig afgedrukt.

Stopzetten printer
De printer kan te allen tijde worden onderbroken door ON in te drukken. De printkop blijft in dat geval staan. Om hem naar de linkermarge te bewegen wordt SHIFT SPACE ingedrukt.

5.4 Vanuit OPL

Zoals gezegd is de instructie LPRINT, het printer-equivalent van PRINT, reeds in OPL aanwezig. Deze instructie wordt op precies dezelfde wijze gebruikt als PRINT. Enkele voorbeelden :

```
LPRINT "Hallo"
```

drukt het woord Hallo op papier. De volgende LPRINT-instructie zal het

papier doorschuiven naar de volgende regel (line feed); de printkop gaat naar de linkermarge (carriage return, net als de cursor op het scherm).

```
LPRINT "Hallo";
```

doet hetzelfde, alleen wordt de volgende LPRINT vlak naast Hallo afgedrukt. Het papier of de printkop bewegen dus niet (wederom : net als de cursor op het scherm).

```
LPRINT "1",
```

drukt een 1 en een spatie. Dit is dus eigenlijk identiek aan

```
LPRINT "1";
```

behalve dat het korter is.

OPGELET : de printer drukt NIETS AF tot er een carriage return of een form-feed ("volgende pagina") wordt doorgegeven, of het einde van de regel bereikt is.

Een carriage return kan gedwongen worden als volgt :

```
LPRINT
```

Een form-feed verkrijgt men zo :

```
LPRINT CHR$(12)
```

5.5 Speciale functies

Selectie van tekenbreedte kan via OPL worden opgelegd, en wel als volgt :

5.5.1 Tekenbreedte

20 kolommen	`LPRINT CHR$(14);`
40 kolommen	`LPRINT CHR$(15);`
60 kolommen	`LPRINT CHR$(23);`
80 kolommen	`LPRINT CHR$(22);`

Op deze manier wordt eenvoudig de tekenbreedte geselecteerd, bijvoorbeeld :

```
LPRINT CHR$(14); "Roger Rabbit"
LPRINT CHR$(23); "60 tekens", CHR$(15), "40 tekens"
LPRINT CHR$(22) : LPRINT "80 tekens"
```

5.5.2 Onderlijnen

Begin onderlijnen LPRINT CHR$(21);
Stop onderlijnen LPRINT CHR$(24);

Voorbeeld :

```
LPRINT CHR$(21); "Dit is onderlijnd"; CHR$(24)
LPRINT "Dit is niet onderlijnd"
LPRINT
```

5.5.3 Allerlei

Lege regel LPRINT
Tabuleer LPRINT CHR$(9);

Carriage return LPRINT CHR$(13);
Volgende pagina LPRINT CHR$(12);

Voorbeeld :

```
LPRINT
LPRINT "1"; CHR$(9); "2"; CHR$(9); "3"
LPRINT ". . . "; CHR$(13); "——"
LPRINT CHR$(12);
```

Dubbele hoogte LPRINT CHR$(17);
Dit drukt de tekens op de huidige lijn dubbel hoog. De tekens op de volgende lijn worden vanzelf weer op de gewone hoogte afgedrukt.

Voorbeeld :

```
LPRINT
LPRINT CHR$(17); "Extra hoog"
LPRINT "Gewoon"
```

De verschillende codes kunnen ook door elkaar worden gebruikt, bijvoorbeeld om zowel dubbel hoog als onderlijnd te drukken :

```
LPRINT CHR$(17); CHR$(21); "HOOG ONDERLIJND"; CHR$(24)
```


5.6 Grafisch printen

Niet alleen kunnen we teksten afdrukken met de Psion Printer II, het is ook mogelijk tekeningen (*graphics*) te drukken. Hiervoor dient de instructie GPRINT.

Syntax : GPRINT: (<exp%1>, <exp%2>)
De eerste expressie (<exp%1>) geeft aan hoeveel bytes er moeten worden afgedrukt. De tweede expressie (<exp%2>) geeft aan waar in het geheugen die bytes precies staan : het is dus een **adres**.

Er zijn evenwel enkele beperkingen. Allereerst is het onmogelijk op één lijn tekst en graphics te mengen. GPRINT zal altijd de nog niet afgedrukte tekens (tekens die nog in het geheugen zitten te wachten tot er een carriage return komt, bijvoorbeeld; zie punt 5.4) uit het geheugen (de zgn. *buffer*) verwijderen. De printkop zal altijd naar de linkermarge verschoven worden. Na het drukken zal GPRINT het papier één regel doorschuiven en wederom de printkop naar de linkermarge brengen.

En nu terug naar de eigenlijke graphics. Zoals gezegd geeft de tweede parameter een adres aan van een reeks (tabel) bytes die moeten worden gedrukt. Iedere byte bestaat uit 8 bits. Iedere bit stelt een puntje voor op het papier. Het is dus nodig, om graphics te printen, te kunnen rekenen in *binair*. Hoe binaire getallen zijn samengesteld staat te lezen in hoofdstuk 2 van deel I van dit boek (*Beknopte inleiding informatica*).

De bytes gaan van links naar rechts, d.w.z. de eerste byte geeft de eerste kolom van 8 bits, de tweede byte de tweede kolom, enzovoort. Een praktijkvoorbeeld zal dat duidelijk maken.

Het volgende voorbeeldprogramma zal het tekentje (de driehoek),die vergroot en omkaderd naast het programma is weergegeven, op de printer afdrukken :

```
VBGPRINT:
LOCAL T%, G%(8)
G%(1) = 1
G%(2) = 3
G%(3) = 7
G%(4) = 15
G%(5) = 31
G%(6) = 63
G%(7) = 127
G%(8) = 255
GPRINT: (8, ADDR(G%()))
```

De eerste byte in de tabel heeft als waarde 1, omdat de meest linkse kolom van het grafiekje het binaire patroon 00000001 weergeeft. De decimale waarde van het binaire getal 00000001 is immers 1. De tweede byte heeft als waarde 3, omdat de tweede kolom (van links) het binaire patroon 00000011 weergeeft, wat 3 is in decimaal, enz. Uiteindelijk geeft de rechtse kolom (11111111) het getal 255 weer. De eerste parameter naar GPRINT is 8 omdat we 8 bytes willen afdrukken.

Eén regel op de printer kan 256 van deze grafische kolommen naast elkaar afdrukken.

Merk tot slot nog op dat het drukken in grafische modus (dus met GPRINT) aanzienlijk trager gaat dan de gewone LPRINT.

5.7 Printen naar de COMMS LINK

Zoals in hoofdstuk 1 van deel V van dit boek beschreven wordt is het mogelijk een seriële communicatiekabel aan de Organiser II te koppelen. Aan deze kabel kan bijvoorbeeld een seriële printer of een computer worden aangesloten. Het is dan mogelijk via de instructie LPRINT gegevens door de kabel naar het andere apparaat te sturen.

Op het eerste gezicht lijkt dit problemen op te leveren : LPRINT drukt immers naar de Psion Printer II wanneer deze is aangesloten... Daar is gelukkig aan gedacht. De oplossing vinden we in de instructie XPRINT%.

Syntax : `XPRINT%: (<exp%>)`
Wanneer `XPRINT%: (0)` wordt gebruikt, gaan alle verdere LPRINTs naar de COMMS LINK.
`XPRINT%: (1)` daarentegen stuurt alle LPRINTs naar de printer.

Let op het percentageteken en de dubbele punt!

5.8 Foutmeldingen

De printer kan foutmeldingen opleveren, die niet standaard in OPL voorzien zijn. Welke foutmeldingen dat precies zijn (bijvoorbeeld : PRINTER-BATTERIJ BIJNA LEEG) vinden we terug in Appendices B en C.

Appendix A

De Organiser tekenset

A.1 Inleiding

De Organiser II gebruikt een uitgebreide vorm van de ASCII-code (American Standard Code for Information Interchange). Hij heeft codes vanaf 0 tot 255.

Echter, de codes 128 tot 160 bevatten *lege tekens*. Waarom Psion van deze ruimte geen gebruik heeft gemaakt om bijvoorbeeld Europese tekens, als é of ö op te slaan, is mij een raadsel.

A.2 UDG's

De eerste 8 tekens van de karakterset, 0 tot 7, zijn voorbehouden voor UDG's.

A.3 Controletekens

De karakterset bevat ook enkele speciale tekens, de *controle- tekens*. Deze vormen geen teken op het scherm, maar zorgen voor een codering van bijvoorbeeld het bewegen van de cursortoetsen. Het zijn :

`CHR$(8)`	Veegt het vorige teken uit en beweegt de cursor één plaats naar links.
`CHR$(9)`	Tabuleert : de cursor komt op het midden van de regel of op het begin van de volgende regel terecht.
`CHR$(10)`	Line feed : de cursor gaat naar de volgende regel.
`CHR$(11)`	De cursor gaat naar de positie links bovenaan ophet scherm. Deze plaats wordt de *thuisplaats genoemd.*
`CHR$(12)`	Maakt het scherm leeg, zoals CLS.
`CHR$(13)`	Carriage return : zet de cursor op het begin van de regel.
`CHR$(14)`	Wist de bovenste regel van het scherm en plaatst de cursor op de thuisplaats.

CHR$(15) Veegt de onderste lijn van het scherm uit en plaatst de cursor beneden links, onder de thuisplaats.

CHR$(16) Produceert een bieptoon. Dit teken is bij andere computers meestal CHR$(7).

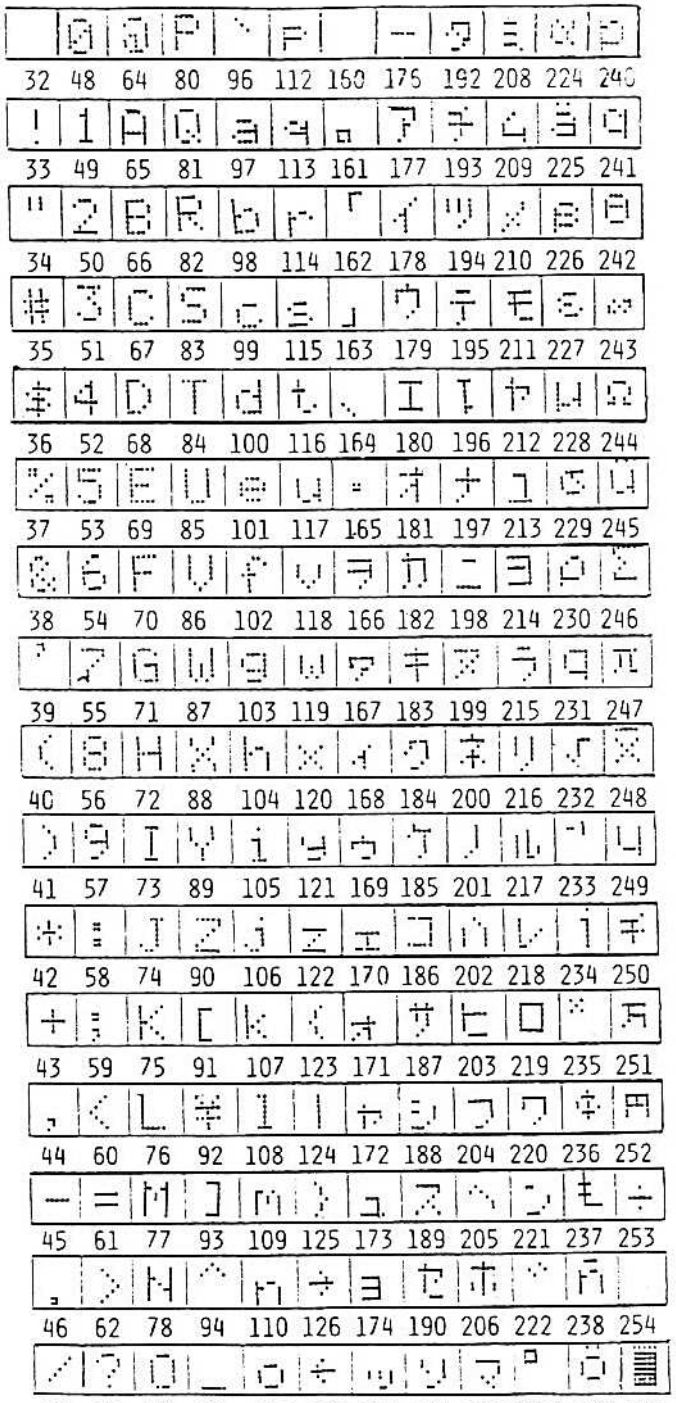
A.4 Speciale toetsen

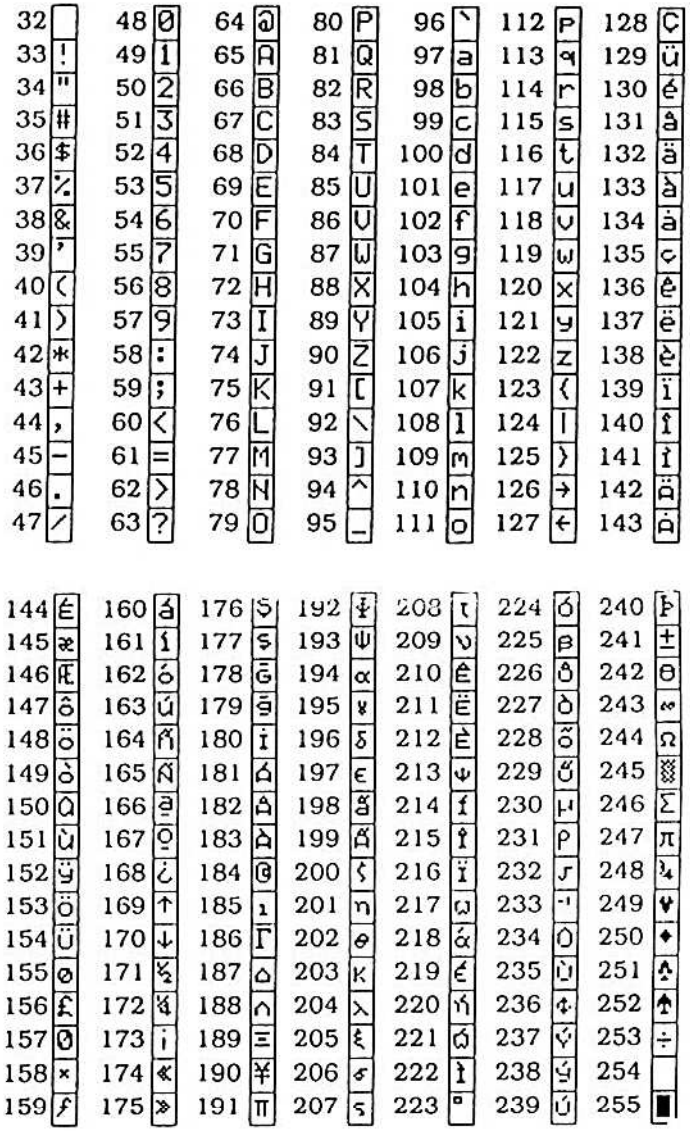
De 'speciale toetsen' van het klavier krijgen een eigen teken toegekend. Wanneer deze toetsen worden ingedrukt, krijgen de functies GET, GET$, KEY en KEY$ deze ASCII-codes.

ON	CHR$(1)
MODE	CHR$(2)
OP	CHR$(3)
NEER	CHR$(4)
LINKS	CHR$(5)
RECHTS	CHR$(6)
SHIFT-DEL	CHR$(7)
DEL	CHR$(8)
EXE	CHR$(13)

A.5 Lijst ASCII-codes

Deze lijst geeft alleen de *afdrukbare* tekens, dus tussen de codes 32 (spatie) en 255 (blok).





Appendix B

Foutmeldingen (op nummer)

255 NO ALLOC CELLS
Een machinetaalprogramma dat wordt uitgevoerd, probeert interne ruimte aan te roepen die niet bestaat.

254 OUT OF MEMORY
Het interne geheugen van de Organiser II is helemaal opgebruikt.

253 EXPONENT RANGE
een getal is buiten de exponentbegrenzing gegaan. Deze moet liggen tussen -99 en +99.

252 STR TO NUM ERR
Er is geprobeerd een string naar een getal om te zetten (met VAL) terwijl die string helemaal geen getal bevatte.

251 DIVIDE BY ZERO
Er is geprobeerd te delen door nul.

250 NUM TO STR ERR
Komt voor wanneer men geprobeerd heeft met de CALCulator een getal om te zetten naar een string (CALC kan niet werken met strings) of tijdens een foutieve berekening in een machinetaalprogramma.

249 STACK OVERFLOW
Een machinetaalprogramma vernietigt de machinestack (zie Appendix D, Hoofdstuk 34).

248 STACK UNDERFLOW
Een machinetaalprogramma vernietigt de machinestack (zie Appendix D, Hoofdstuk 34).

247 FN ARGUMENT ERR
Tijdens het aanroepen van een functie zijn verkeerde parameters gebruikt (bijv. te weinig parameters, een string waar een integer gebruikt moest worden, enz.).

246 NO PACK
De Organiser II probeert een pak te gebruiken dat er niet is, of waarvan niet gelezen kan worden. In het laatste geval is dit pak beschadigd.

245 WRITE PACK ERR
Het is niet gelukt naar een bepaalde pak te schrijven. Misschien is de pak vol of beschadigd.

244 READ ONLY PACK
Er zit een oude datapak in de Organiser II (een zwarte pak van de eerste Organiser). Het is niet mogelijk deze te beschrijven, zelfs niet na formatteren.

243 BAD DEVICE NAME
De enige bestaande paknamen zijn A, B en C. Er is geprobeerd een onbestaande pak aan te roepen.

242 PACK CHANGED
Tijdens een copieerfunctie is een pak verwisseld.

241 PACK NOT BLANK
Er is iets misgelopen tijdens het formatteren van een datapak. Niet alle gegevens zijn uitgewist. Herformatteer de pak.

240 UNKNOWN PACK
Er is een onbekende pak in de datapak drives gestopt. Onmiddellijk verwijderen.

239 PACK FULL
De pak, waarop de Organiser II probeert te schrijven, is reeds vol. Een andere gebruiken of herformatteren.

238 END OF FILE
Er is geprobeerd een record te lezen in een bestand, waarvan het einde reeds bereikt is.

237 BAD RECORD TYPE
Fout in een machinetaalprogramma.

236 BAD FILE NAME
De opgegeven bestandsnaam is niet correct. Die mag max. 8 tekens bevatten en moet beginnen met een letter. Er mogen ook geen tekens boven ASCII-code 122 in voorkomen (zie Appendix A).

235 FILE EXISTS
Er is geprobeerd een bestand te creëren of een programma te schrijven waarvan de naam al gebruikt is. Wis eerst de oude versie of gebruik een andere naam.

234 FILE NOT FOUND
Er is geprobeerd een bestand te OPENen dat niet bestaat.

233 DIRECTORY FULL
Er is geprobeerd een bestand te creëren terwijl er reeds 110 bestanden/programma's op de pak staan. 110 elementen is het maximum.

232 PACK NOT COPYABLE
Er is geprobeerd een pak te copiëren, dat hiertegen beschermd is. Vele commerciële programma's (zoals de Pocket Spreadsheet) kunnen aldus niet worden gecopiëerd.

231 BAD DEVICE CALL
Komt voor wanneer een machinetaalprogramma probeert een pak aan te roepen dat niet bestaat.

230 DEVICE MISSING
Er is geprobeerd een randapparaat, zoals een printer of een modem, te gebruiken terwijl dat randapparaat niet is aangekoppeld.

229 DEVICE LOAD ERR
Terwijl de Organiser II bezig was met 'verificatie' van een pak of randapparaat (correctheidsmaatregel) is er iets met dat pak/randapparaat gebeurd (bijv. verwijderd).

228 SYNTAX ERR
Tijdens vertaling (TRAN) van een procedure kwam de Organiser II een syntaxfout tegen. Kijk de correcte syntax van de functie/instructie na.

227 MISMATCHED ()'s
Er zijn niet evenveel linkse als rechtse haakjes in een expressie.

226 BAD FN ARGS
Er zijn teveel of te weinig parameters doorgegeven aan een functie (bijvoorbeeld 2 terwijl er maar 1 nodig is).

225 SUBSCRIPT ERR
Er is geprobeerd een onbestaand element aan te spreken in een tabelvariabele (bijvoorbeeld A%(20) terwijl deze tabel slechts 15 elementen heeft).

224 TYPE MISMATCH
Er is geprobeerd een verkeerd gegevenstype te gebruiken voor een bepaalde actie, bijvoorbeeld : SIN("ORG II").

223 NAME TOO LONG
Er is geprobeerd een bestandsnaam of variabelenaam te gebruiken die meer dan 8 tekens lang is.

222 BAD QUALIFIER
Een variabele wordt gebruikt met een verkeerde aanduiding van zijn type, bijvoorbeeld A%% of NAAM$$.

221 MISMATCHED "
Er moet steeds een even aantal aanhalingstekens zijn (openen/sluiten). Tel ze na.

220 STRING TOO LONG
Een string is langer dan de plaats die voor die string gereserveerd is (bij LOCAL/GLOBAL). Herzie de plaatsvoorziening of wijzig het programma.

219 BAD CHARACTER
Er is een ongeldig teken gebruikt. Kijk de context van de expressie na.

218 BAD NUMBER
Er is een onmogelijk nummer geproduceerd, bijvoorbeeld 1.2.3.

217 NO PROC NAME
Er is geprobeerd een procedure te gebruiken, die niet in de Organiser II gedefiniëerd is (bijvoorbeeld op een IBM PC) en die geen geldige procedurenaam heeft.

216 BAD DECLARATION
Een variabele is bij LOCAL/GLOBAL verkeerd aangegeven.

215 BAD ARRAY SIZE
Een tabelvariabele is bij LOCAL/GLOBAL aangegeven met een verkeerd aantal elementen, bijvoorbeeld LOCAL TAB$(0).

214 DUPLICATE NAME
Er is geprobeerd een variabele een naam te geven die reeds bestaat in de procedure, bijvoorbeeld LOCAL A%, A%.

213 STRUCTURE ERR
Er is een fout opgetreden in verband met de volgorde van IF/ENDIF, WHILE/ENDWH of DO/UNTIL.

212 TOO COMPLEX
Er is geprobeerd meer dan 8 IF/ENDIF, WHILE/ENDWH of DO/UNTILs in elkaar te gebruiken.

211 MISSING LABEL
Er is geprobeerd naar een onbestaand label te springen.

210 MISSING COMMA
Een lijst parameters mist ergens een komma (bijvoorbeeld OPEN "A:MAIN",A R$).

209 BAD LOGICAL NAME
Er is geprobeerd een andere logische naam te gebruiken dan A, B, C of D bij de opening/creatie van een bestand.

208 BAD ASSIGNMENT
Er is geprobeerd een waarde toe te kennen aan een variabele, die slechts diende als een parameter voor een procedure of functie. Dit is onmogelijk.

207 BAD FIELD LIST
Er is geprobeerd een bestand te openen/creëren met minder dan 1 of meer dan 16 velden.

206 ESCAPE
Het programma is onderbroken met ON-Q. Dit kan worden vermeden met ESCAPE OFF.

205 ARG COUNT ERR
Er zijn teveel/te weinig parameters bij een functie of procedure gebruikt. Kijk de desbetreffende functie/procedure na voor de juiste parameters.

204 MISSING EXTERNAL
Er is geprobeerd een onbestaande variabele aan te roepen (niet bij LOCAL/GLOBAL aangegeven en niet als parameter van een functie of procedure gebruikt). Kijk de spelling van de variabele na.

203 MISSING PROC
Er is geprobeerd een onbestaande procedure uit te voeren.

202 MENU TOO BIG
Er is geprobeerd een menu te creëren met een te lange lijst opties.

201 FIELD MISMATCH
Er is geprobeerd een onbestaand veld van een bestand aan te roepen.

200 READ PACK ERR
Er kan niet van een bepaalde pak gelezen worden. Waarschijnlijk is de pak beschadigd of moet geherformatteerd worden.

199 FILE IN USE
Er is geprobeerd een reeds geopend bestand te openen.

198 RECORD TOO BIG
De maximumlengte van 254 tekens per record is overschreden.

197 BAD PROC NAME
Bij het creëren van een procedure (NEW in het PROGmenu) is een onmogelijke naam opgegeven.

196 FILE NOT OPEN
Er is geprobeerd een nog niet geopend bestand te lezen.

195 INTEGER OVERFLOW
Er is geprobeerd een integere bewerking te doen buiten de grenzen -32768 tot +32767.

194 BATTERY TOO LOW
De batterij is bijna leeg : vervang zo snel mogelijk.

193 DEVICE WRITE FAIL
De pak waarnaar wordt geprobeerd te Saven is beschadigd, niet goed geformatteerd of beschermd tegen beschrijving. Deze fout kan ook voorkomen wanneer in Utils een datapak wordt "geformatteerd" i.p.v. een RAMpak, wat niet mogelijk is. Ook een COMMS LINK die niet in orde is (kijk verbindingskabels en seinprotocol na) kan deze foutmelding opleveren.

192 DEVICE READ FAIL
Het pak waarvan wordt gelezen is waarschijnlijk beschadigd, niet goed geformatteerd of beschermd tegen vermenigvuldiging. Ook een COMMS LINK die niet in orde is (kijk verbindingskabels en seinprotocol na) kan deze foutmelding opleveren.

190 BAD PARAMETER
(Alleen met de COMMS LINK) De bestandsnaam voor de PC is foutief.

189 FILE NOT FOUND
(Alleen met de COMMS LINK) De procedure XTRECV heeft geprobeerd het bestand op de PC te openen maar het bestaat niet of het is van het verkeerde type.

188 SERVER ERROR
(Alleen met de COMMS LINK) Er is iets fout met de PC.

187 FILE ALREADY EXISTS
(Alleen met de COMMS LINK) Het bestand bestaat reeds.

186 DISK FULL
(Alleen met de COMMS LINK) XTSEND probeerde naar een volle schijf te schrijven.

185 RECORD TOO LONG
(Alleen met de COMMS LINK) Het record, dat XTRECV probeerde in te lezen, is langer dan 255 tekens en dit is niet toegestaan.

Appendix C

Foutmeldingen (alfabetisch)

ARG COUNT ERR 205
Er zijn teveel/te weinig parameters bij een functie of procedure gebruikt. Kijk de desbetreffende functie/procedure na voor de juiste parameters.

BAD ARRAY SIZE 215
Een tabelvariabele is bij LOCAL/GLOBAL aangegeven met een verkeerd aantal elementen, bijvoorbeeld LOCAL TAB$(0).

BAD ASSIGNMENT 208
Er is geprobeerd een waarde toe te kennen aan een variabele, die slechts diende als een parameter voor een procedure of functie. Dit is onmogelijk.

BAD CHARACTER 219
Er is een ongeldig teken gebruikt. Kijk de context van de expressie na.

BAD DECLARATION 216
Een variabele is bij LOCAL/GLOBAL verkeerd aangegeven.

BAD DEVICE CALL 231
Komt voor wanneer een machinetaalprogramma probeert een pak aan te roepen dat niet bestaat.

BAD DEVICE NAME 243
De enige bestaande paknamen zijn A, B en C. Er is geprobeerd een onbestaande pak aan te roepen.

BAD FIELD LIST 207
Er is geprobeerd een bestand te openen/creëren met minder dan 1 of meer dan 16 velden.

BAD FILE NAME 236
De opgegeven bestandsnaam is niet correct. Die mag max. 8 tekens bevatten en moet beginnen met een letter. Er mogen ook geen tekens boven ASCII-code 122 in voorkomen (zie Appendix A).

BAD FN ARGS 226
Er zijn teveel of te weinig parameters doorgegeven aan een functie (bijvoorbeeld 2 terwijl er maar 1 nodig is).

BAD LOGICAL NAME 209
Er is geprobeerd een andere logische naam te gebruiken dan A, B, C of D bij de opening/creatie van een bestand.

BAD NUMBER 218
Er is een onmogelijk nummer geproduceerd, bijvoorbeeld 1.2.3.

BAD PARAMETER 190
(Alleen met de COMMS LINK) De bestandsnaam voor de PC is foutief.

BAD PROC NAME 197
Bij het creëren van een procedure (NEW in het PROGmenu) is een onmogelijke naam opgegeven.

BAD QUALIFIER 222
Een variabele wordt gebruikt met een verkeerde aanduiding van zijn type, bijvoorbeeld A%% of NAAM$$.

BAD RECORD TYPE 237
Fout in een machinetaalprogramma.

BATTERY TOO LOW 194
De batterij is bijna leeg : vervang zo snel mogelijk.

DEVICE LOAD ERR 229
Terwijl de Organiser II bezig was met 'verificatie' van een pak of randapparaat (correctheidsmaatregel) is er iets met dat pak/randapparaat gebeurd (bijv. verwijderd).

DEVICE MISSING 230
Er is geprobeerd een randapparaat, zoals een printer of een modem, te gebruiken terwijl dat randapparaat niet is aangekoppeld.

DIRECTORY FULL 233
Er is geprobeerd een bestand te creëren terwijl er reeds 110 bestanden/programma's op de pak staan. 110 elementen is het maximum.

DISK FULL 186
(Alleen met de COMMS LINK) XTSEND probeerde naar een volle schijf te schrijven.

DIVIDE BY ZERO 251
Er is geprobeerd te delen door nul.

DEVICE READ FAIL 192
Het pak waarvan wordt gelezen is waarschijnlijk beschadigd, niet goed ge-

formatteerd of beschermd tegen vermenigvuldiging. Ook een COMMS LINK die niet in orde is (kijk verbindingskabels en seinprotocol na) kan deze foutmelding opleveren.

DEVICE WRITE FAIL 193
De pak waarnaar wordt geprobeerd te Saven is beschadigd, niet goed geformatteerd of beschermd tegen beschrijving. Deze fout kan ook voorkomen wanneer in Utils een datapak wordt "geformatteerd" i.p.v. een RAMpak, wat niet mogelijk is. Ook een COMMS LINK die niet in orde is (kijk verbindingskabels en seinprotocol na) kan deze foutmelding opleveren/

DUPLICATE NAME 214
Er is geprobeerd een variabele een naam te geven die reeds bestaat in het procedure, bijvoorbeeld LOCAL A%, A%.

END OF FILE 238
Er is geprobeerd een record te lezen in een bestand, waarvan het einde reeds bereikt is.

ESCAPE 206
Het programma is onderbroken met ON-Q. Dit kan worden vermeden met ESCAPE OFF.

EXPONENT RANGE 253
Een getal is buiten de exponentbegrenzing gegaan. Deze moet liggen tussen −99 en +99.

FIELD MISMATCH 201
Er is geprobeerd een niet-bestaand veld van een bestand aan te roepen.

FILE ALREADY EXISTS 187
(Alleen met de COMMS LINK) Het bestand bestaat reeds.

FILE EXISTS 235
Er is geprobeerd een bestand te creëren of een programma te schrijven waarvan de naam al gebruikt is. Veeg eerst de oude versie af of gebruik een andere naam.

FILE IN USE 199
Er is geprobeerd een reeds geopend bestand te openen.

FILE NOT FOUND 234
Er is geprobeerd een bestand te OPENen dat niet bestaat.

FILE NOT FOUND 189
(Alleen met de COMMS LINK) De procedure XTRECV heeft geprobeerd het bestand op de PC te openen maar het bestaat niet of het is van het verkeerde type.

FILE NOT OPEN 196
Er is geprobeerd een nog niet geopend bestand te lezen.

FN ARGUMENT ERR 247
Tijdens het aanroepen van een functie zijn verkeerde parameters gebruikt (bijv. te weinig parameters, een string waar een integer moest gebruikt worden, enz.).

INTEGER OVERFLOW 195
Er is geprobeerd een integere bewerking te doen buiten de grenzen -32768 tot +32767.

MENU TOO BIG 202
Er is geprobeerd een menu te creëren met een te lange lijst opties.

MISMATCHED " 221
Er moet steeds een even aantal aanhalingstekens zijn (openen/sluiten). Tel ze na.

MISMATCHED ()'s 227
Er zijn niet evenveel linkse als rechtse haakjes in een expressie.

MISSING COMMA 210
Een lijst parameters mist ergens een komma (bijvoorbeeld OPEN "A:MAIN", A R$).

MISSING EXTERNAL 204
Er is geprobeerd een onbestaande variabele aan te roepen (niet bij LOCAL/GLOBAL aangegeven en niet als parameter van een functie of procedure gebruikt). Kijk de spelling van de variabele na.

MISSING LABEL 211
Er is geprobeerd naar een onbestaand label te springen.

MISSING PROC 203
Er is geprobeerd een onbestaande procedure uit te voeren.

NAME TOO LONG 223
Er is geprobeerd een bestandsnaam of variabelenaam te gebruiken die meer dan 8 tekens lang is.

NO ALLOC CELLS 255
Een machinetaalprogramma dat wordt uitgevoerd, probeert interne ruimte aan te roepen die niet bestaat.

NO PACK 246
De Organiser II probeert een pak te gebruiken dat er niet is, of waarvan niet gelezen kan worden. In het laatste geval is dit pak beschadigd.

NO PROC NAME 217
Er is geprobeerd een procedure te gebruiken, die niet in de Organiser II gedefiniëerd is (bijvoorbeeld op een IBM PC) en die geen geldige procedurenaam heeft.

NUM TO STR ERR 250
Komt voor wanneer men geprobeerd heeft met de CALCulator een getal om te zetten naar een string (CALC kan niet werken met strings) of tijdens een foutieve berekening in een machinetaalprogramma.

OUT OF MEMORY 254
Het interne geheugen van de Organiser II is helemaal opgebruikt.

PACK CHANGED 242
Tijdens een copieerfunctie is een pak verwisseld.

PACK FULL 239
De pak, waarop de Organiser II probeert te schrijven, is reeds vol. Een andere gebruiken of herformatteren.

PACK NOT BLANK 241
Er is iets misgelopen tijdens het formatteren van een datapak. Niet alle gegevens zijn uitgewist. Herformatteer het pak.

PACK NOT COPYABLE 232
Er is geprobeerd een pak te copiëren, dat hiertegen beschermd is. Vele commerciële programma's (zoals de Pocket Spreadsheet) kunnen aldus niet worden gecopiëerd.

READ ONLY PACK 244
Er zit een oude datapak in de Organiser II (een zwarte pak van de eerste Organiser). Het is niet mogelijk deze te beschrijven, zelfs niet na formatteren.

READ PACK ERR 200
Er kan niet van een bepaalde pak gelezen worden. Waarschijnlijk is de pak beschadigd of moet geherformatteerd worden.

RECORD TOO BIG 198
De maximumlengte van 255 tekens per record is overschreden.

RECORD TOO LONG 185
(Alleen met de COMMS LINK) Het record, dat XTRECV probeerde in te lezen, is langer dan 255 tekens en dit is niet toegestaan.

SERVER ERROR 188
(Alleen met de COMMS LINK) Er is iets fout met de PC.

STACK OVERFLOW 249
Een machinetaalprogramma vernietigt de machinestack (zie Appendix A, Hoofdstuk 34).

STACK UNDERFLOW 248
Een machinetaalprogramma vernietigt de machinestack (zie Appendix A, Hoofdstuk 34).

STR TO NUM ERR 252
Er is geprobeerd een string naar een getal om te zetten (met VAL) terwijl die string helemaal geen getal bevatte.

STRING TOO LONG 220
Een string is langer dan de plaats die voor die string gereserveerd is (bij LOCAL/GLOBAL). Herzie de plaatsvoorziening of wijzig het programma.

STRUCTURE ERR 213
Er is een fout opgetreden in verband met de volgorde van IF/ENDIF, WHILE/ENDWH of DO/UNTIL.

SUBSCRIPT ERR 225
Er is geprobeerd een onbestaand element aan te spreken in een tabelvariabele (bijvoorbeeld A%(20) terwijl deze tabel slechts 15 elementen heeft).

SYNTAX ERR 228
Tijdens vertaling (TRAN) van een procedure kwam de Organiser II een syntaxfout tegen. Kijk de correcte syntax van de functie/instructie na.

TOO COMPLEX 212
Er is geprobeerd meer dan 8 IF/ENDIF, WHILE/ENDWH of DO/UNTILs in elkaar te gebruiken.

TYPE MISMATCH 224
Er is geprobeerd een verkeerd gegevenstype te gebruiken voor een bepaalde actie, bijvoorbeeld : SIN("ORG II").

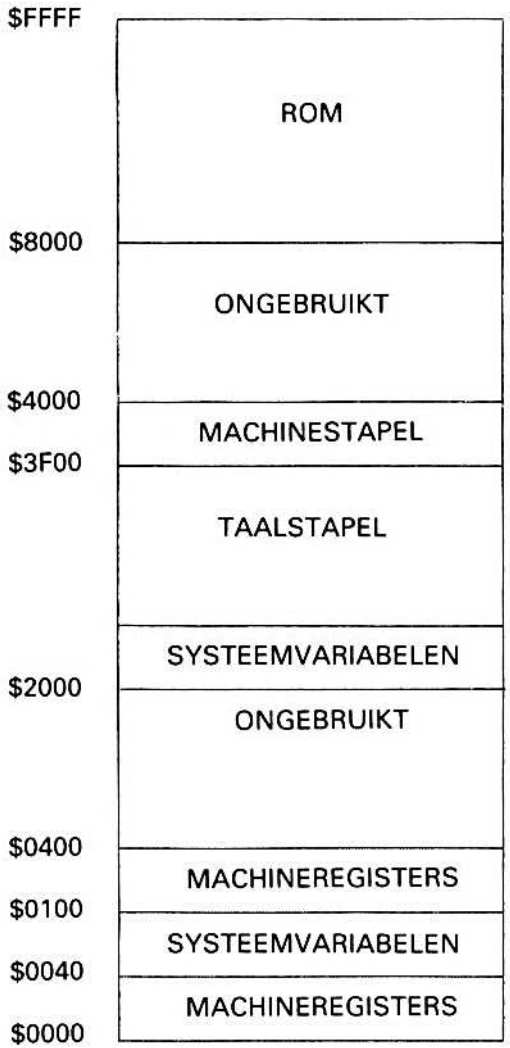
UNKNOWN PACK 240
Er is een onbekende pak in de datapak drives gestopt. Onmiddellijk verwijderen.

WRITE PACK ERR 245
Het is niet gelukt naar een bepaalde pak te schrijven. Misschien is de pak vol of beschadigd.

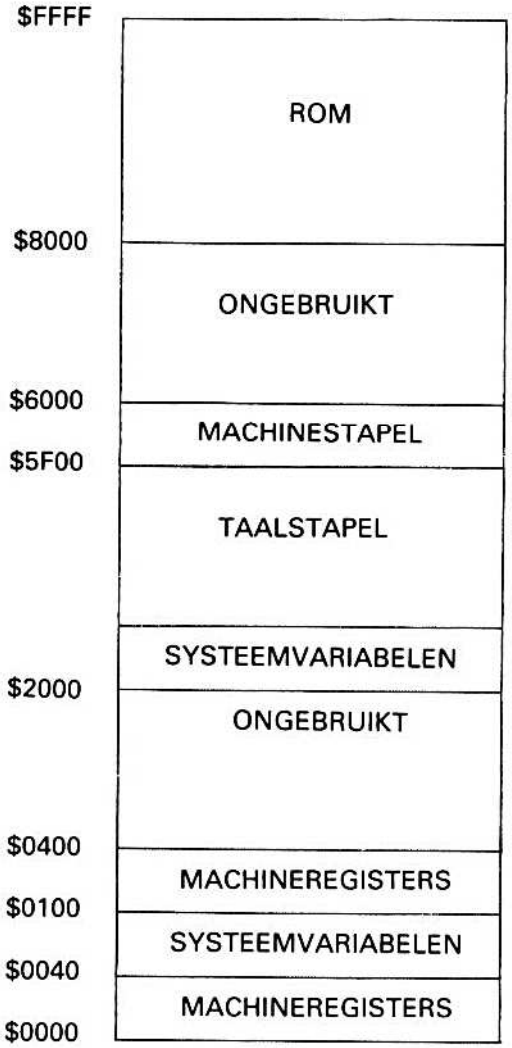
Appendix D
Geheugenkaarten

Aan de linkerkant van de beide histogrammen staat het adres in hexadecimaal aangegeven.

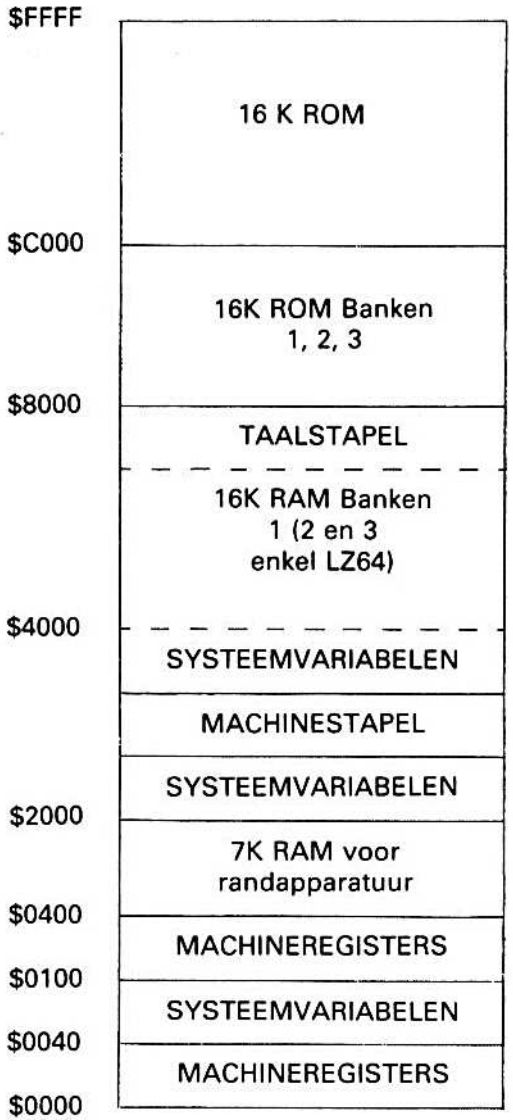
Geheugenkaart Organiser II CM



Geheugenkaart Organiser II XP



Geheugenkaart Organiser II LZ



Appendix E
Woordenlijst

adres: bepaalde plaats in het geheugen. Adressen zijn steeds genummerd (eerste adres : 0). Als het geheugen dus bijvoorbeeld 65536 bytes beslaat, zijn de adressen genummerd van 0 tot 65535.

alfanumeriek: string : bestaande uit tekens. Tekens kunnen zijn : letters, leestekens, cijfers, ... Zie Appendix A.

alkaline: -batterij : batterij op basis van basen. Deze batterij heeft een langere gemiddelde levensduur dan andere batterijen. Sterk aangeraden voor gebruik bij de Organiser II.

argument: zie *parameter*

array: tabel. Ook wel *linked list* (kettinglijst) genoemd.

ASCII: American Standard Code for Information Interchange. Een speciale, vastgelegde code voor gegevensopslag. ASCII-code voorziet een codering voor 128 tekens, genummerd 0 tot 127. Ieder teken heeft dus zijn eigen code. De meeste computers hebben ook nog tekens vanaf codenummer 128 tot 255. In dit geval spreekt men van *extended ASCII* (uitgebreide ASCII). Ook de Organiser II beschikt over extended ASCII (zie Appendix A).

bar-code/streepjescode: een speciale code ontwikkeld door de Amerikaanse firma IBM, die bestaat uit een aantal gegroepeerde vertikale strepen, die samen een getal vormen. De code wordt in de computer 'gelezen' d.m.v. een speciale pen, de zgn. '*bar code reader*'. Het uiteindelijke getal komt dan in de computer terecht. Meestal bestaan die getallen uit enkele reeksen afzonderlijke getallen, zoals bij de meestvoorkomende internationale streepjescode. Daar stellen de eerste twee cijfers steeds het land van herkomst van het produkt voor (alle Belgische produkten beginnen steeds met 54, bijvoorbeeld).

bar-code reader: speciale pen om *bar-code* te 'lezen' in de computer.

bestand: een reeks bij elkaar horende *records*, vaak in een vaste volgorde. Bestanden worden gestockeerd op *datapaks*, **RAMpaks:** of in het interne geheugen.

besturingssysteem: programma dat de computer bestuurt. Soms (niet bij de

Organiser II) staat het besturingssysteem van een computer niet op een ROM, maar moet het van schijf worden gelezen. In dat geval spreekt men van een *disk operating system*, een besturingssysteem dat van schijf komt. Dit heeft als voordeel dat één type computer verschillende besturingssystemen kan hebben. Op de IBM PC, bijvoorbeeld, kunnen zowel de systemen MS-DOS en PC-DOS als XENIX worden 'gedraaid'.

binair: tweetallig. Een binair getal is dus een getal in het tweetallig stelsel, bestaande uit de cijfers 0 en 1.

bit: kleinste eenheid gegeven : 0 of 1. *Bit* is de samentrekking van *BI*nary digi*T*, ofwel binair cijfer. Bits worden meestal gegroepeerd per vier (*nibble*) of per acht (*byte*).

bronpak: pak waarvan gegevens worden gelezen. Zie ook *doelpak*.

bug: programmeerfout : Bugs (Engels voor *luizen*) zijn eigenlijk denkfouten. Het is vaak heel moeilijk bugs uit programma's te halen. Vrijwel *ieder* softwareprodukt bevat bugs, zelfs wanneer de programmeurs één of misschien zelfs twee jaar gewerkt hebben om alle fouten te verwijderen. Zie ook *debuggen*.

byte: groepering van 8 bits. Het geheugen wordt meestal ingedeeld in bytes.

C-programmeertaal. C is een zeer complexe, 'middelhoge' programmeertaal die vaak wordt gebruikt om besturingssystemen te schrijven. Het systeem UNIX, dat wordt gebruikt in grote computers, is geschreven in C. De beroemde PC-programma's *Lotus 1-2-3* en **dBASE:** *III* zijn beiden geschreven in C.

comms link: verbinding tussen de Organiser II en randapparatuur. Zie hoofdstuk 39.

compiler: programma dat een listing van een programma geschreven in de ene taal, omzet naar een andere taal. Meestal is de doeltaal *machinetaal*, een taal die moeilijk te leren maar zeer snel is. De brontaal is meestal een hogere programmeertaal, die veel eenvoudiger aan te leren is en waarmee men veel vlotter werkt. Ook de Organiser II bevat een compiler (*TRAN*) die OPL omzet in *q-code*, een soort tussentaal. Die tussentaal wordt dan door het *besturingssysteem* uitgevoerd.

daisy wheel printer: zie *margrietwielprinter*.

datapak: externe gegevensopslag voor de Organiser en Organiser II handcomputers. Datapaks bevatten een *EPROM* die de gegevens bijhoudt. Da-

tapaks hebben capaciteiten van 16, 32, 64 en 128K. De laatste werkt alleen bij model XP.

dBASE: populair gegevenverwerkingsprogramma op Apple en IBM-computers. De huidige versie (enkel IBM), *dBASE III Plus*, bevat een zeer uitgebreide programmeertaal, specifiek voor gegevensopslag en -verwerking. De OPL- instructie *USE* is afgeleid van de dBASE-instructie met dezelfde naam. dBASE wordt gemaakt door de Amerikaanse firma Ashton-Tate.

debuggen: de fouten (*bugs*) uit een programma halen. Voor de meeste computers en talen zijn hiervoor speciale *debuggers* in de handel; programma's waarmee het debuggen wordt vereenvoudigd. De Organiser Developer, een softwarepakket dat dient om het schrijven van Organiser II-programma's op de PC mogelijk te maken, is ook uitgerust met een debugger.

disk: schijf. Op een disk (Engelse spelling : *disc*), die magnetische sporen bevat (net als magneetband), kunnen gegevens worden opgeslagen. Een disk is ingedeeld in *sporen*, die zijn ingedeeld in *sectoren*. De gegevens worden per sector geschreven of gelezen. Disks zijn niet van toepassing bij de Organiser II.

doelpak: pak waarnaar geschreven wordt. Zie ook *bronpak*.

dot-matrix printer: een *printer* die zijn tekens vormt door *puntjes* af te drukken. Die puntjes worden gevormd door een soort rooster, de *matrix* genaamd, die tegen een lint aandrukt. Dit lint brengt de letters op het papier.

EPROM: Eraseable, Programmable ROM : een chipje voor geheugenopslag dat niet alleen programmeerbaar is (er kunnen gegevens op worden gezet), maar ook uitwisbaar en dat functioneert als een ROMchip. EPROMs worden gebruikt in datapaks. Zij kunnen enkel hun inhoud verliezen wanneer ze langdurig aan ultraviolet licht worden blootgesteld (zoals met de Psion Formatter). De gegevens worden in de chip 'gebrand'.

error code: foutcode : een codering die iedere fout die zich kan voordoen een nummer toekent. Wanneer een bepaalde fout zich dan voordoet, wordt door het besturingssysteem het desbetreffende nummer gebruikt om aan de gebruiker de fout mee te delen. Zie Appendices B en C.

extern geheugen: geheugenopslag buiten de computer. Bij de Organiser II wordt het extern geheugen gevormd door data- en RAMpaks. Bij grotere computers zijn er banden en schijven. Het maximum extern geheugen bij de Organiser II model CM is 128K, bij model XP 256K. Enkel pak B: en

C: bevatten extern geheugen, pak A: is intern geheugen.

field: zie *veld*.

file: zie *bestand*.

floppy disk: zachte schijf : schijf gemaakt van polyvinyl, meestal zwart, die magnetische sporen bevat voor gegevensopslag. Floppy disks, die vierkant zijn, hebben een lengte/breedte van 5.25 of 8 inch. Ze worden tegenwoordig meer en meer verdrongen door de zgn. '*mini disks*', kleine schijfjes van 3.5 inch, die een hogere capaciteit hebben en veiliger in het gebruik zijn (ze worden niet zo makkelijk uitgewist). Zie *disk*.

foutmelding: melding van de computer dat er ergens iets is misgelopen. Zie *error code*.

functie: procedure die een eindwaarde berekent en die dan doorgeeft. Functies worden steeds opgeroepen vanuit procedures (numerieke functies kunnen ook vanuit *CALC* worden opgeroepen). OPL bevat een aantal voorgedefiniëerde functies en biedt ook de mogelijkheid er nieuwe bij te maken.

gegeven: Element waaruit informatie is opgebouwd. Gegevens hebben op zich nooit een betekenis.

globale variabele: variabele die in meer dan één procedure kan worden gebruikt, en overal dezelfde waarde heeft.

hard disk: harde magneetschijf; deze zitten meestal in de computer en kunnen niet verwijderd worden. Harde schijven hebben een zeer korte toegangstijd en hebben capaciteiten van 10M (+10 000 K), 20M, 40M, 80M en meer. Harde schijven kunnen met de Organiser II worden gebruikt via een COMMS LINK en een IBM PC. Zie *disk*, *floppy disk*.

harde schijf: zie *hard disk*.

HD6303X: de *microprocessor* die in de Organiser II zit. Hij wordt geleverd door de Japanse firma Hitachi.

heap: zie *stapel*.

hex: hexadecimaal : zestientallig stelsel. In tegenstelling tot ons dagelijkse decimale (tientallige) stelsel, heeft dit stelsel 16 cijfers : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E en F.

hexadecimaal: zie *hex*.

informatie: één of meer met elkaar verband houdende gegevens die een betekenis hebben gekregen. Een computer kan dus geen informatie opslaan, enkel gegevens.

initialiseren: een eerste (initiële) waarde toekennen.

integer: geheel getal, liggend tussen −32768 en +32767.

intern geheugen: RAMgeheugen dat in de computer zit. Intern geheugen kan steeds direct (zonder wachttijd) worden benut. Het verliest zijn inhoud zodra de stroomtoevoer onderbroken wordt.

K: Kilobyte : 1024 bytes.

KB: zie *K*.

kilobyte: zie *K*.

laserprinter: speciale printer die, met behulp van een laserstraal, tekens en tekeningen kan afdrukken op papier. Laserprinters hebben een zeer hoge resolutie en werken bovendien erg snel. Zie *printer*.

LCD: Liquid Crystal Display : een scherm, waar de puntjes worden gevormd door vloeibare kristallen. De Organiser II heeft een LCD-scherm, net als vele digitale horloges.

lokale variabele: een variabele die slechts in één functie of procedure dezelfde waarde heeft.

M: Megabyte : 1024K.

machinestack: speciale *stack* die door de *microprocessor* wordt gebruikt.

margrietwielprinter: printer die met behulp van een speciaal roterend 'margrietwiel', dat op de bloem met dezelfde naam lijkt, tekens afdrukt op papier. Het wiel bevat de tekenset twee keer en roteert steeds naar de plaats waar het volgende teken staat, dat moet worden afgedrukt. Dat teken wordt dan met een speciaal hamertje tegen het lint, en zo tegen het papier aangedrukt. Margrietwielprinters zijn niet snel, maar leveren kwaliteitswerk af (vergelijkbaar met een elektrische schrijfmachine). Zie *printer*.

MB: zie *M*.

megabyte zie *M*.

microprocessor: elektronisch toestel, dat ontworpen is om een computer te

sturen. De microprocessor wordt geprogrammeerd : hij kent een aantal basisinstructies, de zgn. *machinetaal*, en bevat *registers* (een soort variabelen). Microprocessors kunnen werken met 8, 16 of 32 bits tegelijk.

modem: samentrekking van *moduleren/demoduleren*, een toestel dat gebruikt wordt om over een telefoonlijn gegevens door te sturen. Modems worden aan computers verbonden met een zgn. *seriële interface, die de gegevens bit per bit doorseint. Er zijn digitale* modems, die werkelijk elektronische seinen geven, en de minder betrouwbare *akoestische* modems, die geluidsseinen geven.

MODULA-2: modulaire taal, ontworpen in de late zeventiger jaren door de Zwitserse professor *Niklaus Wirth*. De taal is de opvolger van *Pascal*, waarvan ze slechts in enkele details verschilt.

modulair: opgebouwd uit *modules*.

module: geïsoleerde procedure, of reeks procedures, die vaak in verschillende programma's kunnen worden gebruikt zonder wijziging. Grote programma's zijn vaak opgebouwd uit modules omdat programmawijzigingen zo kunnen worden teruggebracht tot geïsoleerde programmagedeelten, wat het 'programmaonderhoud' vergemakkelijkt.

MS-DOS: Microsoft Disk Operating System, een besturingssysteem voor computers met een Intel 8088-microprocessor (of aanverwanten als de 8086, 80186, 80286 en 80386). MS-DOS wordt gebruikt op IBM PC's en computers die van deze machine zijn afgeleid ('klonen'). De huidige versie, 3.3, kan dubbelzijdige disk drives aan, hard disks tot 32M, en kan netwerken sturen. Hoe meer MS-DOS ontwikkeld wordt, hoe minder het gaat lijken op het systeem waarop het oorspronkelijk gebaseerd was (CP/M) en hoe meer het op grote neef UNIX gaat lijken. Zie ook *besturingssysteem* en *PC-DOS*.

numeriek: bestaande uit cijfers.

octaal: achttallig. Een talstelsel met basis 8, er zijn dus slechts 8 cijfers : 0, 1, 2, 3, 4, 5, 6 en 7. Zie ook *hex* en *binair*.

operating system: zie *besturingssysteem*.

OPL: Organiser Programming Language : de programmeertaal van de Psion Organiser II.

parallel: met 7 of 8 bits tegelijk. Tegengestelde van *seriëel*.

parameter: een waarde die aan een bepaalde functie wordt medegedeeld.

De functie gebruikt dan de parameter om een nieuwe waarde te berekenen. De sinusfunctie bijvoorbeeld heeft 1 parameter : de parameter van `sin(9)` is 9, het resultaat, dat door de functie `sin` wordt berekend, is 0.412118485241. Sommige functies hebben meer parameters (bijv. `left$`).

Pascal: veelal academische programmeertaal ontworpen in de zestiger jaren door professor *Niklaus Wirth*. *Pascal* is gestruktureerd en zeer krachtig. Zie *MODULA-2*.

PC-DOS: speciale versie van *MS-DOS*, voor de IBM PC en echte klonen. *PC-DOS* wordt uitgebracht door IBM, maar gemaakt door Microsoft, die ook *MS-DOS* maakt. Eigenlijk zijn beide systemen precies hetzelfde. Zie *MS-DOS*.

PP3: type batterijen voor gebruik met de Organiser II.

printer: afdrukeenheid : toestel dat wordt gebruikt om gegevens op papier af te drukken. Er zijn verschillende soorten printers, die ieder een totaal ander principe hebben. De meest bekende zijn de *dot-matrix* printers, de *margrietwiel*printers en de *laser*printers.

procedure: programma geschreven in OPL. Procedures kunnen alleenstaand zijn of worden gebruikt in andere procedures. In het laatste geval spreekt men ook van *module*. Zie *module*.

processor: zie *microprocessor*.

programma: reeks bij elkaar horende instructies in een vaste volgorde, die tot doel hebben een zekere taak uit te voeren. Zie *procedure*.

Q-code: een tussencode die door *TRAN* wordt gegenereerd en afgeleid is van OPL. *Q-code* kan echter vrij snel door het besturingssysteem worden uitgevoerd, terwijl dat met OPL niet of nauwelijks mogelijk zou zijn.

RAM: Random Access Memory; vrij toegankelijk geheugen. Geheugen dat vrij kan worden gebruikt, waarvan de inhoud constant mag veranderen. Het interne geheugen van alle computers bestaat uit RAM.

RAMpak: externe gegevensopslag van de Organiser II die bestaat uit RAM. Er zit een klein batterijtje bij, dat de RAMchip 'voedt'. Dit batterijtje gaat 5 jaar mee. De capaciteit van RAMpaks bedraagt 32K. Zie *RAM*.

real: zie *reëel*.

record: bestaat uit bij elkaar horende *velden* in een vaste volgorde, die ieder een zekere, meestal vastgelegde inhoud hebben. Zie *veld*, *bestand*.

reëel: decimaal, tussen de begrenzing 1e-100 en (1e+100)-1.

ROM: Read-Only Memory, geheugen waarvan alleen kan worden gelezen. Het besturingssysteem van de Organiser II staat op een ROMchip, het mag immers niet worden gewijzigd! Ook compact discs en grammofoonplaten kunnen worden gezien als ROMs : er kan wel van worden gelezen, maar er kan niet naar worden geschreven.

RS-232: speciale seriële verbinding. De COMMS LINK van de Organiser II is uitgerust met een RS-232C -verbinding.

schijf: zie *disk*.

seriëel: systeem waarbij de gegevens bit per bit worden overgeseind. De COMMS LINK van de Organiser II bevat een seriële verbinding. Zie *parallel*, *RS-232*.

stack: zie *stapel*.

stapel: speciale geheugenruimte gereserveerd voor tijdelijke opslag van gegevens. Er zijn *machinestapels* en *taalstapels*. Het werkt volgens het LIFO-principe (last-in first-out) : het gegeven dat het laatst op de stapel is 'gelegd', is ook het eerste dat er weer wordt afgenomen. Stapels worden gebruikt om verschillende processen goed te laten verlopen.

string: alfanumeriek gegeven.

systeemvariabele: soort 'variabele' : een adres dat een bepaalde waarde bevat die door het besturingssysteem wordt gebruikt. Zie hoofdstuk 34.

taalstack: taalstapel.

tabelvariabele: variabele die een hele tabel gegevens bevat.

translate: vertalen van *OPL* naar een tussencode. Zie *compiler*, *q- code*.

udg: zie *user-defined graphic*.

user-defined graphic: teken dat door de gebruiker zelf kan worden gedefiniëerd. Bij de Organiser II bestaan user-defined graphics uit 7 bytes, waarvan steeds enkel de eerste 5 bits worden gebruikt. Telkens wanneer de Organiser II wordt gebruikt, moeten de udg's (max. 8) opnieuw worden gedefiniëerd.

variabele: veranderlijk gegeven. Variabelen hebben een naam van maximaal 8 tekens die moet beginnen met een letter en verder letters en cijfers mag bevatten.

veld: indeling van een record. Een veld kan bijvoorbeeld zijn : naam, adres, btw-nummer, leveringsadres, ... Zie *record*, *bestand*.

winchester: Zie *harde schijf*.

WERKEN MET DE ORGANISER II

Tweede, herziene druk

Inclusief het nieuwe model LZ

De Organiser II is een revolutie in de computerwereld : de eerste volwaardige handcomputer! Elk van de drie modellen — CM, XP en LZ — is uitgerust met ingebouwde gegevensverwerking, wetenschappelijke rekenmachine, 8 wekkers, agenda met alarm (tot en met 31 december 1999 voor CM en XP, 31 december 2155 voor LZ) en de krachtige programmeertaal OPL.

Het model LZ is standaard meertalig en kent elke belangrijke stad in de wereld, weet hoe laat het daar is en hoe die stad telefonisch kan bereikt worden. Daarenboven maakt de LZ het mogelijk beschermde bestanden te creëren e.d.m.

Met de vele verkrijgbare softwarepakketten en randapparaten mag de Organiser II gerust tot de 'grote jongens' gerekend worden. Hoe dit alles het beste kan worden gebruikt leest u in 'Werken met de Organiser II' : de eerste nederlandstalige handleiding van dit draagbare wonder. Deze tweede, herziene druk is volledig up-to-date, met volledige documentatie van de nieuwste ontwikkelingen. Geschreven voor iedereen die in de Organiser II is geïnteresseerd en er méér over wil weten!

ISBN 90 6215 227 9
D/1989/1997/6
Dictus, ORGANISER II