

P2000

P2000

P2000

P2000

P2000

P2000

P2000

P2000

P2000

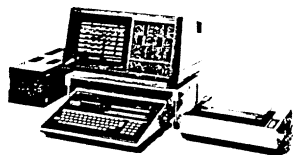
P2000

P2000

P2000

P2000

P2000



6

P2000

P2000

P2000

P2000

P2000

P2000

P2000

P2000

P2000

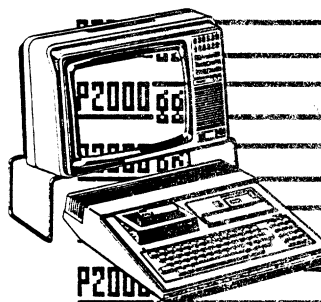
P2000

P2000

P2000

P2000

2



4

GRAFISCHE BEELDSCHERM EDITOR

PRINTERENUM VOOR DE PK-80

PACIFIC

7

P2000

P2000

P2000

P2000

P2000

P2000

P2000

P2000

P2000

P2000

P2000

P2000

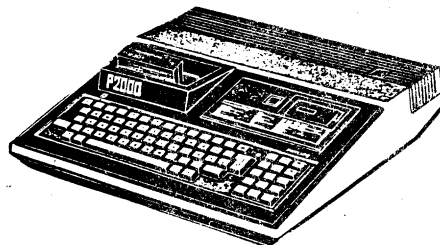
P2000

SAMENVATTING 5 NIEUWSBRIEVEN



P2000

Jaargang 1 no 5



P2000

P2000

P2000

P2000

P2000

P2000

3

TECHNISCHE INFORMATIE OVER HET PK-MODEL

BEHEERPLAATSEN VAN DE SYSTEMSTACK

BASICODE

HYPOTHECAIRE LENING

VISDATA PRINTPROGRAMMA

INSTRUCTIES

SHELLS BEELDSCHERMROUTING

TEKENPROGRAMMA

BANKBEPROGRAMMA

ALLEEN
INRIJK
1982
S 208
ENRUS
SHELLS
TIPS
STAMP
ESSEN

Titel	Type informatie	pag.
<u>Algemeen</u>		
Inhoud		1
Bij deze samenvatting	Redactioneel	3
Inleiding	Redactioneel	4
<u>Nieuwsbrief 1</u>		
Praktische tips	vensterkaartjes	5
Technische informatie over het M-Model	systeem-informatie	6
BASICODE	audiorecorder	8
Hypothecaire lening	BASIC-programma	10
<u>Nieuwsbrief 2</u>		
TV-aansluiting via antennekabel	hardware-tip	11
DEF FN-functie	programmeertip	11
VOLORG	praktische tip	11
RESET tijdens gebruik cassette	praktische tip	11
Beeldschermroutines BASIC UK	systeem-software	12
Conversie van cassette-BASIC naar disk-BASIC	BASIC-programma	15
Demonstratieprogramma routines	systeem-software	16
String arrays naar cassette schrijven	BASIC-programma	18
Printer-menu voor de MX-80	BASIC-programma	19
<u>Nieuwsbrief 3</u>		
Eisen waaraan een programma moet voldoen	programmeertip	20
PRINT CHR\$(5)	programmeertip	21
CLEAR	programmeertip	21
GET-routine voor getallen	programmeertip	21
Graphics ontwerpen	praktische tip	22
Inleiding programmeren in machinetaal	artikel	23
Adressen BASIC-instructies	systeem-software	26
Invoerroutines	(disk-)BASIC-programma	27
Vidgraf	(M-model-)BASIC-programma	28
Snelle beeldschermroutine	hulpprogramma	28
Tekenprogramma	programmabeschrijving	29
Bankgiroprogramma	programmabeschrijving	32
<u>Nieuwsbrief 4</u>		
Viewdata met de P2000	artikel	33
Viewdata-mogelijkheden met de P2000	gebruiks-informatie	34
Viewdata-gebruikerservaringen	artikel	35
Niet-gedocumenteerde toets in 24K BASIC	praktische tip	37
RESET in programma	praktische tip	38
GET-routine in machinetaal	praktische tip	38
Printer-baud	systeem-informatie	39
Programma redden na CLOS ipv CSAVE	praktische tip	39
Praktische tip voor te hete voedingen	hardwaretip	40
Verbergen van programmaregels	programmeertip	41
Random-generator	programmeertip	41
Zo werkt BASIC	systeem-informatie	42
DATA-regels genereren	BASIC-programma	43
Monitor- en cassetteroutines	systeem-informatie	47
Machinetaal voor beginners	artikel	52
Hardware-tips voor de beginner	hardware-tip	53
Plaatsen van componenten	hardware-tip	57
16 K geheugenuitbreiding	hardware-schema	57

<u>Titel</u>	<u>Type informatie</u>	<u>pag.</u>
<u>Nieuwsbrief 5</u>		
Het Philips Disk Operating System	systeem-software	59
Het UCSD-Pascal-systeem	systeem-software	72
Machinetaal achter BASIC zetten	programmeertip	79
Gebruik PRINT USING	programmeertip	81
CLEAR-statement	programmeertip	81
<u>Nieuwsbrief 6</u>		
Mensen en de P2000	Interview met Klaas Robers	82
Het Familiegeheugen	programmabeschrijving	85
Werken met een printer-buffer	gebruiksinformatie	86
VARPTR	programmeertip	87
SCART-plug	hardware-tip	87
Wis-het-scherm-toets in het Viewdata-programma	programmeertip	87
Viewdata: wenken voor de IL	gebruiksinformatie	88
40 tracks	systeeminformatie	92
String-arrays naar cassette schrijven	programmeertip	93
Werken met de computer; soms oppassen ?	artikel	96
Graphics printen in bit image mode	BASIC-programma	97
Subroutine tbv sortering	BASIC-programma	97
Disk lezen/schrijven met cassette-BASIC	BASIC-programma	97
Files vanaf schijf op het scherm zetten	machinetaalprogramma	99
<u>Nieuwsbrief 7</u>		
Data-communicatie	artikel	104
Nieuw ROM-pack: TEXT2000	programmabeschrijving	105
Inventar	programmabeschrijving	108
Opnieuw: onzichtbare listings	programmeertip	109
Variabele RESTORE	programmeertip	110
FORTH op de P2000	systeeminformatie	111
Serial interface	hardware-informatie	116
Printer ready	programmeertip	117
P 2174 I/O-adressen	hardware-informatie	117
Programma terugroepen	praktische tip	117
Machinetaal in programmaregels	programmeertip	118
String binnen een programma opbergen	programmeertip	119
Boekbesprekingen	praktische informatie	120

Bij deze samenvatting

Dit is een nieuwe samenvatting van de Nieuwsbrieven nummers 1 tot en met 7 die de P2000gg tussen 1981 en 1983 heeft uitgegeven. De eerste goede reden om een nieuwe samenvatting uit te geven is dat de oude samenvatting door Ad Schellekens en Peter Janssens is uitverkocht. De tweede is dat er nog steeds belangstelling is voor een aantal artikelen die destijds zijn gepubliceerd.

Deze samenvatting is in grote trekken identiek aan de voorgaande. Slechts enkele artikelen die aperte onjuistheden bevatten of waarvan de inhoud sinds de introductie van BASIC NL niet meer geldig is, zijn aangepast of weggelaten. Verder heb ik enkele overbodigheden weggelaten: Dat geldt met name de overzichten van geheugenadressen die met de regelmaat van de klok, steeds wat verder uitgebreid, werden gepubliceerd. De inhoud en betekenis van deze adressen, waarin alle voorgaande overzichten zijn verwerkt, is bij de PTC verschenen onder de naam P2000 Adresboekje. Hetzelfde geldt voor de "listings" van delen van de Monitor. Ook deze zijn, maar nu volledig, als boekje verkrijgbaar bij de PTC.

Dat er destijds artikelen zijn gepubliceerd met onjuistheden mag u de auteurs en redacteuren van toen niet al te kwalijk nemen. De eerste bezitters van een P2000 tastten volledig in het duister in zaken die nu alom bekend zijn. Sommige suggesties van gebruikers zijn later vervangen door nieuwere en betere. Eigenlijk zou u deze samenvatting en die van de Nieuwsbrieven 8 tot en met 11 dus van achteren naar voren moeten lezen.

Ook al hebben sommige artikelen niet meer dezelfde ontdekkingswaarde als weleer, toch zullen sommigen van u bij het doorbladeren van deze samenvatting een gevoel van nostalgie niet kunnen onderdrukken. Wat was het heerlijke tijd, toen zelfs de kleinste ontdekking nog NIEUWS met louter boterletters was; toen we in voortdurende spanning leefden over wat de mensen van de fameuze NatLab Thuiscomputerclub nu weer hadden gebrouwen; toen het aantal leden van de P2000gg de honderd en wat later de duizend passeerde. Dat is allemaal voorbij. Wat gebleven is, is de verwondering over het feit dat een geweldige thuiscomputer als de P2000, waarschijnlijk de thuiscomputer die het langst op de markt is geweest (en op dit moment nog is!), ondanks al die bewezen kwaliteiten nooit de erkenning en waardering heeft gekregen die hij verdiende. Behalve van de bezitters dan.

Veel plezier met deze samenvatting.

Son, juli 1985
Rob Geutskens
Voorzitter P2000gg

Zo begon het:

"Gevraagd mede leden die een Philips P2000 bezitten.
Om gegevens uit te wisselen en misschien een gg op te
richten.
M. H. Bruins"
(bron: HCC Nieuwsbrief 28, mei 1981)

Een maand later stond in hetzelfde blad:

"27 juni (1981, red.) om 12.00 uur is er een bijeenkomst
voor P2000 bezitters en geïnteresseerden in De Bron te
Utrecht. Dit om een gebruikersgroep op te richten en
informatie uit te wisselen.
M. H. Bruins"

Aldus werd de P2000gg een feit. Op genoemde bijeenkomst meldden
de eerste P2000-gebruikers zich aan als lid, en reeds enkele
maanden later verscheen de eerste Nieuwsbrief met een ledenlijst
van zowaar 34 leden.

Op het moment dat deze inleiding is geschreven mag de P2000gg
zich met meer dan 900 leden de grootste merk-gerichte gebruikers-
groep binnen de HCC noemen.

Met de ontwikkeling en distributie van een nieuwe Basic-versie,
een recente prijsverlaging en de komst van vele nieuwe leden
sluiten we een periode van experimenteren en pionierschap af.
De P2000 heeft inmiddels een gevestigde plaats binnen het vader-
landse huis-computergebeuren ingenomen, en met name in het onder-
wijs is de P2000 een veel gebruikte computer.

Door het samenstellen van deze bloemlezing uit de zeven tot nu
toe verschenen Nieuwsbrieven wil de redactie een aantal bijdra-
gen uit deze periode aan u doorgeven. De artikelen in deze samen-
vatting zijn integraal uit de Nieuwsbrieven overgenomen.
Ze zijn oorspronkelijk geschreven voor Basic 1.0 UK. U zult dan
ook hier en daar informatie aantreffen welke niet zonder meer
geldig is voor de per 1 aug. j.l. geïntroduceerde Basic 1.1 NL.
Dit neemt niet weg dat u met deze samenvatting ongetwijfeld over
een bruikbare verzameling van informatie beschikt. De bijdragen,
die van zeer uiteenlopende aard zijn, bieden vaak een scala van
mogelijkheden met de P2000 waaruit de echte hobbyist zijn eigen
keuze kan maken.

Tot slot danken wij de leden van de P2000gg voor de vele ingezon-
den bijdragen die de redactie tot nu toe (en nog steeds!) van u
heeft mogen ontvangen. Deze dank gaat ook uit naar de leden van
de voormalige Nat.-Lab.-Thuiscomputerclub en de huidige Philips
P2000 Computer Club (P2C2). Wij hopen dat ook in de toekomst nog
veel artikelen van P2000-gebruikers gepubliceerd zullen worden.
Wij wensen dat u met deze bloemlezing net zo veel vreugde zult
hebben als wij aan het samenstellen hebben gehad.

Bunnik / Venray, november 1983,

Ad Schellekens,
Peter Janssens.

In deze rubriek is plaats voor allerlei praktische tips die het werken met de P2000 enigszins kunnen veraangenamen. Niet dat het werken met de P2000 niet aangenaam zou zijn - verre van dat! Maar aangezien hij (nog) niet in staat is om programma's voor je te maken, zul je dat toch zelf moeten doen en daarvoor menig uur achter het toetsenbord door moeten brengen, driftig bladerend in de basic-handleiding en wat dies meer zij. En er zijn daarbij van die kleine ongemakken die met wat vindingrijkheid wellicht enigszins verzacht kunnen worden.

Zo heeft meneer Philips op de P2000 een keurig venstertje gemaakt waarin je een kaartje met gegevens kunt stoppen, waardoor ze bij het maken van programma's overzichtelijk bij de hand zijn.

In de aanvullende documentatie die je dank zij de inspanningen van de onvolprezen "Thuis Computer Club" hebt ontvangen, kun je zo'n kaartje met toetscodes vinden. Uitknippen, stukje plakplastic erop, en klaar is kees! Prima idee! Zo zijn er meer kaartjes te maken. Van de grafische karakters bijvoorbeeld, van de printerinstructies, de ascii-codes, de print chr%-instructies, etc. Handige hulpjes bij het programmeren.

Hierbij alvast een kaartje voor de EDIT-instructies voor het veranderen van programmaregels.

EDIT<regelno.> : start EDIT-mode.	'codetoets' : beëindigen invoegen.
'spatiebalk' : cursor naar rechts (retype).	(i) H : wissen na cursor; INSERT.
'correctietoets': cursor naar links (retype).	(i) D : wissen (i) karakters.
(i) S <kar> : cursor naar (i-de) karakter.	(i) K <kar> : wissen van cursor tot (i-de) kar.
X <tekst> : cursor naar eind; INSERT.	A : regel herstellen; EDIT.
L : regel printen; EDIT.	Q : regel herstellen; BASIC.
(i) C <kar> : veranderen (i) karakter(s).	E : beëindigen edit ; BASIC.
(i) I <tekst> : invoegen tekst/(i-karakters).	'returntoets' : beëindigen edit en print; BASIC.

Wie volgt? Met de Philips printer in smalle karakters kun je zeven regels van honderd karakters op een kaartje zetten (formaat: 156 mm * 36 mm).

Heb je geen printer? Geen nood : wij zetten het wel op een kaartje.

Maar ook anderssoortige programmeerhulp en technische en programmeertips zijn welkom!

Hieronder volgt een artikel geschreven door Gert-Jan v.d. Eijkel
Hoekweg 43, Voorburg :

1) De tweede tekstpagina op het M-model.

Bij het M-model zit van &H5800-&H6000 een stuk geheugen waar
getallen staan. In het T-model is dit stuk geheugen er niet.
Deze getallen, bytes houden een soort overlay, een karaktertype,
voor het karakter dat &H800 bytes in het geheugen terug staat.
Staat er bijv. in &H5000 een byte 65 (=A) dan staat de overlay
op &H5000 + &H800 = &H5800.

Door de mod 16 uit te rekenen van de overlay kan men met behulp
van onderstaande tabel de wijze waarop het karakter &H800 bytes
terug op het scherm komt aflezen:

MOD 16	(1)	(2)	(3)	mode
	[]	[]	[0-127] 128-255]	
0	[]		[geen] flash]	tekst
1	[]	GEWOON	[geen] flash]	grafisch
2	[]		[gewoon] flash]	tekst
3	[]		[gewoon] flash]	grafisch
4	[ZWART]	FLASH	[geen] gewoon]	tekst
5	[]		[geen] gewoon]	grafisch
6	[]		[flash] gewoon]	tekst
7	[]		[flash] gewoon]	grafisch
8	[]		[geen] flash]	tekst
9	[]		[geen] flash]	grafisch
10	[]	GEWOON	[gewoon] flash]	tekst
11	[]		[gewoon] flash]	grafisch
12	[WIT]	FLASH	[geen] gewoon]	tekst
13	[]		[geen] gewoon]	grafisch
14	[]		[flash] gewoon]	tekst
15	[]		[flash] gewoon]	grafisch

Uitleg bij de tabel:

- (1) = achtergrond kleur (zwart, wit)
- (2) = voorgrond type (gewoon, flash)
- (3) = onderlijning (geen, flash, gewoon = stabiele onderlijn)
- Flash = knipperend

2) Regel indeling van een Basic-programma.

Een basic programma wordt door de interpreter op een bepaalde wijze in het geheugen geplaatst:

- L,H : De eerste 2 bytes wijzen naar het geheugen adres waar de volgende regel begint.
- L,H : De 2 daaropvolgende bytes bevatten het regelnummer
 - . : Dan komt er een commando code, de basic opdrachten worden opgeslagen als een bepaald getal tussen 128 en 255, de zg. tokens
 - .. : Dan komen de gegevens die bij dat commando horen. Deze gegevens staan in ascie code.
 - : : Ter onderscheiding van meerdere commando's staat een dubbele punt ook in ascie.
 - 0 : Tenslotte wordt de regel afgesloten met een byte die nul is.
- 000 : Het hele programma wordt afgesloten door 3 bytes die nul zijn.

Variabelen opslag:

De variabelen worden opgeslagen in de variabelen ruimte vlak achter het programma.

Hiebij wordt het volgende systeem gebruikt:

```
*-----*-----*-----*-----*-----*-----*
[   ] [   ] [   ] [   ] [   ] [   ]
[ 1 ] 2 [ 3 ] 4 [ 5 ] 6 [   ]
[   ] [   ] [   ] [   ] [   ] [   ]
*-----*-----*-----*-----*-----*-----*
```

- 1 : Hoeveel bytes data
- 2 : Eerste karakter van het label in ascie.
- 3 : Tweede karakter van het label in ascie.
- 4 : Hoeveel bytes vervolg van het label.
- 5 : vervolg label (max. 38 bytes) - kar +128
- 6 : Data

Bijvoorbeeld de variabele string

```
1 2 3 4      5          6
3 S T 4 R I N G .. .. ..
```

Aan byte 1 is te zien wat voor variabele het is:

- = 2 : integer variabele
- = 3 : string variabele
- = 4 : single precision real variabele
- = 8 : double precision real variabele

Hieronder volgt een artikel geschreven door Gert-Jan v.d. Eijkel
Hoekweg 43, Voorburg :

1) De tweede tekstpagina op het M-model.

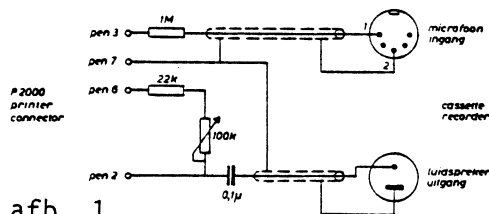
Bij het M-model zit van &H5800-&H6000 een stuk geheugen waar
getallen staan. In het T-model is dit stuk geheugen er niet.
Deze getallen, bytes houden een soort overlay, een karakertype,
voor het karakter dat &H800 bytes in het geheugen terug staat.
Staat er bijv. in &H5000 een byte 65 (=A) dan staat de overlay
op &H5000 + &H800 = &H5800.

Door de mod 16 uit te rekenen van de overlay kan men met behulp
van onderstaande tabel de wijze waarop het karakter &H800 bytes
terug op het scherm komt aflezen:

MOD 16	(1)	(2)	(3)	mode
	[]	[]	[0-127] [128-255]	[]
0	[]		[geen] [flash]	tekst
1	[]	GEWOON	[geen] [flash]	grafisch
2	[]		[gewoon] [flash]	tekst
3	[]		[gewoon] [flash]	grafisch
	[ZWART]			
4	[]	FLASH	[geen] [gewoon]	tekst
5	[]		[geen] [gewoon]	grafisch
6	[]		[flash] [gewoon]	tekst
7	[]		[flash] [gewoon]	grafisch
8	[]		[geen] [flash]	tekst
9	[]		[geen] [flash]	grafisch
10	[]	GEWOON	[gewoon] [flash]	tekst
11	[]		[gewoon] [flash]	grafisch
	[WIT]			
12	[]		[geen] [gewoon]	tekst
13	[]	FLASH	[geen] [gewoon]	grafisch
14	[]		[flash] [gewoon]	tekst
15	[]		[flash] [gewoon]	grafisch

Uitleg bij de tabel:

- (1) = achtergrond kleur (zwart, wit)
- (2) = voorgrond type (gewoon, flash)
- (3) = onderlijning (geen, flash, gewoon = stabiele onderlijn)
- Flash = knipperend



afb. 1

afb. 2

```

10 REM Afregelen cassette interface
20 CLEAR 100,&H9F00
100 REM Proeftonen opnemen
110 DATA 9F20
120 DATA F3,0E,80,3E,C0,D3,10,06,4C,CD
130 DATA 62,9F,3E,40,D3,10,DB,00,EE,FF
140 DATA C2,60,9F,06,49,CD,62,9F,0D,C2
150 DATA 23,9F,3E,C0,D3,10,06,24,CD,62
160 DATA 9F,3E,40,D3,10,DB,00,EE,FF,C2
170 DATA 60,9F,06,21,CD,62,9F,0D,C2,40
180 DATA 9F,C3,21,9F,FB,C9,10,FE,C9,XX,
190 RESTORE 110 : S=-16194 : GOSUB 500
200 DEFUSR1=A : PRINT CHR$(12)
210 PRINT "Neem nu een proefstukje op."
220 PRINT "Afbreken met elke toets."
230 A=USR1(0) : PRINT CHR$(12)
240 PRINT "Nu weerstand afregelen : "
300 REM Sterren boven in het beeld
310 DATA 9F80
320 DATA F3,16,50,26,00,2E,00,06,00,5C
330 DATA 3E,20,12,78,E6,3F,C6,00,5F,67
340 DATA 3E,2A,12,06,00,04,28,24,DB,20
350 DATA E6,01,28,F7,00,00,00,5D,3E,20
360 DATA 12,78,E6,3F,C6,50,5F,6F,3E,2A
370 DATA 12,06,00,04,28,08,DB,20,E6,01
380 DATA 20,F7,18,C9,FB,C9,XX,
390 RESTORE 310 : S=-19559 : GOSUB 500
400 DEFUSR2=A
410 PRINT CHR$(12) : A=USR2(0) : GOTO410
420 END
500 REM Zet machinetaal in geheugen
510 READ P% : I=VAL("&H"+P%) : A:I :T=I
520 READ P% : IF P%="XX" THEN GOTO 550
530 P=VAL("&H"+P%) : POKE I,P : I=I+1
540 T=T+P : GOTO 520
550 IF T=S THEN RETURN
560 PRINT "Typefout gemaakt" : END

```

```

10 REM P2000-programma op audiocassette
20 CLEAR 200,&H9DFF : PRINT CHR$(12)
30 READ P% : I=VAL("&H"+P%) : T=I
40 READ P% : IF P%="XX" THEN GOTO 70
50 P=VAL("&H"+P%) : POKE I,P : I=I+1
60 T=T+P : GOTO 40
70 IF T=10309 THEN GOTO 90
80 PRINT "Typefout gemaakt in DATA":END
90 DEF USR8=&H9E08 : DEF USR9=&H9E28
100 PRINT "?USR8(0) zet P2000-progr. op
cassette."
110 PRINT "?USR9(0) laadt cass.progr. in
P2000."
200 DATA 9E00
210 DATA 00,00,00,00,00,00,00,00,B7,2A,0
5,64,ED,5B,5C,62,ED,52,22,04,9E,CD,5C,9E
220 DATA 36,55,3E,40,CD,8E,9E,CD,69,9E,3
E,20,CD,8E,9E,C9,D5,CD,5C,9E,CD,E6,9E,20
230 DATA 17,2A,5C,62,ED,5B,04,9E,19,22,0
5,64,22,07,64,22,09,64,CD,69,9E,CD,E6,9E
240 DATA E1,36,00,C8,00,36,06,23,36,56,2
3,36,9E,C9,81,45,72,72,6F,72,21,04,9E,22
250 DATA 00,9E,21,06,9E,22,02,9E,C9,2A,5
C,62,22,00,9E,2A,05,64,22,02,9E,C9,3E,C0
260 DATA 06,83,10,FE,D3,50,3D,20,F7,C9,D
5,EB,2A,02,9E,B7,ED,52,EB,D1,23,C9,F3,37
270 DATA 16,00,2A,00,9E,CD,8F,9E,06,27,A
E,32,07,9E,7E,CD,86,9E,CD,82,9E,06,22,3A
280 DATA 07,9E,20,EE,06,22,CD,86,9E,CD,7
6,9E,FB,C9,16,10,17,CD,CC,9E,20,FA,C9,06
290 DATA 4B,CD,D1,9E,20,F9,CE,FE,38,F5,0
6,1F,CD,D1,9E,04,04,10,FE,30,04,06,30,10
300 DATA FE,08,2F,E6,80,F6,40,D3,10,08,0
6,2A,15,C9,F3,2A,00,9E,CD,32,9F,3E,16,CD
310 DATA BF,9E,32,07,9E,CD,32,9F,06,20,C
D,35,9F,38,F9,CD,35,9F,06,35,CD,24,9F,77
320 DATA 3A,07,9E,AE,32,07,9E,CD,82,9E,0
6,30,20,EE,CD,24,9F,21,07,9E,BE,F5,CD,76
330 DATA 9E,F1,FB,C9,16,08,5F,CD,32,9F,7
B,17,06,34,15,20,F5,C9,CD,35,9F,05,DB,20
340 DATA 0F,A9,F2,35,9F,A9,4F,3E,80,B8,C
9,XX, : END

```

3). Dit biedt de mogelijkheid om van programma's die voor normaal gebruik op minicassette staan een extra copie te maken op de veel goedkopere compactcassettes. Voor het inlezen en decoderen van basicodeprogramma's is een apart programma vereist. In het septembernummer van DATABUS is een eerste versie van dit programma gepubliceerd. Op 30 september a.s. wordt echter in het programma Hobbyscoop (19.30 - 20.00 uur op Hilversum 1) een verbeterde versie van dit programma uitgezonden. Als je snel aan de slag wilt loont het dus de moeite dit programma alvast op een normale cassetterecorder op te nemen.

```

10 REM * HYPOTHEEK LENING *
20 REM * HET GEEFT EEN ANNUITEITEN EN EE
N LINIAR TABEL *
30 REM * DIT PROGRAMMA IS OOK GESCHIKT V
OOR EEN PRINTER *
40 DIMI(30),AK(30)
50 PRINTCHR$(12)
60 PR$="":INPUT"printer gebruiken j/n";P
R$:PRINTCHR$(12)
70 PRINT" * HYPOTHECAIRE LENING *"
80 PRINT:PRINT
90 INPUT"geleend kapitaal ":K:PRINT
100 INPUT"rente in % ":I:PRINT
110 INPUT"looptijd in jaren ":J:PRINT
120 A=K*I/100/(1-(1/(1+I/100))^J)
130 AN=A
140 PRINT:PRINT"ANNUITEITENLENING"
150 PRINT"De annuiteit bedraagt fl.":USI
NG"_____";AN:PRINT
160 PRINT"Dat is per halfjaar fl.":USI
NG"_____";AN/2:PRINT
165 PRINT"Dat is per maand fl.":USI
NG"_____";AN/12:PRINT
167 PRINT"LINIAIRE HYPOTHEEK":KK=K/J:PRI
NT"fl.":USING"_____";KK:PRINT
168 PRINT"rente eerste jaar fl.":USIN
G"_____";K*I/100:IFPR$="j"GRPR$="J"
HENPRINTCHR$(5)
170 PRINT:PRINT:INPUT"return voor een ta
bel ":A$
175 INPUT"TABEL VOOR ANNUITEITEN OF LINI
AIR AN/L":L$:KK=K/J
180 PRINT" AFGELOST RENTE
SCHULD"
190 IFPR$="J"THENLPRINTTAB(18)"AFGELOST"
;TAB(33)"RENTE";TAB(45)"RESTART SCHULD";
TAB(60)"JAARBEDRAG"ELSEGOTO200
195 LPRINT
200 PRINT
210 FORN=1TOJ-1
220 I(N)=K*I/100
230 IFL$="AN"THENAK(N)=AN-I(N)ELSEAK(N)=
KK
240 K=K-AK(N)
249 PRINT"na":IFN<10THENPRINT" ";
250 PRINTN;"jr ":PRINTUSING"_____";
AK(N);I(N);K
255 IFPR$="J"THENLPRINT"na ":USING"__";N
::LPRINT" jaar":LPRINTTAB(15)USING"f1_
_____"AK(N)::LPRINTTAB(30)USING"f1_
_____"I(N)::LPRINTTAB(45)USING"f1_
_____"K::LPRINTTAB(60)USING"f1_
_____"AK(N)+I(N)
260 NEXT

```

Hier een programma voor het berekenen van de bedragen van een hypothecaire lening met een sample-run. Heb je een kort programma dat voor andere leden van belang kan zijn, stuur het dan op voor publicatie.

```

270 IFL$="AN"THENPRINT"na ":PRINTUSING"
__";J:PRINT" jr ":PRINTUSING"_____";
K;AN-K;K-KELSEPRINT"na ":PRINTUSING"
__";J:PRINT" jr ":USING"_____";K;K
*I/100;K-K
280 IFPR$="J"THEN300
290 END
300 IFL$="AN"THENLPRINT"na ":USING"__";J
::LPRINT" jaar":LPRINTTAB(15)USING"f1_
_____"K::LPRINTTAB(30)USING"f1_
_____"AN-K::LPRINTTAB(45)USING"_____";
K-K::LPRINTTAB(60)USING"f1_";K+(
AN-K);ELSEGOTO310
305 END
310 LPRINT"na ":USING"__";J:LPRINT" ja
ar":LPRINTTAB(15)USING"f1_";K::
LPRINTTAB(30)USING"f1_";K*I/100:
LPRINTTAB(45)USING"f1_";K-Y::LP
RINTTAB(60)USING"f1_";K+K*I/100

```

* HYPOTHECAIRE LENING *

geleend kapitaal :? 150000

rente in % ? 12.5

looptijd in jaren :? 25

ANNUITEITENLENING

De annuiteit bedraagt fl. 19791.50

Dat is per halfjaar fl. 9895.75

Dat is per maand fl. 1649.29

LINIAIRE HYPOTHEEK

fl. 6000.00

rente eerste jaar fl. 18750.00

Van de hr. Timmermans, Schaapsdijkweg 10, Venlo ontvingen wij de volgende handige tip. Om de P2000-T aan te sluiten op het tv-toestel is het altijd noodzakelijk om de antennekabel aan de achterzijde van het tv-apparaat te verwijderen en dan de UHF-kabel van de computer in te pluggen. Om deze "operatie te vergemakkelijken kunt U beter gebruik maken van een z.g. T-stuk welke steeds in het tv-apparaat blijft zitten. Het biedt dan twee voordelen: het voorkomt veel gefriemel en U kunt tijdens lange wachttijden naar de tv kijken. Het genoemde T-stuk is in iedere radio en tv zaak te koop. Mocht dit niet zo zijn dan kunt U zich tot de hr. Timmermans wenden die voor ongeveer fl 8,50 zorg zal dragen voor een snelle toezending.

Een tip voor het programmeren met de DEFFN functie. Deze functie is erg geschikt wanneer een bepaalde programmaregel met dezelfde definitie regelmatig terugkomt.

```
Voorbeeld: 10 DEFFN C$(A,B,C,D)= CHR$(A)+CHR$(B)+CHR$(C)+CHR$(D)
           20 DEFFN D$(E,F)= CHR$(4)+CHR$(E)+CHR$(F)
           100 PRINT FN D$(5,1);FN C$(141,129,157,135);"TEST
              DEFFN FUNKTIE"
           110 PRINT FN D$(20,1);FN C$(141,130,136,7);"D$ GEEFT
              REGEL POSITIE"
```

De FN D\$(20,10) geeft regelnummer 20 en 10 posities naar rechts op het beeldscherm. De FN C\$(141,129,157,135) geeft dubbele hoogte, kleur rood, balk, letterkleur.

VOLORG

Voor bezitters van 24K BASIC is het handig om voor eigen aanvullingen, veranderingen enz. het programma VO/ORG in onbeschermd toestand te kunnen laden en wegschrijven. Om dit te realiseren moet men de volgende handelingen uitvoeren:

- 1) Exit VO/ORG
- 2) POKE 25513,126: POKE 25514,146: POKE 26400,131:
POKE 26401,190
- 3) Voer 2 regels tekst in: 0 REM
10 REM VO/ORG rel. 1.1
- 4) SAVE "VO/ORG.BAS"

Opmerking: Poke inhouden van 26400 en 26401 zijn de inhouden van resp. 26643 en 26644 verminderd met 6.

ZEER BELANGRIJKE TIP !!!

Tijdens het lezen of schrijven van/naar cassette NOOIT de reset knop indrukken. Uw cassette wordt onherroepelijk onleesbaar. Om eventueel een cassette lees of schrijf routine te onderbreken moet U het klepje van de cassette recorder laten openspringen. Op het scherm verschijnt dan de mededeling "cassette error A".

Na de toetsenbordroutine in de eerste P2000gg-nieuwsbrief volgt hier de monitorlisting van de beeldschermroutine, waarin onder meer een groot aantal mogelijkheden van de tekstverwerker te vinden. Het gaat om zaken als shift/shiftlock, cursorsturing, invoegen, scrollen, etc. etc.

Lambert Knapen.

De listing hiernaast is niet van de monitor, maar van de BASIC interpreter, versie 1 (BASIC UK). BASIC NL zit totaal anders in elkaar. Twee redenen om deze listing te laten staan: om u te helpen machinetaalroutines te doorgronden, en omdat de routines vrijwel identiek zijn aan die van 24K DISK BASIC (adressen &H69D6...6B18).

RG

10E8	C5	PUSH	BC	RED DE REGISTERS
10E9	D5	PUSH	DE	
10EA	E5	PUSH	HL	
10EB	F5	PUSH	AF	
10EC	CD 84 13	CALL	1384H	DOOF KURSOR
10EF	C5	PUSH	BC	
10F0	3A 0F 60	LD	A, (600FH)	SHIFT/SHIFT LOCK
10F3	CB 47	BIT	0,A	SHIFT ?
10F5	28 09	JR	NZ,1100H	GEEN SHIFT
10F7	06 00	LD	B,00H	VERTRAGING VOOR SHIFT
10F9	3E 20	LD	A,20H	
10FB	10 FE	DJNZ	10FBH	TOETS (ong. 40 msec.)
10FD	3D	DEC	A	
10FE	20 FB	JR	NZ,10FBH	
1100	06 01	LD	B,01H	VERTRAGING VOOR
1102	3A A6 60	LD	A, (60A6)	GEHEUGENLOKATIE 60A6H
1105	B7	OR	A	STATUSREGISTER WAARDE
1106	20 01	JR	NZ,1109H	GEVEN
1108	3C	INC	A	
1109	10 FE	DJNZ	1109H	
110B	3D	DEC	A	
110C	20 FB	JR	NZ,1109H	
110E	C1	POP	BC	
110F	3A A7 60	LD	A, (60A7H)	STATUS KURSORKONTROLE
1112	FE 01	CP	01H	CHR*(4) LAATSTE ?
1114	20 0C	JR	NZ,1122H	NEE
1116	3C	INC	A	JA, NU VOLGT REGENUMMER
1117	32 A7 60	LD	(60A7H),A	1:NU VOLGT REGENUMMER
111A	F1	POP	AF	2:NU VOLGT KOLOMNUMMER
111B	F5	PUSH	AF	3:KURSORKONTROLE OKE
111C	32 A8 60	LD	(60A8H),A	REGENUMMER
111F	C3 30 12	JP	1230H	HERSTEL REGISTERS EN RET
1122	FE 02	CP	02H	KURSORKONTROLE HALFWEG
1124	20 0F	JR	NZ,1135H	NEE
1126	3C	INC	A	JA, NU VOLGT KOLOMNUMMER
1127	32 A7 60	LD	(60A7H),A	KURSORKONTROLE OKE
112A	F1	POP	AF	
112B	F5	PUSH	AF	
112C	32 A9 60	LD	(60A9H),A	KOLOMNUMMER KURSOR
112F	CD 6A 12	CALL	126AH	PLAATS KURSOR OP PLAATS
1132	C3 2D 12	JP	122DH	KURSOR AAN + RETURN
1135	F1	POP	AF	
1136	F5	PUSH	AF	

1137	FE 04	CP	04H	KURSORBESTURING ?
1139	20 08	JR	NZ,1143H	NEE
113B	3E 01	LD	A,01H	KURSORKONTROLE OP KOMST
113D	32 A7 60	LD	(60A7H),A	
1140	C3 30 12	JP	1230H	HERSTEL REGISTERS + RET
1143	FE 0C	CP	0CH	FORM FEED?
1145	20 06	JR	NZ,104D	NEE
1147	CD 3C 12	CALL	123CH	WIS HET BEELDSCHERM +HOME
114A	C3 2D 12	JP	122DH	KURSOR,KURSOR AAN + RET
114D	FE 05	CP	05H	BEELDSCHERM OP PRINTER?
114F	20 1C	JR	NZ,116DH	NEE
1151	21 B0 4F	LD	HL,4FB0H	REGEL VOOR BEELDSCHERM
1154	E5	PUSH	HL	
1155	06 18	LD	B,18H	24 REGELS
1157	11 50 00	LD	DE,0050H	REGELLENGTE 80 KARAKTERS
115A	E1	POP	HL	
115B	19	ADD	HL,DE	BEGIN NIEUWE REGEL
115C	E5	PUSH	HL	BEMAAR OP STACK
115D	0E 50	LD	C,50H	REGELLENGTE 80 KARAKTERS
115F	3E 01	LD	A,01H	PRINT 1 REGEL PER KEER
1161	CD 49 14	CALL	1449H	PRINTER READY;PRINT REGEL
1164	DA 39 14	JP	C,1439H	PRINTER ERROR
1167	10 EE	DJNZ	1157H	PRINT VOLGENDE REGEL
1169	E1	POP	HL	
116A	C3 2D 12	JP	122DH	KURSOR AAN + RETURN
116D	FE 0D	CP	0DH	NAAR BEGIN REGEL?
116F	20 09	JR	NZ,117AH	NEE
1171	37	SCF		
1172	F5	PUSH	AF	
1173	AF	XOR	A	KOLOMTELLER OP NUL
1174	32 B3 60	LD	(60B3H),A	
1177	C3 2C 12	JP	122CH	KURSOR AAN + RETURN
117A	FE 0A	CP	0AH	LINE FEED?
117C	20 2A	JR	NZ,11A8H	NEE
117E	37	SCF		CLEAR CARRY FLAG
117F	3F	CCF		
1180	F5	PUSH	AF	
1181	2A B1 60	LD	HL,(60B1H)	BEGIN FYSIEKE REGEL
1184	01 30 57	LD	BC,5730H	EINDE BEELDSCHERM
1187	ED 42	SBC	HL,BC	BEPAAAL VERSCHIL
1189	20 06	JR	NZ,1191H	NUL; DAN EINDE BEELD
118B	CD A6 12	CALL	12A6H	SCROLL BEELDSCHERM
118E	C3 2C 12	JP	122CH	KURSOR AAN + RETURN
1191	2A B1 60	LD	HL,(60B1H)	KURSOR NAAR
1194	06 00	LD	B,00H	
1196	0E 50	LD	C,50H	
1198	09	ADD	HL,BC	
1199	22 B1 60	LD	(60B1H),HL	VOLGENDE REGEL
119C	F1	POP	AF	
119D	FE 0A	CP	0AH	LINE FEED?
119F	20 03	JR	NZ,11A4H	NEE
11A1	22 B4 60	LD	(60B4H),HL	BEGIN LOGISCHE REGEL
11A4	F5	PUSH	AF	
11A5	C3 2C 12	JP	122CH	KURSOR AAN + RETURN
11AB	37	SCF		CLEAR CARRY FLAG
11A9	3F	CCF		
11AA	F5	PUSH	AF	
11AB	FE 08	CP	08H	BACK SPACE?
11AD	C2 DB 11	JP	NZ,11DBH	NEE
11B0	3A B3 60	LD	A,(60B3H)	KOLOM TELLER

11B3	FE 00	CP	00H	GELIJK NUL?
11B5	C2 CD 11	JP	NZ,11CDH	NEE
11B8	3A B0 60	LD	A, (60B0H)	INGESTELDE BEELDBREEDTE
11BB	32 B3 60	LD	(60B3H),A	KURSOR NAAR
11BE	2A B1 60	LD	HL, (60B1H)	
11C1	06 00	LD	B,00H	
11C3	0E 50	LD	C,50H	
11C5	ED 42	SBC	HL,BC	
11C7	22 B1 60	LD	(60B1H),HL	EINDE VORIGE REBEL
11CA	C3 D4 11	JP	11D4H	
11CD	3A B3 60	LD	A, (60B3H)	KURSOR
11D0	3D	DEC	A	EEN POSITIE
11D1	32 B3 60	LD	(60B3H),A	TERUG
11D4	F1	POP	AF	
11D5	CD 62 13	CALL	1362H	VIND KURSOR POSITIE
11D8	C3 2D 12	JP	122DH	KURSOR AAN + RETURN
11DB	FE 0F	CP	0FH	SHIFT IN ?
11DD	20 06	JZ	NZ,11E5H	NEE
11DF	CD FF 12	CALL	12FFH	WIS GETYPTE (REBEL
11E2	C3 2C 12	JP	122CH	KURSOR AAN + RETURN
11E5	FE 07	CP	07H	PIEPER ?
11E7	CC A8 13	CALL	Z,13A8H	JA, PIEPER !
11EA	28 40	JR	Z,122CH	KURSOR AAN + RETURN
11EC	FE 20	CP	20H	BESTURINGS KARAKTER ?
11EE	38 3C	JR	C,122CH	JA; KURSOR AAN + RETURN
11F0	CB 7F	BIT	7,A	
11F2	28 0C	JR	Z,1200H	
11F4	CB BF	RES	7,A	
11F6	21 A6 18	LD	HL,18A6H	AANPASSING
11F9	CD 5A 12	CALL	125AH	GRAPHICS
11FC	CB FF	SET	7,A	
11FE	18 06	JR	1206H	
1200	21 A6 18	LD	HL,18A6H	OP
1203	CD 5A 12	CALL	125AH	BEELDSCHERM
1206	CD 99 13	CALL	1399H	
1209	F5	PUSH	AF	
120A	3A 13 60	LD	A, (6013H)	
120D	E6 01	AND	01H	
120F	20 09	JR	NZ,121AH	
1211	F1	POP	AF	
1212	21 E0 18	LD	HL,18E0H	
1215	CD 5A 12	CALL	125AH	
1218	18 01	JR	121BH	
121A	F1	POP	AF	
121B	CD 62 13	CALL	1362H	VIND KURSOR LOKATIE
121E	77	LD	(HL),A	
121F	3A B3 60	LD	A, (60B3H)	POSITIE KURSOR OP REBEL
1222	21 B0 60	LD	HL,60B0H	WIDTH BEELDSCHERM
1225	BE	CP	(HL)	
1226	28 0D	JR	Z,1235H	
1228	3C	INC	A	
1229	32 B3 60	LD	(60B3H),A	
122C	F1	POP	AF	
122D	CD 6F 13	CALL	136FH	CURSOR AAN
1230	F1	POP	AF	
1231	E1	POP	HL	
1232	D1	POP	DE	
1233	C1	POP	BC	
1234	C9	RET		

```

10 REM VOOR INLICHTINGEN EN PROBLEMEN WE
NDEN TOT L. KNAPEN
20 REM GEEF DIT PROGRAMMA NIET DE NAAM '
CONVERT.FIL'
30 REM DIT PROGRAMMA LEEST EEN PROGRAMMA
VAN CASSETTE ( EXTENDED BASIC ) EN CONV
ERTEERT DIT NAAR DISK BASIC.
40 REM HET PROGRAMMA WORDT ONDER DE NAAM
'CONVERT.FIL' OP DISK A GEZET.
50 REM MEN DIEN DE FILE NA HET KREEEREN
OP DISK, OPNIEUW IN TE LEZEN , WAARNA D
E FILE MET EEN SAVE OPDRACHT EEN ANDERE
NAAM GEGEVEN WORDT.
60 REM DIT IS NODIG OMDAT DAN DE FILE IN
HET GECODEERDE FORMAT WORDT OPGESLAGEN
OP DISK. DE FILE 'CONVERT.FIL' KAN GEWIS
T WORDEN OM MEER RUIMTE OP DE SCHIJF TE
KRIJGEN.
70 REM RESERVEER 20KB VOOR TAPE-PROGRAMM
A.
80 REM RESERVEER 200 BYTES VOOR STRINGSP
ACE
90 REM 2 KB VOOR MACHINETAALROUTINES IN
GEBRUIK.
100 CLEAR,&HA7FF,200
110 REM NAAM VAN DE FILE OP CASSETTE
120 INPUT "FILE NAAM";A$
130 REM DATATRANSFER ADRES
140 POKE &H6030,0:POKE &H6031,176
150 REM NAAM VAN HET PROGRAMMA OP CASSET
TE
160 POKE &H6036,ASC(MID$(A$,1,1))
170 REM BLOKKEN VAN 1 KB
180 POKE &H6033,4:POKE &H6032,0
190 REM BASIC PROGRAMMA OP TAPE
200 POKE &H6041,66
210 REM LAADADRES
220 POKE &H6046,176
230 POKE &H6045,0
240 DATA 245,229,197,213,126,42,48,96,23
7,91,50,96,205,24,0,209,193,225,241,201
250 POKE &H604F,1
260 FOR J=&H800 TO &H813:READ A$:POKE
J,A$:NEXT
270 DEF USRO=&H800
280 C%=0:I=USRO(C%)
290 C%=1:I=USRO(C%)
300 POKE &H6034,0:POKE &H6035,0:POKE &H6
032,1:POKE &H6033,0:POKE &H6036,ASC(LEFT
$(A$,1)):C%=6:I=USRO(C%): IF PEEK(&H6036
)<>ASC(LEFT$(A$,1)) THEN 300
310 POKE &H604F,1:C%=3:I=USRO(C%)
320 POKE &H6030,0:POKE &H6031,176
330 C%=6:I=USRO(C%)
340 RESET
350 OPEN "0",_1,"A:CONVERT.FIL"
360 OPTION BASE 1
370 DIM OPC$(116)

```

Met bijgaand programma kan een Extended Basic-programma geconverteerd worden naar Disk Basic.

```

380 DATA END,FOR,NEXT,DATA,INPUT,DIM,REA
D,LET,GOTO,RUN,IF,RESTORE,GOSUB,RETURN,R
EM,STOP,CONSOLE,WIDTH, ELSE,TRON,TROFF,S
WAP,ERASE,DEFSTR,DEFINT,DEFSGN,DEFDBL,LI
NE,EDIT,ERROR,RESUME,OUT,ON,WAIT,LPRINT
390 DATA DEF,POKE,PRINT,CONT,LIST,LLIST,
DELETE,AUTO,CLEAR,CLOAD,CSAVE,NEW,TAB(,
TO,FN,SPC(,USING,VARPTR,USR,ERL,ERR,STRI
NG$,INSTR,THEN,NOT,STEP,+,-,$/,+,AN
D,OR,XOR,EQV,IMP,MOD,'>,<,<,SGN,INT,ABS
,FRE,INP,LPOS,POS,SQR,RND,LOG,EXP,COS,SI
N,TAN,ATN,PEEK,CINT
400 DATA C$6N,CDBL,FIX,LEN,OCT$,HEX$,STR
$,VAL,ASC,CHR$,SPACE$,LEFT$,RIGHT$,MID$
410 FOR J=1 TO 107:READ OPC$(J):NEXT
420 J=&HB000
430 OFSET=&HB000-&H6547
440 FOR H= J TO &HFFFF
450 REM GET START OF NEXT LINE
460 GOSUB 540
470 REM GET LINE NUMBER
480 GOSUB 590
490 REM DECODE LINE
500 GOSUB 640
510 REM TEST END OF FILE
520 GOSUB 730
530 NEXT
540 REM GET STARTADRES NEXT LINE
550 OLE = PEEK(H)+256*PEEK(H+1)
560 NLE = OLE + OFSET
570 H=H+2
580 RETURN
590 REM GET LINE NUMBER
600 LIN = PEEK(H)+256*PEEK(H+1)
610 H=H+2
620 PRINT _1,LIN;
630 RETURN
640 FOR L=H TO NLE-2
650 F=PEEK(L)
660 IF F=58 AND PEEK(L+1)=146 THEN L=L+1
:F=147:GOTO 680
670 IF F>128 THEN F=F+1
680 IF F<128 THEN PRINT _1,CHR$(F); ELSE
PRINT _1," ";OPC$(F-128);" ";
690 NEXT
700 B$="":PRINT _1,B$
710 H=L
720 RETURN
730 REM TEST END OF FILE
740 IF PEEK(NLE+1)+PEEK(NLE+1)=0 THEN PR
INT:PRINT "END" ELSE RETURN
750 CLOSE
760 REM MAAK GEHEUGEN VRIJ VOOR GROOT PR
OGRAMMA. ANDERS OUT OF MEMORY
770 CLEAR,&HFF00,200
780 END

```


Het nu volgende programma maakt gebruik van een aantal BASIC-rom routines en een monitorroutine.

Deze routines zijn :

- a) Wis het beeldscherm
- b) Maak van het toetsnummer een ASCII-karakter
- c) Display een karakter op het beeldscherm
- d) Scan het toetsenbord

De volgende 'eigenaardigheden' zijn verstopt in het programma.

- a) Het kreeëren van een machineaal programma op tape.
- b) Het inlezen van een machinetaal-tape
- c) Het niet laten scrollen van het beeldscherm

Het programma werkt als volgt:

Wis het beeldscherm.

Scan het toetsenbord.

Display het karakter op het beeldscherm.

Test einde beeldscherm.

Indien einde, wis het scherm; 'HOME' cursor.

Ga terug naar Scan toetsenbord.

```

10 PRINT CHR$(12);CHR$(134);"DIT PROGRAM
MA VEREIST EEN LEGE TAPE. "
20 PRINT "DE WERKING VAN HET PROGRAMMA I
S ALS VOLGT:"
30 PRINT "A: EEN MACHINE-TAAL PROGRAMMA
WORDT IN HET GEHEUGEN GEPOKED"
40 PRINT "B: OP DE VOLGENDE ADRESSEN WOR
DEN DE VOLGENDE WAARDEN GEPOKED"
50 PRINT " 1) &H6034,&H6035: LENGTE V
AN DE FILE"
60 PRINT " 2) &H6043,&H6044: START AD
RES 3) &H6045,&H6046: LAAD ADR
ES 4) &H6030,&H6031: DATA OVE
RDRACHT ADRES
70 PRINT " 5) &H6032,&H6033: LENGTE V
AN DE TAPE
80 PRINT " 6) &H6036 : FILE-NAA
M
90 PRINT " 7) &H6041 : TYPE VAN
DE FILE
100 PRINT "C: MET BEHULP VAN EEN KLEIN M
ACHINE- TAAL PROGRAMMA WORDT HET A
NDERE MACHINE-TAAL PROGRAMMA OP
DE LEGE TAPE GEZET."
110 INPUT "DRUK OP RETURN VOOR VERDERE I
NFOR- MATIE ";D$
120 PRINT CHR$(12);" HET PROGRAMMA DAT O
P TAPE GEDUMPT WORDT, HEEFT DE VOL
GENDE WERKING : "
130 PRINT "A: HET BEELDSCHERM WORDT GEWI
ST."
140 PRINT "B: DE KEYBOARD-ROUTINE VAN DE
MONITOR WORDT AANGEROEPEN; INDIEN
EEN TOEST IS INGEDRUKT, ZAL VIA EEN

```

```

MONITOR AANROEP HET JUISTE ASCII-K
ARAKTER OP HET BEELDSCHERM VERSCHI
JNEN."
150 PRINT "C: INDIEN HET BEELDSCHERM VOL
IS, ZAL HET BEELD NIET SCROLLEN, M
AAR ZAL HET SCHERM DOOR DE EERDER
VERMELDE MONITOR ROUTINE GEWIST WOR
DEN."
160 INPUT "DRUK OP RETURN VOOR MEER INFO
RMATIE ";D$
170 PRINT CHR$(12);"HET NU VOLGENDE PROG
RAMMA WORDT OP TAPE GEZET."
180 PRINT " EI ;MAAK INTER
RUPT AFHANDELIN
G MOGELIJK"
190 PRINT " PUSH HL ;RED DE REG
ISTERS"
200 PRINT " PUSH BC ;"
210 PRINT " PUSH DE ;"
220 PRINT " PUSH AF ;"
230 PRINT "WIS$: CALL 123CH;WIS HET SC
HERM"
240 PRINT "GETK$: RST 38H ;SCAN TOETS
ENBORD
250 PRINT " LD A,(600DH);LAAD TOE
TSCODE "
260 PRINT " CP FFH ;TOETS INGE
DRUKT?"
270 PRINT " JR Z,GETK$;NEE
280 PRINT " CP 58H ;STOP TOETS
?
290 PRINT " JR Z,STOP$;JA, STOP
DAN"

```

```

300 PRINT "      CALL 1465H;ZET TOETSC
ODE                OM IN ASCI
I-CODE"
310 PRINT "      CALL 10EBH;ZET HET AS
CII                KARAKTER O
P DE              JUISTE PLA
ATS OP           HET SCHERM
330 INPUT "DRUK OP RETURN VOOR MEER INFO
RMATIE";D$
340 PRINT CHR$(12);"      CALL 1362H;
LAAD KURSORLOKATIE
IN HL-REGISTER"
350 PRINT "      LD DE,5757
H;HOOGSTE ADRES OP
HET BEELD
360 PRINT "      SBC HL,DE;
KURSOR VOORBIJ HET
EINDE VAN HET BEELD?"
370 PRINT CHR$(4);CHR$(7);CHR$(1);"
JR NC,GETK$;NEE, HAAL VOLGEND
KARAKTER "

380 PRINT "      JR WIS$; J
A, WIS HET SCHERM
390 PRINT "STOP$; POP AF;HERL
AAD DE REGISTERS"
400 PRINT "      POP DE;"
410 PRINT "      POP BC;"
420 PRINT "      POP HL;"
430 PRINT "      RET ;TERU
G NAAR BASIC"
440 INPUT "DRUK OP RETURN VOOR MEER INFO
RMATIE";D$
450 PRINT CHR$(12);
460 PRINT "NU VOLGT DE UITLEG VAN HET MA
CHINE- TAAL PROGRAMMA, DAT HET PROG
RAMMA OP TAPE PLAATST."
470 PRINT " LD A,00H; INITIALISEER DE
CASSETTE RECORDER "
480 PRINT " CALL 0018H;ROEP DE CASSETT
E FUNKTIE AAN "
490 PRINT " LD A,01H; SPOEL DE CASSETT
E TERUG NAAR DE BEGINSTA
ND "
500 PRINT " CALL 0018H;ROEP DE CASSETT
E FUNKTIE AAN "
510 PRINT " LD A,05H; SCHRIJF HET MACH
INE-TAAL PROGRAMMA OP DE
CASSETTE "
520 PRINT " CALL 0018H;ROEP DE CASSETT
E FUNKTIE AAN "
530 PRINT " RET ;TERUG NAAR BASIC

```

```

1000 CLEAR 200,&H9800:REM MAAK GEHEUGEN
VRY VOOR DE MACHINE-TAAL ROUTINE
1010 REM NU VOLGT HET MACHINETAAL PROGRA
MMA DAT OP TAPE GEZET WORDT.
1020 DATA 251,229,197,213,245,205,60,18,
255,58,13,96,254,255,40,248,254,88,40,18
,205,101,20,205,232,16,205,98,19,17,87,8
7,237,82,56,228,24,223,241,209,193,225,2
01
1030 FOR J=&H9800 TO &H982A: READ AZ:POK
E J,AZ:NEXT : REM POKE DE MACHINE-TAAL R
OUTINE IN HET GEHEUGEN
1040 POKE &H6034,46:POKE &H6035,0: REM L
ENGTE VAN DE FILE
1050 POKE &H6036,80: REM NAAM VAN DE FIL
E 'P'
1060 POKE &H6041,80: REM DIT HOUDT IN :M
ACHINE TAAL-PROGRAMMA
1070 POKE &H6043,0:POKE &H6044,152:REM S
TART ADRES VAN HET MACHINE-TAAL PROGRAMM
A
1080 POKE &H6045,0:POKE &H6046,152:REM L
AAD ADRES VAN DE MACHINE-TAAL ROUTINE
1090 POKE &H6032,0:POKE &H6033,4: REM LE
NGTE VAN DE BAND
1100 POKE &H6030,0:POKE &H6031,152: REM
DATA OVERDRACHT ADRES
1110 DEFUSR0=&H9900
1120 REM NU WORDT HETMACHINE-TAAL PROGRA
MMA DAT DE TAPE GAAT SCHRIJVEN, IN HET G
EHEUGEN GEPOKED
1130 DATA 62,0,205,24,0,62,1,205,24,0,62
,5,205,24,0,201
1140 FOR J=&H9900 TO &H990F:READ AZ:POKE
J,AZ:NEXT
1150 PRINT "DRUK OP RETURN-TOETS INDIEN
EEN LEGE TAPE GELADEN IS";
1160 INPUT D$
1165 PRINT CHR$(12);
1170 DEFUSR1=&H385 : REM DIT IS HET STAR
T ADRES IN DE MONITOR OM EEN MACHINE-TAA
L PROGRAMMA TE LATEN LADEN EN STARTEN
1180 DATA 70,117,38,11,39,46,126,46,113,
52: REM I=USR1(1)
1185 POKE &H600F,0
1190 FOR J=&H6000 TO &H6009:READ AZ:POKE
J,AZ:NEXT
1205 I=USR0(1): REM ZET HET PROGRAMMA OP
TAPE "
1207 POKE &H600C,10:REM 10 TOETSCODES IN
INPUT-BUFFER
1210 END

```

Het volgende programma biedt de mogelijkheid om string-arrays naar cassette te schrijven en weer terug in te lezen onder de naam waaronder het array tijdens de executie van een programma is geadministreerd. De methode volgens welke dit kan gebeuren is vooral bruikbaar wanneer het om grote arrays gaat, zoals dat bijvoorbeeld gebeurt bij PIMS (Personal Information Management System, Scelbi).

Eerst iets over strings zoals deze in Microsoft Basic worden gebruikt. In tegenstelling tot arrays van het type 'integer', 'real' en 'double precision', wordt de inhoud van string-arrays (mits niet als 'literal' gebruikt), indirect geadresseerd.

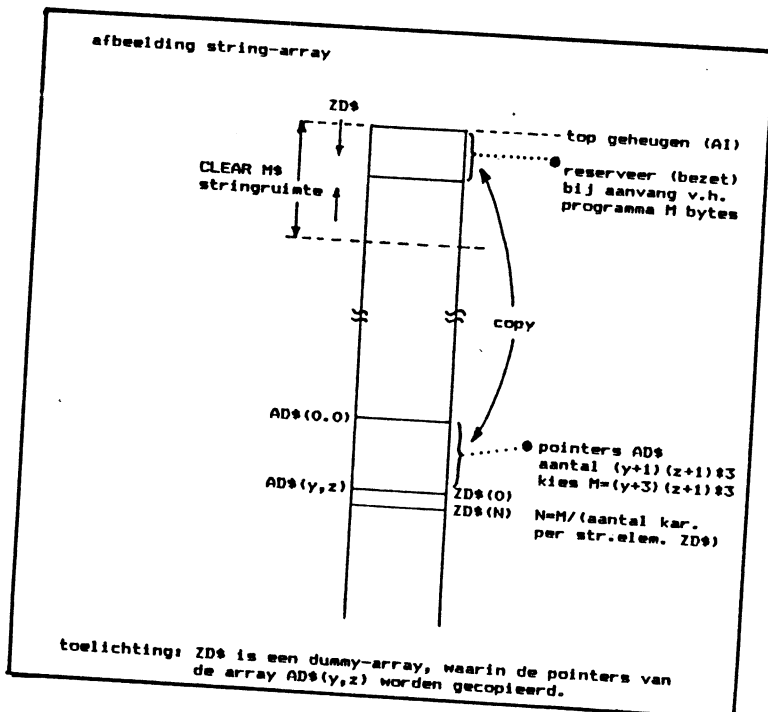
Elk stringarray-element wordt in Basic door een 3-byte pointer gedefinieerd, namelijk een byte welke het aantal karakters van het stringelement aangeeft (LEN(string)) en twee bytes die samen een adres in de algemene stringruimte (te definiëren met CLEAR) vormen.

De methode die hier wordt gevolgd copieert deze pointers in een daarvoor gereserveerd gebied van de stringruimte. Bij het wegschrijven naar cassette worden de strings met de pointers als een aaneengesloten stroom van bytes naar de cassette gecopieerd. Bij het weer inlezen van de gegevens komen de strings weer in het stringgebied terecht en na het teruglezen worden de pointers weer op hun plaats teruggezet.

In het 16K-Basic systeem bestaan twee routines die de hele cassette-data-behandeling voor hun rekening nemen, nl. CALL 157B voor CSAVE en CALL 169D voor CLOAD. Bij het aanroepen van een file dienen in register A de filenaam en in de registers HL en DE de lengte van de file en het aantal bytes te worden geschreven c.q. gelezen.

Een en ander is schematisch weergegeven in de hierbij opgenomen afbeelding. Tevens is een voorbeeld opgenomen in de vorm van een programmalingsting. Dit programma kan in een eigen programma worden toegepast.

Het programma PIMS is aangepast voor de P2000 en opgenomen in de TAPOTHEEK.



Ted Jonker - Naarden.

```

0 REM voorbeeld voor setup van string in
voer/uitvoer op cassette + invoer en uit
voer routine
1 CLEAR 4750:MS=4750
2 DATA 3,40 :REM invoer dimensie van str
ing array AD$(40,3)
3 READ NK,NR: RESTORE 7: Z=NK
5 DIM ZD$(NK):FOR I=0 TO NK:ZD$(I)=SPACE
$(NR+3)*3:NEXT:REM plaats voor pointer
s -> 3 geheugen plaatsen per string el
ement, met wat extra ruimte.
6 DIM AD$(NR,NK)
7 DATA 3A,70,60,2A,72,60,ED,5B,74,60,CD,
9D,16,C9: REM Setup for DSAVE and CLOAD
8 FOR I=&H6076 TO &H60B3: READ J#: POKE
I,VAL("&H"+J#): NEXT I:DEF USR0=&H6076
9 AI=0:I=0:I=VARPTR(ZD$(0)): IF I<0 THEN
I=I+65536
10 AI=PEEK(I)+PEEK(I+1)+PEEK(I+2)*256-1:
REM AI is een vaste waarde, maar verande
rt bij iedere geheugen configuratie. AI
dus vastleggen indien andere configurati
e bij opname en weergave van de array AD
$.
11 REM Print de waarde van AI ix bij bv.
16K RAM; breng deze waarde vast in prog
ramma .Bij een groter geheugen reserveer
meer stringruimte (CLEAR MS+...) .
890 REM
891 REM -----
-----*Laden vanaf cassette
900 PRINT CHR$(12): INPUT "Geef filenaam
": F#
910 FOR J=0 TO NK:FOR I=0 TO NR:AD$(I,J)
=AD$(I,J): NEXT I: NEXT J

```

```

920 POKE &H60B1,&H7B: POKE &H60B2,&H15:
REM call 157B DSAVE
930 POKE &H6070,ASC(F#)
940 IM=0:IB=0:IA=0:I=0:L=0
950 IM=VARPTR(AD$(0,0)): IF IM<0 THEN IM
=IM+65536
955 PRINT "Even geduld."
960 IB=AI-(NR+3)*(NK+1)*3:FOR I=IB TO AI
:POKE I,PEEK(IM):IM=IM+1:NEXT
970 IA=AI-MS+1: IB=&H6072: GOSUB 4000
980 IA=AI: IB=&H6074: GOSUB 4000
990 I=USR0(0)
999 RETURN
1000 REM-----
-----*Laden vanaf cassette.
1010 REM
1020 PRINT CHR$(12):INPUT "Geef filenaam
":F#
1030 POKE &H60B1,&H7D:POKE &H60B2,&H16:
REM CALL 169D ->Lezen
1040 POKE &H6070,ASC(F#)
1050 IM=0:IB=0:IA=0:I=0:L=0:
REM Initialiseer i.v.m. VARPTR
1060 IA=AI-MS+1: IB=&H6072: GOSUB 4000
1070 IA=AI: IB=&H6074: GOSUB 4000
1080 I=USR0(0): IM=VARPTR(AD$(0,0)): IF
IM<0 THEN IM=IM+65536
1085 PRINT "Even geduld."
1090 IB=AI-(NR+3)*(NK+1)*3: FOR I=IB TO
AI: POKE IM,PEEK(I): IM=IM+1: NEXT I
1100 FOR I=0 TO NK: FOR J=0 TO NR: AD$(I
,J)=AD$(J,I)+"":NEXT:NEXT
1105 RETURN
4000 REM
4001 L=INT(IA/256): POKE IB,INT(IA-L*256
): POKE IB+1,L
4002 RETURN

```

P2000

PRINTERMENU VOOR DE MX-80

Onderstaand programma van Marcel Bruyns biedt een gemakkelijke manier om de MX-80 of Philips matrix printer in te stellen. Het menu kan ook als subroutine in een programma verwerkt worden.

```

920 PRINTCHR$(12):REM PRINTERMENU MX 80
930 REM CANCEL CHARACTER CODES EN LINE F
EED
940 LPRINTCHR$(20):CHR$(18):CHR$(27):CHR
$(70):CHR$(27):CHR$(72):CHR$(27):CHR$(65
):CHR$(12):CHR$(27):CHR$(50):POKE24747,
80
950 REM KEUZE CHARACTERS
960 Z#="":INPUT " 1 NORMALE LETTERS. 2.
SMALLE LETTERS.":Z#:PRINT:IFZ#="2"THENL
PRINTCHR$(15):
970 Z#="":INPUT "DUBBELE BREEDTE J/N":Z#:
PRINT:IFZ#="J"THENLPRINTCHR$(14)
980 Z#="":INPUT "VETGEDRUKT J/N ":Z#:PRIN
T:IFZ#="J"THENLPRINTCHR$(27)CHR$(69)

```

```

990 Z#="":INPUT "DUBBEL PRINTEN J/N ":Z#:
PRINT:IFZ#="J"THENLPRINTCHR$(27)CHR$(71)
1000 Z#="80":INPUT "REGELLENSTE 132,80,66
,40,OF SMALLER":Z#:PRINT:POKE24747,VAL(Z
#)
1010 Z#="72":INPUT "AANTAL REGELS PER FOR
MULIER (normaal=72 of 66)":Z#:LPRINTCHR$(
27)CHR$(67)CHR$(VAL(Z#)):
1020 PRINT:PRINT "keuze spatie groote":PR
INT"1. 4.1em (normaal)":PRINT"2. 3.12 m
m":PRINT"3. 2.43 mm":PRINT"4. ZELF INSTE
LLEN n X 0.35 MM":Z#="1":INPUT "HOEVEEL K
EER 0.35 MM":Z#:LPRINTCHR$(27)CHR$(65)CH
R$(VAL(Z#))CHR$(27)CHR$(50):

```

*** Gebruiksaanwijzing ***

Een programma moet van een duidelijke gebruiksaanwijzing voorzien zijn, waarin de bedoeling van het programma wordt uiteengezet en hoe ermee gewerkt moet worden.

Deze gebruiksaanwijzing moet in het programma zelf of in een apart programma op dezelfde cassette gezet kunnen worden. Liever geen gebruiksaanwijzing op papier in verband met distributieproblemen.

Speciale programma's, die van de gebruiker de nodige voorkennis vereisen, moeten toch zoveel aanwijzingen bevatten dat een niet-ter-zake-kundige het programma minstens kan laten lopen, al zal hij er dan geen optimaal gebruik van kunnen maken.

*** Gebruikersvriendelijkheid ***

Een programma moet 'gebruikersvriendelijk' zijn, een mooi woord voor de vraag of de maker van het programma zich verplaatst heeft in de positie van de doorsnee gebruiker van zijn programma. Om maar een paar kleine dingen te noemen:

- kan de gebruiker zowel kleine als grote letters gebruiken om toch de reactie te krijgen die hij verwacht?;
 - is het aantal instructies en het typewerk beperkt gehouden?;
 - wordt de returntoets onnodig of inconsequent gebruikt?;
 - bestaat onnodig risico voor het verlies van ingevoerde gegevens door kleine invoer- of bedieningsfouten?;
- En zo zouden we nog even door kunnen gaan.

*** Invoerbeveiliging ***

Het programma dient bestand te zijn tegen verkeerde invoer. Dat wil zeggen dat het programma niet mag afbreken als de gebruiker per ongeluk een verkeerde invoer geeft, bijvoorbeeld een letter in plaats van een cijfer, of een getal dat groter, kleiner of anders is dan het programma verwacht. Zorg voor goed tegen dit soort zaken beveiligde invoerroutines, waardoor ongewenst afbreken van het programma wordt voorkomen.

*** Situatie herstellen ***

De 'gevorderde' programmamaker heeft al gauw behoefte aan het inbouwen van allerlei fraaie voorzieningen, die met behulp van POKE's in het programma worden ingebouwd. Denk maar eens aan het weghalen van de cursor. Ook het gebruik van CLEAR-statements verandert zaken in het geheugen, die niet automatisch hersteld worden als het programma wordt beëindigd. Dat kan onaangename verrassingen opleveren, wanneer daarna een ander programma gerund wordt. Zorg dus dat dergelijke veranderingen hersteld wordt aan het einde van een programma.

*** STOP-functie ***

Met het herstellen van de situatie hangt samen, dat er in het programma een STOP-functie wordt ingebouwd. Het afbreken van een programma met de shift/STOP-toets heeft immers tot gevolg dat het herstellen van de situatie bij normaal beëindigen van het programma niet plaatsvindt!

We zullen in de toekomst zeker nog terugkomen op dit soort zaken, die de bruikbaarheid en de kwaliteit van programma's enorm kunnen verbeteren.

PRINT CHR\$(5)

De instructie PRINT CHR\$(5) kan worden gebruikt om een pagina die op het beeldscherm staat af te drukken op papier. Je kunt hiervan gebruik maken om bijvoorbeeld de gebruiksaanwijzing die in een programma is opgenomen even uit te printen op papier. Dit kan vaak eenvoudig worden gerealiseerd door de PRINT CHR\$(5)-instructie in te bouwen in de routine voor het 'bladeren' naar de volgende pagina.

Het P2000-viewdata-programma kan hiervoor bijvoorbeeld als volgt worden aangepast:

```
95 IFPEEK(24588)=0THEN95ELSEPOKE24588,0:
  IFPEEK(24576)=90THENPRINTCHR$(5);:
  GOTO95ELSEPOKE&H6013,1:RETURN
```

De gebruiker kan nu door in plaats van op de spatiebalk op de printertoets (shift 00) te drukken een keurige afdruk van de betreffende beeldschermpagina op papier afgedrukt krijgen alvorens de volgende pagina wordt getoond.

CLEAR

Met het CLEAR-statement kun je geheugenruimte reserveren om extra tekst en/of korte machinetaalroutines op te bergen. Normaal zijn voor tekst 50 geheugenposities gereserveerd. Is er meer stringruimte nodig dan gereserveerd is, dan volgt een foutmelding: 'Out of string space'. Door meer ruimte te reserveren wordt dit voorkomen. CLEAR 200,&H97FF bijvoorbeeld reserveert geheugenruimte voor 200 posities tekst en voor &H800 bytes machinetaal. Als je de geheugenruimte aanpast moet je deze aan het eind van het programma weer op 50 posities terugbrengen met de instructie CLEAR 50,&H9FFF. Je voorkomt daardoor dat het volgende programma dat gerund wordt 'raar' gaat doen. Om problemen te voorkomen is het ook niet onverstandig deze instructie altijd aan het begin van je eigen programma's op te nemen. (uit 'Philitel')

GET-ROUTINE VOOR GETALLEN

Wanneer een programma tijdens de uitvoering vaak een getalleninput nodig heeft, kan het vervelend zijn om steeds de ENTER-toets te moeten indrukken.

De zogenaamde GET-subroutine is dan een handige oplossing wanneer het gaat om getallen van 0 - 9. Voor grotere getallen moet iets meer gedaan worden. Eerst moet de code van elke afzonderlijke toetsindruk worden omgezet in de normale waarde. Het getal wordt daarna verkregen door deze waarden samen te voegen.

We geven een voorbeeld voor getallen van 0 - 99 :

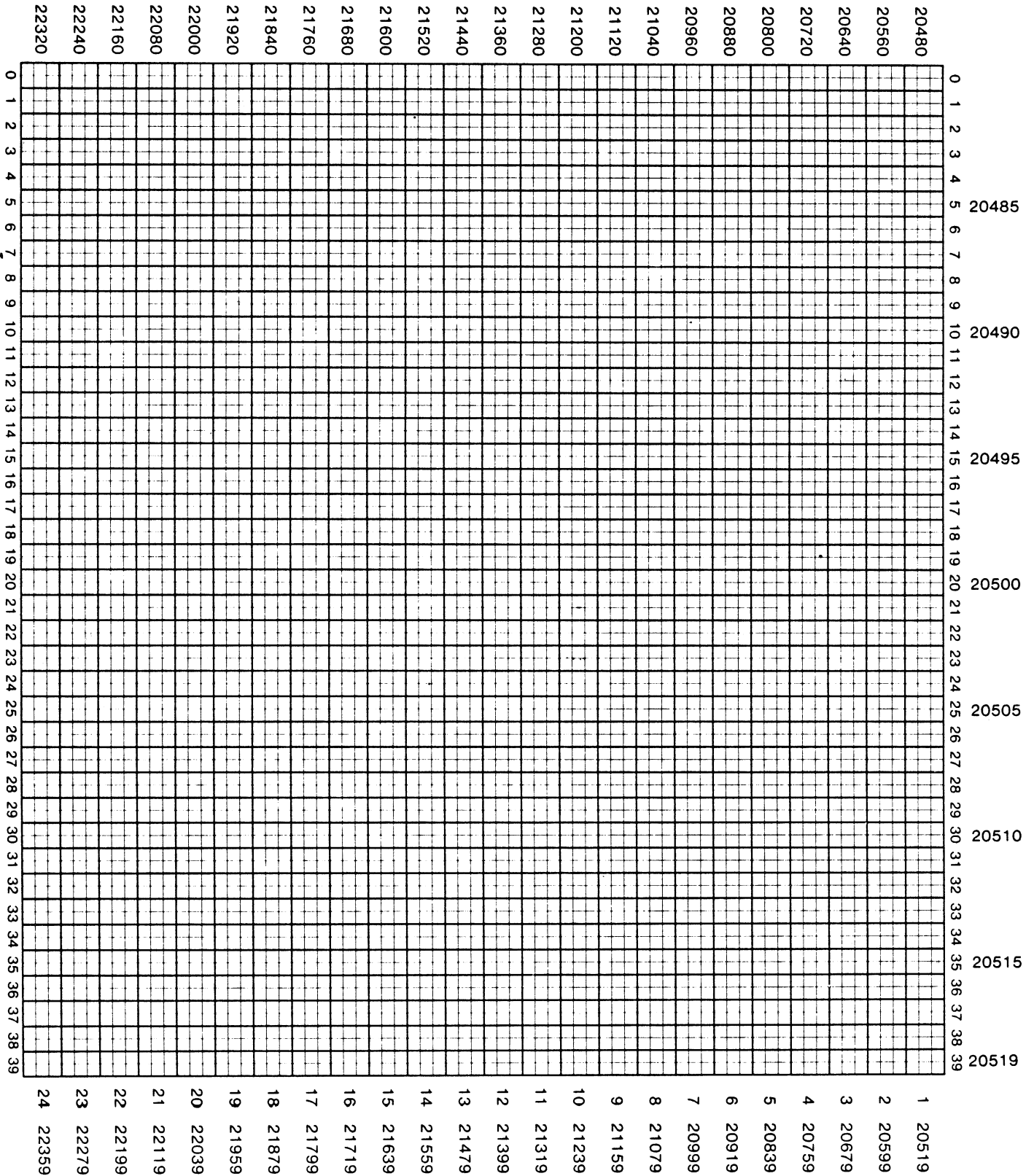
```
10 POKE 24588,0
20 IF PEEK(24588)=0 THEN 20 ELSE G=PEEK(24576)
30 POKE 24588,0:RETURN
```

```
100 GOSUB 10 : REM Normale subroutine
110 T=VAL(CHR$(PEEK(6164+G)))
120 GOSUB 10
130 E=VAL(CHR$(PEEK(6164+G)))
140 X=10*T+E
```

Een klein nadeel is, dat voor getallen onder de 10 toch twee cijfers moeten worden ingetypt, bv. 07. **21**



Op deze pagina vind je een raster voor het ontwerpen van een layout voor het beeldscherm. Langs de randen zijn de horizontale en verticale posities, en de geheugenadressen van de posities in de video-ram weergegeven (voor het poken van karakters). Gebruik het raster als 'origineel' waarvan je regelmatig naar behoefte nieuwe copieën kunt maken.



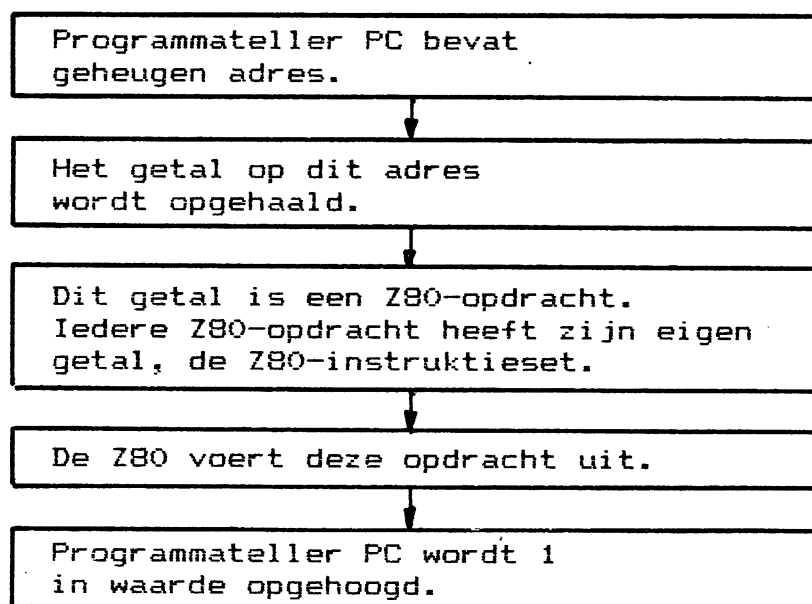
Onder deze titel is op de afgelopen bijeenkomst van 9 januari iets verteld over het programmeren in Z80-machinetaal vanuit BASIC. Een aantal mensen hebben mij gevraagd dit op te schrijven voor de nieuwsbrief zodat ze het nog eens rustig konden nalezen. Daaraan voldoe ik bij deze graag.

Waarom is het programmeren in machinetaal interessant? Het enigste voordeel van programmeren in machinetaal t.o.v. BASIC is de veel grotere snelheid die bereikt kan worden. Deze grotere snelheid is nodig om bijv. muziek te kunnen maken of voor snelle input/output routines zoals Basicode en Viewdata. Er zijn legio nadelen verbonden aan machinetaal zoals onoverzichtelijkheid, geen foutmeldingen en geen basisroutines als vermenigvuldigen en delen.

De beste start om Z80-machinetaal te leren is het kopen van een goed boek. Het boek dat mij uitstekend bevallen is, is "Programming the Z80" geschreven door Rodney Zaks en uitgegeven bij Sybex. Het boek kost ongeveer 50 gulden en is ook in het nederlands te krijgen. Een goedkopere uitgave van dit boek wordt verkocht in de Tandy computer shops. Het boek "Programming the Z80" bestaat uit 3 gedeelten. Het eerste deel behandelt de elementaire onderdelen als binair rekenen, de opbouw van de Z80 en de verschillende groepen instructies. Het tweede deel is een uiterst waardevolle encyclopedie van alle Z80-instructies en het derde deel geeft voorbeelden van Z80-machinetaal-programma's.

Verwijzend naar dit boek volsta ik hier met de uitleg dat de Z80 microprocessor is opgebouwd uit een aantal 8 bits registerparen die worden aangegeven met A/F, B/C, D/E en H/L. Daarnaast zijn er vier 16-bits registers te weten IX, IY, stackpointer SP en programmateller PC. Al deze registers zijn verbonden met geheugen, I/O poorten etc. van de P2000.

Een belangrijk onderdeel van de Z80 is het gedeelte dat zorgt voor de uitvoering van de opdrachten. Dit onderdeel is verbonden met het geheugen. De werking van de Z80microprocessor kan nu met het volgende stroomschema duidelijk gemaakt worden:



Twee voorbeelden uit de Z80-instructieset zijn:

- 5: Verlaag de waarde in register B met 1.
- 74: Copieer inhoud register B in register C.

Stel dat we nu een programma in Z80-machinetaal hebben geschreven, hoe moeten we dat dan laten uitvoeren door de P2000. Het is mogelijk om vanuit BASIC een machinetaalprogramma te laten uitvoeren. Daarvoor gaan we als volgt te werk:

- 1) Geef BASIC de opdracht om een deel van het geheugen niet te gebruiken. Dit gaat met de opdracht

```
CLEAR 50,40191.
```

In deze opdracht geeft het getal 50 aan hoeveel stringruimte gereserveerd wordt. Mocht U ooit de foutmelding "Out of string space" krijgen, verhoog dan dit getal 50 totdat de foutmelding niet meer terugkomt.

Het getal 40191 is een geheugen adres. Hiermee krijgt BASIC de opdracht om alle geheugenlokatie tussen adres 40191 en de hoogste geheugenlokatie die aanwezig is, niet te gebruiken. Hier kan nu dus het machinetaalprogramma neergezet worden zonder dat het overschreven wordt.

- 2) Plaats de machinetaalinstructies op de goede plaats in het geheugen. Vanuit BASIC gaat dit het handigste als U een BASIC-programma schrijft waarin de machinetaalinstructies in DATA-regels geplaatst worden. Met een eenvoudige lus leest U dan steeds een instructie en plaatst deze met een POKE opdracht op de goede geheugenlokatie. Voorbeeld in pseudo-BASIC:

```
DATA .....,.....  
  adres=40192  
FOR I=1 TO (aantal instructies in machinetaal)  
  READ instructie  
  POKE adres,instructie  
  adres=adres+1  
NEXT I
```

- 3) Machinetaalprogramma laten uitvoeren.

Om het Z80-machinetaalprogramma dat nu vanaf geheugenlokatie 40192 in het geheugen staat te laten uitvoeren maken we gebruik van de USR-functie.

- a) Als eerste moet het startadres van de machinetaal routine vastgelegd worden. In dit geval is het startadres 40192.

Opdracht:

```
DEF USR = 40192
```

- b) Uitvoeren gaat nu met de opdracht:

```
v = USR(0)
```

In deze opdracht is v een willekeurige variabelenaam. Na deze opdracht springt BASIC naar geheugenadres 40192 en begint het daar geschreven programma uit te voeren.

- 4) Terug naar BASIC.

U keert terug naar BASIC door aan het einde van uw machinetaal programma de instructie 201 of &H C9 op te nemen. In Z80-machinetaal is dit de instructie voor RETURN. Het BASIC programma gaat dan verder met de eerste opdracht na v = USR(0).

Tot slot van dit verhaal een aantal praktische tips:

- * Werkwijze:
 - BASIC programma intypen
 - Op cassette wegschrijven.
 - Cassette verwijderen (!!!). Als er verkeerde sprongen in uw programma staan kunt U in de P2000 cassette routines terecht komen die uw cassette kunnen gaan wissen.
 - RUN

* Bij het ontwikkelen van een machinetaalprogramma eerst het disassemblerprogramma laden. Daarachter dan uw BASIC-programma. U kunt dan in de veel gemakkelijker leesbare Z80-mnemonics lezen wat U in het geheugen geschreven heeft.

* De STOP-toets werkt niet tijdens de uitvoering van een machinetaalroutine. Een op-hol-geslagen machinetaalprogramma is alleen met de RESET-toets tot de orde te roepen. U moet dan echter wel weer helemaal van voren af aan beginnen met inlezen van het programma, vandaar de raad om het programma altijd eerst op cassette te schrijven.

Als laatste volgt hier een listing van een voorbeeld programma waarin al het voorgaande geïllustreerd wordt. Het programma vult het beeldscherm met de letters Z van Z80, eerst in machinetaal en daarna in BASIC zodat U het verschil in snelheid kunt beoordelen.

Gerben Mooiweer
Martinusgaarde 26
3436 RE Nieuwegein.

```

10          REM
           Voorbeeld machinetaal programmeren.
           Gemaakt door G.Mooiweer, jan. 1982
20 PRINT CHR$(12)
30 PRINT CHR$(4);CHR$(22);CHR$(1)
40 PRINT TAB(6);"LET OP !! Eerst in mach
   inetaal."
50 CLEAR 200,&H9CFF
60 REM Deze opdracht zorgt ervoor dat
   basic geen gebruik maakt van de
   geheugenplaatsen boven &H9CFF
70 REM Ons machinetaal programma wordt
   weggeschreven vanaf &H9D00.
80 REM We beginnen met het in het
   geheugen zetten van het Z80-machine
   taal programma.
90 REM Startadres is hexadecimaal 9D00
           decimaal 40192.

100 A=&H9D00
110 READ IN$
120 IF IN$="XX" GOTO 240
130 REM Met XX markeer ik het einde
   van het inlezen.
140 CD=VAL("&H"+IN$)
150 REM De DATA-regels bevatten de
   machinetaal codes in hexadecimale
   vorm, echter niet de vorm waarin de
   P2000 hexadecimale getallen herkent
   Via opdracht 3100 worden ze omgezet
   in het equivalente getal.
160 POKE A,CD
170 A=A+1
180 GOTO 110
190 DATA 06,18,0E,50,11,00,50,3E,5A
200 DATA 12,13,0D,C2,09,9D
210 DATA 0E,50,05,C2,09,9D

```

```

220 DATA C9
230 DATA XX
240 REM Het machinetaal programma staat
   nu in het geheugen en kan aange-
   roepen worden.
250 REM Eventueel kunt U de disassem-
   bler aan dit programma koppelen
   (zie P2000gg cassette nr.1 kant B)
   en controleren.
260 REM Het aanroepen van dit machine-
   taal programma gaat met de USR_
   functie waarbij _ een getal tussen
   0 en 9 is, bijv USR1,USR5 enz.
270 REM Eerst definiëren we het start-
   adres van de USR functie, in dit
   geval heb ik USR8 genomen.
280 REM
290 DEF USR8=&H9D00
300 REM
310 REM Nu kunnen we door deze USR8
   in een BASIC opdracht aan te roepen
   het machinetaal programma laten
   uitvoeren.
320 REM
330 D=USR8(0)
340 REM Het hele beeldscherm staat nu
   vol met Z's (van Z80). Tot slot nog
   een eenvoudige vertraging om het
   resultaat rustig te kunnen bekijken
350 FOR I=1 TO 2000:NEXT I
360 PRINT CHR$(12)
370 PRINT CHR$(4);CHR$(22);CHR$(1)
380 PRINT TAB(6);"En nu in basic...."
390 FOR I=&H5000 TO &H577F
400 POKE I,90
410 NEXT I
420 FOR I=1 TO 2000:NEXT I
430 REM Zet de P2000 terug in zijn uit-
   gangstoestand.
440 PRINT CHR$(12)
450 CLEAR 50,&H9FFF
460 END

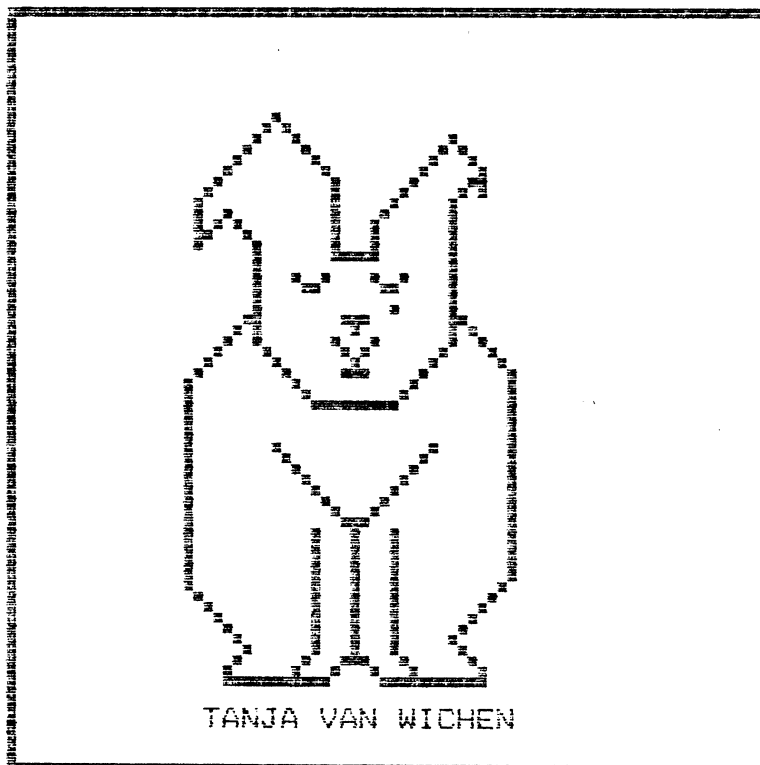
```

Het volgende korte programma van Ted Jonker uit Naarden geeft een lijst van adressen in het geheugen, waar de uitvoerroutines voor diverse instructies beginnen.

```

10 REM SOME 16K-BASIC ENTRY POINTS (P200
066 TED JONKER NAARDEN)
15 CLEAR:A=8620:B=8220:C=8427:PRINTCHR$(
12)
20 LPRINT "FUNCT";TAB(10);":"TAB(12);"AD
DRESS":LPRINT
25 FORX=BTOC:Y=PEEK(X):IFY>127 THEN Y=Y-
128
30 Z=X+1:IFPEEK(Z)>127THEN LPRINTCHR$(Y)
;TAB(10);":";ELSEGOTO45
35 A=A+2
40 LPRINTTAB(12);HEX$(PEEK(A+1)*256+PEEK
(A)):GOTO50
45 LPRINT CHR$(Y);
50 NEXT
60 IF F=-1THEN GOTO70ELSEA=8156:B=8509:C
=8619:F=-1:GOTO25
70 REM *END PROGRAM *M *
    
```

FUNCT	: ADDRESS
END	: 1D33
FOR	: 272C
NEXT	: 1E98
DATA	: 293A
INPUT	: 2BF1
DIM	: 4791
READ	: 2C27
LET	: 295F
GOTO	: 28F4
RUN	: 28D4
IF	: 2AA4
RESTORE	: 1D16
GOSUB	: 28E3
RETURN	: 2912
REM	: 293C
STOP	: 1D31
CONSOLE	: 32CD
WIDTH	: 32D0
ELSE	: 293C
TRON	: 1D96
TROFF	: 1D97
SWAP	: 1D9C
ERASE	: 1DE4
DEFSTR	: 284D
DEFINT	: 2850
DEFSNG	: 2853
DEFDBL	: 2856
LINE	: 2BB4
EDIT	: 1A48
ERROR	: 2A63
RESUME	: 2A19
OUT	: 32A9
ON	: 29CC
WAIT	: 32AF
LPRINT	: 2ADE
DEF	: 30E5
POKE	: 3411
PRINT	: 2AE7
CONT	: 1D82
LIST	: 3336
LLIST	: 3331
DELETE	: 33D5
AUTO	: 2A6E
CLEAR	: 1E4C
CLOAD	: 4E7B
CSAVE	: 4E5D
NEW	: 1C8B
etc.	



Van Maarten Groosman uit Hengelo ontvingen wij een aantal korte programma's voor het M-model, en een beeldschermroutine voor het M- en T-model.

P2000 gg

INVOER-ROUTINES

Met de routine die start op 5100 kan je decimale getallen invoeren. Op 5300 start de routine voor invoer van integer getallen, op 5400 voor strings (alleen tekens) en op 5500 invoer van strings (alleen letters en spaties).

Het voorbeeld programma roept achtereenvolgens de 4 routines aan, waarbij Q de lengte van de string is, D de min. waarde en E de max. waarde.

Terugkeer naar het hoofdprogramma volgt pas wanneer een getal of string binnen de grenswaarden is ingetypt.

```

10 D=5:E=10:Q=5:GOSUB 5100:PRINT:PRINT:G
OSUB 5300:PRINT:PRINT:GOSUB 5400:PRINT:P
RINT:GOSUB 5500:STOP
5100 REM 'SUBDECIM' A40, INPUT ROUTINE DE
CINALE GETALLEN, D(<G<=E, MAX. LENGTE Q
5104 REM BEKEND:Q(MAX. LENGTE VAN G%),D(
MIN. WAARDE),E(MAX. WAARDE)
5108 G$="":T=0:IFQ=0 THEN Q=5
5112 T$=INKEY$:IFT$=""THEN GOTO 5112 ELS
E IF ASC(T$)=13 THEN GOTO 5132
5116 IF ASC(T$)=8 AND LEN(G%)>0 THEN PRI
NT CHR$(8);CHR$(132);CHR$(8);:Z=LEN(G%):
IF Z=0 THEN PRINT CHR$(7);:GOTO 5112 ELS
E IF ASC(T$)=8 THEN T$=MID$(G%,Z,1):IF A
SC(T$)=46 THEN T=0:G$=MID$(G%,1,Z-1):GOT
O 5112 ELSE G$=MID$(G%,1,Z-1):GOTO 5112
5120 IF ASC(T$)=46 THEN IF T=0 THEN T=1:
GOTO 5128 ELSE PRINTCHR$(7);:GOTO 5112
5124 IF ASC(T$)<48 OR ASC(T$)>57 THEN PR
INTCHR$(7);:GOTO 5112
5128 G$=G$+T$:IF LEN(G%)>Q THEN PRINTCHR
$(7);:Z=LEN(G%):G$=MID$(G%,1,Z-1):GOTO51
12 ELSE PRINTT$;:GOTO 5112
5132 G=VAL(G%):IF G<D THEN PRINTCHR$(7);
:GOTO 5112
5136 IFG>E THEN PRINTCHR$(7);:GOTO 5112
ELSE RETURN

```

```

5300 REM 'SUBINTEG' A40, INPUT ROUTINE GE
HELE GETALLEN, D(<G<=E, MAX. LENGTE Q
5310 REM BEKEND:Q(MAX.LENGTE G%),D(MIN.
WAARDE) EN E(MAX. WAARDE)
5320 G$="":IF Q=0 THEN Q=5
5330 T$=INKEY$:IF T$="" THEN GOTO 5330 E
LSE IF ASC(T$)=13 THEN GOTO 5370
5340 IF ASC(T$)=8 AND LEN(G%)>0 THEN PRI
NTCHR$(8);CHR$(132);CHR$(8);:Z=LEN(G%):I
F Z=0 THEN PRINTCHR$(7);:GOTO 5330 ELSE
G$=MID$(G%,1,Z-1):GOTO 5330
5350 IF ASC(T$)<48 OR ASC(T$)>57 THEN PR
INTCHR$(7);:GOTO 5330
5360 IF LEN(G%)>Q-1 THEN PRINTCHR$(7);:G
OTO 5330 ELSE G$=G$+T$:PRINT T$;:GOTO 53
30

```

```

5370 G=VAL(G%):IF G<D THEN PRINTCHR$(7);
:GOTO 5330
5380 IF G>E THEN PRINTCHR$(7);:GOTO5330
ELSE RETURN
5400 REM 'LENSTRIN' INPUT ROUTINE STRING
S, MAX. LENGTE Q
5405 G$="":IF Q=0 THEN Q=5:REM BEKEND:Q(
MAX. LENGTE VAN G%)
5410 T$=INKEY$:IF T$="" THEN GOTO 5410 E
LSE IF ASC(T$)=13 THEN GOTO 5435
5415 IF ASC(T$)=8 AND LEN(G%)>0 THEN PRI
NTCHR$(8);CHR$(132);CHR$(8);:Z=LEN(G%):G
$=MID$(G%,1,Z-1):GOTO 5410
5420 IF ASC(T$)=8 AND LEN(G%)=0 THEN PRI
NTCHR$(7);:GOTO5410
5425 IF ASC(T$)=32 THEN PRINTCHR$(7);:GO
TO 5410
5430 IF LEN(G%)>Q-1 THEN PRINTCHR$(7);:G
OTO5410
5432 G$=G$+T$:PRINTT$;GOTO5410
5435 RETURN
5500 REM 'LENLETSP' INPUT ROUTINE STRING
S,MAX. LENGTE Q, ALLEEN LETTERS EN SPATI
ES
5505 G$="":IF Q=0 THENQ=5:REM BEKEND:Q(M
AX. LENGTE VAN G%)
5510 T$=INKEY$:IF T$="" THEN GOTO 5510 E
LSE IF ASC(T$)=13 THE GOTO 5545
5515 IF ASC(T$)=8 AND LEN(G%)>0 THEN PRI
NTCHR$(8);CHR$(132);CHR$(8);:Z=LEN(G%):I
F Z=0 THEN PRINTCHR$(7);:GOTO 5510 ELSE
G$=MID$(G%,1,Z-1):GOTO 5510
5520 IF ASC(T$)>64 DDFSTR ASC(T$)<91 THE
N GOTO 5540
5525 IF ASC(T$)>96 AND ASC(T$)<123 THEN
GOTO 5540
5530 IF ASC(T$)=32 THEN GOTO 5540
5535 PRINTCHR$(7);:GOTO 5510
5540 G$=G$+T$:IF LEN(G%)>Q THEN PRINTCHR
$(7);:Z=LEN(G%):G$=MID$(G%,1,Z-1):GOTO 5
510 ELSE PRINT T$;:GOTO 5510
5545 POKE 20479+(X-1)*80+Y+LEN(G%),17:RE
TURN

```

De routines 6000 t/m 6142 geven een string weer op een manier zoals in REM is aangegeven. De lengte van de string is Q, X is het regelnummer en Y het kolomnummer op de monitor.

```
6000 REM 'VIDGRAF1' VANAF X,Y OVER LENGTE Q GRAFICS VIDEO
6002 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,1:NEXTI:RETURN
6020 REM 'VIDGRAF2' VANAF X,Y OVER LENGTE Q VIDEO NORMAAL UNDERLINED
6022 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,2:NEXTI:RETURN
6030 REM 'VIDGRAF3' VANAF X,Y OVER LENGTE Q VIDEO GRAFICS UNDERLINED
6032 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,3:NEXTI:RETURN
6040 REM 'VIDGRAF4' VANAF X,Y OVER LENGTE Q VIDEO NORMAAL KNIPPEREND
6042 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,4:NEXTI:RETURN
6050 REM 'VIDGRAF5' VANAF X,Y OVER LENGTE Q VIDEO GRAFICS KNIPPEREND
6052 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,5:NEXTI:RETURN
6060 REM 'VIDGRAF6' VANAF X,Y OVER LENGTE Q VIDEO NORMAAL UNDERLINED KNIPPEREND
```

```
6062 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,6:NEXTI:RETURN
6070 REM 'VIDGRAF7' VANAF X,Y OVER LENGTE Q VIDEO GRAFICS UNDERLINED KNIPPEREND
6072 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,7:NEXTI:RETURN
6080 REM 'VIDGRAF8' VANAF X,Y OVER LENGTE Q VIDEO REVERSE
6082 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,8:NEXTI:RETURN
6100 REM 'VIDGRAF10' VANAF X,Y OVER LENGTE Q VIDEO NORMAAL UNDERLINED REVERSE
6102 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,10:NEXTI:RETURN
6120 REM 'VIDGRAF12' VANAF X,Y OVER LENGTE Q VIDEO NORMAAL KNIPPEREND REVERSE
6122 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,12:NEXTI:RETURN
6140 REM 'VIDGRAF14' VANAF X,Y OVER LENGTE Q VIDEO UNDERLINED KNIPPEREND REVERSE
6142 A=22527+(X-1)*80+Y:FOR I=A TO A+Q:POKE I,14:NEXTI:RETURN
```

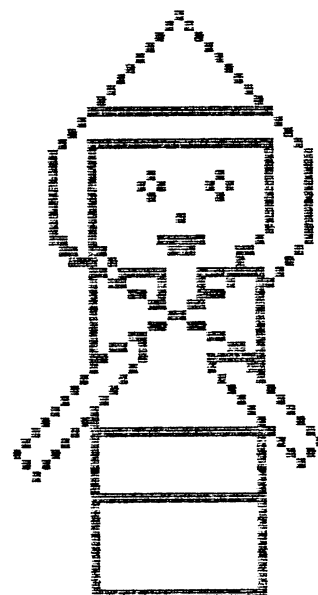
Het volgende programma van Maarten Groosman zet een string in het geheugen, haalt deze weer op, verhoogt de ascii-waarde met 1 en vult vervolgens het hele scherm met deze string. Het gaat zo snel dat de opeenvolgende strings nauwelijks kunnen worden waargenomen. Wil je zien wat er gebeurt, zet dan in regel 80 voor next een pauze-loop. Regel 80 wordt dan als volgt:

```
80 FORI=32TO255:X=VARPTR(A(1)):DEFUSRO=X:POKE X+10,I:Z=USR0(0):FORB=1TO200:NEXT:NEXT
```

Hier volgt het oorspronkelijke programma.

```
10 DATA&H21,0,&H50,1,&H7F,7,&H11,1,&H50,&H3E,127,&H77,&HED,&HB0,&HC9,0
20 DEFINTA-Y:DIMA(22):FORI=1TO8:READX:READY:Z=X+Y*256
30 IFZ>32767THENZ=Z-64*1024
40 A(I)=Z:NEXT
50 FORI=22528TO22548+1919:POKEI,1:NEXT
60 DEFUSRO=VARPTR(A(1))
70 X=USR0(0)
80 FORI=32TO255:X=VARPTR(A(1)):DEFUSRO=X:POKE X+10,I:Z=USR0(0):NEXT
90 END
```

MADELENE



Toelichting

Met dit programma kunnen tekeningen worden gemaakt met de toetsen 4 5 6 1 3 0 00 en , op het numerieke toetsenbord. Als een tekening klaar is kan deze op papier worden afgedrukt en ook op cassette worden bewaard.

De tekeningen komen als tabel op de cassette onder naam A, B, C, enz tot en met Z. Er kunnen dus 26 tekeningen per cassette weggeschreven worden.

De tekeningen hebben een nummer 1-26.

Als je het programma zelf aan het begin van de band zet met als naam een kleine letter, b.v. "t", dan kan er niets fout gaan.

Pas op met veranderingen in het programma.

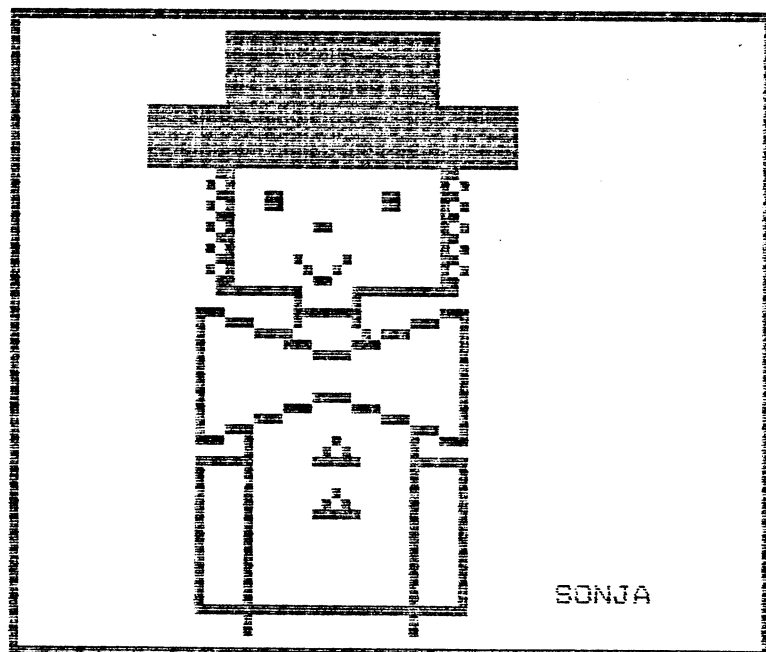
Als je daaraan begint kan soms (zelden) een aanpassing van y in regels 2, 40000, 40063 en 40073 nodig zijn.

Daar staan nl. de routines om het wegsschrijven onder naam correct te laten verlopen. De naam wordt gepoked in het programma op de betreffende plaatsen.

Als het programma is gestart en het scherm is schoon, dan kun je links-onder beginnen met tekenen.

Met CODE wordt de tekenlijn zwart. Opnieuw CODE maakt de lijn weer wit. Met de tekenlijn op zwart kun je lijnen wissen of over het scherm gaan naar een andere plaats om daar verder te tekenen.

Dit programma maakt gebruik van de subroutines uit de blauwe map



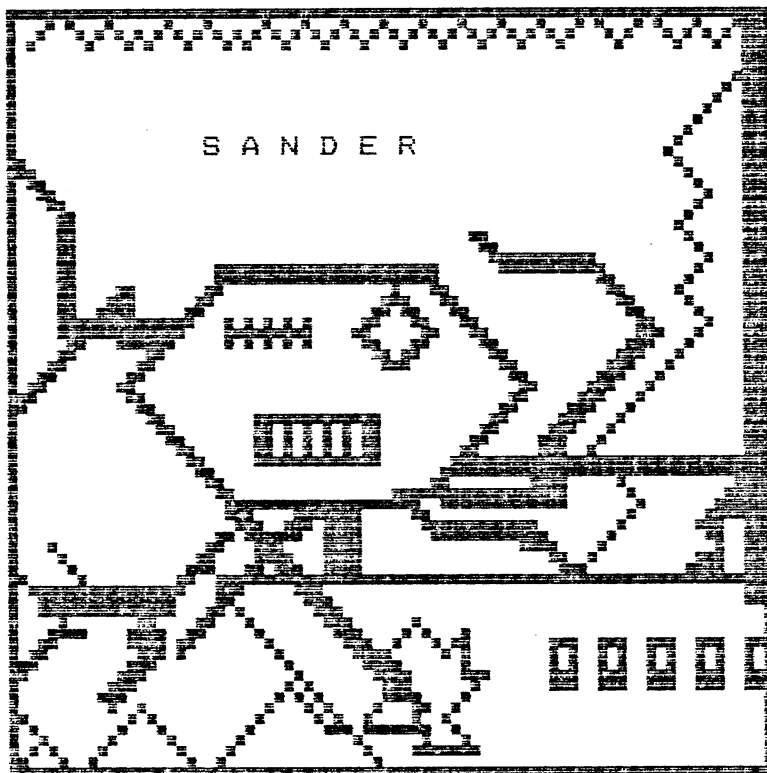
Tekst:

Stuur het blokje naar de plaats waar de tekst moet komen te staan. Tik dan de tekst liefst in hoofdletters i.v.m. printen. Daarna RETURN en dan verschijnt de tekst. De tekst kan daarna wel overschreven worden door spaties of een andere tekst; maar niet overtekend.

Zet de tekst dus liefst als slot in de tekening.

Printen.

Als een printer is aangesloten kan het scherm worden uitgeprint. Toets daarvoor shift en het printteken (numeriek bord de toets 00).

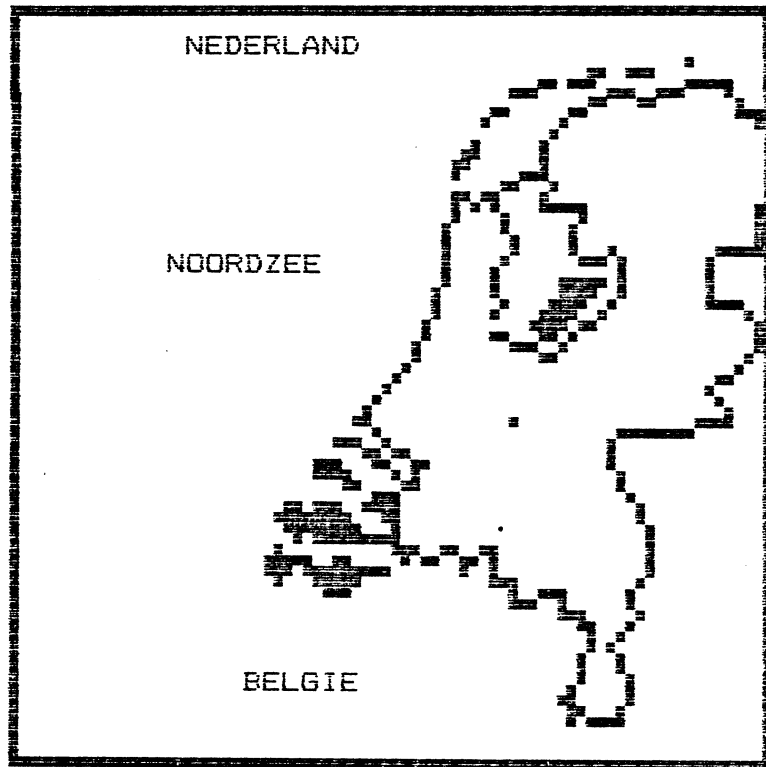


Opnemen.

Toets shift 7 (cassettesymbool). Dan duurt het even voordat de scherminhoud in een tabel is gezet. Dan verschijnt de vraag of de tekening wel of niet op de cassette moet worden gezet.

strookje voor onder het klepje.

TEKENEN: numeriek toetsenbord	4	5	6			
CODE: tekenlijn wit/zwart				☐		
PRINTEN: SHIFT 00 (printerteken)	1		3	↙	↑	↘
OPNEMEN: SHIFT 7 (cassettesymbool)				←		→
TEKST : intypen en RETURN	0	00	,	↙	↓	↘



Tip voor gebruikers van het tekenprogramma (zwart/wit).

=====

In een programma zijn nogal eens grafische karakters nodig, soms voor op het scherm en soms voor op papier.

Het is een hele klus om eerst een tekening op een beeldschermformulier te ontwerpen en dan alle codes in de lijst op te zoeken.

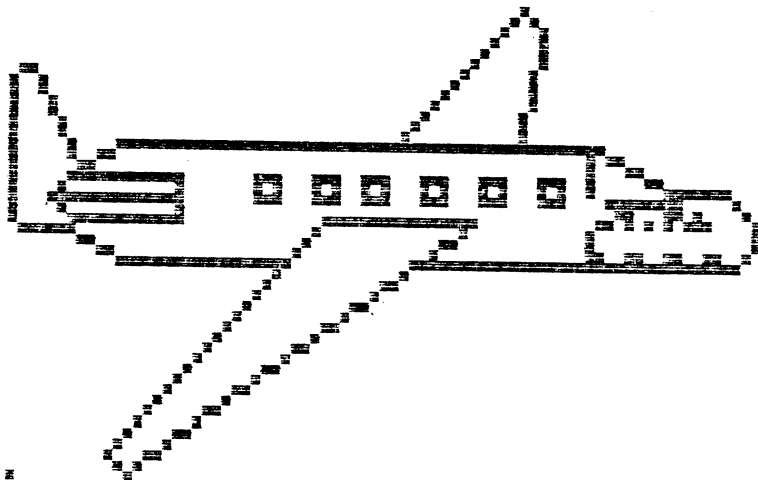
Met het tekenprogramma gaat dit een stuk makkelijker.

Je tekent de grafische figuren met het tekenprogramma op het scherm. Dan wordt de tekening in een tabel gezet en op cassette weggeschreven.

Vervolgens lees je de tekening weer in met een ander programmaatje dat het volgende doet.

1. inlezen van het array
2. op papier printen van de inhoud van het beeldscherm.
in ASCII-codes.

Je hebt dan een lijst van de codes voor je grafische karakters.



Louis Naus
Schoutsakker 5
1871 CZ SCHOORL
tel. 02209 - 2508

(cassette 102 kant B programma "G".)

Idee en oorspronkelijk programma: Chr.E.de Boer.
Ingepast in het bestandsprogramma met de T.Jonker taperoutines
door Louis Naus te Schoorl.

Toelichting.

De Postcheque en girodienst en ook o.a. de Rabobank heeft voor
het onderwijs lesmateriaal beschikbaar in de vorm van een boekje
en voorbeeldformulieren.

Het gebruik van dit lesmateriaal wordt zinvoller als de P 2000
wordt ingeschakeld met dit programma.

Hoe kan een bankgiroproject verlopen in b.v. de 6e klas
basisonderwijs of in de brugklas voortgezet onderwijs?

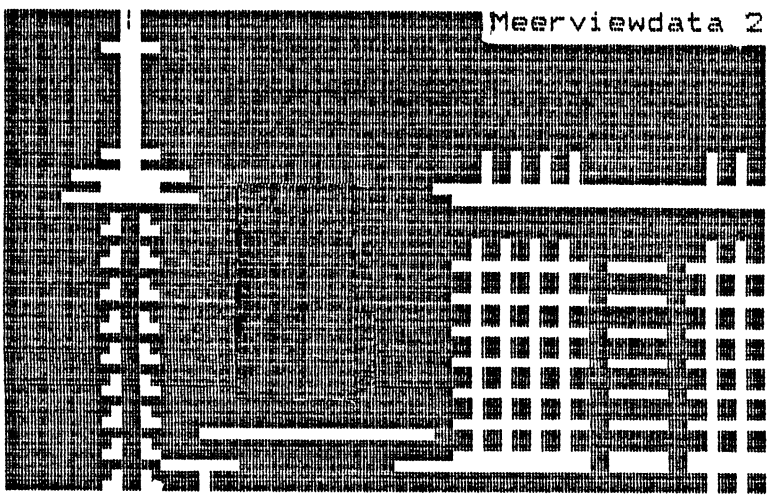
1. alle leerlingen krijgen de beschikking over een set
formulieren en het lesboekje (gratis).
2. alle leerlingen openen een rekening door het invullen van het
juiste formulier.
3. de leraar (of een ervaren leerling) typt na school de
gegevens in de computer. De computer maakt van iedere mutatie
een dagafschrift of rekeningoverzicht. Dit overzicht komt op het
beeldscherm. Rechtsonder in beeld verschijnt het getalteken (#).
Er zijn dan twee mogelijkheden:
 - a. getalteken intypen (#) : het programma wordt vervolgd.
 - b. printerteken intypen (shift en 00) : het dagafschrift
wordt op papier afgedrukt.
4. de volgende les krijgen de leerlingen hun eerste
dagafschrift.
5. Nu is het formulier aan de beurt om geld te storten op de
eigen rekening.
6. Dat wordt weer door de computer verwerkt en de volgende les
krijgt iedere leerling weer het 2 e dagafschrift.
7. Nu staan er veel mogelijkheden open naar keuze.
 - b.v. * cheques uitschrijven
 - * overschrijven naar elkaar
 - * giroloterij met prijzen
 - * betaalkaarten en pas gebruiken
 - * wat te doen bij saldotekort
 - * geld storten
 - * acceptgiro betalen
 - * enz. enz.
8. Dagelijks kan er op het prikbord een bijgewerkte lijst van
rekeninghouders worden opgehangen.
9. Suggesties voor andere mogelijkheden zijn welkom bij Louis
Naus te Schoorl (tel.02209-2508).

Regelmatige viewdatagebruikers zullen inmiddels de weg wel hebben gevonden naar zowel de Nat.Lab.-'PHILITEL'-viewdatacomputer met het inmiddels beroemde P2000-viewdatagebeuren van de gebruikersgroep, als naar de Nova Viewdata 3000 computer met onder meer het HCC-viewdatabestand. Beide systemen hebben hun eigen 'trekkers' zoals het eerste en enige echte P2000gg-prikbord bij PHILITEL en de probleemloos functionerende telesoftware-service, geesteskinderen van Klaas. Anderzijds de TELEBUS-service van VIEWDATA 3000, waarvan veel P2000-gebruikers een dankbaar gebruik maken voor het verzenden van 'prive'-berichten naar elkaar.

Het is duidelijk dat PHILITEL de eerste en belangrijkste bron van informatie voor de P2000-gebruiker is en blijft. In het bestuur is daarom gesteld dat er maar een P2000-prikbord zal zijn, nl. in PHILITEL. Immers, hoe meer P2000-prikborden, hoe minder een dergelijk informatiesysteem functioneert. Inmiddels wordt echter ook in de Viewdata 3000-computer informatie over de P2000 gegeven en wel op meerdere plaatsen. De bedoeling is dat er - overigens zonder particuliere initiatieven te willen ontmoedigen - door het bestuur naar wordt gestreefd om een P2000-informatiebestand te creëren binnen het HCC-bestand. Daarin komt zowel informatie over de P2000gg voor (nog) niet-P2000-gebruikers als voor P2000-gebruikers. Wat het laatste betreft denken we bijvoorbeeld aan een permanent overzicht van de P2000-software-service en produktinformatie. Wat dit laatste betreft gaat het om objectieve informatie die van belang is voor P2000-gebruikers, en niet om reclame.

Hoe een en ander zich zal ontwikkelen is mede van jullie reacties afhankelijk. Laat dus maar horen als je kritiek, - of beter nog - goede ideeën hebt. Toets *3015599_ voor informatie en reacties.

Meerviewdata 2



PHILIPS NATUURKUNDIG LABORATORIUM

Telefoon: 040 040 741154
Gebruikersnummer: 2075
Codewoord: 2075
Inlichtingen: 040 743072

Na de verschijning van Nieuwsbrief 3 zijn de viewdata-mogelijkheden met de P 2000 weer uitgebreid. Met een Viditel-modem (te koop of te huur) en het P 2000 viewdata-programma kunt U via de telefoon verbinding maken met diverse Data-banken. Reeds velen maken hiervan gebruik voor het uitwisselen van gegevens, het stellen van vragen het inlezen van programma's of om zomaar door de pagina's te 'bladeren'.

Hieronder een overzicht van de mogelijkheden:

Pagina 18
Prikbord
VIEWDATA-mogelijkheden met P 2000

- * Philitel (Philips Nat Lab)
tel. 040-741154. Inloggen met 2075-2075.
- * Viditel (PTT) 020-318318 of 070-151515.
- * NOVA (o.a. HCC).
tel.03402-36364/38264/38364/41514.
Inloggen met 005500-5500.
Vraag een eigen usernummer aan als HCC-lid.
Pag.no *30114 of *789900131(voor niet HCC-leden.)
- * ABC-Datavision tel. 020-435824.
Inloggen met 5678-1234.
- * TRS-80 Vidiboard. tel.1880-30039.
(Werkt nog niet volgens Viewdata-protocol.)
- * IYS-3(Belgie)Tel.093226 409255
- * BELL(Belgie) Tel.093231 370000
- * PRESTEL (U.K)Tel.0944 1 6001261
- * EUROSOUND 030/517199 (timetable in
NOVA *789870_ 123456 1234 wo + zat)
Voor PRESTEL-codenummer bel
010/330844 (hfl 125,-- per maand +
hfl 1,-- per min.
- * BILDSCHIRMTEXT (Oostenrijk)
Tel.0943 22908. Aanvraag usernr.
Tel.09222 784705
prestel keatscomp. 09-4421618111Ben g.

Den Haag Amsterdam
Tel: 070 151515 Tel: 010 318318
Niet kosteloos !

Toegangsnummers aanvragen bij:
Centrale Directie der PTT
Bureel Viditel
Antwoordnummer 6000
2500 GA 's Gravenhage



PRESTEL.

Prestel is vanaf het begin van het jaar oproepbaar via een directe verbinding in Nederland. Men kan als men een toegangsnummer heeft toegang krijgen tot Prestel door een nummer in Rotterdam te bellen. Dit kan dus zonder internationale gesprekskosten, maar zeker niet gratis (125 gulden per maand en een gulden per minuut). Bel voor nadere informatie 010-330844.

Na een bezoek aan de Viewdata-demonstratie in Baexem heb ik mijzelf meteen aangemeld als Viditel-lid en kwam zodoende, voor een huurbedrag van f. 11,80 per maand, in het bezit van de benodigde modem.

Terugkijkend naar de voorbije periode, waarin vrij intensief gebruik gemaakt werd van de diverse Viewdata-mogelijkheden, kom ik tot de slotsom dat deze uitbreiding van de P2000-mogelijkheden een absolute 'must' is voor iedere P2000-kollega die deze stap (nog) niet heeft genomen.

Veel waardering heb ik voor ons eigen P2000-gebruikersbestand van de Nat.Lab Thuiscomputerclub met een zeer actieve Klaas als beheerder. Als P2000-gebruiker vind je bij Nat.Lab voor elk wat wils.

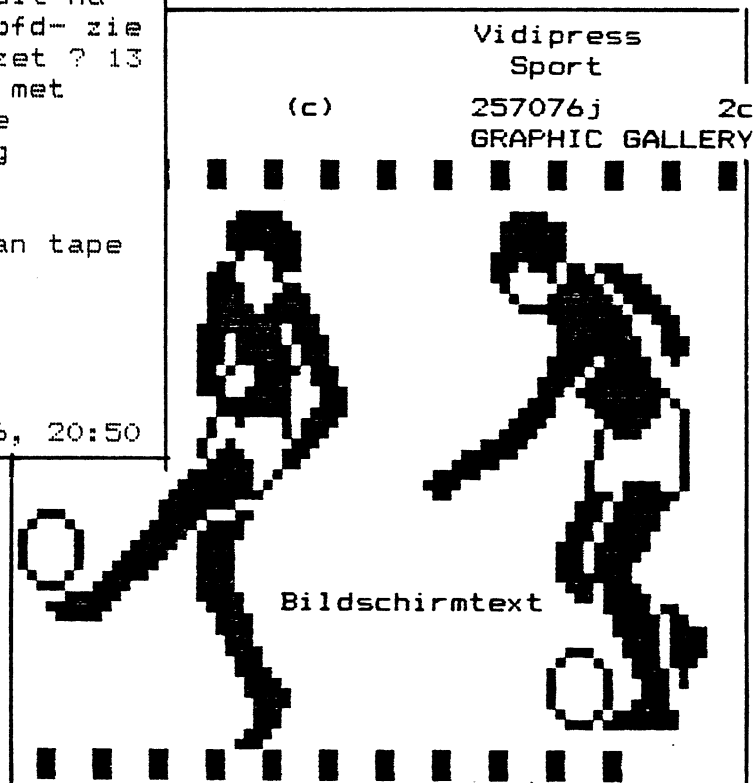
Wat te denken bijvoorbeeld van rubrieken als: Beginners, Gevorderden, Programma's, Bibliotheek enz.

En laten we vooral het Prikbord-gedeelte niet vergeten, waarop iedereen zijn vragen, problemen en mededelingen kwijt kan, die dan binnen (een) enkele dag(en) door kollega's worden beantwoord c.q. opgelost.

```

Prikbord                Pagina 12
Wie weet welke ROM-routine ik
aan moet roepen om de cassette
enkel en alleen terug te laten
te spoelen                Joep W. 17/6

joep, daarvoor moet jezelf een heel
klein machinetaal programmaatje maken:
LD A,(HL)    7E        wie heeft dit nu
CALL 0018    CD 18 00  weer in hoofd- zie
RET          C9        letters gezet ? 13
je kunt de routine nu aanroepen met
x=usr(n), waarin met de volgende
waarden, het volgende tot gevolg
heeft :
n =  |  gevolg =
1    |  spoel terug naar begin van tape
2    |  spoel blok vooruit
3    |  spoel blok achteruit
4    |  wis een stukje
5    |  schrijf data
6    |  lees data
7    |  ???                ronald, 17-06, 20:50
    
```



Mijn eerste indruk van Viditel (PTT) is ook erg positief gezien de schat aan informatie die hierin ligt opgeslagen, als is het vanuit kostenoverwegingen echt wel raadzaam om vooraf in de Viditel-gids de verlangde pagina-nummers op te zoeken.

Minder prettig is het feit dat het gezamenlijke Viewdata-experiment van de Nederlandse dagbladen onder de naam 'Krantel' sterk zal worden ingekrompen. Het is te hopen dat deze tendens zich niet verder doorzet ! Daartegenover staat echter dat er nu, ook op het technische vlak, overeenstemming bereikt is over één Europese Viewdata-standaard voor ondermeer Nederland (Viditel), Engeland (Prestel), Frankrijk (Téletel) en West-Duitsland (Bildschirmtext).

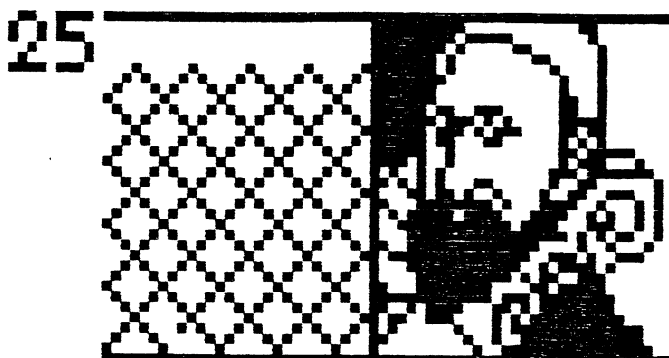
Ook ABC-Datavision en NOVA VIEWDATA 3000 (HCC-bestand) zijn zeker de moeite van het 'inloggen' waard !

Tenslotte nog dit:

Zou het zin hebben om de informatie (vragen, antwoorden en mededelingen) op het Prikbord regelmatig over te nemen (uitprinten) t.b.v. een daarvoor aan te leggen archief, zodat ook nieuwe P2000-kollega's achteraf over deze informatie kunnen beschikken.

Ik denk daarbij bijvoorbeeld aan een regelmatige publikatie van een trefwoorden-lijst over de op het Prikbord behandelde vragen en onderwerpen in onze nieuwsbrief, zodat aan de hand van deze lijst kopiën over het betreffende onderwerp tegen kostprijs besteld kunnen worden bij dit archief.

Ik wens iedere (toekomstige) Viewdata-gebruiker verder nog veel plezier toe !!!



FINANCIËEL ADMINISTRATIE
SYSTEEM



Raet Software & Computerservice
(2296)

Vidipress (c)
Beeld van de maand
november '81
2570715a 2c
GRAPHIC GALLERY

Drunen, 3 juni 1982
Jan de Wit

Op het numerieke toetsenbord van de P2000 wordt de shift-mogelijkheid van de cijfertoetsen vaak gebruikt voor functies. Zo gebruikt het Tekstbewerkingspakket P2301 <shiftM> (toets 9) om een blok van tot 9 regels in een buffer te zetten die op een of meer andere plaatsen in het geheugen tussengevoegd kan worden.

Het blijkt nu, dat de <shiftM> toets in Disk Basic toegang geeft tot de buffer, waarin tekst input en output voor de Interpreter wordt bewaard. Als je <shiftM> indrukt, verschijnen op het scherm een uitroepteken en een spatie. Je kunt dan met de L of de X de inhoud van de buffer zichtbaar maken: alle EDIT-functies blijken gewoon te werken.

Wat kun je met die toets doen?

Denk je eens in: een lange regel ingetikt, je wilt <ENTER> geven en je ziet heel aan het begin een fout staan. Als je nu teruggaat met <BACKSPACE> kun je mooi opnieuw beginnen. Met 16K Basic (Tape Basic) zou je gewoon <ENTER> en EDIT. intikken en de fout corrigeren. EDIT. werkt in Disk Basic echter niet. Maar <ENTER> <shiftM> doet hetzelfde, en zelfs nog iets vlugger. Alleen als je de regel rechtstreeks hebt ingetikt, staat het regelnummer ook in die buffer; als je uit de AUTO Mode bent gekomen met de <STOP> toets is het regelnummer verdwenen en zijn de eerste tekens van de buffer overschreven met . Ctrl/C.

Een paar praktische toepassingen:

1. Een commando herhalen, eventueel na een wijziging:

- a. FOR I=0 TO 15:PRINT I CHR\$(&H20+I) " ";:NEXT <ENTER>
- b. <shiftM> S 2 C 3 <ENTER>
- c. <shiftM> S 3 C 4 <ENTER> enzovoort.

2. Een algoritme uitproberen en dan in het programma opnemen.

- a. 1000 INPUT "Een letter ";A\$
- b. 1010 B\$="ACDEF"
- c. RUN
- d. PRINT INSTR(B\$,A\$) <ENTER>.
- e. <shiftM> X <>0 <ENTER>
- f. <shiftM> S < H +1 <ENTER>
- g. <shiftM> I 1020 <SPATIE> <CODE> 2 C ON 3 D X GOSUB 2000, 2100 <ENTER> LIST <ENTER>
- h. <shiftM> I 1500 <SPATIE> <CODE> 2 S 2 C 3 <ENTER> LIST

<shiftM> werkt met EDIT en met LIST. De laatste regel blijft in de buffer staan zolang geen andere nieuwe toets dan <shiftM> of <STOP> wordt ingedrukt.

Je kunt <shiftM> gebruiken om een regel te kopiëren of te splitsen: oproepen met LIST <regelno>, <shiftM> geven, het nieuwe regelnummer invoegen, eventueel statements verwijderen en sprongadressen aanpassen, en met <ENTER> gewoon wegzetten. Bij splitsen moet je dan natuurlijk ook de oude regel aanpassen.

Een file op meer dan één disk SAVEn gaat ook prima: SAVE "PROG" <ENTER> - na Ok nieuwe disk erin en <shiftM> <ENTER> geven enz. Dit werkt niet met SAVE "PROG",A, omdat dan alle output net als bij LIST door de buffer loopt, en je de laatste regel van het geSAVEde programma terugvindt.

Het loont de moeite om hiermee te experimenteren, omdat deze functie de kracht van de Editor aanzienlijk vergroot. Denk er wel om, dat de buffer gewist wordt, zodra je een andere toets dan <shift M> of <STOP> hebt gebruikt.

J. Vermeulen



FRAKTISCHE TIPS

RESET IN PROGRAMMA

Een RESET kan worden gegenereerd door de volgende regel in te voeren:

```
DEF USR = 0 : X = USR(0)
```

Dit kan zowel direct als in een programma. (Ronald v.d. Wal)

GET-ROUTINE IN MACHINETAAL

Onderstaande routine in machinetaal geeft de ASCII-waarde van een ingedrukte toets in geheugenplaats 9FFF. Als er geen toets is ingedrukt staat hier een 0.

```
GET      LD  A,(600C)
         CP  00
         LD  (9FFF),A
         RET Z
         DEC A
         LD  (600C),A
         LD  H,60
         LD  L,A
         LD  A,(HL)
         ADD 14
         LD  H,18
         LD  L,A
         LD  A,(HL)
         LD  (9FFF),A
         RET
```

Deze routine verschilt niet wezenlijk van de gebruikelijke GET-routine, maar is sneller door het gebruik van machinetaal. (R.v.d.W)

PRINTER BAUD

Door op geheugenplaats &H6016 een getal te zetten kan de Baud-snelheid van de printer-uitgang worden veranderd.

Dit werkt als volgt:

POKE &H6016,0 -> 2400 Baud

POKE &H6016,1 -> 1200 Baud

POKE &H6016,7 -> 300 Baud

POKE &H6016,21-> 110 Baud

De instelling na inschakelen van de P 2000 is 1200 Baud (dus een 1).

Door echter op de printerconnector pen 2 met pen 6 te verbinden start de computer op met 300 Baud (dus een 7).

Andere snelheden moeten door het programma worden ge-POKE-d.

PROGRAMMA REDDEN BIJ "CLOAD" i.p.v. "CSAVE"

Het komt wel eens voor dat je een programma op cassette wilt vastleggen, maar dat je in plaats van CSAVE intypt CLOAD. Het op te bergen programma is dan ogenschijnlijk uit het geheugen van de computer verdwenen! Als je het echter tijdig ontdekt en de cassette stopt of er stond geen programma onder die letter, dan kan het 'verdwenen' programma met een eenvoudig foefje worden teruggehaald. Het staat namelijk nog wel in het geheugen, maar door CLOAD te geven heeft de basic de indicatie voor het einde van het programma aan het begin van de basic-ram gezet. Dit is tweemaal 0. Normaal staat hier het adres van de volgende programmaregel.

Door nu een programma altijd te beginnen met een lege REM-regel is het adres van de volgende regel altijd bekend en kan met twee POKE-opdrachten hersteld worden. We geven een voorbeeld.

Voorbeeld:

De eerste regel is: 5 REM

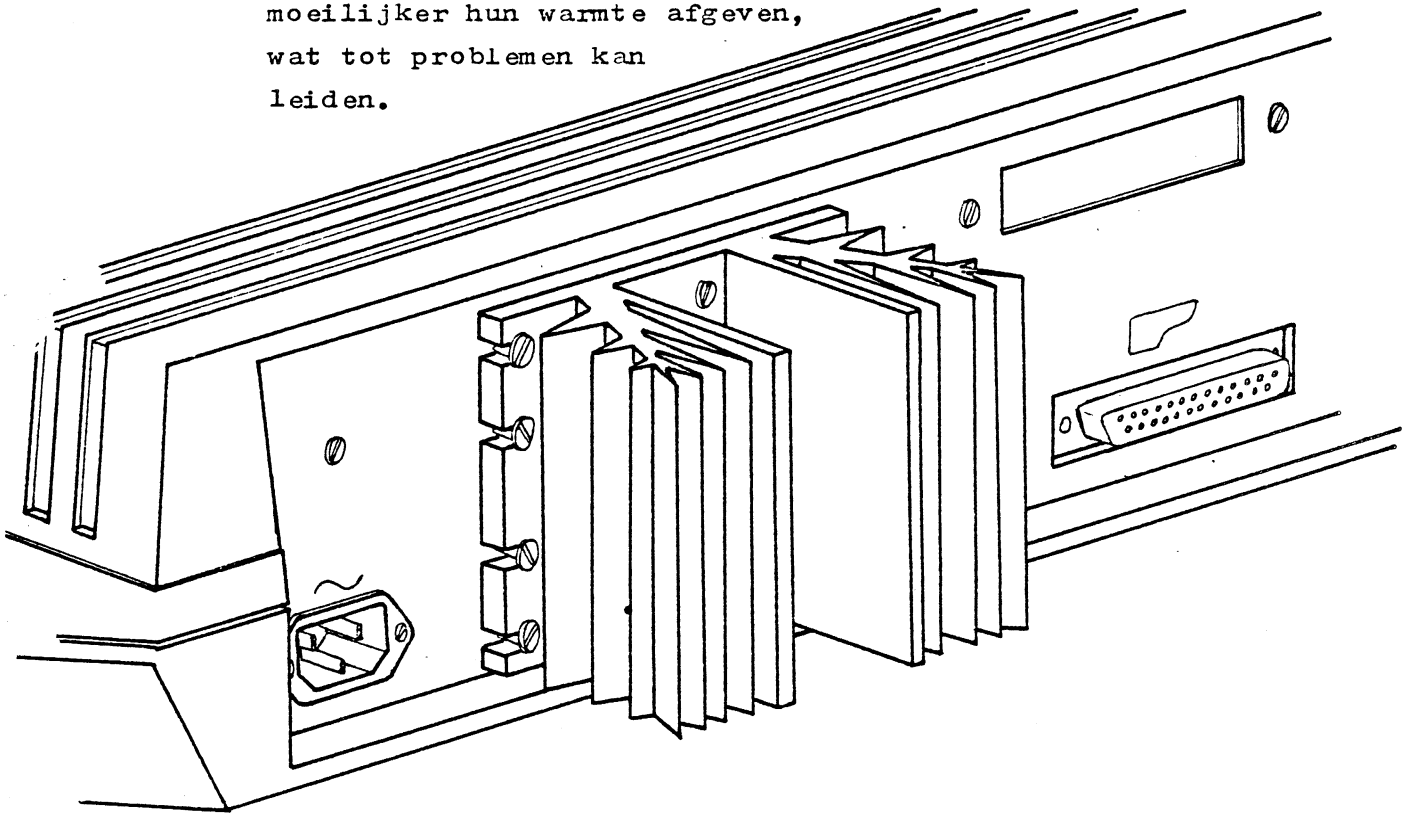
In het geheugen staat nu:

6547	4D	lage byte beginadres volgende regel
6548	65	hoge byte " " " "
6549	05	lage byte regelnummer
654A	00	hoge byte " "
654B	8E	token voor REM
654C	00	einde regel
654D	58	begin nieuwe regel
	65	
	64	regelnummer 100
	00	

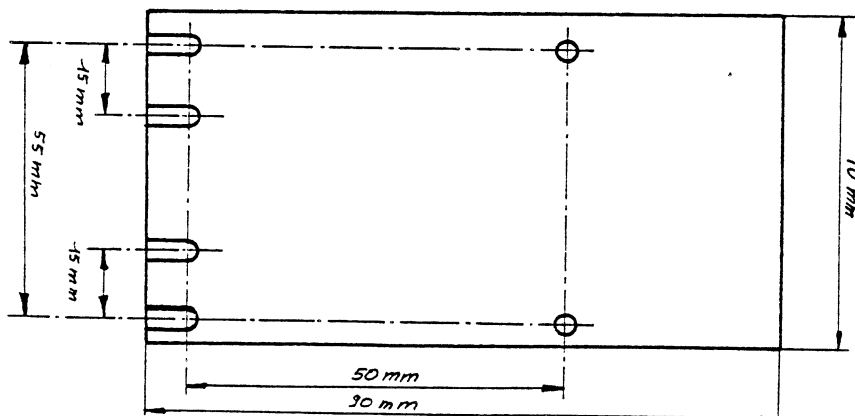
Met POKE &H6547,&H4D:POKE &6548,&H65 komt het programma weer te voorschijn en kan alsnog worden ge-CSAVE-ed.

Ton Hilgersom.

Het blijkt dat de koelplaat van de P2000 nogal krap bemeten is wat betreft de warmte-afgifte. Dit is vooral het geval bij een volledig opgezette versie (met 48kB). De achterkant wordt dan heel heet en de transistoren kunnen moeilijker hun warmte afgeven, wat tot problemen kan leiden.



Men kan de warmte-afgifte op een eenvoudige manier een heel stuk verbeteren, wat zeker ten goede komt aan de voeding. Hiertoe plaatst men op de achterzijde van de P2000 een koelvin. De montage kan gebeuren met de reeds aanwezige schroeven zoals in de figuur is afgebeeld. De figuur vertelt in dit geval veel meer dan kan geschreven worden, daarom laat ik het aan ieders verbeelding over, hoe men de blok monteert. De afmetingen voor het boren van de gaten (of sleuven) staan hieronder afgebeeld.



VERBERGEN VAN PROGRAMMAREGELS

Door een truc kunnen delen van het BASIC-programma bij het listen onzichtbaar worden gemaakt zonder dat dit de uitvoering van het programma beïnvloedt. Twee voorbeelden:

```
10 A=10:B=2          10
20 FOR I = 1 TO 10   20 FOR I = 1 TO 10
30 A=A+1:B=2*A       30 A=A+1
40 PRINT A;B         40 PRINT A;B
50 NEXT              50 NEXT
```

Beide programma's geven bij RUN hetzelfde resultaat.

Dit wordt bereikt door de volgende truc. Voer op die plaatsen, waarna de listing onzichtbaar moet worden gemaakt vijf maal een door BASIC niet herkend karakter in, bijvoorbeeld # (hekje). Regel 10 en 30 in het tweede programma zijn bijvoorbeeld als volgt ingevoerd:

```
10 #####A=10:B=2
30 A=A+1#####B=2*A
```

Zoek vervolgens op waar deze vijf karakters staan en POKE op het eerste karakter ervan een 0. Dus als het eerste hekje staat op &H6647 maak hier dan een 0 van.

Met het volgende programma kun je naar willekeur delen van een programma beveiligen.

```
0 I = &H6547
1 A = 0 : FOR J = 0 TO 4 : IF PEEK(I+J) = ASC("#") THEN A =
  A+1 : NEXT
2 IF A = 5 THEN POKE I,0
3 I = I+1
4 IF I >= PEEK(&H6407) + 256*PEEK(&H6404) THEN END
5 GOTO 1
```

Plaats dit programma voor een ander programma met daarin de vijf karakters. Het programma werkt goed, maar is betrekkelijk traag. Het kan sneller gemaakt worden door bv. eerst een # op te zoeken en vervolgens de andere vier karakters. Zijn alle vijf karakters gevonden, dan wordt de programmaregel uitgevoerd. Zoniet, dan wordt het zoeken naar het volgende karakter voortgezet.

Jan Wind, Hellevoetsluis.

RANDOM GENERATOR

Zoals bekend geeft RND(1) steeds dezelfde reeks getallen af nadat de P 2000 is aangezet of nadat op de RESET is gedrukt. Dat heeft o.m. tot gevolg dat allerlei spelprogramma's steeds hetzelfde beginnen (en doorgaan).

Remedie:

Zet d.m.v. een POKE-opdracht een willekeurig getal in de randomgenerator bij de start van een programma. Geschikt is daarvoor de stand van de klok-timer. Als U dat doet geeft RND (1) telkens een andere reeks getallen.

De juiste opdracht is: POKE 25144,PEEK(24592).

Misschien hebt U zich wel eens afgevraagd hoe de P 2000 Uw programma's in z'n geheugen neerzet.

Dat valt het beste uit te leggen aan de hand van een klein voorbeeld.

We gaan kijken hoe het volgende programmaatje in het geheugen staat.

```
10 PRINT"TEST"
20 INPUT A
30 PRINT A
```

De P 2000 schrijft programma's neer vanaf geheugenplaats 25927. De eerste 2 plaatsen bevatten getallen die verwijzen naar de geheugenplaats waarop de volgende programmaregel begint. De volgende twee plaatsen bevatten het regelnummer. Dan volgt de code voor het basic statement met daarop de tekst in ASCII-codes. Om aan te geven dat de programmaregel af is volgt dan nog een nul.

Bekijk onderstaand schema maar eens goed. De eerste kolom bevat de geheugenplaats en in de tweede kolom staat de inhoud van die geheugenplaats. U kunt dit zelf ook onderzoeken door het voorbeeldprogramma te verlengen met:

```
50 FOR I= 25927 to 25954:PRINTI;PEEK(I):NEXT
```

Met RUN 50 krijgt U dan dezelfde lijst cijfers.

25927	83	{	NEXT LINE POINTER	
25928	101	{	101*256+83 = 25939	
25929	10	{	REGELNUMMER	
25930	0	{	256*0+10 = 10	
25931	165		CODE VOOR 'PRINT'	
25932	34		ASCII VOOR AANHALINGSTEKEN	
25933	84		ASCII VOOR 'T'	
25934	69		ASCII VOOR 'E'	
25935	83		ASCII VOOR 'S'	
25936	84		ASCII VOOR 'T'	
25937	34		ASCII VOOR AANHALINGSTEKEN	
25938	0		DE NUL BETEKENET HET EINDE VAN DE REGEL.	
25939	91	{	NEXT LINE POINTER	
25940	101	{	256*101+91 = 25947	
25941	20	{	REGELNUMMER	
25942	0	{	256*0+20 = 20	
25943	132		CODE VOOR INPUT	
25944	32		ASCII VOOR SPATIE	
25945	65		ASCII VOOR 'A'	
25946	0		NUL VOOR HET EINDE VAN DE REGEL	
25947	0	{	NEXT-LINE-POINTER	
25948	0	{	TWEE NULLEN BETEKENEN 'DIT IS DE LAATSTE REGEL'	
25949	30	{	REGELNUMMER	
25950	0	{	256*0+30 = 30	
25951	165		CODE VOOR PRINT	
25952	32		ASCII VOOR SPATIE	
25953	65		ASCII VOOR 'A'	
25954	0		NUL VOOR HET EINDE VAN DE REGEL	Louis Naus

In veel programma's komen regels voor met DATA. Als U b.v. een adressenbestand bij wilt houden kunt U DATA-regels maken met daarin de gegevens. Met het READ-statement kan het programma die gegevens lezen en verwerken.

Het is echter niet zo gemakkelijk om op deze manier gegevens in te voeren. Een manier die gebruikersvriendelijker is is het invoeren van gegevens met het INPUT-statement. De gebruiker kan dan antwoorden intypen op vragen die de P 2000 stelt. De P 2000 kan niet zonder meer string-arrays op cassette bewaren. Daarom zijn we er niet als we met INPUT-statements een string-array vullen.

In dit artikel beschrijf ik twee manieren om gegevens die met INPUT-statements zijn ingevoerd, om te zetten in DATA-regels. Als dat gebeurd is kunnen de gegevens naar cassette worden geschreven met een gewone CSAVE opdracht.

Dat kan met alle mogelijke gegevens. Ik gebruik deze manier zelf op school in een Multiple Choice programma. De leerkrachten kunnen vragen invoeren door te antwoorden op INPUT-vragen van de P 2000. Daarna worden de vragen met antwoorden door de P2000 in DATA-regels gezet en er is weer een test klaar. Die gaat op de cassette tot hij nodig is.

De eerste versie van de leerlingenadministratie heb ik ook op deze manier gemaakt. De gemaakte DATA-regels komen per klas op cassette te staan. Met een koppelprogramma lees ik dan de DATA-regels aan het programma dat de gegevens verwerkt.

In alle gevallen waarin DATA-regels nodig zijn kunnen onderstaande routines worden gebruikt.

Zoals ik al schreef kan het op twee manieren. Beide hebben voor- en nadelen. U kunt het zelf uitproberen, want ik heb ze beide in hetzelfde programma gezet. Het is een gedeelte uit het Bestandsprogramma van P 2000 cassette no 4 (A-kant pr. "B".) Uit dit programma heb ik de invoer-routine genomen en de routine die de DATA-regels maakt. Het volledige programma vindt U op cassette 4.

Wie belangstelling heeft voor het Multiple Choice-programma kan het beste een cassette aan mij sturen (met gefrankeerde antwoordersveloppe.)

Maar dan nu de routines.

Manier A.

Op regel 900 begint de DATA-POKE-routine.

Als nog geen gegevens ingevoerd zijn staat op regel 1000 een 0. Dat leest de P 2000 in regel 35 (N1). We kunnen daarna gegevens gaan invoeren. In dit geval naam, adres, postcode en woonplaats. Als we klaar zijn met gegevens invoeren kiezen we '2' uit de menu en dan gaat de P 2000 naar regel 900.

In die regel worden de ingevoerde gegevens per record achter elkaar gezet in A\$.

Dan wordt de subroutine in regel 910 aangeroepen. Daar begint dan het belangrijkste stuk van de routine.

Om dit te begrijpen moet U weten hoe een basic regel in het geheugen wordt opgeslagen. (Zie elders in dit blad.)

In regel 920 gaat de P 2000 op zoek in het geheugen. Vanaf adres SA (27000) kijkt de P 2000 of er ergens twee keer achter elkaar een machtsverheffingspijltje staat. Dat pijltje heeft de ASCII-waarde 94. De pijltjes staan op regel 998.

Op het scherm ziet U getallen voorbijflitsen tot de pijltjes gevonden zijn. Als U deze routine in een lang programma inbouwt duurt het lang voor de pijltjes gevonden zijn. Na een RUN kunt U op het scherm zien op in welk adres ze staan. U kunt dan SA in regel 910 verhogen.

Als de pijltjes gevonden zijn gaat de P 2000 naar regel 930. In regel 930 en 940 wordt het volgende regelnummer uitgerekend. Dat gaat zo. In dit voorbeeld staan de pijltjes op regel 998. In 940 wordt RG gelijk aan 998 en daarna met 2 verhoogd. We krijgen dan regelnummers 1000, 1002 enz. Als we de pijltjes op regelnummer 3000 zouden zetten krijgen we 3000, 3002 enz. Als we de 2 in regel 940 veranderen in b.v. 5 dan krijgen we 3000, 3005 enz. Daarmee is gelijk een voordeel van manier A duidelijk geworden. We kunnen aangeven welke regelnummers de DATA-regels moeten krijgen.

In regel 950 wordt de zogenaamde Next-line-pointer gepoked. Die geeft het adres aan waarop het volgende regelnummer in het geheugen begint. In regel 960 wordt eerst het regelnummer zelf gepoked en daarna de code voor DATA (131). In regel 980 wordt dan A\$ letter voor letter gepoked.

Regel 990 zet een '0' in het geheugen. Dat betekent het einde van de basic-programmaregel. Tenslotte wordt berekend waar de volgende regel moet beginnen.

Er moet wel ruimte zijn om de DATA-regels te gaan maken. Daarvoor dienen de regels met al die :::::. U moet er zoveel intypen (met hoge regelnummers) dat Uw gegevens er in ieder geval in passen.

Als alle gegevens in DATA-regels zijn gezet gaat de P 2000 naar regel 903. Dat is een belangrijke regel ; want die zorgt ervoor dat de gemaakte data-regels netjes gaan aansluiten aan de 'deels vernielde' regels met :::::

Dan stopt het programma met de mededeling dat U moet typen 1 STOP (ENTER) 1 (ENTER). Als U dat niet doet wil het programma niet listen. U kunt het wel op de cassette saven. Maar U wilt natuurlijk zien of er nu echt data-regels gemaakt zijn. Als U het bovenstaande intypt doet U in feite niets anders als regel 1 maken met STOP erin en vervolgens regel 1 weer verwijderen. U mag gerust ook een andere regel maken en weghalen. Dan werkt het ook.

Op de listing kunt U zien wat er is veranderd voor en na de invoer van gegevens. Als U nu een RUN geeft worden in regel 35 de net gemaakte DATA ingelezen.

Als U later meer gegevens wilt toevoegen dan kan dat. Als U echter in de DATA-regels wilt veranderen moet dat in de EDIT-mode gebeuren.

```

10 REM DATA GENEREREN
20 CLEAR2000:READZ:DIMAD$(40,Z)
30 FORY=0TOZ:READR$(Y):NEXT:READA$
35 READN1:IFN1=0THEN40ELSEFORI=1TON1:FORY=0
TOZ:READAD$(I,Y):NEXT:NEXT
40 PRINTCHR$(12);" I N H O U D":PRINT
100 PRINT" 1.gegevens invoeren.":PRINT:PRIN
T" 2.gegevens in DATA-regels zetten.":PRINT
:PRINT" 3.gegevens printen.":PRINT
120 DATA3,naam,straat,postcode,woonplaats
160 INPUT" geef een opdracht";N
180 ONNGOSUB195,900,300:GOTO40
195 N1=N1+1:FORI=N1TO100

```

```

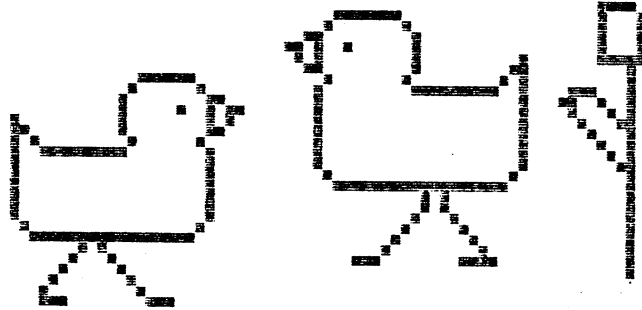
200 PRINTCHR$(12):PRINT"TYPE 'STOP' OM TE S
TOPPEN MET INVOER":PRINT:PRINT"record";I:FQ
RY=0TOZ:PRINTR$(Y)+SPACE$(12-LEN(R$(Y)));:I
NPUTAD$(I,Y):IFAD$(I,Y)="STOP"GOTO260ELSE N
EXTY
210 PRINT:INPUT"alles goed j/n ";G$:IFG$="N
"THEN200ELSENEXTI:N1=I-1:RETURN
260 N1=I-1:RETURN
300 PRINTCHR$(12);"Inhoud van dit bestand":
PRINT:FORI=1TON1:PRINTI;:FORY=0TOZ:PRINTAD$
(I,Y):NEXT:PRINT:NEXT
320 INPUT"Enter voor inhoud";A$:RETURN

```

```

899 REM DATA GENERATOR MANIER A
900 PRINTCHR$(12):INPUT"klaar? ";Z$:A$=STR$
(N1):GOSUB910:FORI=1TON1:A$="":FORY=0TOZ-1:
A$=A$+AD$(I,Y)+CHR$(44):NEXT:A$=A$+AD$(I,Z)
:PRINTA$:JJ=FRE(""):GOSUB910:NEXT
903 PRINT:PRINTCHR$(141)"TYPE NU '1 STOP (E
NTER) 1 (ENTER) '":FORI=XTOX+255:IFPEEK(I)=0
THEN904ELSENEXT:STOP
904 POKEX,(I+1)MOD256:POKEX+1,(I+1)'256:RG=
RG+2:POKEX+2,RGMOD256:POKEX+3,RG'256:POKEX+
4,142:STOP
910 IFSW=1THEN940ELSESA=27000:SW=1
920 FORX=SATOSA+8000:PRINTX,PEEK(X):IFPEEK(X)
=94ANDPEEK(X+1)=94THEN930ELSENEXT
930 RG=PEEK(X-2)*256+PEEK(X-3):X=X+3
940 RG=RG+2:F=X+5+LEN(A$):FF=INT(F/256)
950 POKEX,F-FF*256:POKEX+1,FF
960 RR=INT(RG/256):POKEX+2,RR-RR*256:POKEX+
3,RR:POKEX+4,131
980 FORK=1TOLEN(A$):POKEX+4+K,ASC(MID$(A$,K
,1)):NEXTK
990 POKEX+4+K,0:X=X+5+K:RETURN
998 DATA^^
1000 DATA 0:REM aantal records
50000 REM:.....
:.....
:.....
:.....
:.....
:.....
:.....
50002 REM:.....
:.....
:.....
:.....
:.....
:.....
:.....

```



```

NA INVOEREN GEGEVENS:
998 DATA^^
1000 DATA 2
1002 DATANaus L.,Schoutsakker 5,1871 CZ,Sch
aarl
1004 DATAOosterkim D.,Bovenweg 6,1871 VN,Sc
haarl
1006 REM:.....
:.....
:.....
:.....
:.....
:.....
:.....
50002 REM:.....
:.....
:.....
:.....
:.....
:.....
:.....

```

Manier B.

Vanaf regel 900 staat weer de DATA-generator. Regel 35 moet ook aangepast worden. Bij deze manier moet U achter het programma DATA-regels schrijven. Deze worden voorlopig gevuld met In regel 995 staat eerst 0,.... De nul geeft aan dat er nog geen gegevens zijn ingevoerd. Daar wordt later het aantal records neergeschreven. U moet in elke regel zoveel punten typen dat de DATA er zeker in passen!!!

In de listings kunt U zien wat het verschil is voor en na het invoeren van gegevens. Als U goed kijkt ziet U dat aan het einde van iedere regel ,..... staat (na invoer van gegevens). Dat is het restant van de punten. Om toch in regel 35 alle DATA in de goede volgorde te lezen is een extra READ A\$ ingelast. Die leest de aan het einde van de regel in.

Regel 900 is bijna gelijk aan regel 900 uit manier A. De routine die in regel 910 begint zoekt uit welke geheugenplaats volgt op het laatst gelezen DATA-statement. In de stack wordt op de adressen 25611 en 25612 bijgehouden welk DATA-statement aan de beurt is om gelezen te worden. Bij RESTORE worden deze adressen ook aangepast. Deze DATA-READ-pointer geeft het adres aan dat meteen volgt op de laatst gelezen DATA. Er moet dan 6 bij worden opgeteld om te komen bij het begin van de volgende DATA (2 voor de nextline-pointer, 2 voor het regelnummer, 1 voor het DATA-token en nog een om op de eerste '.' achter DATA te komen. Precies op de achter DATA wordt dan met regel 930 character voor character A\$ gepoked.

Deze manier is wat eenvoudiger. Er komen ook geen problemen met listen zoals bij manier A. Wel oppassen dat er voldoende punten staan in de DATA-regels.

Succes ermee!

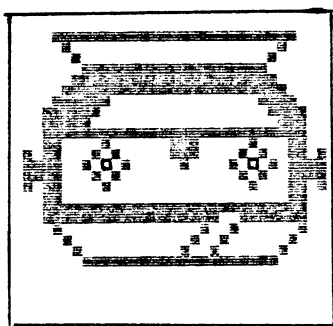
Louis Naus.

```

MANIER B

35 READN1:IFN1=0THEN40ELSEREADA$:FORI=1TON1
:FORY=0TOZ:READA$(I,Y):NEXT:READA$:NEXT

899 REM DATA GENERATOR MANIER B
900 RESTORE995:PRINTCHR$(12):INPUT"klar? "
:A$:A$=STR$(N1)+CHR$(44):PRINTA$:GOSUB910:F
ORI=1TON1:A$="":FORY=0TOZ:A$=A$+AD$(I,Y)+CH
R$(44):NEXT:PRINTA$:GOSUB910:NEXT:STOP
910 XA=PEEK(25611)+PEEK(25612)*256+6
920 X=LEN(A$):FORXI=1TOX
930 POKEXA,ASC(MID$(A$,XI,1)):XA=X+1:NEXT
940 IFVAL(A$)=N1THENREADA$:READA$:RETURN
950 FORIX=0TOZ+1:READX$(IX):X$(IX)="":NEXT
960 RETURN
990 DATA TEST
995 DATA 0,.....
1000 DATA.....
.....
1002 DATA.....
.....
1004 DATA.....
.....
  
```



```

NA INVOEREN INGEVEVENS

990 DATA TEST
995 DATA 2,.....
1000 DATANaus L.,Schoutsakker 5,1871 CZ,Sch
aarl,.....
1002 DATAOosterkim D.,Bovenweg 6,1871 VN,Sc
haarl,.....
1004 DATA.....
.....
  
```

De volgende entry adressen zijn ontdekt in de monitor.
Alle adressen worden gegeven in hex.

Adres	functie	parameters
0005	PDOS	De meeste volgen de aanroepen van CP/M
0018	cassette	Hierover in de volgende nieuwsbrief meer. 0,1,2,3,4,5,6,7 0 initialisatie 1 terugspoelen 2 vooruitspoelen n blokken 3 terugspoelen n blokken 4 einde tape schrijven 5 schrijf geheugen op band 6 lezen geheugen van band 7 opvragen status cassette

Hiervoor dienen de volgende adressen van getallen voorzien te worden:

Adres	Omschrijving
6030	Adres waarna het eerste byte geschreven wordt, c.q. het adres waarvan het eerste byte gelezen wordt.
6032	Lengte van het geheugenblok (in bytes) dat ge transporteerd wordt.
6034	In sommige gevallen staat hier de lengte van een blok op de tape (standaard 1 K bytes). In andere gevallen staat hier ook de lengte van het geheugenblok dat getransporteerd wordt.
6036	Hier staat de naam van de 'file'. Bij cassette-basic is dit slechts een karakter, bij andere rompacks zijn dit 6 karakters voor de naam van de 'file', gevolgd door 3 karakters voor de extensie.
6041	Hier staat het filetype: B voor het extended basic W voor wordprocessor V voor viewdata P voor assemblerprogramma O voor ander filetype
	Ik stel voor om diskbasic aan te geven met de letter C, om onderscheid te kunnen maken met tape-basic.
6042	Bytecode. Dit moet een 0 zijn. (eventueel kan dit gebruikt worden om onderscheid te maken tussen de verschillende nationale keyboards.)
6043	Startadres van een assemblerprogramma.
6045	Laadadres van een assemblerprogramma.
6047	?
604E	Gereserveerd voor toekomstige toepassingen.
604F	Relatieve teller voor de positie van een 1 K blok in de file. De inhoud van dit adres bepaalt hoe ver de tape heen- of terugspoelt bij de cassette acties 2 en 3. (aantal blokken)

AANROEP:	<ol style="list-style-type: none"> 1) Eerst het cassette-controlblock op de juiste manier vullen. 2) De accumulator vullen met de waarde van de te verrichten cassettefunctie. 3) De subroutine aanroepen, die op adres 0018 begint.
001B	Initialiseer het toetsenbord.
0020	Enable het toetsenbord.
0023	Disable het toetsenbord.
0026	Ophalen van toetscode uit de wachtrij.
	Indien er geen toets in de wachtrij is, wordt gewacht tot er een toets wordt ingedrukt.
0029	Opvragen van keyboard-status.
	Indien carry gezet, is de stop-toets ingedrukt.
	Indien zero vlag is gezet is er geen key aanwezig.
002C	Wis keyboard wachtrij.
	Alle ingedrukte toetsen gaan verloren.
002F	Systeemfout. Zet Z80 processor stop.
0032	Beeper routine. Laat 'pieper' piepen.
0035	Wis geheugenblok. Inhoud accumulator * 80 (decimaal) bytes worden gewist, vanaf locatie aangegeven door het register HL.
0038	Toetsenbord routine.
	Het resultaat staat in de accumulator.
	Indien dit FF is, is er geen toets ingedrukt; anders staat de toetscode in de accumulator.
	(Pas op ! De shift-toets werking is omgekeerd. Zonder shift-toets krijgt men kleine letters.)
01A5	Op stack zetten van de registers HL,DE,BC,AF gevolgd door interrupt disable.
01AB	Van stack halen van de registers AF,BC,DE,HL gevolgd door interrupt enable.
01BC	Het initialiseren van de interrupt mode 2 van de Z80-CTC
01BE	Het plaatsen van de speciale karakters (upper-case, printer, diskette en cassette) op het M-model.
	AANROEP : Register B vullen met ASCII-waarde van L,D,T,P. Geheugenplaats 6014 vullen met gewenste geheugenlokatie.
024D	Plaats 'CALL SERVICE' op beeldscherm.
0388	Lees stand-alone programma van cassette.
	(Dit betekent zonder ROM pack)
0431	Plaats een uit meerdere gedeelten bestaande tekst, op verschillende plaatsen op het scherm.
	AANROEP : Laad register HL met start adres bericht. De eerste twee bytes bevatten het adres waar dit gedeelte van het bericht komt te staan. Het derde byte bevat de lengte van dit gedeelte.
	Vervolgens komt het gedeelte van het bericht. Hierna weer een adres voor het volgende gedeelte enz.
	De totale string afsluiten met byte FF.
	OPGELET ! Bezitters van een M-model dienen er rekening mee te houden dat het werkelijke adres waar de boodschap komt te staan twintig (dec.) geheugen plaatsen verder ligt.

Tot zover deze lijst als aanvulling op de eerder gepubliceerde lijst met adressen. Daar is overigens nog een fout in geconstateerd. Adres 60AB geeft de maximale breedte (aantal karakters) die de printer aankan. Hogere waarden resulteren in dubbel printen.

We zullen nu enige voorbeelden behandelen van het op cassette wegzetten en teruglezen van een stuk geheugen.

De uitgangspositie van de tape is willekeurig.
De tape bevat programma's die gewist dienen te worden.

Het wegzetten van een stuk geheugen (&H1000 tot &H5000) gaat als volgt :

In bewoordingen : spoel de tape terug naar het begin
 formatteer de tape
 spoel 1 blok terug
 schrijf het geheugenblok naar tape
 schrijf einde tape marker.

In programmatuur:

;	.ORG	E000H	;startadres &HE000
;	LD	A,01H	;spoel tape terug
;	RST	18H	;voer functie uit
;			formatteer tape
;	LD	A,04H	;formatteer tape
;	RST	18H	;voer functie uit
;			spoel tape 1 blok terug
;	LD	A,01H	;1 blok terugspoelen
;	LD	(604FH),A	;
;	LD	A,03H	;spoel n blokken terug
;	RST	18H	;voer functie uit
;			laad filecontrolblock met de
;			juiste informatie
;	LD	HL,1000H	;begin te verplaatsen
;			;adressen
;	LD	(6030H),HL	;data overdracht adres
;	LD	(6045H),HL	;laad adres
;			het startadres is ook 1000H
;	LD	(6043H),HL	;start adres
;			lengte van de file op cassette
;	LD	HL,4000H	;lengte van de file
;	LD	(6032H),HL	;
;			lengte van een blok op de tape
;	LD	HL,400H	;lengte 1 Kbyte
;	LD	(6034H),HL	;

:	filenaam opgeven
:	
:	LD A,46H ;filenaam is 'F'
:	LD (6036H),A ;
:	
:	file type is assembler programma
:	
:	LD A,50H ;assembler programma
:	LD (6041H),A ;
:	
:	file controlblock is nu gereed
:	voor het op tape zetten van het geheugen
:	
:	LD A,05H ;schrijf geheugen naar tape
:	RST 18H ;voer functie uit
:	
:	schrijf einde tape marker
:	
:	LD A,04H ;schrijf einde tape marker
:	RST 18H ;voer functie uit
:	
:	terug naar aanroeper
:	
:	RET

We gaan nu het 'programma' weer inlezen.

Hierbij wordt uitgegaan van de volgende veronderstellingen :

De uitgangspositie van de tape is willekeurig.

De positie van het programma is willekeurig.

We gaan het programma op een andere plaats in het geheugen zetten, om eventueel het programma op diskette te zetten.

IN WOORDEN :

- a) spoel tape terug naar het begin
- b) lees header van de tape
- c) indien dit niet het gezochte programma is, spoel dan n blokken vooruit en vervolg bij stap b.
- d) verander het data transferaddress, zodat het programma op de juiste plaats in het geheugen komt.
- e) lees de file van tape en zet deze in het geheugen.

IN PROGRAMMATUUR :

:	.ORG A800H	;startadres &H800
:		
:	LD A,01H	;spoel tape terug
:	RST 18H	;voer functie uit
:		
:	lees de header van het blok	
:		
:	LD HL,0000H	;valid record length
:	LD (6034H),HL	;
:	INC HL	;
:	LD (6032H),HL	;
:	LD A,06H	;lees header

```

RST      18H      ;voer functie uit
JR      NZ,ERROR ;fout gekonstateerd
;
;
;
test op korrekte filenaam
LD      A,'F'    ;filenaam
LD      HL,6036H ;filenaam van tape
CP      (HL)    ;vergelijk
JR      NZ,NEXTHD ;lees volgende header
LD      A,'P'    ;assemblerprogramma ?
LD      HL,6041H ;file type van tape
CP      (HL)    ;vergelijk
JR      Z,REWIND ;zoek een assemblerprog
;
;
spoel zoveel blokken vooruit als nodig
is om de header van de volgende file te kunnen lezen
;
;
de lengte van de file staat in lokatie 604FH
en staat op de juiste waarde
;
;
LD      A,02H   ;spoel n blokken vooruit
RST     18H    ;voer functie uit
JR      NEXTHD ;lees next header
;
;
spoel een blok terug
;
;
REWIND:  LD      A,01H   ;1 blok terugspoelen
LD      (604FH),A ;
LD      A,03H   ;n blokken terugspoelen
RST     18H    ;
;
;
;
plaats nu het gewenste datatransfer adres
in datatransfer adres.
;
;
LD      HL,B000H ;datatransfer adres
LD      (6030H),HL ;
;
;
de resterende gegevens zoals lengte van de file,
het aantal tape-blokken enz. staat door het lezen
van de header nog steeds in het geheugen.
;
;
;
We kunnen nu de file gaan lezen.
;
;
LD      A,06H   ;lees file
RST     18H    ;voer functie uit
RET     Z      ;indien zero foutloos
;
;
;
afhandelen van opgetreden fouten
de foutcode (ASCII waarde) staat in de accumulator.
;
;
ERROR:  PUSH     AF      ;bewaar foutcode
LD      HL,CASMES ;cassette error
CALL    PRMES    ;schrijf op scherm
INC     HL      ;schrijf een spatie
INC     HL      ;
POP     AF      ;foutcode
LD      (HL),A  ;schrijf op scherm
RET
;
;
CASMES: .WORD     5028H ;lokatie op scherm
        .BYTE     0EH  ;lengte mededeling
        .ASCII    "cassette error" ;bericht
        .BYTE     0FFH ;einde bericht

```

Tot zover de cassette-funkties.

Lambert Knapen.

In P 2000 gg nieuwsbrief no 3 schreef Gerben Mooiweer het artikel "Inleiding programmeren in machinetaal."

Als U dat hebt uitgeprobeerd dan hebt U gemerkt dat in machinetaal het hele beeldscherm in een klap vol staat met letters. Daarna wordt hetzelfde gedaan in basic en dan duurt het veel langer.

Machinetaal schrijf je meestal in DATA-regels; hoewel het ook kan met speciaal geschreven programma's. Bij sommige programma's uit de Tape-o-theek worden machinetaalprogramma's aan BASIC gekoppeld door de eindpointers van het programma te verzetten. Als U die programma's runt worden de machinetaalprogramma's weer op hun plaats gezet. U merkt dat doordat U niet hoeft te wachten op het inlezen van de data. Het programma begint dan dus sneller te werken.

Voor de beginners is de manier die Gerben heeft beschreven de duidelijkste en best hanteerbare.

Hieronder volgen de machinetaalgegevens van twee subroutines die voor bijna alle programma's gebruikt kunnen worden.

Probeer eens om het programma uit Nieuwsbrief 3 zo te veranderen dat de onderstaande machinetaal-subroutines gaan werken.

Als U alle aanwijzingen in het bedoelde artikel volgt kan het haast niet mislukken.

Het gaat om twee GET-routines.

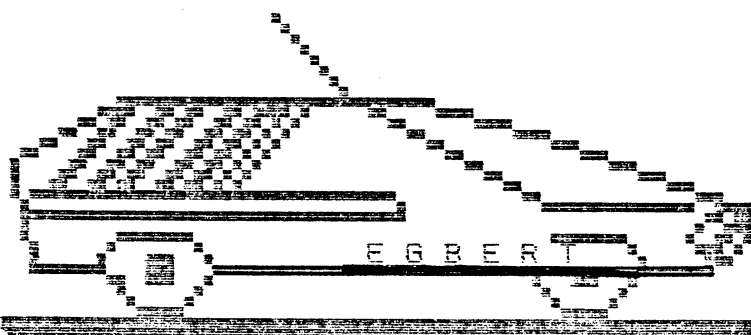
Als U deze routines met USR aanroept wacht de P 2000 tot U een toets indrukt. Als een toets is ingedrukt geeft de eerste routine de ASCII-waarde van de ingedrukte toets.

De tweede routine geeft de toetscode.

USR1 CD 4D 10 77 C9 (geeft ASCII)

USR2 CD 26 00 77 C9 (geeft toetscode)

SUCCES. (L.N.)



Men kan tegenwoordig heel veel apparaten kopen en ook (dure) accessoires ervoor. Denk bij voorbeeld aan een cassetdeck en een afstandsbediening ervoor. Of denk in ons geval aan de P2000 en als aanvulling al die leuke schakelingetjes (zoals lichtpen, joy-stick, PIO enz). Een nadeel is wel dat men al deze schakelingetjes zelf moet bouwen. Een voordeel is echter dat de schakelingen er veel goedkoper door worden dan dat men ze kant en klaar bij één of andere fabrikant zou kopen. In dit artikelje gaan we beschrijven hoe men te werk gaat om zo'n schakeling te bouwen. We gaan ervan uit dat er een gedrukte schakeling (print) bestaat voor de toepassing die we wensen te maken. (over het ontwerpen en maken van gedrukte schakelingen gaan we het nu niet hebben)

HET GEREEDSCHAP.

Welke werktuigen hebben we nu allemaal nodig? Eerst en vooral een goede soldeerbout. Dit betekent een type van 15 à 35 Watt (het best is 20 à 25 Watt). Het veiligst (wat betreft het vast solderen van CMOS-IC's) is een type van soldeerbout die werkt op een lage spanning en die gescheiden is van het net (met batterijen of met een nettransformator). Natuurlijk moeten we ook iets hebben om mee te solderen. Dit heet "soldeer". Het is te verkrijgen in 1001 uitvoeringen. Men neemt een soort die een vloeimiddel bevat en een dikte heeft van max. 1mm. Het vloeimiddel bevindt zich in de kern van de soldeerdraad en het dient om de koperoppervlakken te reinigen terwijl men soldeert. Potjes of tubes met zogenaamde "soldeer pasta" komen absoluut niet in

aanmerking en zouden het best in een vuilnisemmer gedeponneerd worden. Ze zijn chemisch agressief en tasten alle elektronische componenten aan. Verder heeft men nog een zijknip-tang nodig (om de draden van de gesoldeerde onderdelen op lengte af te knippen) en een bektang met fijne, platte bekken (om de draden van weerstanden ed. om te buigen). Enkele schroevendraaiers en eventueel een printhouder kunnen nuttige diensten bewijzen.

DE SOLDEERBOUT.

Als men reeds een soldeerbout heeft, dan kun je gerust dit gedeelte overslaan. Indien je echter een nieuwe bout wenst te kopen (of pas gekocht hebt) lees dit dan eerst voor hem in gebruik te nemen. Bij de aankoop van een soldeerbout kan men meestal kiezen tussen twee soorten soldeerstiften. De goedkoopste zijn de gewone, koperen stiften.

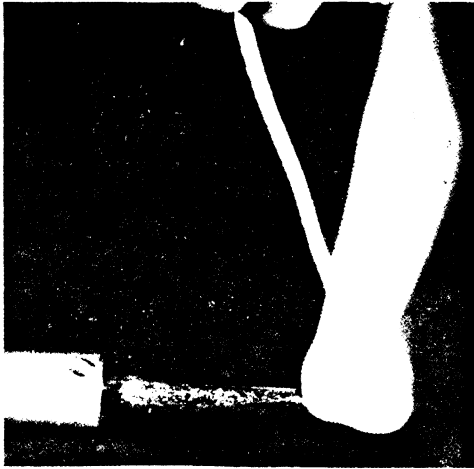
De iets duurdere, maar veel betere, zijn de zg. "long-life" stiften. Deze zijn in feite ook koperen stiften, maar ze zijn bedekt met een laagje nikkel. Dit maakt dat ze veel langer meegaan dan de gewone, koperen stiften. Een koperen stift moet men na verloop van tijd weer in de juiste vorm vijlen, omdat de punt gaat oxidieren en dus inbranden.

Een long-life stift gaat zeker tot twee jaar mee zonder dat er een merkbare slijtage op komt. Let wel op: omwille van het laagje nikkel dat erop ligt, mogen deze stiften zeker niet bijgevijsd worden! Anders vijl je er gewoon het nikkel-laagje van af en zijn het gewone koperen stiften geworden.

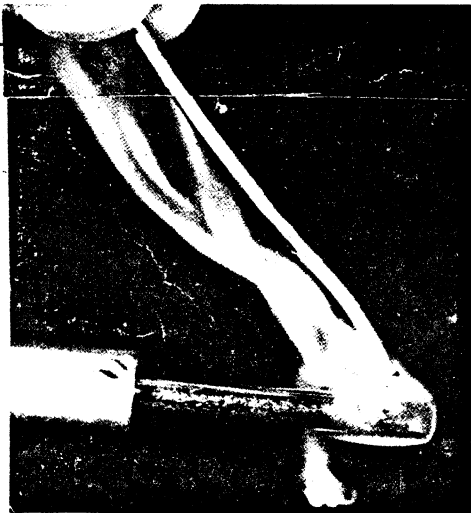
Elke bout moet voor het gebruik ook eerst vertind worden. Dit moet men maar eenmaal doen.

Het gaat als volgt:

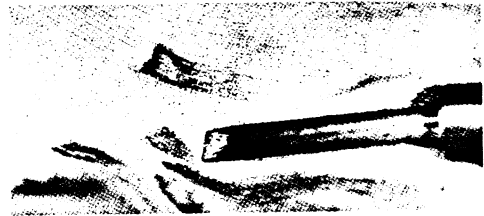
- * Sluit de bout aan en laat hem gedurende een drietal minuten op temperatuur komen.
- * Neem nu een droge doek en veeg de punt van de bout mooi af. (let op dat je je vingers niet verbrandt)



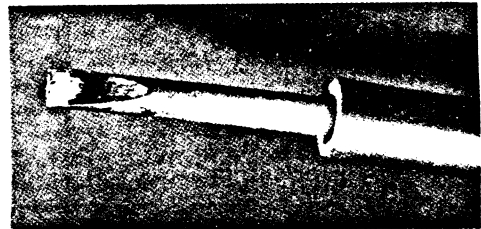
Afb.1: Ongeveer drie minuten na het inschakelen heeft de soldeerbout de juiste werkt temperatuur bereikt. Nu moet de spits worden "vertind". Daarvoor wordt de soldeerdraad tegen de hete spits gehouden. Zoals u ziet, verdampt het zich in de soldeerdraad bevindende vloeimiddel. Daarbij wordt de soldeerpunt gereinigd.



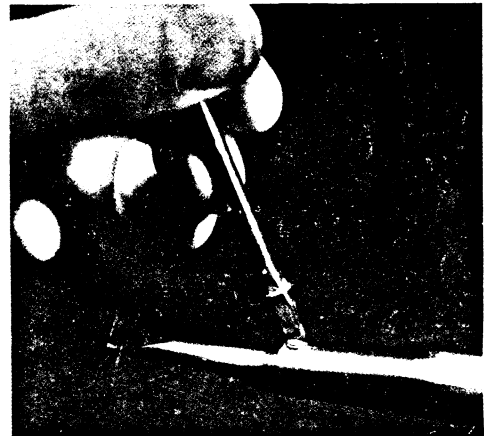
Afb.2: Het is eigenlijk volkomen normaal, wanneer de soldeerstift de eerste keer bij het vertinnen in zekere zin "nog geen soldeer wil aannemen". Het gesmolten soldeertin druipt eraf. Alleen al daarom is een geschikte ondergrond ter bescherming van het werktafelblad belangrijk.



Afb.3: Omdat het vloeimiddel niet altijd volledig verdampt, vormt zich op de soldeerstift tenslotte een zwarte korst. Deze moet - het beste met een droge, zachte doek - steeds weer even worden afgeveegd.



Afb.4: Hier is het afgeschuinde deel van de soldeerstift ongeveer voor de helft vertind. De soldeerbout is daarmee voor gebruik gereed.



Afb.5: Voor het onderhoud van de soldeerbout behoort men zo nu en dan de bevestigingsschroef los te draaien om de soldeerstift eruit te nemen. Alleen zo kan ook het zich in het binnenste van het verwarmingselement bevindende deel van de soldeerstift worden schoongemaakt, dus worden ontdaan van de door de hitte ontstane koperoxide schilfers. Daarna moet de bevestigingsschroef natuurlijk weer goed worden vastgedraaid.

- * Neem nu het soldeer en breng het aan op de punt van de bout. Het soldeer zal smelten en het hars verdampt. Er zal een soldeerdruuppel op de stift blijven liggen. Deze druppel wrijft men eraf met met de droge, katoenen doek. Er blijft nu een vertind plaatsje achter op de punt van de stift. Men herhaalt dit nu tot de punt voor de helft mooi vertind is.
- * Als dit gebeurd is, volstaat het om, telkens als er zwarte korstjes op de punt komen, deze met een doek af te vegen om de punt proper te houden. Weet dat de kwaliteit van de soldeerstift bepalend is voor de kwaliteit van de soldeerverbinding. Dus onderhoud uw punt goed en veeg hem telkens goed af!
- * Nu is uw bout klaar voor gebruik.

HET MONTEREN VAN PRINTJES.

Een voor het solderen geschikt werkblad is belangrijk om het oppervlak van de werktafel tegen inwerking van hitte te beschermen. Het eigelijke solderen moet snel geschieden. De aansluitdraden worden het beste meteen met soldeer voorvertind. Het gevoel voor de juiste soldeerduur krijgt men met de tijd. Die soldeerduur mag noch te lang, noch te kort zijn. (ongeveer 4 à 6 sec.) Het soldeer moet zonder moeite vloeien en tot "rust" komen. Daarbij hoort natuurlijk ook een vaste hand.

Montagevolgorde:

Wij beginnen bij de montage met de lage, dan de middelhoge en tenslotte de hoge onderdelen. Dit uitgezonderd de halfgeleiders, deze worden het laatst gemonteerd. De print ligt bij deze volgorde steeds nagenoeg gelijkmatig op de onderdelen. Wij hoeven deze dan bij het vast solderen ook niet nog eens vast te houden. Bijvoorbeeld dus eerst de weerstanden of dioden. De aansluitdraden met een stevig pincet of tang op de goede maat omgebogen.

Dat mag niet zomaar vlak bij de weerstand gebeuren, omdat de aansluitdraden kunnen afbreken. Wanneer we alle even hoge onderdelen hebben doorgestoken, draaien we de print op de soldeerzijde, de weerstanden blijven op hun plaats. Indien nodig buigen we de lange aansluitdraden een klein beetje naar buiten, zodat ze vóór het solderen al contact maken. De draadeinden worden aan de geleidebaan vastgesoldeerd en met een zijkniptang kort afgeknipt. Bij verkeerde montage moet men zonder moeite de onderdelen weer los kunnen solderen zonder de print te beschadigen.

Nu komen de iets hogere onderdelen aan de beurt, waaronder condensatoren, trimmers enz. Doch nog geen halfgeleiders. Waarop moet worden gelet: de elektrolytische condensatoren (elco's) moeten natuurlijk voor de aangelegde spanning geschikt zijn en slechts met de juiste polariteit worden vastgesoldeerd.

Als laatste monteren we de halfgeleiders, waaronder transistoren, IC's, FET's, enz... Hierbij moeten we voor één ding oppassen. Halfgeleiders, (IC's daaronder begrepen) zijn gevoelig voor warmte en te hoge spanning. De solderbout moet dus, zoals reeds gezegd, een onberispelijk aardcontact hebben om kruipstromen uit te sluiten. Een bout die aangesloten is op het net, mag zeker niet gebruikt worden om CMOS-IC's te solderen. Hiervoor moet een laagspanningsbout gebruikt worden, die goed geaard is (of veel goedkoper: gebruik IC-voetjes). Over MOS-IC's later meer.

De aansluitingen van de transistoren mogen niet vlak onder het huis worden omgebogen, want het inwendige zou, afhankelijk van het type, mechanisch kunnen worden vernield. Tussen het huis en de ombuigplaats houden we een tangetje met fijne, platte bek en buigen de aansluitdraden over de kant van het tangetje om. De afstand tussen het transistorhuis en de soldeerplaats moet ongeveer 10 mm bedragen, opdat we de halfgeleider thermisch niet overbelasten. Het vast solderen mag niet langer dan 5 sec duren. Wie helemaal zeker wil zijn, houdt tijdens het solderen, vlak boven de soldeerplaats, de betreffende aansluitdraad bovendien nog met een tangetje of pincet vast en leidt zo een deel van de warmte af.

Wat de thermische belasting betreft wordt in samenhang hiermee nog op het volgende gewezen. Halfgeleiders met stromen in de grootteorde van een paar mA tot ongeveer 400mA vragen een koelvin of een koelster. Bij grotere stromen zijn sterker geprofileerde koellichamen vereist. Heel handig is het de kant en klaar gemonteerde print te voorzien van soldeerbare aansluitstiftjes. Bedrijfsspanningsleidingen en andere toevoerleidingen kunnen zo op ieder moment worden los- of vastgemaakt.

Tenslotte loopt men de montage nog een keer heel accuraat, aan de hand van het schema, stap voor stap na. Hebben we er ons van overtuigd, dat bij het solderen niet bij vergissing schakelbanen met tin werden kortgesloten, dan kan de voedingsspanning (met juiste polariteit) worden aangesloten.

Vertoont de schakeling geen enkele reactie of geeft hij zelfs "rooksignalen" af, dan is er een fout in. Het is bekend, dat men een bepaalde fout keer op keer over het hoofd kan zien. Of misschien hebben we ons al te zeer op de juistheid van het schema of de montage verlaten.

HET BEHANDELEN VAN CMOS-IC's

Om het stroomverbruik van de schakeling laag te houden, worden steeds meer CMOS-IC's toegepast. Deze schakelingen gebruiken slechts heel weinig stroom, maar ze zijn heel gevoelig voor statische ladingen! Een mens kan gemakkelijk enkele kilovolt aan statische elektriciteit bevatten. (denk aan het "knetteren" van wollen of nylon kledingstukken bij het uittrekken). Als men in zo'n toestand een CMOS-IC vastneemt met de hand, dan is hij 9 kansen op 10 kapot, omdat de statische lading van uw hand naar de pennen van het IC afgeleid wordt.

De duizenden poortschakelingen die zich in het IC bevinden, zijn niet bestand tegen zo'n grote spanning en zullen inwendig doorslaan. Daarom zijn dergelijke IC's steeds verpakt in aluminiumfolie of ze zitten met hun pennen in een zwart "mousje" geduwd.

Zo'n mousje bevat koolstof dat geleidend is voor elektrische spanning, zodat, net als bij de aluminiumfolie, alle pennen van het IC met elkaar verbonden zijn. Daardoor kan er geen spanningsverschil optreden tussen twee pennen van het IC. Deze methode vermijdt dat er schade aangericht wordt aan het inwendige van het IC door transport en opslag.

Daarom moet men speciale voorzorgsmaatregelen in acht nemen als men zo'n IC in een print wil steken.

* Eerst en vooral, draag geen wollen of nylon kledingsstukken en ontlaad uw lichaam van statische elektriciteit door even een metalen voorwerp aan te raken (vb. een waterkraan, radiator, aluminium ramen of deuren, enz..).

* Leg de print nu met zijn koperzijde op een grote koperen of aluminium plaat. Druk de print eventjes goed op de plaat en raak de plaat ook even goed aan met de handen, zodat alles op hetzelfde potentiaal staat.

* Neem nu de aluminiumfolie of het mousje met de IC erin of erop en wrijf het ook even goed op de plaat. Nu heeft het IC dezelfde potentiaal als u en de print. Neem het IC uit zijn bescherming en breng hem aan op de print. (Steek hem in een voetje of soldeer hem voorzichtig met een geschikte bout.)

* Eens dat de IC in zijn schakeling zit, is hij tegen statische lading beveiligd door de componenten die errond op de print zitten. Nu is hij dus veilig voor aanrakingen.

Ik hoop dat ik wat meer uitleg heb gegeven over het bouwen van printjes. Ik wens iedereen verder veel succes bij het maken van elektrische schakelingen.

Voor degenen onder ons die weinig of niets weten van electronica hebben we hieronder een aantal tips. Als je een electronica plaatje bekijkt zie je aan de ene zijde meestal alleen gaatjes, dit is de componenten zijde, en aan de andere kant gaatjes met sporen, dit is de soldeer zijde. Let erop dat aan de soldeer zijde de rondjes, waarin een gaatje is geboord, blank zijn. De laklaag kan je verwijderen met een fijn schuurpapiertje of een pannespons. Gebruik een soldeerbout niet zwaarder dan 25 W en gebruik een harskern soldeer van een of anderhalve mm².

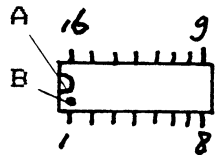


fig.1



fig.2



fig.3

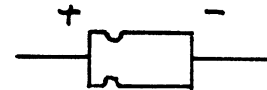


fig.4

Op het schema zie je de componenten opstelling. Let op dat de componenten zo geplaatst worden als in het schema want anders worden de componenten vernield.

Een IC (fig.1) heeft altijd een aanduiding om pootje nr.1 aan te geven, dit kan een inkeping zijn (A), of een stip (B). Leg de inkeping of de stip aan de linker kant en je hebt links onder pootje 1.

De zenerdiode (fig.2) heeft een anode en een kathode, het zwarte deel is de kathode, en op de zenerdiode zelf is de kathode aangegeven met een zwarte ring.

Een elco (fig.3) heeft ook een positieve en negatieve kant en is op de component ook aangegeven met met + of - of een inkeping rondom (fig.4).

Om een IC makkelijk te vervangen kan je hem op een voetje monteren d.w.z het voetje soldeer je vast en het IC klem je erin. Heb je bijvoorbeeld een condensator nodig van 0.1uF/16V en je hebt er een van 0.1uF/63V dan kan 0.1uF/63V ervoor in de plaats gezet worden. De waarde uF moet gelijk zijn maar de aangegeven spanning op de condensator mag gerust hoger zijn.

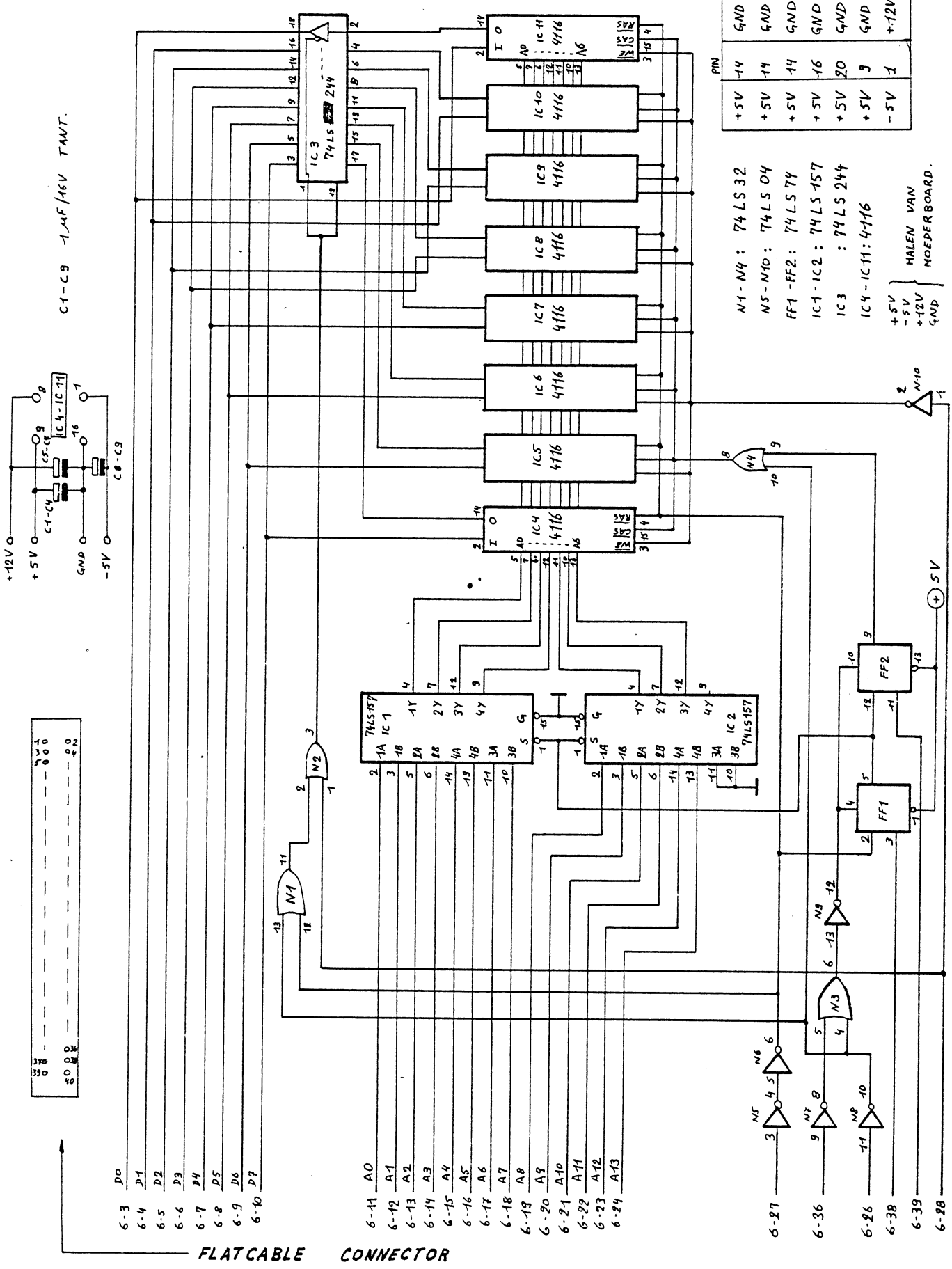
We hopen dat het monteren van componenten op een plaatje door deze tips iets makkelijker gaat.

Hans Oudshoorn

Op de volgende pagina het schema voor een 16 K geheugenuitbreiding waarvoor niet het normale 32 k extension board niet vereist is. Deze uitbreiding wordt met een flat cable in de P2000 aangesloten op het betreffende slot voor geheugenuitbreiding.

Getekend door: **CASTELEIN Patrick**
M. Sabbestraat, 3
B 8400 OOSTENDE
☎ 059/80.35.39

P2000 16kB RAM



C1-C9 1µF/16V TANT.

PIN	PIN
+5V	GND
+5V	GND
+5V	GND
+5V	GND
+5V	GND
+5V	GND
-5V	+12V

N1-N4 : 74LS32
N5-N10 : 74LS04
FF1-FF2 : 74LS74
IC1-IC2 : 74LS157
IC3 : 74LS244
IC4-IC11 : 4116
+5V } HALEN VAN
-5V } MOEDERBOARD.
+12V }
GND }

FLATCABLE CONNECTOR

Het Philips Disk Operating Systeem.

I. PDOS organisatie.

PDOS is opgebouwd uit 4 delen.

Deze delen zijn :

- a) Entry points + parameters
- b) Monitor I/O voor seriële randapparatuur.
- c) Het BDOS (Basic Disk Operating system)
- d) Het gebruikersprogramma

I.1 Geheugengebruik bij PDOS

Het is mogelijk om PDOS tijdelijk uit het geheugen te verwijderen, om een extra groot reken programma te laten draaien.

Men kan namelijk PDOS overschrijven met bijv. een aantal eigen assembler routines, die bij het laden van diskette, eerst op een andere plaats gebufferd worden en vervolgens naar het PDOS gebied overgeplaatst worden. Wanneer men nu gebruik wenst te maken van PDOS, dan dient men dit eerst weer in het geheugen te laden.

Dit opnieuw booten van PDOS gebeurt door het uitvoeren van een CALL 0E90H vanuit BASIC of machinetaal.

Het enige dat gebeurt bij dit booten is, dat PDOS in het geheugen komt te staan, zonder dat verder het systeem geïntialiseerd wordt.

Men dient bij dit laden wel er aan te denken, dat de systeemdisk in drive A zit, en dat het machine taalprogramma niet in de tweede geheugenbank zit.

I.2 Hoe dient men PDOS aan te roepen ?

Het aanroepen van PDOS gaat als volgt :

- a) Geef een funktienummer op in het Cregister.
- b) Definieer een pointer naar een FCB (file controll block) in het registerpaar DE.
- c) Geef een CALL instructie naar het entryadres
- d) Controleer aan de resultaten of de werking korrekt was.

I.3 Welke functies zijn mogelijk bij PDOS ?

Er zijn 25 functies mogelijk nl.

- a) Reset systeem disk
- b) Selecteer disk
- c) Open file
- d) Sluit file (close)
- e) Zoek eerste file (Search for first)
- f) Zoek volgende file (Search for next)
- g) Verwijder file (Delete file)
- h) Lees sequentieel (Read sequential)
- i) Schrijf sequentieel (Write sequential)
- j) Maak file (Make file)
- k) Herbenaem file (rename file)
- l) Initializeer DMA-adres (Set DMA-address)
- m) Wis directory (Delete disk)
- n) Lees file sequentieel (Read file sequential)
- o) Schrijf file sequentieel (Write file sequential)
- p) Lees directory (Read directory)
- q) Lees random (Read random)
- r) Schrijf random (Write random)
- s) Kopieer disk (Copy disk)
- t) Kopieer file (Copy file)
- u) Bepaal vrije ruimte op disk (Free disk space)
- v) Zet motor aan (Motor on)
- w) Zet motor uit (Motor off)
- x) Maak schrijven mogelijk (reset writeprotect)
- y) Maak schrijven onmogelijk (set writeprotect)

I.4 Uitleg bij deze functies

- a) Reset system disk.

Entry parameters : Register C : 0DH

Return parameter : none

Reset alle drives.

Zet alle drives in lees/schrijftoestand en selecteer drive A.

Het default DMA adres wordt op de default waarde gezet. Bij deze functie vindt geen terugmelding plaats.

- b) Select disk

Entry parameters : Register C : 0EH
Register E : drive nummer
(0 t/m 15)

Return parameter : none

Deze functie kiest een bepaalde drive voor en plaatst deze in een on-line toestand.

Indien een leesopdracht gegeven wordt met drive nummer 0, dan wordt de geselecteerde disk aan gestuurd.

Geeft men echter een ander drive nummer op, dan gebruikt men de andere disk, zonder dat deze geselecteerd wordt.

c) Open file

Entry parameters : Register C : OFH
Register DE : FCB adres

Return parameter : Register A : directory cod

De bedoeling van deze funktie is het "inlezen" van alle parameters, die te maken hebben met de plaats van de file op de diskette. Het is zelfs mogelijk om met vraagtekens een meer algemene naam te kreeeren. Indien er meerdere entries zijn in de directory, die aan deze naam voldoen, wordt de eerste file geopend. Het OPENEN van een file is dus NODIG, voordat er gelezen of geschreven kan worden van of naar het bestand. Het resultaat geeft bij een succesvolle aktie het volgnummer van de file in de directory. Indien de file niet bestaat vindt men in de accumulator de waarde OFFH (255).

Indien men een file wil gaan lezen, dan moet men er wel aan denken om de huidige record pointer op 0 (nul) te zetten.

d) Close file

Entry parameters : Register C : 10H
Register DE : FCB adres

Return parameter : Register A : directory code

Een CLOSE operatie heeft als doel, de huidige plaatsing van de file op de diskette, in de directory te doen opnemen. Indien dit niet gebeurt, blijft de file de positie innemen, die hij innam bij het openen van de file, of indien het een make-commando was, kan men de file zelfs niet terugvinden op de diskette. Hieruit kan men afleiden, dat wanneer een file alleen gelezen dient te worden, een open-commando noodzakelijk is, maar een close-commando niet.

Het resultaat is het zelfde als voor het open-commando. (255 voor een niet bestaande file)

e) Search for first

Entry parameters : Register C : 11H
Register DE : FCB adres

Return parameter : Register A : directory code

Zoekt de eerste voorkomende entry in de directory, die voldoet aan de filenaam in het FCB.

Indien er geen file voldoet aan het FCB dan is het resultaat OFFH (255).

Het is mogelijk om met vraagtekens te werken, waardoor meerdere entries in de directory kunnen voldoen aan het FCB.

f) Search for next

Entry parameters : Register C : 12H

Return parameter : Register A : directory code

De werking van deze functie is gelijk aan de functie Search for first, alleen start deze functie op de plaats, waar Search for first is opgehouden. De functie die voor Search voor next is aangeroepen, dient Search for first te zijn.

g) Delete file

Entry parameters : Register C : 13H
Register DE : FCB adres

Return parameter : Register A : directory code

De DELETE FILE functie verwijdert ALLE entries uit de directory, die voldoen aan het FCB. De filenaam in het FCB mag vraagtekens bevatten.

Return parameter:

0 : succesvol
1 : file niet gevonden
2 : disk niet gereed
3 : directory vol
4 : disk vol
5 : fout in file
6 : fout bij sluiten van de file
7 : CRC fout bij directory lezen
8 : einde van de file bereikt
9 : diskette write protected
A : niet aangesloten device nummer
B : file write protected
C : fout op systeem disk

h) Read sequential

Entry parameters : Register C : 14H
Register DE : FCB adres

Return parameter : Register A : directory code

Indien het FCB adres geactiveerd is door een OPEN of MAKE kommando, leest het READ SEQUENTIAL kommando het volgende 256 bytes record in het geheugen vanaf het huidige DMA adres.

Wanneer het einde van een extent bereikt wordt, wordt de extentcounter opgehoogd en de record counter gereset.

Indien gewenst dient men zelf de extent of record counter aan te passen.

In het geval dat het resultaat ongelijk nul is, betekent dit dat er geen volgende leesopdracht gegeven mag worden. (Resulteert in foutmelding)

Return parameter :

0 : succesvol
1 : voorbij einde van de file
2 : nog niet geschreven record in random access file

Het DMA adres is standaard tenzij gezet m.b.v. de functie Set DMA (1AH)

i) Write sequential

Entry parameters : Register C : 15H
Register DE : FCB adres

Return parameter : Register A : directory code

Indien het FCB adres geactiveerd is door een OPEN of MAKE kommando, schrijft het WRITE SEQUENTIAL kommando het volgende 256 bytes record naar de schijf vanaf het huidige DMA adres.

Wanneer het einde van een extent bereikt wordt, wordt de extentcounter opgehoogd en de record counter gereset.

Indien gewenst dient men zelf de extent of record counter aan te passen.

In het geval dat het resultaat ongelijk nul is, betekent dit dat er geen volgende schrijfoopdracht gegeven mag worden. (Resulteert in foutmelding)

Return parameter :

0 : succesvol
1 : fout in extending van de file
2 : geen ruimte meer op de diskette

Het DMA adres is standaard tenzij gezet m.b.v. de functie Set DMA (1AH)

j) Make file

Entry parameters : Register C : 16H
Register DE : FCB adres

Return parameter : Register A : directory code

Het Make-commando is het zelfde als het Open-commando, met als enige uitzondering dat een niet in de directory voorkomende naam opgegeven dient te worden.

Na het Make-commando hoeft dus geen Open-commando te worden uitgevoerd.

PDOS kreeert de file en initialiseert de directory en geheugen. De programmeur dient er zelf zorg voor te dragen, dat er geen dubbele filenaam gekreeerd wordt. Indien nodig dient men eerst een Delete file commando uit te voeren.

Indien het resultaat OFFH (255) bevat, dan is er geen ruimte meer in de directory.

k) Rename file

Entry parameters : Register C : 17H
Register DE : FCB adres

Return parameter : Register A : directory code

Het Rename commando verandert alle entries met de naam zoals deze staat in de eerste 16 bytes, in de naam zoals deze staat in de tweede 16 bytes. Het drivenummer wordt bepaald door byte 0 van het FCB, terwijl het drivenummer van het tweede ge deelte nul (0) moet zijn.

Return parameter :

0 : Rename commando succesvol uitgevoerd.
1 : File niet in directory.
2 : Disk niet gereed.
3 : Directory vol.
4 : Disk vol.
5 : Fout in de file.
6 : Fout tijdens het sluiten van de file.
7 : CRC fout tijdens het lezen van de directory.
8 : Voorbij het einde van de file.
9 : Diskette tegen schrijven beveiligd.
A : Ongeldig drivenummer.
B : File tegen schrijven beschermd.
C : Fout op de systeemdiskette.

1) Set DMA address

Entry parameters : Register C : 1AH
Register DE : DMA adres

DMA is een afkorting voor Direct Memory Access. Dit is een principe waarbij een aparte chip gedurende enige tijd, toegang krijgt tot het geheugen, waarbij de processor gedurende enige tijd gestopt wordt.

Deze toepassing wordt vooral gebruikt bij die processen waarbij met een zeer grote snelheid data getransporteerd wordt. (bijv. Disk access).

In PDOS heeft DMA een andere betekenis. Hier is het DMA adres het adres vanaf waar de eerst volgende 256 bytes gelezen of waarnaar de eerste 256 bytes geschreven worden.

Bij het RAM-PDOS (Philips) is het standaard DMA-adres 6200H.

De SET DMA adres functie heeft als doel dit adres te wijzigen.

Het DMA-adres blijft deze gewijzigde waarde behouden tot een volgend Set DMA commando, Reset system disk, koude start of warme start.

m) Wis directory van diskette.
(Format diskette)

Entry parameters : Register C : 28H
Memory 5000H: disk adres (A8H)
Register E : drive nummer

Return parameter : Register C : Directory code

Het Format diskette commando is door Philips omgebouwd tot het Wis directory commando.

Het is mogelijk om eventueel het Wis directory commando weer terug te brengen tot het Format diskette commando.

Indien het formatteren succesvol afgesloten wordt, is het resultaat 0 in register C.

Resultaat:

'0' : Succesvol
'1' : File niet gevonden
'2' : Disk niet gereed
'3' : Directory vol
'4' : Disk vol
'5' : Fout in file
'6' : Fout tijdens Close commando
'7' : CRC fout tijdens lezen van directory
'8' : Voorbij einde van de file
'9' : Diskette tegen schrijven beschermd
'A' : Verkeerd drivenummer
'B' : File tegen schrijven beschermd
'C' : Fout op systeem disk

n) Read file sequential

Entry parameters : Register C : A9H
Memory 5002H : FCB adres
Register HL : DMA adres

Return parameter : Register C : PDOS errorcode

Indien het de file op disk geadresseerd is door een OPEN commando, leest de READ FILE SEQUENTIAL commando, de volgende 256 bytes van de file in het geheugen vanaf adres DMA. Het FCB wordt automatisch aangepast voor een volgende leesopdracht. (Dit houdt in het automatisch ophogen van de recordcounter of extentcounter).

Return code :

'0' : leesopdracht succesvol uitgevoerd
'1' : file niet gevonden
'2' : disk niet gereed.
'3' : directory vol
'4' : disk vol
'5' : fout in file
'6' : fout tijdens close opdracht.
'7' : CRC four tijdens lezen van data
'8' : einde van de file bereikt.
'9' : disk beveiligd tegen schrijven.
'A' : niet bestaand drive-nummer.
'B' : file beveiligd tegen schrijven.
'C' : fout op systeemdisk.

o) Write file sequential

Entry parameters : Register C : AAH
Memory 5002H : FCB adres
Register HL : DMA adres
Register IY : Lengte data

Return parameter : Register C : PDOS errorcode

Indien het FCB geïntialiseerd is door een OPEN of MAKE commando, dan worden er 256 bytes naar de diskette geschreven. De werking en foutmeldingen zijn het zelfde als voor het READ FILE SEQUENTIAL commando.

p) Read directory

Entry parameters : Register C : ABH
Memory 5000H : Disk adres

Return parameter : Register C : PDOS errorcode

De directory van de disk aangegeven in geheugen lokatie 5000H, wordt weergegeven op het scherm. Voorts wordt de vrije capaciteit weergegeven.

Return parameter :

'0' : read directory succesvol beëindigd.
'1' : file niet gevonden.
'2' : Drive niet gereed.
'3' : directory vol.
'4' : diskette vol.
'5' : fout in file.
'6' : fout tijdens close opdracht.
'7' : CRC fout tijdens lezen data.
'8' : einde van de file bereikt.
'9' : diskette beveiligd tegen schrijven.
'A' : niet aanwezig drivenummer.
'B' : file tegen schrijven beveiligd.
'C' : fout op systeemdisk.

q) Read random

Entry parameters : Register C : ACH
Memory 5002H : FCB adres
Register HL : DMA adres
Register IX : record number

Return parameter : Register C : PDOS error code

De READ RANDOM functie is gelijk aan de READ SEQUENTIAL FILE functie, behalve dat de leesopdracht plaats vindt van een bepaald record, aangegeven door het hoge byte van het IX-register, (extent nummer) en het lage byte van het IX-register (sector nummer in het extent).

De waarde in het IX-register bevat een byte-paar, welk aangeeft welk record gelezen wordt.

Om een correcte werking te verkrijgen, is het STRIKT NOODZAKELIJK dat eerst het EXTENT 0 geopend wordt. Het is niet noodzakelijk dat dit extent data bevat, maar het openen van extent 0 zorgt ervoor dat de directory entry correct bijgehouden wordt.

Na het aanroepen van PDOS bevat register C een fout-code of de waarde 0. Indien succesvol afgesloten bevat het DMA adres het gelezen record. LET OP : Het record nummer wordt in tegenstelling tot de READ SEQUENTIAL functie NIET OPGEHOOGD.

Return parameter :

'0' : Succesvol afgesloten
'1' : File niet gevonden
'2' : Disk niet gereed
'3' : Directory vol
'4' : Diskette vol
'5' : Fout in file
'6' : Fout tijdens CLOSE
'7' : CRC fout tijdens lezen directory
'8' : Lezen voorbij het einde van de file
'9' : Diskette tegen schrijven beveiligd
'A' : Niet bestaand drivenummer
'B' : File tegen schrijven beveiligd
'C' : Fout op systeem diskette

r) WRITE RANDOM

Entry parameters : Register C : ADH
Memory 5002H : FCB adres
Register HL : DMA adres
Register IX : record nummer
Register IY : data lengte

Return parameter : Register C : PDOS fout code

De WRITE RANDOM instructie wordt op dezelfde manier geïntialiseerd als de READ RANDOM opdracht, behalve dat er data naar disk geschreven wordt vanaf het DMA adres.

Ook wanneer een nieuw extent geadresseerd wordt, zal PDOS er voor zorgen dat dit extent in de directory opgenomen wordt.

Zoals bij de READ RANDOM opdracht, zal geen automatische ophoging van de recordcounter plaats vinden. Ook bij het overschrijven van een extent vindt geen automatisch ophoging van her extent nummer plaats.

De PDOS fout code is dezelfde als voor de READ RANDOM opdracht.

s) COPY DISK

Entry parameters : Register C : AEH
Memory 5000H : Disk adres
(source)
Memory 5002H : Disk adres
(destination)

Return parameter : PDOS foutcode

De diskette in de drive aangegeven door de geheugenlokatie 5000H wordt gekopieerd naar de diskette in de drive aangegeven door de geheugenlokatie 5002H.

De nummering is 0 voor drive A
1 voor drive B
etc.

Return parameter :

'0' : Copy succesvol
'1' : File not found
'2' : Disk niet gereed
'3' : Directory vol
'4' : Diskette vol
'5' : Fout in file
'6' : Fout tijdens close
'7' : CRC fout tijdens lezen data
'8' : Voorbij einde van de file
'9' : Diskette tegen schrijven beveiligd
'A' : Ongeldig drivenummer
'B' : File tegen schrijven beveiligd
'C' : Fout op systeemdiskette

s) COPY FILE

Entry parameters : Register C : AFH
Memory 5000H: FCB adres
source
Memory 5002H: FCB adres
destination

Return parameter : Register C : PDOS foutcode

Het COPY FILE kommando copieert een file
aangegeven door het FCB (5000H) naar een file
aangegeven door het FCB (5002H).
De foutmeldingen zijn gelijk aan het COPY DISK
kommando.

t) FREE DISK SPACE

Entry parameters : Register C : B0H
Memory 5000H : Drivenummer

Return parameters : Register C : PDOS foutcode
Register A : Free disk space
in Kbyte

De FREE DISK SPACE funktie berekent de vrije
ruimte in Kbyte op de schijf geadresseerd door
adres 5000H
Indien de funktie succesvol beëindigd wordt, bevat
register A de vrije ruimte in Kbyte.

Return parameter :

'0' : Succesvol afgesloten
'1' : File niet gevonden
'2' : Drive niet gereed
'3' : Directory vol
'4' : Diskette vol
'5' : Fout in file
'6' : Fout bij Close
'7' : CRC fout tijdens lezen data
'8' : Voorbij einde van de file
'9' : Diskette tegen schrijven beveiligd
'A' : Drivenummer bestaat niet
'B' : File tegen schrijven beveiligd
'C' : Fout op systeemdiskette

u) MOTOR ON

Entry parameter : Register C : 32H

Alle motoren van de drives worden gestart.

w) MOTOR STOP

Entry parameter : Register C : 33H

Alle motoren van de drives worden gestopt.

x) Write disable file

Entry parameters : Register C : B4H
Memory 5002H : FCB adres

Return parameter : Register C : PDOS foutcode

Deze functie zet in de directory entry een vlag waardoor aangegeven wordt, dat deze file writeprotected is. Het FCB moet een unieke file adresseren (geen ??? in het FCB).

Return parameter :

'0' : Succesvol afgesloten
'1' : File niet gevonden
'2' : Drive niet gereed
'3' : Directory vol
'4' : Diskette vol
'5' : Fout in file
'6' : Fout tijdens Close
'7' : CRC fout tijdens lezen data
'8' : Voorbij einde van de file
'9' : Diskette tegen schrijven beveiligd
'A' : Niet bestaand drivenummer
'B' : File tegen schrijven beschermd
'C' : Fout op systeemdiskette

w) Write enable file

Entry parameters : Register C : B5H
memory 5002H : FCB adres

Return parameter : Register C : PDOS foutcode

Deze functie reset in de directory een vlag, waardoor aangegeven wordt, dat de file write enable is.

De foutmeldingen zijn de zelfde als bij de vorige functie (B2H)

De opbouw van het FILE CONTROLL BLOCK

Het FCB heeft de volgende opbouw :

field	FCB positie	doel
ET	0	Entry type
FN	1-8	Filenaam , aangevuld met spaties
FT	9-11	File type, aangevuld met spaties
EX	12	Nummer van het extent binnen de file
WP	13	Write protect
	14	Niet gebruikt
RC	15	Aantal records in het extent
DM	16-31	Disk allocatie map (gevuld door PDOS bij open commando en verder door PDOS bij te houden)

Het FCB is dus eigenlijk een stuk geheugen dat voor PDOS geschikt is om de informatie in op te bergen zodat deze informatie tijdens een CLOSE opdracht naar de schijf geschreven kan worden.

Het gebruik van PDOS onder het EXTENDED BASIC (cassette basic)

Wist U dat tijdens het opstarten van deze BASIC indien de systeemdiskette van het DISK-BASIC systeem in drive 1 zit, PDOS geladen wordt. Dit houdt dus in dat U onder het 16K systeem ook de beschikking heeft over PDOS.

De bijdrage voor de volgende nieuwsbrief zal een aantal voorbeelden behandelen, over het gebruik van PDOS.

Lambert Knapen

Het U.C.S.D. PASCAL systeem.

Het UCSD p systeem (Total systemset) bestaat uit de volgende onderdelen :

- a) 1 ROM pack (part 1)
- b) 4 systeem schijven (part 2 t/m 5)
- c) 1 PASCAL COMPILER

Deze schijven bevatten de volgende programmas :

Part 2: SYSTEEMSCHIJF

bevat de volgende programmatuur :

- a) Het operatingsysteem (SYSTEM.PASCAL)
- b) Een aantal datafiles voor de aanpassing op de verschillende toetsenborden
- c) een programmabibliotheek (SYSTEM.LIBRARY)

Part 3: EDITOR/FILER SCHIJF

bevat de volgende twee programma's

- a) De editor.
De editor is scherm georiënteerd. Dit houdt in, dat de cursor steeds in het beeld blijft. De editor wordt gebruikt om alle gebruikers-programma's mee in te voeren.
- b) De filer.
De filer is een programma om de communicatie te regelen tussen allerlei randapparatuur. Ook wordt de filer gebruikt voor het onderhouden van bestanden op schijven. De filer is nodig voor het verplaatsen van een bestand naar een (andere) schijf onder een (zelfde) naam, of voor het controleren van schijven op slechte sectoren.

Part 4: ASSEMBLERSCHIJF

bevat de volgende programmatuur :

- a) Z80 assembler
- b) 8080 assembler
Beide assemblers genereren Z80 c.q. 8080 machine codes.
- c) Een programma dat het mogelijk maakt om onder PASCAL te werken met assemblerprogrammas.

Part 5: TOEPASSINGSPROGRAMMA'S

bevat de volgende programmatuur

- a) LIBRARY : een programma waarmee de gebruiker, een aantal routines kan samenvoegen tot een bibliotheek.
- b) SETUP : Hiermee kan een nieuwe file SYSTEM.MISCINFO gekreeerd worden.

- TABEDIT : Hiermee wordt de file system.tables gelezen en er kunnen eventueel veranderingen mee aangebracht worden.
- c) DECODER : Een programma om p-code files te lezen en te decoderen naar pseudo-machine instructies.
- d) YALOE : Een regel-georiënteerde editor. Deze editor is afgeleid van de EDIT editor op de PDP 11 minicomputer van DEC. Deze editor zal in principe nagenoeg nooit gebruikt worden op de P2000.
- e) DEBUGGER: Hiermee kan men fouten opsporen in een PASCAL programma. Dit gebeurt door het controleren van de p-code met de programma-listing. Dit vereist echter een behoorlijke kennis van machinetaal en PASCAL.
- f) COPYDUPDIR en MARKDUPDIR zijn programma's om een kopie van de directory op schijf bij te houden en aan te maken.
- g) SCREENTEST : Dit programma is nodig om te testen of veranderingen in systeem tabellen korrekt zijn aangebracht. Ook nodig voor het testen van eigen screenroutines.
- h) XREF : Crossreferencer. Dit is een bijzonder bruikbaar programma , omdat het eventueel waarschuwingen en foutmeldingen kreeert bij het "verwerken" van een PASCAL programma. XREF geeft verschillende tabellen omtrent gebruik van variabelen en declaraties in het PASCAL programma.
- i) RECOVER: Geeft in bepaalde gevallen de mogelijkheid om een als verloren gegane schijf of een verloren gegaan programma te redden.
- j) BOOTCOPY: Hiermee kan men de bootstrap van de ene schijf kopiëren naar een andere schijf. Dit is nodig om een systeemschijf te maken.
- k) PATCH : Een programma om op byte-niveau een file te lezen en eventueel te wijzigen.

Het opstarten van het systeem.

Het voor de eerste keer opstarten van het systeem levert een aantal moeilijkheden op.

Handel als volgt :

- a) Plaats in drive 1 de systeemschijf (part 2).
- b) Plaats in drive 2 de editor/filer (part 3).
- c) Plaats PASCAL prom blok in slot 1.
- d) Zet de P2000 aan.

Na enige tijd verschijnt een mededeling op het scherm, die er aan herinnert dat alle diskettes geïntialiseerd dienen te worden.

Men wordt verzocht #5:init in te typen.

Dit levert echter problemen op, omdat de P2000 geacht wordt een Duits toetsenbord te bezitten.

Men dient hierom nu in te typen :

X

#5.init (er verschijnt #5:init)

Nu gaat drive 2 lopen en wordt de editor/filer geïntialiseerd.

Het is nu het beste om de P2000 gebruik te laten maken van het Engelse toetsenbord.

Hiervoor kan men het beste gebruik maken van de volgende kommando's :

F

C

5.szstem.tables,germ.m.tables (indien T-model

5.uk.m.tables,szstem.tables lees.t. ipv.m.)

Q

I

Het systeem start zich nu opnieuw op, en zal na enige tijd verschijnen met de opstartboodschap.

Nu zal indien men een bepaalde toets indrukt, de corresponderende letters op het beeldscherm verschijnen.

Vervolgens kunnen alle schijven geïntialiseerd worden door in te typen :

X

#5:init

Nu heeft men de beschikking over alle schijven. Wat is er nu gebeurd ?

In het ROM-pack staat een nummer, dat door het uitvoeren van de file init.code, in belangrijke systeemprogramma's wordt geschreven.

Het resultaat is nu, dat de schijven alleen nog maar werken op het eigen systeem.

Wat te doen met de systeemschijven ?

Eerst dienen alle schijven gecopieerd te worden.

Dit doet men door gebruik te maken van de utility bootcopy en de filer.

Bootcopy copieert de eerste track van drive 4 naar drive 5.

Vervolgens de schijf waarvan de eerste track gekopieerd is, plaatsen in drive 5.

Plaats de editor filer schijf in drive 4 en geef de volgende commando's

F

T

Plaats nu de schijf waarop de eerste track reeds gekopieerd is in drive 4, en geef het volgende commando :

#5:,#4:

x

De schijf wordt nu gekopieerd van drive 5 naar drive 4.

Men dient deze handelingen te verrichten voor alle systeemsschijven.

Indien men nu besluit om te gaan werken met het systeem, dan komt men na enige tijd tot de konklusie, dat men meer op een disc-jockey lijkt, dan dat men een programmeur is.

Hoe kunnen we er nu voor zorgen, dat we steeds met veel plezier van het systeem gebruik maken?

We kunnen de programmatuur op een bepaalde manier gaan reorganiseren.

Het verdient de voorkeur dat bepaalde programma tuur op een schijf komt en andere programmatuur weer op een andere schijf.

Het verdient aanbeveling om op een schijf de volgende programmatuur te plaatsen :

- a) system.pascal
- b) system.tables
- c) system.miscinfo
- d) system.filer
- e) system.library
- f) system.compiler
- g) system.syntax
- h) userlib.text (gevorderde PASCAL users)

Dit is een schijf waarmee het mogelijk is PASCAL programma's te compileren en te executeren.

De volgende schijf is de assembler/linker schijf. Plaats op deze schijf :

- a) system.pascal
- b) system.tables
- c) system.miscinfo
- d) system.filer
- e) system.assemblr
- f) system.linker
- g) Z80.opcodes
- h) Z80.errors
- i) asmhost.code
- j) userlib.text

U zult tot de ontdekking komen, dat er nagenoeg geen vrije ruimte meer is op beide schijven. In principe kunt U op beide schijven de file system.startup plaatsen, maar dit heeft tot nadeel dat uitbreiding in de vorm van een bibliotheek niet mogelijk is. Men kan altijd nog beslissen om de file te verwijderen.

Voorts verdient het de voorkeur, om op beide schijven ervoor te zorgen dat de files :

- a) system.pascal
- b) system.tables
- c) system.miscinfo
- d) system.filer

op de zelfde plaats staan.

Dit biedt het voordeel dat men de compilerschijf kan verwisselen met de assemblerschijf, zonder dat het systeem opnieuw opgestart dient te worden.

Op welke schijf komt nu de editor te staan ?

De editor (system.editor) komt op de schijf te staan waarop alle programmatuur wordt ontwikkeld.

Deze handelswijze vertoont echter wel 2 nadelen.

- a) op elke schijf verliest men 16 Kbyte aan ruimte
- b) men verliest het voordeel van de workfile

Hier tegenover staat echter wel het grote voordeel dat het wisselen van schijven nagenoeg niet meer nodig is.

DE MOGELIJKHEDEN VAN HET SYSTEEM.

De mogelijkheden van het systeem zijn enorm. Men kan met namelijk assembler programma's interfacen met PASCAL programma's, waardoor het mogelijk is PASCAL programma's te schrijven die bijv. gebruik maken van eigen randapparatuur. Ook is het mogelijk om eigen functies te definiëren onder PASCAL.

Een andere mogelijkheid is het ontwikkelen van assembler programma's, die naderhand niet meer onder het PASCAL systeem draaien, maar als stand-alone programma toepassing kunnen vinden. Het is mogelijk om zeer grote PASCAL programma's te schrijven en te verwerken. Deze programma's mogen in totaal groter zijn dan de totaal beschikbare geheugenruimte.

Het opbouwen van een programma-bibliotheek biedt de mogelijkheid tot het vereenvoudigen van het programmeerwerk, doordat alleen de aanroep van de betreffende routine uit de bibliotheek in het gebruikersprogramma behoeft opgenomen te worden.

DE NADELEN VAN HET SYSTEEM.

Enige nadelen zijn er te noemen.

PASCAL is geen gemakkelijke taal.

Het is nagenoeg onmogelijk om een PASCALprogramma te veranderen, zodanig dat het programma na de verandering ook weer foutloos loopt.

Voor het T-model geldt nog een bijzonder nadeel:

De breedte van het scherm is slechts 40 karakters.

Dit betekent dat wanneer een assemblerlisting op het scherm gelist wordt, nagenoeg niets van de source-code te zien is .

Ook wanneer men een file aan het creëren is c.q. veranderen is heeft men slechts de beschikking over 40 karakters. Het is namelijk mogelijk om meer dan 40 karakters op een regel in te typen, maar bij het 40e karakter verschijnt een uitroep teken op het scherm, en voor de rest typt men in den blinde.

Dit is wel de grootste beperking voor het T-model en ook een bijzonder lastige.

Dit geldt ook voor het M-model, maar hier heeft men de beschikking over 80 karakters. Dit is ruim voldoende voor de gemiddelde regel.

DOCUMENTATIE

De documentatie is zeer goed verzorgd en zeer uitgebreid.

De documentatie bestaat uit de volgende boekwerken :

- a) P2350 UCSD p-SYSTEM (TKS) operator manual
- b) P2351 UCSD p-SYSTEM (TSS) programmer's guide
- c) P2351 UCSD p-SYSTEM (TSS) internal architecture
guide
- d) P2351 UCSD p-SYSTEM (TSS) reference manual
- e) PASCAL USER MANUAL AND REPORT

De documentatie is in het engels geschreven.

In de boekhandel zijn boeken verkrijgbaar die eenieder kunnen helpen bij het werken met het UCSD p-SYSTEM.

SAMENVATTING

Het UCSD p-SYSTEM is een bijzonder krachtig systeem en heeft weinig begrenzingsen.

Ik wil er echter wel op wijzen dat men zich niet in het systeem moet vergissen. Het is mooi, maar wil men er effectief gebruik van maken, dan is een behoorlijke kennis van programmeren VEREIST.

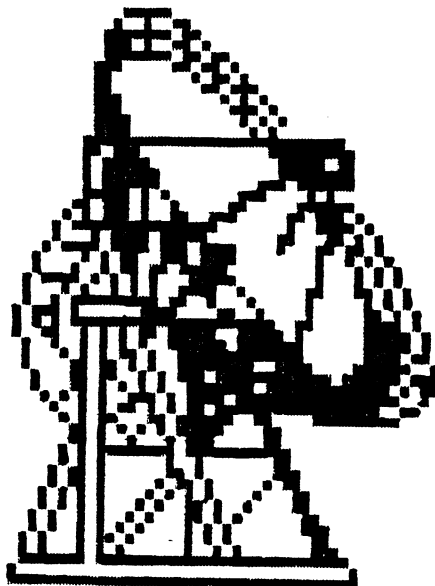
Het UCSD p-SYSTEM is verkrijgbaar in 3 uitvoeringen

- a) Het TKS systeem.
Het TKS systeem bestaat uit
ROM pack
TKS system disk
P2000 TKS operator's manual
Lege diskette

Met het TKS systeem is het niet mogelijk om zelf programma's te ontwikkelen.

- b) Het TSS systeem. (ong. f 1000.-)
Het TSS systeem bestaat uit :
ROM pack
TSS systeem diskettes : systeem
editor/filer
assembler
utilities
Documentatie : Internal Architecture guide
Reference manual
Programmers guide
- c) PASCAL compiler (ong. f 800.-)
Documentatie : PASCAL User manual and report

Lambert Knapen



De P-2000 een
blik in de toekomst !!!

Machinetaal programma's achter BASIC zetten.

In P2000gg nieuwsbrief nr.3 is beschreven hoe met behulp van de BASIC opdrachten DATA, READ en POKE een machinetaal programma in het geheugen van de P2000 geplaatst kan worden. Het nadeel van deze methode is de lange duur als het machinetaal programma wat groter is. Dit nadeel kan overwonnen worden door de bytes die het machinetaal programma vormen direkt achter het BASIC programma te zetten. Onder deze bytes bevindt zich dan een routine die de bytes vanachter het BASIC programma terugzet naar hun oorspronkelijke plaats. In een fraktie van een seconde staat het machinetaal programma op de goede plaats en kan aangeroepen worden. Bekende programma's waar deze methode gebruikt wordt zijn 'Chess' (ong. 14000 bytes machinetaal!!) 'Viditel' en 'Basicode'.

Voor het ontwikkelen van een machinetaal programma gebruiken we de methode beschreven in P2000 nieuwsbrief 3. Dit omdat dan gemakkelijk wijzigingen aangebracht kunnen worden. Als het programma gereed is willen we een definitieve versie maken waarbij de machinetaal bytes achter de BASIC staan. Daarvoor hebben we twee machinetaal routines nodig namelijk een die voor het copieeren achter BASIC zorgt en een tweede die de zaak terugbrengt naar zijn oorspronkelijke plaats.

Voordat deze routines worden besproken een paar algemene zaken.

Centraal in het geheel staan de geheugenadressen &H6405 en &H6406.

Deze bevatten de pointer die het einde van het BASIC programma aanwijst. De waarde van deze pointer verhogen we zoveel als we nodig hebben om ons machinetaalprogramma kwijt te kunnen. Het verdient aanbeveling hiervoor veelvoud van 256 bytes te kiezen.

Stel dat we een machinetaal programma hebben van 600 bytes dat vanaf &H9D00 in het geheugen staat. We schrijven dan $3 * 256 = &H300$ bytes achter BASIC. Voordat we dit doen verhogen we de waarde die in &H6406 staat met het aantal blokken van 256 bytes, in dit geval dus 3. Dan wordt het volgende stukje machinetaal aangeroepen dat alle bytes van &H9D00 tot &H9FFF in de op deze wijze vrij gemaakte ruimte achter het BASIC-programma zet:

9FB0	E5	:PUSH HL	zet register HL op stack.
9FB1	AF	:XOR A	maak Carry-flag nul voor aftrekken
9FB2	2A 05 64	:LD HL, (6405)	waarde einde BASIC pointer in HL
9FB5	01 00 03	:LD BC, 3	aantal te copieeren bytes in BC
9FB8	ED 42	:SBC HL, BC	bereken beginadres bestemming co-
9FBA	EB	:EX DE, HL	pieeren en plaats in DE
9FBB	21 00 9D	:LD HL, 9D00	beginadres te copieeren bytes
9FBE	ED B0	:LDIR	blockmove, het eigenlijke copieeren
9FC0	E1	:POP HL	register HL herstellen
9FC1	C9	:RET	terug naar P2000.

Uit het BASIC programma kunnen nu desgewenst alle DATA en andere statements verwijderd worden die voor het in het geheugen zetten van het machinetaal programma zorgden. Het blijkt dat de BASIC interpreter geen enkel probleem heeft met de niet-BASIC staart die we achter het programma geplakt hebben. Dit komt omdat direkt voor onze machinetaalbytes drie nullen staan. Zodra de BASIC interpreter deze tegen komt wordt de uitvoering gestopt. Bij het verwijderen of tussenvoegen van BASIC regels wordt naar de waarde van de eindpointer in &H6405, &H6406 gekeken en het stuk machinetaal in de wijzigingen betrokken. Evenzo geldt dit voor het inlezen en wegschrijven naar cassette.

Tot slot moeten we een routine hebben die de machinetaalbytes vanachter BASIC terugzet naar hun oorspronkelijke plaats. Daarvoor gebruiken we het volgende stukje machinetaal:

9FD0	E1	:PUSH HL	berg register HL op
9FD1	AF	:XOR A	maak Carryflag nul voor aftrekken
9FD2	11 00 9D	:LD DE,9D00	beginadres terugzetten
9FD5	01 00 03	:LD BC,3	aantal te copieeren bytes
9FDB	2A 05 64	:LD HL,(6405)	waarde BASIC eindpointer in HL
9FDB	ED 42	:SBC HL,BC	bereken beginadres ophalen bytes
9FDD	ED B0	:LDIR	blockmove,het eigenlijke terugzetten
9FDF	E1	:POP HL	register HL herstellen.
9FE0	C9	:RET	return

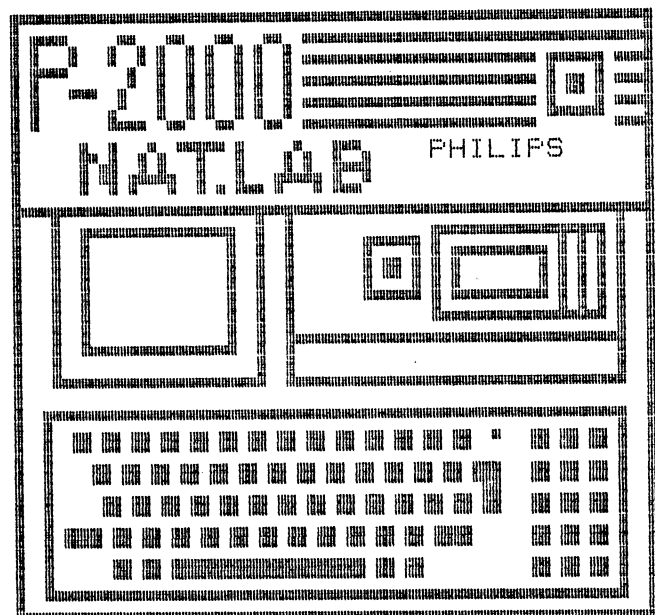
Deze routine wordt dus ook mee weggeschreven achter BASIC. Zijn beginadres is dan ook afhankelijk van de waarde van de BASIC-eindpointer. Daarom laten we het BASIC programma beginnen met de volgende regels:

```
10 CLEAR 50,&H9D00
20 DEF USR=PEEK(&H6405)+256*PEEK(&H6406)-&H20
30 X=USR(0)
```

Nu staat het machinetaal programma weer op zijn oude plaats en kan aangeroepen worden.

Bovenstaand verhaal geldt voor 16K BASIC en een P2000 met 16K RAM-geheugen. Voor andere configuraties zijn kleine wijzigingen nodig.

Gerben Mooiweer



Gebruik PRINT USING:

Het is gebleken dat niet iedereen weet hoe de PRINT USING gebruikt kan of dient te worden.

In een string wordt het gehele beeld opgebouwd, waarbij de plaats van getallen met ##### aangeeft, en eventueel de plaats van strings met ' '.

Om een voorbeeld te geven;

```
10 D$ = "Deze ' ' kost ##.## gulden"  
20 PRINT USING D$; "vaas"; 12.5  
30 PRINT USING D$; "pot"; 7
```

dit in tegenstelling tot het slordige:

```
10 PRINT "Deze "; USING " ' "; "vaas"; " kost "  
USING "##.##"; 12.5; "gulden"
```

Uit het prikbord van Philitel.

CLEAR-statement:

Opdat een programma na executie de P-2000 in de oorspronkelijke staat achterlaat, is het gewenst dat een gegeven CLEAR-statement rekening houdt met de in de machine aanwezige geheugengrootte. Dat staat vermeld in locatie &H605c:

```
1 = 16K RAM  
2 = 32K RAM  
3 = 48K RAM
```

Het is daarmee mogelijk om in een enkele expressie de machine in de goede staat terug te brengen:

```
CLEAR 50, &H5FFF+&H4000*PEEK (&H605C)  
+&Y2000* (3=PEEK (&H605C)).
```

Voor de P-2000 printer:

```
LPRINT CHR$(30) ".." voor underline;  
LPRINT CHR$(31) voor stop underline.
```

"Als straks allemaal mensen, allemaal computers thuis hebben, en die mensen denken: O, wat is het toch fijn om met een computer om te gaan, dan zal ik heel blij zijn als ik daar aan heb kunnen bijdragen":

Klaas Robers

Curriculum vitae:

Klaas Robers is in Amsterdam, in september 1944 geboren. Zijn middelbare schooltijd bracht hij door op de HBS te Hengelo. Al gauw bleek dat het voor hem de B-richting moest worden: "Ik ben niet iemand die in hogere sferen verkeert, maar een beetje met de benen op de grond, dus de technische dingen en de wat exactere zaken vielen mij wat beter dan de wat zweverige zaken en de wat abstracte dingen; dat zag ik allemaal nooit zo erg zitten." Hierna ging hij naar de TH in Delft, waar hij electro-techniek studeerde omdat hij dit een leuk vak vond. Na zijn studie, die bijna elf jaar geleden is afgerond, ging hij werken in het Natuurkundig Laboratorium van Philips in Eindhoven, in de zogenaamde TV-groep. Vijf jaar later ging hij over naar het Nat. Lab. in Geldrop, waar hij in de educatie-groep terecht kwam, en waar hij nog steeds werkt.

Geleidelijk aan is zijn werk zich gaan richten op het onderwijs thuis. Hiernaar is veel vraag, maar er is een sterke wil nodig om het tot een goed eind te brengen. Men tracht de educatie thuis prettiger te maken, waardoor men b.v. op het idee gekomen is de VLP (Laservision) te koppelen aan de P2000. Over hoe het 'lesgeven' thuis moet gebeuren zei Klaas het volgende: "De stof die je moet onderwijzen, of die je over wilt brengen, moet je zodanig verpakken dat het leuk is. Het lesnemen thuis of iets leren thuis, moet niet in eerste instantie nuttig zijn, maar in eerste instantie leuk zijn; Zo leuk dat je zegt: He getsie, de televisie vanavond is helemaal niet leuk, weet je wat, ik ga weer verder met dat lesprogramma van mij, want dat is zo hartstikke leuk... Nou, zo moet dat eigenlijk zijn."

Klaas is van mening dat men er nu een idee van begint te krijgen hoe men dit alles moet realiseren. Zijn taak in het Nat. Lab. is te omschrijven als: Niet het verbeteren van de computer, maar kijken wat je met zo'n computer kunt doen op het gebied van onderwijs.

De persoon Klaas:

Een van de eerste dingen die aan Klaas opvallen is zijn luide stem. Dit wijst Klaas aan het feit dat hij zendamateur is; Hierbij is het nodig om een constante, luide stem te gebruiken, wat Klaas zich ook in het dagelijks leven is aan gaan wennen. Een tweede opmerkelijk iets is dat zodra Klaas begint te praten er zoveel mensen naar hem komen luisteren. Op de vraag of dit de commerciële zijde van Klaas is antwoordde Klaas dat het voor hem niet het belangrijkste is iets te verkopen, maar om zijn kennis over te dragen. Daar komt bij dat hij zelf vaak enthousiast is, waardoor hij zijn verhaal nog boeiender weet te brengen.

Hieruit vloeit automatisch voort dat Klaas wat arrogant lijkt: "Het is echt niet zo dat ik de mensen wil laten zien hoe het moet, om te laten zien hoe knap ik nou ben, en hoe goed ik het wel weet, maar mijn belangrijkste doel daarmee is om informatie die je daar bij hebt over te dragen aan anderen.

De hobby's van Klaas:

Deze zijn: de P2000, zendamateurisme en electronica. Vaak combineert hij die drie hobby's om het doel wat hij zich gesteld heeft te realiseren.

Klaas en de P2000:

De eerste kennismaking met computers had Klaas op zijn werk. Zijn groepsleider (Pieter v. d. Avoort) vond dat Klaas met micro-processors moest gaan werken, maar hiervoor voelde Klaas niet veel, want hij zag bij zijn college's dat dit er erg moeilijk uitzag.

Na een tijdje kon Klaas echter goedkoop een kapotte Apple op de kop tikken die hij heeft kunnen repareren. Hij begon toen voorzichtig te programmeren in Basic. Het bleek veel gemakkelijker te zijn dan hij dacht. Na ongeveer 9 maanden nam Klaas de stap naar assembleer- en machinetaal: "Dan sluit zich ineens de kloof tussen wat je als electronicus leert van poortjes, flip-floppen en nullen en enen, en het programmeren in Basic, en dan beheers je ineens het hele gebied."

Met zijn kennis op het gebied van de hardware kon hij kiezen tussen wat hij softwarematig en wat hij hardwarematig met zijn computer deed in tegenstelling tot diegenen die alleen software of hardware gericht zijn.

Pas 3 jaar geleden leerde Klaas de P2000 kennen. Toen was de P2000 nog niet op de markt, maar in verband met zijn werk was er een prototype op het Lab. De groepsleider kwam toen op het idee om de Nat. Lab. Thuiscomputerclub op te richten, omdat hij wist dat er software moest komen, zodat de P2000-verkoop goed zou kunnen starten. Hiermee begon de P2000 een van Klaas zijn beroepsaspecten en een van zijn hobby's te worden.

Klaas heeft er veel plezier in te werken met de P2000, omdat hier weer het aspect van de kennisoverdracht naar voren komt: "Ik heb daar altijd een zwak voor gehad. Op de middelbare school heb ik altijd leraar willen worden om te laten zien dat het ook goed kon, en ik heb in mijn studententijd een jaar lang natuurkunde-les gegeven in Rotterdam. Daar heb ik ook mijn aantekening gehaald als leraar natuurkunde. Dat vond ik hartstikke leuk, maar uiteindelijk heb ik het toch maar niet gedaan, want ik vond het zonde van mijn studie. Je doet zo'n studie voor electrotechnisch ingenieur niet om leraar te worden, maar ik vind het leuk om kennis over te dragen; dat is iets belangrijks."

Klaas wil het gebruik van de computer gemakkelijk en voor iedereen toegankelijk maken door zeer gebruikers-vriendelijke software te maken. Zo kan iedereen de computer gebruiken. De eerste stap hier naar toe is het "familiegeheugen".

Verder vindt Klaas het erg plezierig om 'experimenten' te doen met zijn P2000. Hierbij krijgt hij een idee en wordt 'uitgedaagd' door de vraag of dat mogelijk is of niet. Een paar van die experimenten zijn bijvoorbeeld:

- De frequentie-teller: Dit idee kreeg Klaas toen hij met Basicode bezig was, en zag dat de 'sterretjes' zich bewogen als functie van de frequentie van het geluid.
- Het viewdata-programma: "Er bestaat een viewdata-moduul, maar dat betekent dat iedereen die met de P2000 viewdata wilt doen zo'n doos moet kopen en daarbij weer een programma moet hebben, en dat is niet leuk. Het probleem was nu: Is het mogelijk om met die printerconnector, geheel in software, 1200 baud te ontvangen en 75 baud te zenden en het toetsenbord te scannen en lettertjes op het scherm te zetten, wat helemaal dwars door elkaar heen loopt, zonder dat het elkaar hindert. Nou, dat is een uitdaging, en ik heb 4 keer opnieuw moeten beginnen. En op gegeven moment bouw je een hele gekke maar slimme constructie, maar dan is het wel zo dat als iemand dat programma disassembleert er kop nog staart aan te vinden is. Een heel gek programma is het ineens, maar ja, dat was nodig om het te laten doen wat het doet. Uiteindelijk is het dan iets dat werkt, en dat alleen nog maar een hoop gemagnetiseerd roest is op een plastic bandje, en dat is dan aardig.
- De morsecursus: Deze is aanvankelijk ontwikkeld door Peter Lundahl, en wel voor de computer van het Evoluon. Klaas heeft hiervan een ponsband kunnen maken en op die manier het programma in zijn Apple kunnen inlezen. Later heeft hij het programma via het Basicode-programma vanuit de Apple in de P2000 kunnen laden.

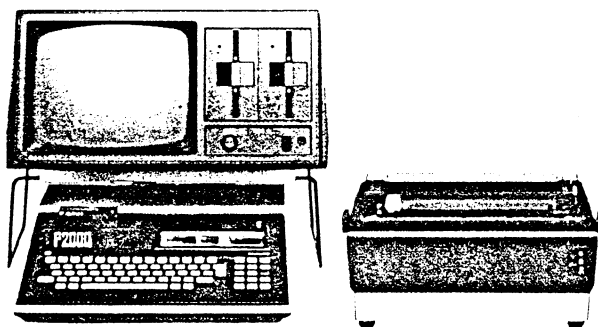
Klaas en viewdata:

Zoals aan velen onder de P2000-gebruikers bekend is, is Klaas de databaas van de Philips Nat. Lab. database. Ook hier steekt Klaas nogal wat tijd in. Zo vergt het inladen van telesoftware erg veel tijd, maar Klaas doet dit met plezier, omdat hij zelf gelooft dat dit de beste manier is om programma's te verspreiden. Ook bevat deze database een "prikbord". Dit idee is afkomstig van Karel de Jong, die dit tijdens een demonstratie naar voren bracht. Klaas wist toen nog niet of het idee uitvoerbaar was, en zo ja of het dan zo zou zijn dat er alleen maar onzin op zo'n prikbord zou komen te staan. Toen hij het technische gedeelte had gerealiseerd bleek dat zo'n prikbord goed functioneert, zoals men kan zien in deze database. Er heerst een goede discipline op het prikbord. Hieraan hoeft Klaas dus eigenlijk geen tijd te besteden.

Klaas' toekomstvisie:

Klaas gelooft dat de computer niet tegen te houden is, maar hoe snel alles gaat gebeuren hangt er onder andere van af hoe snel er programma's komen voor de 'gewone' man, die over die programma's kan zeggen: Dat vind ik interessant, daar heb ik wat aan.

Anita Janssens.



Op de Firato werd een nieuwe ROM-MODULE voor de P 2000 T geïntroduceerd: HET FAMILIEGEHEUGEN. Voor f 350 inclusief BTW krijg je een pakket met de Rom-module, een handleiding en 6 cassettes. De Module moet in slot 1 geplaatst worden en dan gedraagt de P 2000 zich als een soort elektronische kaartenbak. De cassette wordt gebruikt als opslagmedium.

Hieronder een korte beschrijving van de mogelijkheden. Met het familiegeheugen kun je allerlei bestanden bijhouden. B.v. je boeken, je platen, recepten, adressen enz. enz. Als het programma start heb je de keus uit:
N = nieuw bestand opbouwen
L = lezen van een bestand dat op cassette staat
I = informatie (korte gebruiksaanwijzing)

Bij het opbouwen van een bestand typ je gegevens in tot een onderwerp klaar is. De onderwerpen zijn nauwelijks aan een formaat gebonden. De gebruiker heeft alle vrijheid om gegevens in te typen. Een onderwerp kan niet langer zijn als een beeldscherm.

De verschillende onderwerpen in een bestand hoeven ook niet allemaal even lang te zijn. B.v. in een bestand over boeken kun je over het eerste boek b.v. 10 regels intypen en over het tweede boek b.v. 3 regels.

Een onderwerp wordt afgesloten met CODE K(laar). Als het laatste onderwerp is ingetypt kun je het bestand wegschrijven naar de cassette onder een zelf te kiezen naam.

De zoekmethode om een onderwerp terug te zoeken is erg veelzijdig. Je kunt een zoekwoord opgeven. Het programma "leest" dan alle onderwerpen door en geeft die onderwerpen weer waarin het zoekwoord voorkomt. Ook kunnen twee zoekwoorden gegeven worden en zelfs kan een zoekwoord gegevens worden met een niet-zoekwoord. Het programma zoekt dan naar een onderwerp waarin het eerste woord wel voorkomt en het tweede niet.

Met D(isplay) kan het hele bestand bekeken worden. Er zijn uitgebreide Edit-mogelijkheden om onderwerpen te wijzigen of weg te laten.

Met P kan een onderwerp op papier worden afgedrukt. Dan wordt de scherminhoud naar de printer gestuurd. Met shift 2 kan de printerbaud worden ingesteld. Dit staat overigens niet in de handleiding.

Het programma is ook geschikt voor mensen die niet zelf programmeren. De handleiding is duidelijk gericht op de 'computerleek'. Bij het pakket worden 6 cassettes geleverd met enkele voorbeeldbestanden: boeken, platen, recepten en adressen.

Voor heel wat bestanden is dit een geschikt programma. De gebruiker moet er echter wel rekening mee houden dat hij het programma moet nemen zoals het is. Je kunt het niet aanpassen aan je eigen wensen.

Al werkend met de P 2000 vond ik het hinderlijk bij printwerk steeds te moeten wachten tot de printer klaar was. In HCC nieuwsbrief 43 las ik een advertentie van de firma Infotheek te Leiden over Microfazer Printerbuffers. De prijs was lager dan ik elders had gezien en ik besloot een exemplaar te bestellen van het type serieel/parallel met een buffer van 8k die uit te breiden is tot 64 k.

Bevindingen.

De buffer is in een net metalen kastje gebouwd. Er zit ruimte in om door insteken van chips de buffer uit te breiden tot 64 k (64 k ram chips type 4164). Aan de ene kant zitten drie rode leds en twee drukknopjes. Er is een knop voor reset en een copyknop. Met die laatste knop kun je de inhoud van de buffer telkens opnieuw laten afdrukken. Zolang er nog gecopieerd kan worden brandt een led. De tweede led geeft aan of er nog data worden geaccepteerd en de derde geeft een printerfout aan. Achterop het kastje bevindt zich een 25 polige aansluiting zoals op de P 2000. De P 2000 moet met deze connector worden verbonden. Het was eerst een heel zoek om de juiste kabel te maken. Dat lukte na veel zoek met de voltmeter en na diverse telefoontjes met de mensen van Infotheek.

Zo moet het:

P 2000 Microfazer Printerbuffer

pen	pen
2-----	3 buffer stuurt data
3-----	2 P 2000 stuurt data.
7-----	7 aarde
20-----	6 handshake

Bij afgeschermd kabel kan pen 1 met pen 1 worden verbonden.

Met een flatkabel wordt de buffer met een paralelle printer verbonden. Ik gebruik de MX-80 zonder ser. interface.

De buffer kan op twee manieren voeding krijgen.

1. extern 9 Volt AC of DC via een trafo of lichtnetadapter.
2. vanaf de printer 5 Volt door de flatkabel. In de handleiding is aangegeven waar de voeding uit de printer gehaald kan worden.

Binnen in de buffer kan de baudrate worden ingesteld van 110 tot 9600 door het verzetten van een bruggetje.

Het werken met dit kleine apparaat van f 590 is erg prettig. De P 2000 zendt continue informatie naar de buffer. De printer werkt nu veel sneller omdat hij via de paralelle ingang informatie krijgt. De P 2000 is heel vlug weer beschikbaar om b.v. een volgende file klaar te maken. Vooral bij de tekstbewerking is dit erg tijdsbesparend.

Als je een file, die groter is als 8 k naar de buffer stuurt is er geen enkel probleem. De buffer leegt zich en vult zich volgens de regel 'first in first out'. De handshake regelt dat alles. Met deze buffer heb je zo ook een universele serieel-paralelle interface. (incl. LN tel. 02209-2508)

VARPTR

Met deze instructie kun je er achter komen waar de P 2000 de waarde van een variabele in het geheugen plaatst. Bij strings werkt dat als volgt: Met PRINT VARPTR(A\$) krijg je het eerste adres van de zogenaamde STRINGDESCRIPTOR. Het eerste byte geeft de lengte van A\$. De twee volgende bytes geven het adres waar de string zelf begint. Dat kan een stukje van het programma zelf zijn of een stukje in de gereserveerde stringruimte.
B.v. A=VARPTR(A\$):A=PEEK(A+1)+256*PEEK(A+2). A bevat dan het adres waar de string begint.

SKART-PLUG

=====

Hieronder een schakeling waarmee de P 2000 met een televisietoestel met SKART-PLUG kan worden verbonden. Deze schakeling is iets anders als eerder gepubliceerde schema's. Bij deze schakeling hoeft in de computer geen extra transistor geplaatst te worden. Het voordeel is dat er tijdens het werken met de computer heel gemakkelijk kan worden overgeschakeld naar televisie kijken. Dat kan door een schakelaar in de TV om te zetten. Dat is handig als de computer niet zo lang van de TV gebruik hoeft te maken.

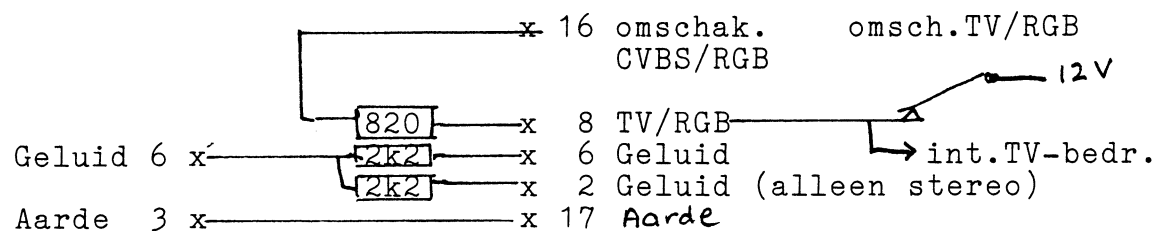
In sommige TV-toestellen moet de 12 volt van de schakeling nog naar punt 8 gebracht worden. In sommige andere TV-toestellen moet het schakelaartje nog geplaatst worden.

P 2000

SKART-TV

Rood	4	x	—	150	—	x	15	Rood
Groen	1	x	—	150	—	x	11	Groen
Blauw	5	x	—	150	—	x	7	Blauw
Sync	2	x	—	560	—	x	20	Video

(opm. bij sync: zonder ombouw moet de weerstand niet 560 maar 150 zijn)



Wis-het-scherm-toets in het Viewdata-programma.

Rechtsboven op het kleine toetsenbord zit de wis-het-scherm-toets. In VD/3000 is het handig als deze toets gebruikt kan worden. Bij Viewdata versie 1.8 werkt deze toets naar behoren, maar bij Viewdata versie 1.5 moet je de toets zelf "repareren".

Dat kan met de verkorte versie als volgt:

Laad het programma vanaf de cassette. Niet RUN-nen!

Typ nu (zonder regelnummer):

```
'FOR I=&H7000 TO &H7FFF: IF PEEK(I)=30 AND PEEK(I-38)=30 AND PEEK(I-1)=210 THEN ?(I): NEXT ELSE NEXT' en dan 'ENTER'.
```

Op het beeldscherm komt even later een getal, bijvoorbeeld '31289'. Typ nu 'POKE 31289,12' en dan 'ENTER', en schrijf het programma weer weg naar cassette.

1 WEES KORT, DUIDELIJK EN KRACHTIG.2 ORDE IS BELANGRIJK

Een juiste indeling van het beeld draagt bij tot betere zichtbaarheid, betere leesbaarheid. Sniijdt U binnen hetzelfde beeld een ander onderwerp aan, toon dit dan door een regel over te slaan en de nieuwe alinea zo mogelijk met een tussenkopje te beginnen.

Een nieuwe alinea (zonder onderwerpwisseling) is eenvoudig aan te geven door de eerste regel ervan niet helemaal rechts, maar na drie spaties inspringen te laten beginnen. Eventueel met "inspringen" te combineren is het in een andere kleur zetten van de nieuwe alinea-tekst. Zie ook 5.

Opsommingen? Een puntsgewijze opsomming oogt beter als de diverse onderdelen ook inderdaad door een (gekleurd) blokje, een liggend streepje, een cijfer of een letter worden voorafgegaan. Spring bij een opsomming niet in als dit niet perse hoeft. Of als U vindt dat dit wel moet, hanteer dan een maat voor inspringen en niet 2, 3 of meer, zoals in voorbeeld 1.

voorbeeld 1

1. Aanlegkosten
- 1a. Aanlegkosten fietspad
- 1a.1 Voorbereiding fietspad
- 1a.2 Grondverwerving
- 1a.3 Grondverzet, speciale voorzieningen en aanpassingen
- 1a.4 Verharding-drie alternatieven
- 1b. Aanlegkosten trottoir (enzovoorts)

voorbeeld 2

2. Aanlegkosten
- 2a. Aanlegkosten fietspad
- 2a.1 Voorbereiding fietspad
- 2a.2 Grondverwerving
- 2a.3 Grondverzet, speciale voorzieningen en aanpassingen
- 2a.4 Verharding-drie alternatieven
- 2b. Aanlegkosten trottoir (enzovoorts)

voorbeeld 3

3. Aanlegkosten
- 3a. Aanlegkosten fietspad
- 3a.1 Voorbereiding fietspad
- 3a.2 Grondwerving
- 3a.3 Grondverzet, speciale voorzieningen en aanpassingen
- 3a.4 Verharding-drie alternatieven
- 3b. Aanlegkosten trottoir (enzovoorts)

Op het eerste gezicht, hier zwart op wit afgedrukt, lijkt variant 3 overzichtelijker dan 2; in ieder geval "rustiger". Het zal een kwestie van smaak en van functioneel kleurgebruik zijn om Uw definitieve keuze te maken.

3 HOOFDLETTERS

Wijdverbreid misverstand is dat een in hoofdletters opgestelde tekstregel beter zou worden gelezen dan "kleine letters". NIETS IS MINDER WAAR. Pardon, niets is minder waar. Naast elkaar geplaatste hoofdletters zijn juist minder goed leesbaar, ook al vallen ze bij beperkt gebruik wel op....

Als U een tekstdeel extra wilt benadrukken, kunt U dat beter doen door of de tekst zelf in een andere kleur te zetten of de achtergrondkleur te wijzigen. (Zie onder 5 en 6). Het onderstrepen van tekst is bij viewdata niet mogelijk.

4 DUBBELE HOOGTE-LETTERS

Letters van dubbele hoogte nemen twee regels tekst in beslag in plaats van een regel. Ze zijn even "breed" als gewone letters. Door hun dubbele hoogte zijn ze moeilijk te mixen met de standaard tekstletters, of U moet zinnen of opsommingen van een regel hebben waarin zo'n dubbelhoog woord voorkomt, en een extra regelspatie niet gebruiken. Het gebruik van dubbelhoge hoofdletters als in 3 omschreven moet worden afgeraden.

Dubbelhoge letters lenen zich wel voor bijvoorbeeld een (korte) kopregel of tussenkop. Ook als U een erg beknopte boodschap op een viewdata-beeld zet.

5 HET GEBRUIK VAN KLEUR IN TEKST

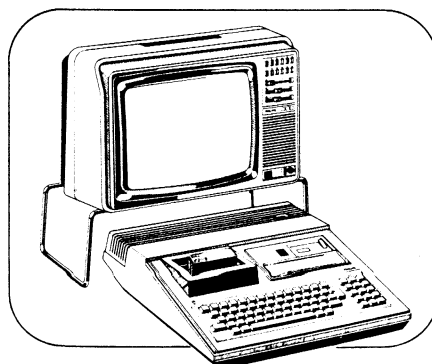
"Met mate" is hier het devies. Tekst kan naast wit ook in geel, lichtblauw en eventueel ook groen worden ingevoerd op zwarte achtergrond. Kleur moet echter altijd een functie hebben:

voor een (tussen)kop.

voor een onderwerp-wisseling of alinea-wisseling.

voor een woord of regel(deel) waarop de lezer extra moet letten.

voor een opsomming, zeker als daarbij hoofd- en sub-opsommingen zijn te onderscheiden.



Nu zien ogen niet alle kleuren even helder. Dat ziet U het beste als U het kleurentestbeeld van de NOS even zwart-wit bekijkt:
donkerblauw heeft een relatieve helderheid van 11 %,
rood een relatieve helderheid van 30 %,
groen 59 %.

Relatief wil zeggen: ten opzichte van wit dat 100 % helder is, de som van deze over elkaar heen geprojecteerde drie basis-kleuren. Telt U de percentages maar bij elkaar op en U ziet dat het klopt.

Dan zijn er ook mengkleuren:

violet (donkerblauw en rood samen), rel. helderheid 41 %,

geel (rood en groen samen), rel. helderheid 89 %,

lichtblauw (donkerblauw en groen samen) rel. helderheid 70 %

Welke mogelijkheden staan dus tot onze beschikking, in aflopende graad van helderheid?

wit	100 %
geel	89 %
lichtblauw	70 %
groen	59 %
violet	41 %
rood	30 %
donkerblauw	11 %
zwart	0 %, echter niet voor tekst

Wat doen we met deze wijsheid omtrent relatieve helderheid? Veel, als we bedenken dat een aantal viewdata-abonnees geen kleurenbeeld hebben, maar een compact zwart-wit kastje op het bureau. Een zwartwit kijker ziet nauwelijks een helderheidsverschil tussen wit en geel, of geel en lichtblauw. Dat wil zeggen: hij merkt het helderheidsverschil niet als de tekstletters tussentijds van kleur veranderen (wit-geel of geel-lichtblauw: grotere contrasten zijn in zwart-wit wel zichtbaar, maar in kleur niet acceptabel). De zwart-wit kijker merkt geringe helderheids-nuances wel op bij (grotere) vlakken. Dat is de reden waarom we blijven zeggen: kleurverschillen in tekst "met mate". Er zijn immers meer mogelijkheden voorhanden om iets te accentueren.

6 HET GEBRUIK VAN ACHTERGRONDKLEUR

Behalve zwart als achtergrondkleur is ook donkerblauw goed te gebruiken. Kleurenkijkers zullen een gele tekst erop waarderen, zo blijkt uit onderzoek. Voor de zwart-wit kijkers maakt het weinig uit mits het contrast maar groot genoeg is. Voor teksten met hoge rel. helderheid moeten andere achtergrondkleuren worden ontraden. Witte tekst op rood fond zou qua contrast wel gaan, maar U kunt de kleurenkijker ook op een andere manier een klap in de ogen geven.

Het kan ook anders. U kunt de combinatie toch omdraaien? Met uitzondering van zwarte tekst, want dat is in de huidige viewdata-techniek onmogelijk:

donkerblauwe tekst op witte achtergrond

donkerblauwe tekst op gele achtergrond

groene tekst op witte achtergrond,

om de meest gereede kleurencombinaties op te noemen.

Bedenk wel dat het omdraaien van tekst-/achtergrondcombinaties ook een medicijn met bijverschijnselen is! Stelt U zich voor: steeds consequent 'n blauw fond met een gele tekst en ineens een helgeel oplichtend vakje waarin een donkerblauw woord -nou ja- wel leesbaar is, maar daar is ook alles mee gezegd. De lezer knipperde met zijn ogen voordat hij het woord kon onderscheiden en lezen; en dat was niet de bedoeling.

Ook hier is het dus oppassen geblazen. U hoeft niet naar de schoonheidsprijs te streven; leesbaarheid is en blijft het belangrijkste.

7 KNIPPEREN

Alleen als andere methoden om de aandacht van de kijker op een specifieke plaats van het beeld te richten falen. Alleen dan is knippen een redmiddel. Het irriteert snel dus gebruik het als laatste redmiddel of met een Zeer Bijzonder Doel. Beperk de oppervlakte van het knipperend element tot het hoogstnoodzakelijke: een cijfer, een letter of -als U een woord of zin(sdeel) wilt benadrukken- twee rechthoekige vlakjes voor, resp. achter dat zo belangwekkende element.

8 CONSEQUENT ZIJN

Bij het samenstellen van teksten en ideeën voor viewdata-beelden moet U vooral consequent zijn. U hoeft beslist niet te doen wat Uw buurman doet, maar een eenmaal gekozen gedragslijn omtrent tekst-stijl en visuele presentatie moet U terwille van de lezer rigoureuus volhouden. Want herkenbaarheid van Uw specifieke (wijze van) informatieverschaffing staat hoog op het verlanglijstje van elke viewdata-abonnee.

9 DURVEN

Na het lezen van zoveel "do's" en "don'ts" vergaat bij U misschien de lust tot experimenteren. Dat was niet de bedoeling van dit verhaal; het toonde alleen maar de gulden middenwegen, zonder U te verplichten tot platgetreden paden...

Durf dus gerust iets te doen. In het allerergste geval is het onmogelijk. In het allerbeste geval is het zonder enige wijziging mogelijk. Elk gedeeltelijk haalbaar idee moet voor U een uitdaging zijn.

Nu er door verschillende dealers alternatieve floppy drives (TEAC, OLIVETTI, BASF,) met meer dan 35 tracks geleverd worden (bijv. 40 tracks), is het interessant gebleken om de overige 5 tracks (netto 20 Kb!) in het systeem op te nemen. Ook sommige door PHILIPS zelf geïnstalleerde drives in de P2000 (dit zijn de SHUGART SA 400L i.p.v. SA 400) bezitten 40 tracks.

Ervaring is opgedaan met BASIC 1.4 en UCSD. Binnenkort zullen ook BASIC 1.6 en CP/M aangepast kunnen worden.

De aanpassing kan op twee manieren gebeuren:

- Na het opstarten van het systeem.
(BASIC: 'poken', UCSD: 'debug')
- Door middel van veranderingen, direct op de schijf aan te brengen. Dit laatste zou kunnen door een 'patch'-cassette uit te geven, die de BASIC of UCSD bytes op de schijf veranderen en eventueel weer terug zetten.

Veranderingen voor BASIC 1.4:

PDOS (Ram bank 2, zie voorbeeld poken in 2e bank FORMAT programma op P2000gg-diskette 1):

Location in RAM	Old	New	Track no	Byte offset on track	Comment
E3F3	23	28	1	03F3	Number of tracks
E543	23	28	1	0543	Number of tracks
E68D	88	9C	1	068D	MAXBYT: 139K
F6F6	88	9F	2	06F6	MAXALL: 140K
Veranderingen voor UCSD 4.03:					
<u>PBIOS:</u>					
63B4	23	28	1	004D	Number of tracks

Opmerking voor UCSD:

In feite kunnen er drives met meer dan 40 tracks toegepast worden aangezien het operating system hierin voorziet (voor FILER zero-ing meer blocks opgeven, bijv. 40 tracks: 312 blocks)

Opmerking algemeen:

De floppies moeten opnieuw geformatteerd worden; gebruik hiervoor het FORMAT-programma op P2000gg-diskette 1.

Ted Jonker, Naarden

NOG EEN MANIER OM STRING-ARRAYS NAAR CASSETTE TE SCHRIJVEN

Het is niet mogelijk een string-array in BASIC direct naar cassette te schrijven. Het string-array AB\$ b.v. kan niet weggeschreven worden met de instructie CSAVE*AB\$. Dit is wel mogelijk met numerieke arrays. We kunnen dus wel de opdracht geven CSAVE*AB%. Er zijn verschillende pogingen ondernomen om string-arrays toch naar cassette te kunnen schrijven, zoals b.v. in het PIMS-programma, maar de ideale methode is nog niet gevonden. In dit artikelje wordt een heel eenvoudige methode beschreven, die wel is waar niet ideaal, maar toch zeer bruikbaar is. Tot beter begrip van de toegepaste truc, eerst wat achtergrond.

HOE STAAN VARIABELEN IN HET GEHEUGEN?

Achter het programma begint de variabelenruimte. De variabelen worden aangemaakt "in volgorde van opkomst" en elke variabele beslaat een vast aantal bytes. Het eerste byte geeft het soort variabele aan. Het tweede en derde byte geven de naam van de variabele (b.v. AB) en daarna komt de waarde. Hiervoor gebruikt de P2000 2 bytes voor een integer, 4 voor een real, 8 voor een dubbele precisie en 3 bytes voor een string.

Voor arrays geldt iets dergelijks. Eerst komt het soort array, dan de naam, de lengte (b.v. 101) en dan de 101 array-elementen elk weergegeven door zoveel bytes als boven beschreven.

Er worden dus 3 bytes gebruikt om de "waarde" van een string-variabele aan te geven. Nu kan een string meestal niet in 3 bytes worden weergegeven. Een string kan immers tot 255 bytes lang zijn. Kennelijk staat een string niet in het geheugen, zoals de "gewone" variabelen. Waarom niet? Daarvoor kunnen we een aantal redenen bedenken. Als strings direct in de variabelen-ruimte zouden worden geplaatst, dan zouden we voor elke string de maximale lengte, 255 bytes, moeten reserveren. Het geheugen zou dan gauw "op" zijn, terwijl er misschien bijna niets in zou staan. Zouden we de ruimte, die een string nodig heeft iedere keer aanpassen, (b.v. na de instructie AB\$=A\$+B\$), dan zouden we alle variabelen, die na AB\$ staan moeten verplaatsen, wat verschrikkelijk veel tijd kan kosten. (Misschien hebben we het daar nog wel eens over in een volgende P2000 Nieuwsbrief).

Om deze en mogelijk ook nog om andere redenen, gebruikt de P2000 de zgn "string-descriptor". Dat is een blokje van 3 bytes, waarvan het eerste byte de lengte aangeeft van de string en de twee volgende bytes het begin-adres van de string en deze stringdescriptor staat achter de naam van de stringvariabele. Vandaar de drie bytes.

Als er dus in de variabelen-ruimte een blokje van 6 bytes staat, dat er uitziet als:

3 65 66 25 24 121

dan betekent dit: Dit is een string-variabele, naam AB (65,66), lengte 25 bytes en het begin is te vinden op geheugenplaats $121 \cdot 256 + 24 = 31000$.

Wat er in het geheugen vanaf 31000 staat is niet belangrijk. De P2000 zal adres 31000 t.m. 31024 zien als de string AB\$. Deze geheugenplaatsen kunnen dus liggen: In het programma, in de "stringspace" (een stuk geheugen, waar alle strings terecht komen, waarmee de P2000 geen raad weet), maar het kan ook zijn: EEN STUK VAN EEN INTEGER ARRAY. Hierop is het nu volgende principe van stringopslag en behandeling gebaseerd: Als je geen string arrays kan wegschrijven, zorg er dan voor, dat je ze ook niet hebt!!

Onze string-administratie komt hierop neer:

1. Zet een string byte voor byte weg in een integer (of real) array
2. Laat de P2000 naar het array kijken alsof er strings in staan
3. Schrijf het array op de gewone manier weg (CSAVE*AB%)

Alvorens deze punten verder te behandelen eerst wat DIM's en DEF's. DIM een integer array, b.v. 2000 bytes lang (DIM AB%(999)). De plaats van een byte in het array vinden we met een DEF FN voor de plaats P% vanaf het begin
 DEF FN PL% (P%) = VARPTR (AB%(0)) + P%

1. INVOEREN

We kunnen een string b.v. invoeren met LINEINPUT of met een GET. Daarna POKE-n we de string in het array. Echter, we moeten straks kunnen herkennen, waar de vorige string ophoudt en de volgende begint. Daarom voegen we vooraan de string z'n lengte toe. Dit kan op verschillende manieren:

```

LINEINPUT AB$ : AB$ = CHR$( LEN( AB$))
+ AB$ : FOR I = 1 TO LEN( AB$) : POKE
FN PL% (P%), ASC( MID$( AB$, I, 1)) :
P% = P% +1 : NEXT
  
```

Of op elke andere, elegantere manier. Als we de string met GET binnenhalen, kunnen we ieder karakter meteen na het intypen wegzetten in het array. Na afsluiten van de invoer zetten we de lengte er voor.

Na een tijdje invoeren hebben we dan een array gekregen, dat er ongeveer zo uitziet:

```
3aap4noot4mies3wim3zus3jet4teun4vuur
```

en bovendien kunnen we aflezen, dat P% = 36, de eerste vrije plaats is in het array.

2. HET ZIEN ALS STRINGS

Stel we willen alle strings onder elkaar afgedrukt hebben. Dan gebruiken we het volgende programma'tje:

```

10 PX%=0 : T$=""
20 POKE VARPTR(T$),PEEK (FNPL% (PX%)) :
POKE VARPTR (T$)+1,FNPL%(PX%)MOD256 :
POKE VARPTR (T$)+2,FNPL%(PX%)'256
30 PRINT T$
40 PX% = PX% + PEEK(FNPL%(PX%))+1 :
IF PX% < P% THEN 20 ELSE END
  
```

Aangezien op T\$ alle string-operaties kunnen worden toegepast, kunnen we ook vragen een bepaalde string op te zoeken (in regel 30 b.v. IF R\$ = T\$ THEN) of om een woord in het bestand te vinden (IF INSTR (T\$,"uu") = 0 THEN 40 ELSE)

Vergeleken met "echte" string arrays heeft deze methode van stringbeheer wel enkele nadelen, speciaal als het gaat om het veranderen of verwisselen van strings. Daarvoor komen we dan wel in de machinetaal terecht. Nu hoeft dat ook weer niet zo erg veel pijn te doen. De Z80 kent n.l. een "blocmove" instructie, waarmee snel een "gat" in het array AB% gevuld kan worden. Bij het veranderen van een string kunnen we dan de nieuwe string achteraan bijschrijven en vervolgens alle bytes opschuiven over de "oude string" heen. Voor de niet-machinetaal programmeurs onder ons is het wellicht een goed begin tot het machinetaal-programmeren. De hele blocmove routine beslaat 12 bytes!

Voordelen van de beschreven methode zijn o.m.

-zonder meer toe te passen op 16K, 32K en 48K machines. Data kunnen worden ingelezen in elke andere machine. (Dit in tegenstelling tot de PIMS methode)

-zuinig met geheugen. Per string is slechts 1 byte extra nodig. De strings worden pas string als ze als string gebruikt worden.

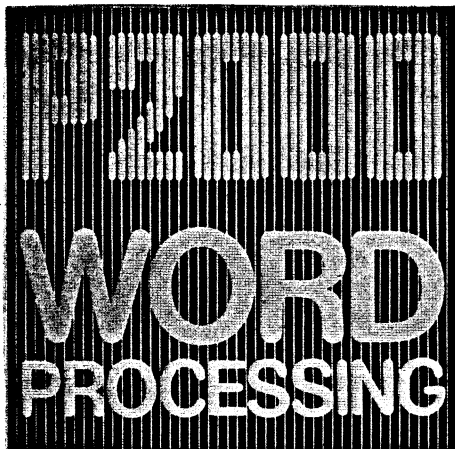
Wat kan er mis gaan ?

Let vooral op het volgende: Als een stringdescriptor is ingePOKEd, (zoals op regel 20) kunnen we in principe alles doen met T\$. Bedenk evenwel, dat T\$ geen "echte" string is. Zo gauw we zeggen T\$ = T\$ + "gijs", of zoiets, vinden we T\$ niet meer in het array AB%, maar in de stringospace. Willen we zoiets doen, dan moeten we T\$ weer opnieuw "bijschrijven" in het array. Verder nog dit: Als we tijdens ons handelen met T\$ een variabele gebruiken, die nog niet eerder is voorgekomen, dan gaat alles mis. In dat geval schuift het array AB% een stukje op om plaats te maken voor de nieuwe variabele. Deze moet nl tussen het array en de laatste variabele worden gezet. De stringdescriptor, echter, "kijkt" nog naar de oude plaats. Niet alleen veranderen begin en eind van T\$ hierdoor, de berekening van het begin van alle volgende strings (regel 40) gaat ook fout.

Daarom: zet eerst ALLE te gebruiken variabelen even op 0.

Laatste opmerking: Bij lange arrays wordt FN PL% negatief. Er moet dus gekeken worden of er mogelijk 65536 bij moet worden opgeteld.

D.J. Kroon



Werken met de computer; soms oppassen?

Kunnen we zonder schadelijke gevolgen kinderen achter de computer zetten en kunnen we zelf uren naar het beeldscherm zitten turen zonder dat het kwaad kan? Deze vragen hielden mij intensief bezig na zelf lichamelijke klachten gekregen te hebben.

In de buurt van Schoorl ligt het Energie-onderzoek-Centrum Nederland (ECN te Petten). Verschillende ouders van onze school werken daar. Via deze ouders heeft de medische dienst van het ECN wat adviezen gegeven over het werken met computers. In Petten zitten vele medewerkers achter beeldschermterminals; maar bovenal wordt daar veel onderzoek gedaan naar gevolgen en effecten van allerlei soorten straling.

De medische dienst gaf mij enkele adviezen, waarvan hieronder een overzicht:

De Röntgenstraling die in elke beeldbuis wordt opgewekt is bij een KTV 500 keer zo groot als bij een zwart-wit TV. Of een TV of een RGB-monitor wordt gebruikt maakt niet uit. De straling komt uit de buis. Zet de TV zover mogelijk weg als het om een KTV gaat. De straling is overigens erg klein en kan normaal geen kwaad. Gevolgen zijn overigens pas op heel lange termijn merkbaar. Bloed en weefsels kunnen worden aangetast. Dat kan alleen gebeuren bij defekte of ondeugdelijke apparaten. Philips werd genoemd als een veilig merk omdat de beeldbuizen in de fabriek worden getest op straling. Ze voldoen aan de IEC 65 norm.

Waar we wel voor moeten oppassen is iets heel anders als straling. De medische dienst had gegevens over dampen die bij warmte afgegeven worden door kunststoffen. Deze dampen kunnen irritaties veroorzaken van de slijmvliezen (ogen, luchtwegen) en kunnen huidandoeningen veroorzaken. In het algemeen moet gezorgd worden voor goede ventilatie als er met kunststoffen wordt gewerkt. In de klas zal dat wel niet veel problemen geven; maar velen van ons zijn hobbyisten die mogelijk thuis hun P2000 met toebehoren op een benauwd zolderkamertje gebruiken. Dan is het wel ventileren geblazen. Alle elektronische apparaten zitten vol met allerlei kunststoffen en zitten vaak ook nog in een kast van kunststof. De toestellen worden warm en geven dampen af die moeten worden afgevoerd.

ECN adviseerde om elk nieuw apparaat een hele dag aan te laten staan in een goed geventileerde ruimte. De ergste dampen zijn dan weg.

Louis Naus, Schoorl

```

540 REM Programma voor het uitprinten van Viewdata
    karakters met een Epson MX printer in Bit Image Mode.
545 REM Auteur Ton Hilgersom
550 RESTORE 590: FOR I=&H9FEF TO &H9FF3
560 READ A$: POKE I,VAL("&H"+A$): NEXT
570 DEFUSR=&H9FEF: X=USR(27): X=USR(64): RETURN
590 DATA 4E,CD,5D,0E,C9
600 X=USR(27):X=USR(64):FOR R=&H5000 TO &H5800 STEP80:
    X=USR(10): R1=R: FOR D=R1 TO R1+40 STEP40: D1=D: F=0:
    G=0: FOR C=D1 TO D1+39: AZ=PEEK(C)
605 IF AZ=0 THEN IF F<>0 THEN GOSUB 620: X=USR(32):
    NEXT C,D,R: RETURN ELSE X=USR(32): NEXT C,D,R: RETURN
610 IF AZ<32 THEN 700 ELSE IF G=0 THEN X=USR(AZ):
    NEXT C,D,R: RETURN
615 F=F+1: NEXT C: GOSUB 620: NEXT D,R: RETURN
620 FH=6*F: FLZ=FHM0D256: FHZ=FH'256: X=USR(27):
    X=USR(65): X=USR(8): X=USR(27): X=USR(75): X=USR(FLZ):
    X=USR(FHZ)
625 FOR I=C-F TO C-1: A=PEEK(I): IF A>127 THEN A=255-A
630 IF (A AND 1)=1 THEN AZ=224 ELSE AZ=0
640 IF (A AND 4)=4 THEN AZ=AZ+24
650 IF (A AND 16)=16 THEN AZ=AZ+7
660 X=USR(AZ): X=USR(AZ): X=USR(AZ)
670 IF (A AND 2)=2 THEN AZ=224 ELSE AZ=0
680 IF (A AND 8)=8 THEN AZ=AZ+24
685 IF (A AND 64)=64 THEN AZ=AZ+7
690 X=USR(AZ): X=USR(AZ): X=USR(AZ): NEXT I: F=0:
    RETURN
700 IF F<>0 THEN GOSUB 620
710 X=USR(32): IF AZ>0 AND AZ<14 OR AZ=24 THEN G=0:
    NEXT C,D,R: RETURN
720 IF AZ>16 AND AZ<28 THEN G=1
730 NEXT C,D,R: RETURN

```

```

15000 REM *** Subroutine tbv sortering
15010 REM Array P(I)
15015 REM 'LAST' = Aantal elementen
15020 IDZ=4
15030 IF IDZ>LAST GOTO 15050
15040 IDZ=IDZ*2: GOTO 15030
15050 IDZ=IDZ-1
15060 IDZ=IDZ*2
15070 IF IDZ<16 GOTO 15180
15080 ND=LAST-IDZ
15090 FOR L=1 TOND
15100 I=L
15110 IF P(I,0V)<=P(I+IDZ,0V) GOTO 15160
15120 SWAPP(I),P(I+IDZ)
15140 I=I-IDZ
15150 IF I>=16 GOTO 15110
15160 NEXT L
15170 GOTO 15060
15180 RETURN
15190 REM *** Einde subroutine

```

```

10 REM DISK LEZEN en/of SCHRIJVEN track
15 REM voor track onder 16K Cassette-
20 REM BASIC.
25 REM DISK HEADER &H6070 tot &H608F
30 REM Job Van Broekhuijze Assurantien
35 REM Rijsingel 13
40 REM 2987 SB Ridderkerk
45 REM tel. 01804-11221
50 REM versie dd 20/12/82
55 :
60 REM &H6070 BUFFER Low Byte
65 REM &H6071 BUFFER High Byte
70 REM &H6072 Lengte opdracht moet
75 REM 09 zijn
80 REM &H6073 Lees (&H42) of schrijf
85 REM (&H45)
90 REM &H6074 Drive nummer. Ook op
95 REM &H607E en &H6082
100 REM &H6075 Track nummer. Op &H607F
105 REM track nummer -1
110 REM &H6076 Head Adress (kant van
115 REM de DISK) moet 00 zijn
120 REM &H6077 Sector moet 01 zijn
125 REM &H6078 Bytes code moet 01 zijn
130 REM &H6079 Aantal sectors moet &H10
135 REM zijn
140 REM &H607A Gap lengte moet &H0E
145 REM zijn
150 REM &H607B Code DATA lengte moet
155 REM 00 zijn
160 REM &H607C Lengte opdracht moet
165 REM 03 zijn
170 REM &H607D Code opdracht moet &H0F
175 REM zijn
180 REM &H607E Drive nummer
185 REM &H607F Track nummer -1
190 REM &H6080 Lengte opdracht moet 02
195 REM zijn
200 REM &H6081 Code opdracht moet &H07
205 REM zijn
210 REM &H6082 Drive nummer
215 REM &H6083 to &H608F worden door
220 REM PHILIPS wel goed gezet
225 REM *****
250 DATA F3: REM DI
252 DATA ED,73,38,60: REM LD (603B).SP
254 DATA 3E,01: REM LD A.01
256 : REM Regel 360 t/m 400
258 DATA D3,88: REM OUT 88
260 : REM Disable Z80 contr.

```

```

262 DATA D3,89:      REM OUT 89
264 DATA D3,8A:      REM OUT 8A
266 DATA D3,8B:      REM OUT 8B
268 DATA CD,E2,0E:   REM CALL 0EE2
270 :                 REM Initialiseren
272 DATA CD,8B,0F:   REM CALL 0FB8
274 :                 REM Zet motor aan
276 DATA CD,7D,0F:   REM CALL 0F7D
278 :                 REM step
280 DATA CD,25,C1:   REM CALL C125
282 :                 REM WDTR/RDTR
284 DATA 3E,03:      REM LD A.03
286 :                 REM regel 450 t/m 460
288 DATA D3,8B:      REM OUT 8B
290 :                 REM Re-enable CTC
292 DATA ED,7B,38,60: REM LD SP.(603B)
294 :                 REM Herstel L.SP
296 DATA E7:         REM RST 20H
298 :                 REM Re-enable keyboard
300 DATA C9:         REM RET
302 DATA FD,21,72,60: REM LD IY.6072
304 :                 REM Regel 500 t/m 820
306 :                 REM subroutine
308 DATA 21,5A,0F:   REM LD HL.0F5A
310 :                 REM voor WRITE/READ
312 DATA 22,20,60:   REM LD (6020).HL
314 DATA AF:         REM XOR A
316 DATA 3C:         REM INC A
318 DATA FD,77,05:   REM LD (IY+05).A
320 DATA 21,72,60:   REM LD HL.6072
322 DATA CD,A5,0F:   REM CALL 0FA5
324 DATA 3E,C5:      REM LD A.C5
326 DATA D3,89:      REM OUT 89
328 DATA 3E,01:      REM LD A.01
330 DATA D3,89:      REM OUT 89
332 DATA 2A,70,60:   REM LD HL.(6070)
334 :                 REM buffer adres
336 DATA 0E,8D:      REM LD C.8D
338 DATA 3E,0D:      REM LD A.0D
340 :                 REM disk motor
342 DATA D3,90:      REM OUT 90
344 DATA 1E,10:      REM LD E.10
346 DATA 3E,00:      REM LD A.00
348 :                 REM Select bank 0 or 1
350 DATA D3,94:      REM OUT 94
352 DATA DB,90:      REM IN A.90
354 DATA 1F:         REM RRA
356 DATA 30,FB:      REM JR NC.C151
358 :                 REM regel 690
360 DATA ED,FF:      REM FF= A2 of A3
362 :                 REM schrijven (&HA3) of
364 :                 REM lezen (&HA2) op
366 :                 REM &HC157
368 DATA 1B,F7:      REM JR NC.C151
370 :                 REM regel 690
372 DATA 1D:         REM DEC E
374 DATA DB,90:      REM IN A.90
376 DATA 1F:         REM RRA

```

```

378 DATA 30,FB:      REM JR NC.C15B
380 :                 REM regel 750
382 DATA ED,FF:      REM zie 360
384 :                 REM schrijven of lezen
386 DATA 20,F7:      REM JR NZ.C15B
388 DATA 3E,00:      REM LD A.00
390 :                 REM Select bank 0 or 1
392 DATA D3,94:      REM OUT 94
394 DATA C9:         REM RET
400 REM *****
405 PRINTCHR$(12);CHR$(4):PRINT:RESTORE2
50:FORI=&HC100TO&HC168:READI$:POKEI,VAL(
"&H"+I$):PRINT USING"  ";I$;NEXT
415 CLEAR 50,&HC000:DEFUSR=&H42E:X=USR(0
):DEFUSR=&HC100:DEFSTRW:DEFNW(X,Y)=CHR$(
4)+CHR$(X)+CHR$(Y):POKE&H6070,0:POKE&H6
071,&HD0:GOTO425
420 PRINTCHR$(12)
425 PRINTFNW(5,25)"WILT U LEZEN (1) OF S
CHRIJVEN (2)";:LINE INPUTW
430 IF W="1"THEN POKE&H6073,&H42:POKE&HC
157,&HA2:POKE&HC161,&HA2:GOTO465
435 IF W="2"THEN POKE&H6073,&H45:POKE&HC
157,&HA3:POKE&HC161,&HA3:GOTO465
440 GOTO420
445 FORI=&HD000TO&HE000:POKE I,&H32:NEXT
:RETURN
450 JZ=1:J=&H5000:FORI=&HD000TO&HE000:PO
KEJ,PEEK(I):J=J+1:JZ=JZ+1
455 IFJ=&H5000+24*80THENPRINTFNW(24,1):J
=&H5000+23*80
460 NEXT:PRINTFNW(24,50)"AANTAL INGELEZE
N BYTES IS "JZ:RETURN
465 AZ=0:BZ=0:PRINTCHR$(12)FNW(10,5);:IN
PUT"Geef het TRACK nummer (1-35) ";AZ
470 IFAZ=0THEN420
471 INPUT" Geef het DISK nummer (1-2)
op ";BZ
475 IFAZ>35THENAZ=35:REM Drives met meer
dan 35 tracks een hoger nummer
480 IF BZ<1THENBZ=1
485 IFBZ>2THENBZ=2:REM bij vier drives
is BZ max 4
490 POKE&H6075,AZ:POKE&H607F,AZ-1:REM
select track
495 POKE&H6074,BZ:POKE&H607E,BZ:POKE&H60
82,BZ:REM select drive
500 X=USR(0):IFW="1"THENGOSUB450
505 LINE INPUT WW:WW="":OUT&H90,0:REM ze
t motor af
510 GOTO465
515 JJ=1:J=5:FORI=&H6070TO&H608F:PRINTFN
W(J,JJ);HEX$(I),HEX$(PEEK(I)):J=J+1
520 IFJ=21THENJ=5:JJ=40
525 NEXT:RETURN
999 END
1000 RESET:SAVE"DISK"
1001 END

```

```

0000      ;      DUMP   PROGRAM READS INPUT FILE AND DISPLAYS HEX DATA
0000      ;
A000      ;      ORG    0A000H
A000      ;
A000      ;
A000      ;      EQUATES FOR SEVERAL FUNCTIONS
A000      ;
0005      B005   EQU    5
6916      SCREEN EQU    6916H ;DISPLAY ROUTINE
6CBE      ASCII  EQU    6CBEH ;GET ASCII CODE FROM KEY
0026      GETKEY EQU    26H   ;KEYBOARD INPUT ROUTINE
0029      KSTAT  EQU    29H   ;KEYBOARD STATUS ROUTINE
000F      OPENF  EQU    15    ;OPENFILE COMMAND
0014      READF  EQU    14H   ;READ SECTOR COMMAND
A000      ;
A000      ;      NON-GRAPHIC CHARACTERS
A000      ;
000D      CR     EQU    13    ;CARRIAGE RETURN
000A      LF     EQU    10    ;LINE FEED
A000      ;
A000      ;      SAVE OLD STACK
A000      ;
A000 ED73CAA1 LD     (OLDSP),SP
A004      ;
A004      ;      DEFINE NEW STACK AREA
A004      ;
A004 3162A3   LD     SP,STKTOP
A007      ;
A007      ;      GET FILENAME AND OPEN FILE
A007      ;
A007 CDD7A0   CALL   SETUP
A00A FEFF     CP     0FFH    ;OPEN NOT OK !
A00C 2009     JR     NZ,OPENOK
A00E      ;
A00E      ;      FILE NOT HERE, GIVE ERROR MESSAGE AND RETURN
A00E      ;
A00E 115BA1   LD     DE,OPNMSG
A011 CDA2A0   CALL   ERR
A014 C36DA0   JP     FINISH
A017      ;
A017      ;      OPEN OPERATION OK, SET BUFFER INDEX TO END
A017      ;
A017 3E00     OPENOK:  LD     A,0
A019 32CBA1   LD     (IBP),A      ;SET BUFFER POINTER TO
A01C 3221A3   LD     (LINE),A
A01F 3C       INC     A
A020 32CCA1   LD     (FLAG),A
A023 210000   LD     HL,0        ;START AT ADDRESS 0
A026      ;
A026      ;      HL CONTAINS NEXT ADDRESS TO PRINT
A026 E5       BLOOP:  PUSH  HL
A027 CDACA0   CALL  GNB         ;RECALL LINEPOSITION
A02A E1       POP   HL         ;RECALL LINE POSITION
A02B 3840     JR     C,FINISH    ;CARRY SET BY GNB IF EN
A02D 47       LD     B,A
A02E      ;

```

```

A02E ; PRINT HEX VALUES
A02E ; CHECK FOR LINE FOLD
A02E ;
A02E 7D LD A,L
A02F E607 AND 7H ;CHECK LOW 4 BITS
A031 202D JR NZ,NONUM
A033 ;
A033 ; PRINT LINENUMBER
A033 ;
A033 CD79A0 CALL CRLF
A036 ;
A036 ; CHECK FOR BREAK KEY
A036 ;
A036 CD2900 CALL KSTAT
A039 3832 JR C,FINISH ;DONT PRINT ANY MORE
A03B ;
A03B 3A21A3 LD A,(LINE)
A03E FE17 CP 23
A040 2012 JR NZ,NXLINE
A042 11AFA1 LD DE,MESCON
A045 CDA2A0 CALL ERR
A048 CD2600 CALL GETKEY
A04B 3E0C LD A,12
A04D CD1669 CALL SCREEN
A050 AF XOR A
A051 3221A3 LD (LINE),A
A054 3C NXLINE: INC A
A055 3221A3 LD (LINE),A
A058 7C LD A,H
A059 CD95A0 CALL PHEX
A05C 7D LD A,L
A05D CD95A0 CALL PHEX
A060 ;
A060 23 NONUM: INC HL ;TO NEXT LINENUMBER
A061 3E20 LD A,' ' ;SPACE
A063 CD75A0 CALL PCHAR
A066 78 LD A,B
A067 CD95A0 CALL PHEX
A06A C326A0 JP GLOOP
A06D ;
A06D ; END OF DUMP, RETURN TO BASIC
A06D ;
A06D CD79A0 FINISH: CALL CRLF
A070 ED7BCAA1 LD SP,(OLDSP)
A074 C9 RET
A075 ;
A075 ;
A075 ; SUBROUTINES
A075 ;
A075 ;
A075 CD1669 PCHAR: CALL SCREEN
A078 C9 RET
A079 ;
A079 3E0D CRLF: LD A,CR
A07B CD1669 CALL SCREEN
A07E 3E0A LD A,LF
A080 CD1669 CALL SCREEN
A083 C9 RET
A084 ;
A084 ; PRINT NIBBLE
A084 ;
A084 E60F PNIB: AND 0FH ;LOW 4 BITS
A086 FE0A CP 10 ;
A088 3005 JR NC,PL10

```

```

A08A      ;
A08A      ;          LESS THAN OR EQUAL 9
A08A      ;
A08A C630      ADD    A,'0'
A08C C391A0    JP     PRN
A08F      ;
A08F      ;          GREATER THAN OR EQUAL 10
A08F      ;
A08F C637      PL10:  ADD    A,37H
A091 CD75A0    PRN:   CALL   PCHAR
A094 C9       RET
A095      ;
A095      ;          PRINT HEX CHARACTER IN REGISTER A
A095      ;
A095 F5       PHEX:   PUSH   AF
A096 0F       RRCA
A097 0F       RRCA
A098 0F       RRCA
A099 0F       RRCA
A09A CDB4A0    CALL   PNIB   ;PRINT NIBBLE
A09D F1       POP    AF
A09E CDB4A0    CALL   PNIB
A0A1 C9       RET
A0A2      ;
A0A2      ;          PRINT (ERROR) MESSAGE
A0A2      ;          D,E ADDRESSES MESSAGE ENDING WITH '$'
A0A2      ;
A0A2 1A       ERR:   LD     A,(DE)
A0A3 FE24      CP     '$'
A0A5 C8       RET    Z
A0A6 CD1669    CALL   SCREEN
A0A9 13       INC    DE
A0AA 1BF6      JR     ERR
A0AC      ;
A0AC      ;          GET NEXT BYTE, IF NECESSARY READ NEXT SECTOR
A0AC      ;
A0AC 3ACCA1    GNB:   LD     A,(FLAG)
A0AF B7       OR     A
A0B0 2006      JR     NZ,READ
A0B2 3AC8A1    LD     A,(IBP)
A0B5 47       LD     B,A
A0B6 1011      DJNZ  GLO
A0B8 CDF6A0    READ:  CALL   DISKR
A0BB B7       OR     A
A0BC 2009      JR     NZ,CLOSE   ;END OF FILE
A0BE AF       XOR    A
A0BF 32CCA1    LD     (FLAG),A
A0C2 32C8A1    LD     (IBP),A
A0C5 1B02      JR     GLO
A0C7      ;
A0C7 37       CLOSE  SCF
A0C8 C9       RET
A0C9      ;
A0C9 5F       GLO:   LD     E,A
A0CA 1600      LD     D,0
A0CC 3D       DEC    A
A0CD 32C8A1    LD     (IBP),A

```

```

A0D0 ;
A0D0 ; POINTER IS INCREMENTED
A0D0 ; SAVE CURRENT FILE ADDRESS
A0D0 ;
A0D0 21FEA1 LD HL,BUFF
A0D3 19 ADD HL,DE
A0D4 ; ABSOLUTE CHARACTER ADDRESS IN HL
A0D4 7E LD A,(HL)
A0D5 B7 OR A ;RESET CARRY
A0D6 C9 RET
A0D7 ;
A0D7 ; DISPLAY DUMP MESSAGE
A0D7 1143A1 SETUP: LD DE,SIGNON
A0DA CDA2A0 CALL ERR
A0DD ; SETUP FILENAME
A0DD 1180A1 LD DE,MESFN
A0E0 CDA2A0 CALL ERR
A0E3 ; GET FILENAME
A0E3 CD05A1 CALL FNAME
A0E6 AF XOR A
A0E7 321EA3 LD (FCBCR),A
A0EA CD31A1 CALL SETDMA
A0ED 11FEA2 LD DE,FCB
A0F0 0E0F LD C,OPENF
A0F2 CD0500 CALL BDDS
A0F5 ; 255 IN ACCUMULATOR IF OPEN ERROR
A0F5 C9 RET
A0F6 ;
A0F6 ; READ DISK FILE RECORD
A0F6 ;
A0F6 ;
A0F6 E5 DISKR: PUSH HL
A0F7 D5 PUSH DE
A0F8 C5 PUSH BC
A0F9 11FEA2 LD DE,FCB
A0FC 0E14 LD C,READF
A0FE CD0500 CALL BDDS
A101 C1 POP BC
A102 D1 POP DE
A103 E1 POP HL
A104 C9 RET
A105 ;
A105 ; GET FILENAME
A105 ;
A105 ;
A105 21CDA1 FNAME: LD HL,INPUT
A108 060D LD B,13
A10A C5 NEXT PUSH BC
A10B CD2600 CALL GETKEY
A10E CD8E6C CALL ASCII
A111 77 LD (HL),A
A112 23 INC HL
A113 CD1669 CALL SCREEN
A116 C1 POP BC
A117 10F1 DJNZ NEXT
A119 21CDA1 LD HL,INPUT
A11C 7E LD A,(HL)
A11D B7 OR A
A11E D641 SUB 41H
A120 32FDA1 LD (DRVNUM),A
A123 32FEA2 LD (FCB),A
A126 23 INC HL
A127 23 INC HL

```

```

A128 11FFA2          LD      DE,FCB+1          A1C8          ;
A12B 010B00          LD      BC,11             A1C8          ;          VARIABLE AREA
A12E ED80            LDIR                     A1C8          ;
A130 C9              RET                      A1C8          IBP:  DEFS  2
A131 11FEA1  SETDMA: LD      DE,BUFF          A1CA          OLDSP: DEFS  2
A134 0E1A            LD      C,1AH            A1CC          FLAG:  DEFS  1
A136 CD0500          CALL   B00S              A1CD          INPUT: DEFS  30H
A139 3AFDA1          LD      A,(DRVNUM)       A1FD          DRVNUM: DEFS  1
A13C 5F              LD      E,A              A1FE          BUFF:  DEFS  100H
A13D 0E0E            LD      C,14             A2FE          FCB:   DEFS  32
A13F CD0500          CALL   B00S              A31E          FCBCR: DEFS  3
A142 C9              RET                      A321          LINE:  DEFS  1
A143 ;
A143 ;               FIXED MESSAGE AREA          A322          ;
A143 ;               A322          ;          STACK AREA
A143 ;               A322          DEFS  40H
A143 46494C45  SIGNON: DEFM  'FILE DUMP VERSION 0.0'  A362          STKTOP:
A147 2044554D          0000          END
A148 50205645          0000
A14F 5253494F
A153 4E20302E
A157 30
A158 0D              DEFB  CR
A159 0A              DEFB  LF
A15A 24              DEFB  '$'
A15B 0A              OPNMSG: DEFB  10
A15C 0D              DEFB  13
A15D 4E4F2049          DEFM  'NO INPUT FILE PRESENT ON'
A161 4E505554
A165 2046494C
A169 45205052
A16D 4553454E
A171 54204F4E
A175 20544849          DEFM  ' THIS DISK'
A179 53204449
A17D 534B
A17F 24              DEFB  '$'
A180 47495645  MESFN:  DEFM  'GIVE FILENAME : '
A184 2046494C
A188 454E414D
A18C 45203A
A18F 0D              DEFB  CR
A190 0A              DEFB  LF
A191 443A4646          DEFM  'D:FFFFFFFFEXT'
A195 46464646
A199 46464558
A19D 54
A19E 0D              DEFB  CR
A19F 24              DEFB  '$'
A1A0 57524F4E  WRONG:  DEFM  'WRONG FILENAME$'
A1A4 47204649
A1A8 4C454E41
A1AC 4D4524
A1AF 54595045  MESCON: DEFM  'TYPE ANY KEY TO CONTINUE$'
A1B3 20414E59
A1B7 204B4559
A1BB 20544F20
A1BF 434F4E54
A1C3 494E5545
A1C7 24

```

ASCII	6CBE	B00S	0005
BUFF	A1FE	CLOSE	A0C7
CR	000D	CRLF	A079
DISKR	A0F6	DRVNUM	A1FD
ERR	A0A2	FCB	A2FE
FCBCR	A31E	FINISH	A06D
FLAG	A1CC	FNAME	A105
GETKEY	0026	GLO	A0C9
GLOOP	A026	GNB	A0AC
IBP	A1C8	INPUT	A1CD
KSTAT	0029	LF	000A
LINE	A321	MESCON	A1AF
MESFN	A180	NEXT	A10A
NONUM	A060	NXLINE	A054
OLDSP	A1CA	OPENF	000F
OPENK	A017	OPNMSG	A15B
PCHAR	A075	PHEX	A095
PL10	A05F	PNIB	A084
PRN	A091	READ	A088
READF	0014	SCREEN	6916
SETDMA	A131	SETUP	A0D7
SIGNON	A143	STKTOP	A362
WRONG	A1A0		

=====

Onder data communicatie wordt hier verstaan het uitwisselen van gegevens tussen twee computers. Voor een P2000 bezitter is viewdata wel het meest bekende voorbeeld van datacommunicatie. Geprobeerd zal worden in een aantal stukjes de basisbeginselen van datacommunicatie uit te leggen.

We gaan uit van de situatie dat de P2000 via de telefoon verbonden is met een andere computer. Wat gebeurt er als de P2000 de letter A over wil sturen? De letter A wordt overgestuurd als een getal waarvan de waarde overeenkomt met de ASCII waarde van A. Dit is 65 of in binaire codering 1000001. Een 0 of 1 in de binaire codering wordt een bit genoemd. Deze 7 bits kunnen niet tegelijkertijd worden verzonden omdat we maar 1 telefoondraad hebben. Daarom worden de 7 bits van de letter A stuk voor stuk na elkaar verstuurd, te beginnen met de meest rechtse. Om het de andere computer mogelijk te maken de verschillende bits onderling te onderscheiden worden afspraken gemaakt hoelang een bit aanwezig blijft op de telefoonlijn. Deze tijd per bit wordt gegeven door de baudrate. De baudrate is gedefinieerd als het aantal bits per seconde. Een baudrate van 1200 baud betekent dus 1200 bits per seconde, iedere bit blijft $1/1200$ sec is ongeveer 0.83 milliseconde op de lijn.

We bekijken nu eerst het versturen van een bit. Op de P2000 zijn twee outputpoorten aanwezig via welke bits naar buiten gestuurd kunnen worden. Dit zijn de printerkonnektor en het 2e I/O slot. We beperken ons even tot de printerkonnektor. Met behulp van het OUT kommando kunnen we pen 3 van de printerkonnektor naar believen 0 of 1 maken. Door op de juiste tijdstippen met dit OUT kommando pen 3 van 0 naar 1, resp. van 1 naar 0 te veranderen, kunnen we de 7 bits van de letter A versturen met de gewenste baudrate.

We zijn hier echter nog niet mee klaar want de nullen en enen uit de printerkonnektor van de P2000 kunnen niet rechtstreeks op de telefoonlijn gezet worden. Deze is namelijk ontworpen voor het versturen van spraak, geluid. Daarom wordt altijd een modem gebruikt. Het woord modem is een samenvoeging van MODulator/DEModulator. De taak van het modem is het omzetten van de nullen en enen in een toon met een bepaalde frequentie en deze tonen op de telefoonlijn te zetten. Daarnaast is het zijn taak om de tonen die van de andere computer komen weer terug te vertalen in de oorspronkelijke nullen en enen. Het volgende probleem is het verbinden van het modem met de printerkonnektor van de P2000. In principe zijn 5 draden nodig, een ontvang en een verstuur draad, een draad die de P2000 vertelt dat het modem verbinding heeft en omgekeerd een draad die het modem vertelt dat de P2000 aanwezig is, en tot slot een aarde. Er moet een vaste afspraak zijn tussen modem en computer hoe een bit 0 of 1 verstuurd wordt. Deze afspraak staat bekend als de RS232 standaard. Volgens deze standaard hoort bij bit=1 een spanning tussen -3 en -15 Volt en bij bit=0 een spanning tussen +3 en +15 Volt. De printerconnector van de P2000 werkt volgens deze RS232 standaard en kan dus direkt met het modem verbonden worden. Het 2e I/O slot voldoet niet aan deze standaard.

Samenvattend:

=====

Het versturen van een letter gebeurt in de vorm van een getal waarvan de waarde bepaald wordt door de ASCII kode. De 7 bits van dit getal worden 1 voor 1 na elkaar volgens de afgeproken baudrate naar de printerkonnektor gestuurd. De printerkonnektor is volgens de RS232 standaard elektrisch verbonden met een modem. Dit modem zet de nullen en enen uit de printerkonnektor om in tonen met een bepaalde frequentie en verstuurd deze tonen over de telefoonlijn naar de andere computer.

Daar staat hij dan, de P2000. Een echte computer. Volop belangstelling van vrienden en bekenden. Laat eens zien wat ie kan. Dus verschijnt er een spelletje op het scherm, nog een ander, nog een, ..., en dan? Kun je er ook iets praktisch mee doen?

Op dat moment blijkt het praktische gebruik van de P2000 erg beperkt te zijn. De oorzaak hiervan ligt niet bij de computer zelf, maar vooral in het ontbreken van geschikte programma's (software). De in de folders aangeprezen mogelijkheden, zoals tekstverwerking en administratie, zijn er eenvoudigweg niet of werken in Basic, waardoor ze (te) traag zijn, te veel geheugen innemen, etc. Programmapakketten die dit wel mogelijk maken zijn meestal alleen verkrijgbaar op duurdere modellen computers (met discdrive), en tegen niet geringe prijzen. Kortom, wil je meer met de P2000 doen dan is er een grote drempel.

Een voorbeeld van een uitgebreider gebruik van de P2000 is tekstverwerking. Eenvoudige tekstverwerkers kunnen het beste omschreven worden als typemachines met een beeldscherm, waarbij de gebruiker de tekst op het beeldscherm samenstelt. Dit heeft grote voordelen boven het werken op papier: foute letters, woorden of zinnen zijn heel simpel over te typen of uit te wissen. De tekstverwerker is niet regel-georiënteerd, zoals bij de Basic-editor, maar "full-screen": de gebruiker kan over het hele scherm intypen. Ook is het mogelijk stukken tekst te verplaatsen of nieuwe zinnen tussen te voegen zonder dat de hele tekst opnieuw moet worden getypt. Geavanceerdere mogelijkheden zijn het zoeken van trefwoorden in de tekst of het uitlijnen van de rechterkant van de tekst.

Wat kan zo'n tekstverwerker nu allemaal als resultaat opleveren?

- Het meest voor de hand liggende voorbeeld is een brief. Bij het opmaken van een brief zijn alle mogelijkheden van het boven beschreven 'luxe typen' te gebruiken: gemakkelijk intikken, fouten herstellen (zonder alles weer helemaal over te moeten typen zoals op papier) en daarbij een nette, zelf bepaalde pagina-opzet (layout) maken.
- Ook langere werkstukken als scripties of verslagen behoren tot de mogelijkheden. Het is dan prettig dat elke pagina automatisch

genummerd wordt, eventueel voorzien van een standaard titel (kop) zoals bijvoorbeeld in de P2000 nieuwsbrief. Het hele werkstuk kan natuurlijk (op cassette) bewaard worden om er later verder aan te werken.

- Met een tekstverwerker kan de gebruiker ook een 'database' maken, bijvoorbeeld een lijst van namen, adressen en telefoonnummers. De gegevens voor zo'n adressenbestand worden gewoon ingetikt, maar er kunnen ook nieuwe namen tussengevoegd worden, iets wat op papier niet mogelijk is. De opgeslagen gegevens hoeven zich natuurlijk niet te beperken tot adressen, te denken valt ook aan een lijst van grammofoonplaten, examenvragen van de laatste 5 jaar: alles kan er per soort in gezet worden. Uit zo'n lijst kan dan ook een selectie gemaakt worden en met een druk op de knop afgedrukt worden.
- Een volgende toepassing vormen standaard-brieven (bewaard op cassette), waar alleen een paar kleine toevoegingen of veranderingen nodig zijn. Denk bijvoorbeeld aan een sollicitatiebrief.
- Helemaal gemakkelijk is het als in een standaard-brief de plaats waar de naam van de geadresseerde moet komen met een teken is aangegeven, waarna de computer bij het afdrukken van de brief de namen een voor een uit het adressenbestand haalt en invult. Een simpel voorbeeld hiervan is het maken van etiketten.

Voor de P2000T was er tot nog toe geen programma-pakket dat dit alles (tegen een schappelijke prijs) mogelijk maakt. Hiervoor nu is TEXT2000 ontwikkeld, een module die alle bovengenoemde toepassingen realiseert voor minder dan 180 gulden.

De module is een ROMpack dat ipv. Basic in het eerste slot wordt gestoken en dan de besturing van de computer overneemt. De volle geheugenruimte (16, 32 of 40k) is beschikbaar. Een zeer compact overzicht van de mogelijkheden van TEXT2000 is op de volgende pagina te vinden. Schrik niet van de uitgebreidheid ervan: in de praktijk blijkt TEXT2000 heel gemakkelijk te gebruiken. Zelfs met een geringe kennis van de commando's is al goed te werken, en er is geen enkele kennis van een programmeertaal vereist. Iedereen die kan typen, kan er mee werken! Anderzijds zijn er voor de fijnproevers allerlei geavanceerde mogelijkheden ingebouwd.

<PREON>

Lou Somers
Sumatraplein 52
6524 KH Nijmegen
tel. 080-227049

Paul Timmers
Staringstraat 10
6521 AJ Nijmegen
tel. 080-227998

FUNCTIE OVERZICHT

- * Filemanagement.
 - Meerdere bestanden kunnen tegelijk behandeld worden.
 - Bestanden hebben een naam van maximaal 11 letters.
 - Nieuwe bestanden kunnen gecreeerd worden.
 - Bestanden kunnen op cassette "gesaved" en weer in het geheugen geladen worden.
 - De inhoudsopgave van een cassette kan opgevraagd worden.
 - De cassette kan gewist worden.
- * Schermindeling.
 - 23 regels voor input, met instelbare breedte (maximaal 80).
 - 1 commando/status regel
- * Volledige schermcontrole.
 - Cursor een positie omhoog, omlaag, naar rechts, naar links.
 - Cursor geheel naar boven, onder, rechts, links.
 - "Terug-wagen" toets (carriage return).
 - Instelbare tabulatorstops. Woord-tabulator.
 - Wis letter, wis woord, wis tot einde regel.
 - Voeg letters tussen (insert).
- * Functietoetsen voor regelcommando's, deze werken op de regel waar de cursor staat.
 - Wis regel. Voeg lege regel tussen. Dupliceer regel.
- * Functietoetsen voor blokcommando's, met deze commando's kunnen willekeurige stukken tekst verplaatst, gecopieerd of gewist worden.
 - Markeer begin van een blok.
 - Markeer eind van blok en copieer naar buffer.
 - Idem en wis hét blok.
 - Markeer eind en wis blok zonder naar buffer te copieren.
- * Functietoetsen voor schermbeweging ("scrollen").
 - Een regel/pagina verder/terug.
 - Ga naar begin/eind van bestand.
- * Functietoetsen voor "knip- en plakwerk" (formatting).
 - Splits regel vanaf cursorpositie.
 - Plak volgende regel achter cursorpositie.
 - Format het hele bestand: schuif woorden binnen een alinea aan, zodanig dat de regels gevuld zijn.
- * Functietoetsen voor commandoregel.
 - Wis commandoregel.
 - Haal laatst uitgevoerde commando terug.
- * Commando's voor scrollen.
 - Opgegeven aantal regels verder/terug.
 - Geef nummer van huidige regel.
 - Ga naar opgegeven regelnummer.
- * Commando's voor vinden en veranderen van tekst.
- * Commando's voor "settings".
 - Tabulatorstops.
 - Breedte van scherm.
 - Printer baudrate.
 - Printer type.
- * Commando's voor printen.
 - Print het hele bestand; uitgebreide mogelijkheden voor linker en rechter kantlijn, centreren, instellen regel- en paginalengte, titeldefinitie, etc. Voeg items uit een database in.

INVENTAR is een programma waarmee men een beperkt bestand kan opbouwen. Het bestand kan maximaal 350 regels van 120 tekens bevatten. De gegevens worden in tabelvorm op papier afgedrukt. Daarom is het noodzakelijk om over een p2123-printer te beschikken.

Het bestand is opgebouwd uit een zestal kolommen die men zelf kan benoemen. Deze kolommen kunnen vervolgens gevuld worden met informatie. Men heeft dan de mogelijkheid deze informatie op verschillende manieren te sorteren en af te drukken op papier.

CODE 1 het benoemen van de kolommen.

Op het scherm staan een zestal regels waarop men de omschrijving van een kolom kan zetten. Men beschikt over verschillende EDIT mogelijkheden. Dit zijn de pijltjes naast de spatiebalk, de wistoets en de entertoets. Als men een regel gevuld heeft kan men met de entertoets de nog aanwezige puntjes op de regel wissen. Als alle regels gevuld zijn, gaat men met CODE naar de opdrachtregel. De vraag 'delete old data Y/N' dient men met 'N' te beantwoorden. Als hier met 'Y' geantwoord wordt wordt ALLE informatie inclusief de zojuist ingevoerde gewist.

CODE 2 het vullen van de kolommen.

De met CODE 1 ingevoerde gegevens verschijnen bovenaan het scherm. Daaronder bevinden zich de zes invoerregels. Deze worden op dezelfde manier gevuld als bij CODE 1. Is een totale invoer klaar dan dient men deze te beëindigen met CODE #. De gegevens worden nu weggeschreven op diskette en er verschijnen opnieuw zes lege invoerregels. Om de invoer te beëindigen toetst men nu CODE 0 in.

CODE 3 het wijzigen van reeds ingevoerde gegevens.

De met CODE 1 ingevoerde gegevens verschijnen bovenaan het scherm. Met de pijltjes naast de spatiebalk springt men naar die regel waarvan men de inhoud weet en waarvan men een van de zes reeds ingevoerde regels wil wijzigen. Druk nu op de entertoets. De gegevens worden nu opgezocht en verschijnen op het scherm. Men kan nu wijzigingen aanbrengen en verder zoeken met CODE #. Hoeft er echter verder niets gewijzigd te worden ga dan weer terug met CODE 0.

CODE 4 het wissen van gegevens.

Op dezelfde manier als bij CODE 3 kunnen ook gegevens opgezocht worden die men wil wissen. Toets als bevestiging van de wis opdracht bij de vraag 'remove Y/N' de 'Y' toets als de gegevens inderdaad gewist dienen te worden en anders de 'N' toets. Dit zoeken en wissen gaat net zolang door totdat CODE 0 ingetoets wordt.

CODE 5 sorteren van het gehele bestand.

Opnieuw verschijnen de omschrijvingen van de kolommen in beeld. Men kan nu een keuze maken waarop het bestand gesorteerd dient te worden. Als men dus wil sorteren op de combinatie van kolom 3 en 4 dan voert men deze getallen in, wil men sorteren op alleen kolom 5 toets dan 5 en 0 in. Het bestand wordt nu van de diskette gehaald en gesorteerd. De tijdsduur van het sorteren bedraagt maximaal ca. 5 minuten als het gehele bestand (350 regels) gevuld is. Na het sorteren van het bestand verschijnt de menu pagina op het scherm. Zie verder bij CODE 7.

CODE 6 sorteren van een geselecteerd bestand.

Met de pijltjes naast de spatiebalk springt men naar die regel waarvan men een geselecteerd bestand wil hebben. Voer nu die tekst in waarop gezocht dient te worden. Dus bv. zoek op regel 5 naar het woord "EINDHOVEN" doet men als volgt. Spring naar regel 5 met de pijltjes, toets EINDHOVEN in, en druk vervolgens op de entertoets. Nu kan men als bij CODE 5 een keuze van sorteren doen. Het bestand wordt vervolgens geselecteerd van diskette gehaald en gesorteerd. Na het sorteren verschijnt de menu pagina op het scherm. Zie verder bij CODE 7.

CODE 7 het printen van het bestand.

Zorg ervoor dat het papier in de printer A4 formaat heeft en schakel de printer in. Men heeft verschillende mogelijkheden om het bestand te printen, vanuit CODE 5 of CODE 6 via de menu pagina of direkt vanaf de menu pagina na een andere CODE. Indien het bestand gesorteerd is wordt het gesorteerde bestand afgedrukt en anders het bestand zoals dat door u is ingevoerd. Na het intoetsen van CODE 7 kunt u een keuze maken uit het aantal en de volgorde van de kolommen die men afgedrukt wil hebben. Voor niet gebruikte kolommen voert men een 0 in. De gegevens worden vervolgens afgedrukt. Na afdrucken is het eventueel gesorteerde bestand niet meer aanwezig in het geheugen van de computer.

CODE 9 beëindigen van het programma.

Na intoetsen van CODE 9 is het programma beëindigd.

P2000

Opnieuw: onzichtbare listings

Hoe BASIC in het geheugen van de P2000 staat is uitgelegd door Louis Naus in Nieuwsbrief 4 op blz. 2.34. In dit verhaal refereer ik aan dit artikel.

Indien U het laatste gedeelte van Uw programma onzichtbaar wilt maken en in dit gedeelte geen referenties pleegt aan regelnummers (GOSUB nnnnn of GOTO nnnnn) dan hoeft U alleen maar via een EDITOR, zoals het programma MONITOR van het NAT. LAB. (binnenkort op cassette A 212 te verkrijgen), het eerste te verbergen regelnummer te verhogen tot 65535 dit door in het voorbeeld van Louis adres 25929 en 25930 beide een waarde van 255 (&HFF) te geven.

RUN : Werkt gewoon
LIST : Werkt niet

Veel plezier met deze gekke fout in de BASIC !

In eerste instantie zal ik even uitleggen wat RESTORE precies doet.

Het is een opdracht die te maken heeft met het inlezen van DATA uit BASIC-regels een voorbeeld:

```

20 DATA Aap , Noot
30 DATA Mies
40 DATA XX

60 READ A$
70 IF A$ = "XX" THEN PRINT "Einde demo" : END
   ELSE PRINT A$ " " ; : GOTO 60

```

Als U dit "programma" uitvoert ziet U het volgende:

```

-----
Aap Noot Mies
-----

```

Indien U nu regel 50 toevoegt:

```

50 RESTORE 30

```

wordt er alleen "Mies" afgedrukt.

Dit betekent dat RESTORE nnnnn inhoudt dat de eerst volgende READ opdracht moet geschieden vanaf nnnnn.

Nu is het in BASIC niet mogelijk om nnnnn te vervangen door een variabele waardoor een relatie kan worden gelegd tussen een Getalswaarde en in te lezen data.

De routine hieronder maakt het mogelijk (op "vieze" wijze) om RESTORE variabele uit te voeren:

```

-----
1000 RESTORE 1070
1010 IF RN < 0 OR RN > 65529 THEN RETURN
1020 RN$ = RIGHT$( STR$( RN ), LEN( STR$( RN ) ) - 1 )
1030 RA = PEEK( &H640B ) + PEEK( &H640C ) * 256 + 6
1040 RN$ = SPACE$( 5 - LEN( RN$ ) ) + RN$
1050 FOR I = RA TO RA + 4
1060 POKE I, MID$( RN$, I - RA + 1, 1 ) : NEXT I
1070 RESTORE.....
1080 RETURN
-----

```

Bij de aanroep dient in RN het gewenste regelnummer meegegeven te worden. De punten in REGEL 1070 dienen spaties te zijn.

Deze routine is ook gemakkelijk om MACHINETAAL verborgen in BASIC-regels te declareren omdat met een RESTORE en twee PEEK's het startadres te vinden is !

Voor diegenen, die hun P2000 met de CP/M aanpassing hebben uitgerust was al een onder CP/M draaiende Forth verkrijgbaar.

Nadeel van CP/M is dat het een op diskettes gebaseerd Operating Systeem is, waarvoor een 48K systeem met een speciale hardware aanpassing vereist is.

P2000 Forth werkt zowel met cassettes als diskettes en draait al op een 16K systeem. De op een 16K systeem beschikbare gebruikersruimte van 5800 Byte (exclusief buffers, inclusief Stack) legt geen al te grote beperkingen op, omdat Forth zeer compact compileert, in veel minder ruimte dan voor Basic nodig is.

P2000 Forth wordt geleverd op een initialisatiecassette die men gebruikt om een eigen systeem-cassette aan te maken. Bij initialisatie moet een keus gemaakt worden tussen een Forth systeem dat alleen met cassettes werkt en een systeem dat met cassettes en diskettes werkt. De systeem-cassette kan echter op elk gewenst moment opnieuw geconfigureerd worden.

Ook wordt bij initialisatie vastgesteld met welk toetsenbord gewerkt wordt (Nederlands/Engels, Duits, Frans, Italiaans, Zweeds of Spaans). Op de systeem-cassette worden dan de bij die uitvoering behorende tabellen vastgelegd.

Voor de disk uitvoering wordt dan vastgesteld, hoeveel drives er aanwezig zijn en wat de capaciteit van elke drive is (1 of 2 koppen en aantal tracks -- P2000 Forth voor disk is voorbereid voor 4 drives en 256 tracks per drive -- aanpassen aan hard disks tot 32 Megabyte is mogelijk).

Bij het opstarten van het systeem met de systeem-tape worden de bij het model en de printer behorende tabellen geladen. Toepassingsprogramma's kunnen controleren of zij op een cassette-systeem of op een disk-systeem draaien.

Tussen Forth en het Philips Operating Systeem is een aanvullend Operating Systeem gebouwd dat in elk geval bevat:

- toetsenbordroutines
- schermfunktieroutines
- een snel tape operating systeem
- afhandelingsroutines voor funktietoetsen
- een eenvoudige monitor hetzij voor cassette hetzij voor disk.

Het standaard Forth-model heeft foutboodschappen op een vaste plaats op de schijf staan. Om het wachten op spoelen van de cassettes te vermijden zijn voor een cassette-systeem deze boodschappen in de RAM opgenomen.

De Cassette Monitor maakt het mogelijk, cassettes te formatten, te kijken welk cassette type in de drive zit, en om een applicatie op tape te zetten of in gecompileerde vorm van tape te laden en te starten.

Ook het Disk Operating Systeem is op snelheid gebouwd. Daardoor kan het CP/M Forth schijven die op de P2000 zijn aangemaakt niet lezen. De Disk Monitor is daarom uitgerust met een conversieroutine, die CP/M schijven naar P2000 Forth omzet en omgekeerd.

Met de Disk Monitor kunnen ook schijven en cassettes voor Forth worden geformat, en kan het type van een cassette worden gecontroleerd. Verder kunnen Runfiles op disk aangemaakt en van Disk geladen worden.

De volgende funktietoetsen zijn beschikbaar:

- langzame output naar het beeldscherm
- output naar het beeldscherm stoppen en weer starten
- de cursor aan- en afzetten
- omschakelen van alleen hoofdletters naar hoofdletters en kleine letters en weer terug
- beeldscherm output parallel naar de printer
- de cassette terugspoelen
- disk drive aanzetten en weer uitzetten

Applicatieprogramma's kunnen deze funktietoetsen desgewenst onbereikbaar maken voor de gebruiker. Zij kunnen ook van de niet geïmplementeerde funktietoetsen gebruik maken.

Op de P200T kunnen in Forth kleuren en graphics gegenereerd worden.

De meegeleverde Installation Guide bevat:

- aanwijzingen omtrent het aanmaken van de systeem tape
- inlichtingen over de werking van de tussengebouwde routines zoals Terminal, Tape Operating Systeem, Disk Operating Systeem en de beide Monitoren
- inlichtingen omtrent voor de gebruiker belangrijke geheugenplaatsen en welke daarvan veranderd mogen worden
- een naar soort functie ingedeeld overzicht van de geïmplementeerde Forth woorden
- korte handleidingen voor het gebruik van de meegeleverde Editor en Assembler routines

Behalve de standaard Forth Editor worden, met commentaar en toelichting in de Source, een Screen Editor en twee Dump Routines (een naar het scherm en een naar een Source File) meegeleverd.

De Installation Guide is geen boek om Forth te leren. Daarvoor kan vooral aangeraden worden "Starting Forth" van Leo Brodie (ca. f. 60,-), eventueel ook "Forth Ok" van Lemaire en Meijer (ca. f. 30,-, Nederlands) samen met de aanvullingen en wijzigingen in "Het Vijgeblad", het orgaan van de HCC Forth Interesse Groep.

(Als kleine demonstratie van de mogelijkheden van Forth) Ok

(Gaan we een Dump routine bouwen:) Ok

Ok

(Met de Printer toets heb ik de Printer bijgeschakeld,) Ok

(zodat een verslag van deze sessie meteen afgedrukt wordt.) Ok

(De haakjes maken dat het Forth systeem dit als commentaar) Ok

(beschouwt en niet probeert te compileren.) Ok

Ok

(Om de inhoud van een geheugenplaats op het scherm of papier) Ok

(te zetten tikken we meestal in:) HEX 6014 @ . 0 Ok

(maar onder de 10H krijgen we dan geen nul, zoals je hier-) Ok

(boven ziet. We moeten dus dieper Forth in.) Ok

```

( We bouwen onze routine als volgt op: ) Ok
: BYTE          ( de naam van de routine )
  HEX  0        ( zet de machine in hexadecimaal, en maak van )
              ( het getal op de stack een 32 bits waarde )
  <#          ( zet alles klaar voor numerieke output )
  BL HOLD     ( Achteraan komt een BLank, een spatie )
  # #        ( Dan wil ik twee hexadecimale cijfers hebben)
  #>         ( zet het adres en de lengte van de numerieke )
              ( output buffer op de stack)
  TYPE       ( Stuurt de output naar het scherm )
;           ( Sluit de definitie van het Forth "woord" af) Ok
( Kijk: nu we klaar zijn met definieren komt "Ok" terug! ) Ok
Ok
( Even testen:) Ok
CR DECIMAL 0 BYTE DECIMAL 16 BYTE DECIMAL -1 BYTE
OO 10 FF Ok
( Dat werkt dus: nu kunnen we eenvoudig op dezelfde manier ) Ok
( de output van een adres met 4 Bytes definieren: ) Ok
: ADDRESS HEX- 0 <# BL HOLD # # # # #> TYPE ; Ok
0 ADDRESS -1 ADDRESS 0000 FFFF Ok
DECIMAL ( en nu dat getal waar Basic altijd zo lastig mee doet:) Ok
32768 ADDRESS 8000 Ok
Ok
( Ook willen we bij een Dump graag zien of we met tekst te maken ) Ok

( hebben. Daarvoor gaan we de IF ... ELSE ... THEN structuur Ok
( gebruiken: ) Ok
Ok
: ASCII
  DUP BL -    ( Dupliceer het byte op de stack en trek er )
              ( de ASCII waarde van de spatie van af )
  OK IF      ( Als het resultaat kleiner dan nul is )
    DROP 2E  ( gooiën we het byte van de stack en zetten we )
              ( daarvoor een punt neer )
    ELSE     ( Anders moeten we kijken of byte>'z' is:)
    7A OVER  ( Dat geeft op de stack: byte 'z' byte )
    - OK    ( kijk nu of 'z'-byte kleiner is dan nul; zo ja )
    IF DROP ( dan doen we hetzelfde als hierboven)
    2E
    THEN    ( sluit de binnenste IF loop af)
  THEN     ( sluit de IF ... ELSE loop af)
  EMIT ;   ( EMIT stuurt het teken weg ) Ok
Ok
CR HEX 30 ASCII SPACE 40 ASCII SPACE 50 ASCII SPACE
O @ P Ok
CR HEX IF ASCII SPACE 20 ASCII SPACE 7A ASCII SPACE 7B ASCII SPACE
. z . Ok
( Dat was wat ik wou hebben ) Ok
Ok
( Om nu een aantal bytes op het scherm te krijgen kunnen we van ) Ok
( een lus gebruik maken; als we daarvoor een apart woord ge- ) Ok
( bruiken kunnen we daarvan nog plezier hebben ) Ok
: BYTES
  ( op de stack staat het beginadres onder het )
  ( aantal )
  O DO      ( met dat aantal zetten we een loop op )
    DUP I + ( We dupliceren het adres en tellen er de loop- )
              ( index van 0 tot aantal-1 bij op )
    C@ BYTE ( We zetten de inhoud van de geheugenplaats op )
              ( de stapel en roepen daarmee BYTE aan)
  LOOP     ( Verhoogt de index met 1 en springt terug naar )
              ( het gedeelte dat bij DO begint zolang index en )
              ( aantal niet aan elkaar gelijk zijn )
  DROP ;   ( Anders gooiën we het adres van de stack) Ok

```

```
( Meteen weer testen: ) CR HEX 0 20 BYTES
F3 C3 5A 02 00 C3 4F 04 C3 00 0D C3 67 04 E4 01 C3 13 10 C3 7A 04 E4
01 C3 F1 04 C3 67 01 90 0E 0k
```

Ok

```
( Op dezelfde manier maken we een "letterwoord": ) Ok
: CHARS 0 DD DUP I + C@ ASCII LOOP DROP ; Ok
CR HEX 223 29 CHARS
```

```
P H I L I P S . R . . . M I C R O C O M P U T E R . S . . . P 2 0 0 0 0 k
```

```
( Dit kunnen we nu allemaal gaan gebruiken in een woord DUMP dat ) Ok
```

```
( op de stack een beginadres onder het eindadres verwacht: ) Ok
```

```
: DUMP >R ( Het eindadres bewaren we op de Return Stack)
  B / B * ( Hierdoor eindigt het beginadres altijd op een )
          ( nul of een acht )
```

```
CR 16 EMIT ( Nieuwe regel en cursor uit)
```

```
BEGIN
```

```
DUP ADDRESS ( duidelijk? )
```

```
DUP 8 BYTES
```

```
DUP 8 CHARS
```

```
CR ( nieuwe regel )
```

```
B + ( verhoog adres met 8 )
```

```
R OVER ( haal eindadres op en kopieer beginadres daar )
        ( OVER heen )
```

```
- 0< ( verschil kleiner dan nul? )
```

```
?TERMINAL ( of de stopstoets ingedrukt? )
```

```
OR
```

```
UNTIL ( zo niet, terug naar begin, anders eruit )
```

```
CR 15 EMIT ( nieuwe regel, cursor weer aan )
```

```
DROP ( Beginadres van de stack )
```

```
R> DROP ; ( Eindadres van de Return stack ) Ok
```

Ok

```
HEX 200 230 DUMP
```

```
0200 EE CD AB 01 C9 06 80 10 .....
```

```
0208 FE C9 7E 06 00 4F 23 ED .....0#.
```

```
0210 80 C9 43 41 4C 4C 20 53 ..CALL S
```

```
0218 45 52 56 49 43 45 EC 51 ERVICE.0
```

```
0220 0F 06 0D 50 20 48 20 49 ...P H I
```

```
0228 20 4C 20 49 20 50 20 53 L I P S
```

```
0230 DC 52 0F 06 0D 4D 49 43 .R...MIC
```

Ok

```
( Voor een versie die ook op het M-model werkt kunnen ) Ok
```

```
( we de regellengte aan de hand van de vlag in 6013H ) Ok
```

```
( berekenen en in een variable LLEN opslaan ) Ok
```

```
( Dat is in Scr # 9 van de volgende listing gedaan: ) Ok
```

Ok

```
8 LIST 9 LIST
```

```
SCR # 8
```

```
0 ( Dump Routine -1-
```

JV 82apr12)

```
1
```

```
2 FORTH DEFINITIONS HEX
```

```
3
```

```
4 : BYTE ( n --- ; formatted hexadecimal output)
```

```
5   HEX 0 <# BL HOLD ( trailing) # # #> TYPE ;
```

```
6 : ADDRESS ( n --- )
```

```
7   HEX 0 <# BL HOLD # # # # #> TYPE ;
```

```
8 : ASCII ( n --- )
```

```
9   DUP BL - 0< IF DROP 2E ( a dot )
```

```
10   ELSE 7A OVER -
```

```
11   0< IF DROP 2E THEN
```

```
12   THEN EMIT ;
```

```
13 : BYTES ( addr n ---- )
```

```
14   0 DD DUP I + C@ BYTE LOOP DROP ; ( hex output)
```

```
15 -->
```

SCR # 9

```
0 ( Dump Routine -2-                               JV 82apr12 )
1 : CHARS ( addr n ---- )
2   0 DO DUP I + C@ ASCII LOOP DROP ;           ( ascii output)
3 6013 C@ 1 AND 1+ 8 * VARIABLE LLEN
4
5 : DUMP ( from to ---- )
6   >R LLEN @ / LLEN @ * CR 16 EMIT ( Adjust start address *)
7   BEGIN DUP ADDRESS ( ---- addr addr )
8     DUP LLEN @ BYTES ( ---- addr addr )
9     DUP LLEN @ CHARS ( ---- addr addr )
10    LLEN @ 8 = IF CR THEN ( New line if count is only 8 )
11    LLEN @ + R OVER - ( Check for last address *)
12    OK ?TERMINAL OR ( End reached or STOP pressed? *)
13    UNTIL ( If so, leave the structure )
14    CR 15 EMIT DROP R> DROP ; ( and clear the stacks *)
15 ;S
Ok
```

(De gecompileerde woorden maken nu deel uit van het Forth) Ok
(systeem tot we ze met FORGET weer verwijderen:) Ok

```
FORGET BYTE Ok
HEX 230 20 CHARS CHARS? Not found
Ok
: AFSCHIED CR CR CR CR
  ." Dit was Forth. Goedenavond !"
  CR CR CR CR ; Ok
AFSCHIED
```

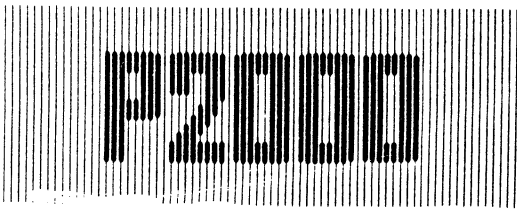
Dit was Forth. Goedenavond !

Ok

Aanvullende inlichtingen kunt U verkrijgen op de gg-dag in augustus en schriftelijk bij:

Jan Vermeulen
Riouwstraat 55
1521 SC Wormerveer

Jan Vermeulen
en
Frans v.d. Markt



NIEUWSBRIEF

JUNI 1983

Serial Interface (V24).

Onder nummer P2174 brengt Philips een V24 seriele interface voor de P2000 op de markt. Het is een insteekdoos voor in sleuf 2 met een kabel eraan van 2,50 meter lengte met aan de andere kant een 25 polige Cannon steker. In de achterzijde van de module bevinden zich een aantal miniatuur wipschakelaars. De V24 interface wordt begeleid door een duidelijk, zij het Engelstalig, handboek van 45 pagina's.

UART module.

In de insteek-module zit een UART IC van het type 8251 met alle benodigde omringende electronica. Een UART is een schakeling die geheel zelfstandig de door de P2000 parallel aangeboden bytes in bits ontleedt en deze in de tijd achter elkaar plaatst. Eveneens kan hij een bit-treintje ontvangen en als compleet byte weer aan de P2000 aanbieden.

De ons bekende printer-connector geeft ook een serieel signaal, maar hier worden de bits achter elkaar gezet door een stukje machinetaal. Het gebruik van een UART heeft als voordeel dat tijdens het werken van dit circuit de P2000 iets anders kan doen. Het verzenden en ontvangen van een byte is voor de P2000 nu niet meer dan een simpele OUT of INP-instructie en de UART doet de rest.

In de module kan de bitsnelheid met schakelaartjes worden ingesteld op diverse waarden, als laagste 75 Baud en dan steeds 2 maal zo hoog tot 9600 Baud. De enigszins verouderde maar toch nog wel in gebruik zijnde snelheid van 110 Baud kan niet worden ingesteld en is ook niet door een eenvoudige wijziging te verkrijgen. Wel is het door een simpele ingreep mogelijk hoger te gaan dan 9600 Baud in stappen van een factor 2. Het is standaard alleen mogelijk met dezelfde snelheid te zenden als waarin ontvangen wordt, maar door het solderen van een draadje is het wel mogelijk de module aan te passen voor bijvoorbeeld de Videotext standaard van 1200/75 Baud.

De print in de module is al voorbereid om met synchrone seriele transmissie te werken, waarbij een klokpuls wordt meegestuurd. Dit in tegenstelling tot de asynchrone systemen waarbij met start en stop impulsen gewerkt wordt.

Verder in de module.

In de module bevinden zich nog een achttal schakelaartjes die door de P2000 zijn uit te lezen. Dit is voor een eenvoudige toepassing nauwelijks zinvol, maar wanneer een aantal van deze module's afwisselend in gebruik zijn kan men in deze schakelaartjes een code zetten, zodat het programma hieraan kan zien of de juiste UART-module op dat moment in de machine zit. Het lijkt erop dat deze schakelaartjes ooit bestemd zijn geweest om bij interruptbedrijf een deel van het adres van de interruptroutine mee te geven. Zoals de module nu geleverd wordt is hij niet geschikt om op deze manier te gebruiken.

Op de schuine bovenkant van de module steekt een rode LED naar buiten. Deze gaat branden als het aangesloten apparaat (via de draad DSR) aangeeft dat het gereed is om gegevens te ontvangen.

Documentatie.

De bij de module geleverde documentatie is niet bestemd voor de pure beginner. Maar wie al iets van machinetaal afweet en enig idee heeft van de werking van een UART vindt hierin alles wat hij weten wil.

De volgende onderwerpen worden behandeld: instellen van de seinsnelheid; aansluiten van de modem; initialisatie van de UART; gebruik vanuit BASIC; machinetaal hulproutine; direct gebruik in machinetaal; gebruik vanuit PASCAL. In de appendices treft men ook het complete principeschema aan van de insteekmodule.

V24-connector aansluitingen

Dit zijn de aansluitingen van de 25 polige connector. Achter de naam van de verbinding staat het V24-nummer. Daar achter staat een 0 als het een uitgang van de module is, of een I als het een ingang is.

1. Veiligheids aarde
2. Transmit Data (103) 0
3. Receive Data (104) I
4. Request To Send (105) 0
5. Clear To Send (106) I
6. Data Set Ready (107) I
7. Signaal aarde
8. Carrier Detect (109) I
9. Select Standby (116) 0
20. Data Terminal Ready (108) 0
22. Calling Indicator (= de bel) (125) I
23. Data Singalling Selector (111) 0
24. Transmitter Clock (113) 0

Printer ready.

Zoals wellicht bekend is kan men op bit 1 van inputpoort &H20 zien of de printer ready is. Hiermee kan het programma zien of het zinvol is iets naar de printer te sturen of niet. Nu wees Robbert Backer erop dat dit verkeerd gaat wanneer de matrixprinter zojuist is uitgezet. Het lijkt dan nog geruime tijd alsof de printer aan staat en de P2000 stuurt vrolijk bitjes naar de printer die allang uit staat. Nadere beschouwing leert ons dat dit een ontwerpfout van de matrixprinter is. Als de netschakelaar hiervan wordt uitgezet doven de groene lampjes op de printer heel langzaam en het blijkt dat het ready-sigitaal nog een hele tijd voldoende spanning heeft om de P2000 te laten denken dat de printer nog aanstaat. Na een minuut is de spanning voldoende gedaald, het is dus geen blijvend verschijnsel. Blijkbaar moet men het bij Epson doen zonder ledentest.

Redactie.

P2174 I/O-adressen.

De V24 module gebruikt de volgende adressen voor input en output:

&H40 UART data input en output register
&H41 UART statusregister en controlregister
&H61 bit 0: niet gebruikt
bit 1: soldeerdruppel
bit 2: Call indicator
bit 3: Carrier Detect
&H62 de 8 schakelaartjes

De adressen zijn niet geheel uitgedecodeerd. Dit houdt in dat alle even adressen tussen &H40 en &H4F hetzelfde doen als &H40 en alle oneven adressen hetzelfde als &H41. Evenzo zijn alle oneven adressen tussen &H60 en &H6F equivalent aan &H61 en komt &H62 overeen met &H66, &H6A en &H6E. Op de adressen &H63, &H67, &H6B en &H6F vindt men een mengseltje van &H61 en &H62. Niet gebruiken dus. .e

Programma terugroepen.

Het invoeren van CLOAD"A" terwijl CSAVE"A" bedoeld was leidt tot het verlies van het programma, dat in het geheugen stond. Indien U de fout opmerkt voordat de cassetterecorder de gelegenheid heeft gekregen het geheugen met een ander programma te vullen, is het nog mogelijk het oude BASIC-programma terug te krijgen.

Allereerst moet de eindpointer van het programma verzet worden. Deze dient een geheugenplaats aan te wijzen, waarvan U zeker weet dat deze achter het oude BASIC-programma ligt. Kies bij een 16K machine &H9F00, bij 32K &HDF00 en bij 48K &HFF00. Met POKE &H6405,&H00 : POKE &H6406,&H9F wordt de eindpointer verzet naar &H9F00.

De volgende handelingen moeten hierna worden verricht:

1. plaats een kladbandje in de recorder
2. geef in CSAVE"A"
3. type het onderstaande programma in en RUN
4. laad het programma A van het kladbandje
5. geef in ?USR(0)

LIST laat U zien dat het programma er weer helemaal is.

Als de machinetaal-routine al in het geheugen stond voordat U abusievelijk CLOAD invoerde, kunt U het oude programma direct terugkrijgen via de aanroep van punt 5. In het in ontwikkeling zijnde gereedschapskist programma zal deze routine standaard aanwezig zijn.

Waarschuwing! Sommige programma's hebben een stuk machinetaal gekoppeld aan BASIC. Dit stuk machinetaal verliest U met deze routine. Hij is wel in staat het einde van de BASIC te vinden, maar verder niet.

Het programma.

```
1 CLEAR 20,(2*PEEK(&H605C)+(PEEK(&H605C)
=3))*&H2000+&H5FBF
2 A=PEEK(&H605C) :IF A=1 THEN A=&H9FC0
ELSE IF A=2 THEN A=&HDC00 ELSE A=&HFFC0
3 FOR I=0 TO54:READ A$:POKE A+I,VAL("&H"
+A$):NEXT
4 DEFUSR=A
5 PRINT"Laad programma en type ?USR(0)"
6 DATA 2A,5C,62,01,03,00,09,23,7E,FE
7 DATA 00,20,FA,23,DD,2A,5C,62,DD,75
8 DATA 00,DD,74,01,2B,2B,23,7E,FE,00
9 DATA 20,FA,23,7E,FE,00,20,F4,23,7E
10 DATA FE,00,20,EE,23,22,05,64,22,07
11 DATA 64,22,09,64,C9
12 DELETE 1-12
```

John Compter.



Deze bijdrage beschrijft enkele manieren, om machinetaal routines rechtstreeks in een programmaregel op te nemen, waarbij de eindwijzer rustig op zijn plaats mag blijven staan. De routine moet dan wel aan bepaalde eisen voldoen. Een programmaregel waar machinetaal in moet komen wordt eerst in BASIC voorbereid. Op de plaats van de machinetaal zetten we b.v. allemaal uitroeptekens, evenveel als er straks bytes nodig zijn. Met de MONITOR zien we dan een hele reeks 21'en staan en daarheen MOVEn we de routine. Er zijn de volgende mogelijkheden:

1. In een DATA regel. Syntax:

```
300 DATA"L","!!!!!!!!.....!!!!!"
```

Aan te roepen als volgt:

```
860 RESTORE 300:READ L$:F=VARPTR(L$):DEFUSR=F+4:F=USR(0)
```

De L fungeert als lokstring om de VARPTR op de goede plaats te krijgen.

2. In een string. Syntax:

```
300 M$="!!!!!!!!.....!!!!!"
```

Aanroep:

```
860 F=VARPTR(M$):DEFUSR=F:F=USR(0)
```

3. In een REM regel. Syntax:

```
300 REM!!!!!!!!!!!!.....!!!!!!
```

Aanroep:

```
860 RESTORE 300:F=PEEK(&H640B)+256*PEEK(&H640C):DEFUSR=F+6:F=USR(0)
```

In al deze gevallen mogen de regels niet ge-edit worden, omdat de EDITOR alles door elkaar gooit. Bij het listen kunnen vreemde dingen gebeuren en een printer kan zich ook merkwaardig gedragen, maar er gaat niets kapot. Achter een REM staat de machinetaal waarschijnlijk nog het beste beschermd.

De machinetaal routine moet verder aan de volgende eisen voldoen:

1. Zij moet compleet relocatable zijn.
2. Er mag geen 00 in voorkomen.
3. De totale lengte van de regel mag niet groter zijn dan 255 tekens.

De laatste beperking is op te heffen door meerdere regels te reserveren en de positiewijzer in de eerste regel te verzetten.

In Nieuwsbrief 170 e.v. is een methode gegeven waarmee door de gebruiker ingevoerde waarden in een programma opgeslagen kunnen worden. Die methode leent zich niet zonder meer voor strings. We presenteren nu een methode om strings, die door een gebruiker zijn ingevoerd binnen het programma te bewaren.

Ook hierbij maken we weer gebruik van de procedures van de P2000 zelf. De VARPTR van een ingelezen DATA string wijst n.l. naar die DATA regel zelf en wel naar het item dat aan de beurt is om gelezen te worden.

In het onderstaande voorbeeld zijn 4 strings van elk max. 8 bytes gereserveerd, die door een gebruiker onder controle van het programma naar willekeur gewijzigd kunnen worden. Ze worden dan meteen naar de DATA regel terug gepoke'd. Het aardige is, dat de bewuste gegevens op die manier slechts één keer in de machine voorkomen. Het kan echter alleen met strings.

Let op! De op deze wijze opgeslagen strings zijn READ-ONLY. Zodra ermee gemanipuleerd gaat worden zet de P2000 ze alsnog in de stringruimte.

```
10 DATA "Aap      ", "Noot      ", "Mies      ", "Wim      "
20 PRINTCHR$(12)
30 FOR I=1 TO 4:READ A$:PRINT I, A$:PRINT: NEXT
40 INPUT "Welk nr veranderen "; N:PRINT
50 RESTORE 10:FOR I=1 TO N:READ A$:NEXT
60 LINEINPUT "Geef nieuwe tekst  : "; B$
70 IF LEN(B$)>8 THEN B$=LEFT$(B$,8)
80 IF B$="" THEN B$=A$:PRINT B$
90 B$=B$+STRING$(8-LEN(B$),32)
100 FOR I=0 TO 7
110 POKE PEEK(VARPTR(A$)+1)+256*PEEK(VARPTR(A$)+2)+I,
ASC(MID$(B$, I+1, 1)):NEXT
130 PRINTCHR$(12):PRINT:PRINT "LIST":PRINT:LIST-130
```

BOEKBESPREKING

Chr. de Boer en P.Potjegort
Computersystemen

Philips Data Systems Nederland BV

200 pag.

F. 24,50

Een vlot geschreven werkje, dat beoogt leken bekend te maken met de meest gangbare begrippen op het gebied van computertoepassingen. De nadruk ligt op computers voor administratief gebruik. De auteurs schenken ook aandacht aan aspecten als het inpassen van de computer in een organisatie, de verandering van persoonlijke werkomstandigheden en het privacy-vraagstuk. Uitermate geschikt voor toekomstige computergebruikers. Binnen de P2C2 zal echter eerder behoefte bestaan aan meer gedetailleerde en meer diepgaande informatie.

H.H.Schutte

BASIC programma's voor wetenschap en techniek

Kluwer software reeks

110 pag.

F. 29,50

De opzet van dit boekje, om leken te helpen aan een serie technisch/wetenschappelijke programma's is maar zeer ten dele geslaagd. Het voornaamste punt van kritiek is de veel te summiere, meestal onbegrijpelijke en soms zelf onjuiste toelichting bij de programma's. De programma's zelf zijn rommelig, slordig vertaald en als ze al werken bijzonder gebruikers-onvriendelijk. Misschien voor haastige programmeurs, die geen tijd willen besteden aan het oplossen van wiskundige problemen maar snel een routine willen hebben.

Boek besprekingen.

Praktische inleiding tot de microprocessor
K de Wever en H Loobuyck
Uitg. De Sikkel
380 pag. f 43,50

Een Vlaamse uitgave van 1979. Het geeft een vrij complete opsomming en uitgebreide beschrijvingen van al wat langer bestaande microprocessors. Ook de Z80 komt aan bod. Instructieset, assembleertaal en getalrepresentatie worden besproken, hogere programmeertalen worden alleen genoemd.

Het is typisch kost voor mensen, die wat meer over hardware van de micro willen lezen, ook al is dit beperkt tot typen, die al geruime tijd op de markt zijn.
Jan Gieles.

Begin cursus Z80 assembleertaal
K de Wever
Uitg. De Sikkel/Sciento
24 pag. f 10,95

Overdruk van een serie artikelen uit Percosoft, waarin architectuur en instructieset van de Z80 op een zeer heldere manier worden uitgelegd. In dit beperkte formaat kon dat natuurlijk niet volledig gebeuren en er ontbreken ook uitgewerkte toepassingsvoorbeelden van b.v. de verschillende adresseermethoden. Ondanks deze beperkingen toch aanbevolen vooral voor beginners, die hier een uitstekende introductie aan zullen hebben.

Jan Gieles.

Basic-Probeerboek
Chr E de Boer
uitg. Educaboek (ISBN 90 11 920538)
300 pag. f 26,50

Dit is het boek dat we allemaal allang wilden hebben: De handleiding voor gebruikers van de P2000T. Het ^{is} precies wat de beginnende gebruiker nodig heeft n.l. een gids, die hem op duidelijke en begrijpelijke manier over de eerste moeilijkheden heen helpt. Het boek is zo geschreven, dat de gebruiker alles meteen kan uitproberen op zijn P2000. Vrijwel alle BASIC functies komen aan de orde, compleet met foutmeldingen, voorbeelden en opgaven. Achterin bevindt zich nog een register op trefwoorden en een overzicht van het videogeheugen.

Een voortreffelijke en verzorgde uitgave, een must voor elke P2000 bezitter, al was het maar voor zijn familieleden, die het ook eens willen proberen.
Jan Gieles.

Cursus Z80 Assembleertaal
Roger Hutty
Uitg: Academic Service, Den Haag
186 pag. f 35,-

Voor iedereen, die met Rodney Zaks gewerkt heeft, moet het een verademing zijn dit boek ter hand te nemen om al oefeningen uitwerkend te merken hoe men zich relatief snel vertrouwd kan maken met de methoden om 157 Z80 instructies zodanig in programma's te verwerken, dat deze doen, wat de maker ervan verwacht. Het boek is prettig geschreven en geeft wel veel minder details dan Zaks maar dat kan ook niet anders in dit korte bestek. Toch is het vrij volledig en uitstekend geschikt voor zelfstudie. Doordat in het begin al een schermroutine wordt gegeven (die helaas niet klopt op de P2000 maar daar wel voor kan worden aangepast) geven alle opgaven, mits goed uitgewerkt bij het invoeren in de computer een visuele beloning. Bij gebruik van een assembler moet er rekening mee worden gehouden dat commando's als de diverse DEF's worden omgezet. Als men eenmaal doorheeft hoe dat alles precies moet is met Hutty in de hand, de P2000 op tafel met de assembler erin erg plezierig te werken.

Wim van den Eijnde.

Programmeren in LISP
E W Dijkstra
Uitg: Academic service, Den Haag
190 pag. f 39,50

Hoewel voor de P2000 bezitter nog niet zo actueel, is dit boek over LISP, de tot gestructureerd programmeren uitnodigende standaardtaal voor kunstmatige intelligentie een beschouwing waard. Het boek behandelt de stof op een bijzonder instructieve manier, door telkens aan de hand van een stuk tekst een groot aantal vragen te stellen waarna de antwoorden, die men kan afdekken, direct volgen. Als men het boek heeft doorgewerkt, dan moet men in staat zijn LISP-programma's te schrijven en programma's van anderen te lezen.

Tot slot een uitspraak van de strijdbare schrijver: "Bij wijze van grap heeft iemand LISP ooit de intelligentste manier genoemd om een computer te misbruiken. Ik vind deze beschrijving een groot compliment, omdat het goed weergeeft dat LISP een echte bevrijding is. Het heeft enkele zeer talentvolle mensen geholpen om gedachten te hebben die daarvoor onmogelijk waren."

Als deze schrijver zoiets van een computertaal zegt, dan moet het ook wel iets bijzonders zijn.

Wim van den Eijnde.