DISCIPLE

3e Jaargang nr. 14

Met o.a.

SECTOR ERROR REPARATIEKIT

OPHUS>DISCIPLE CONVERTER

TURBO TRACKS < >

84 TRACKS FORMAT

AUTOLINE KILLER

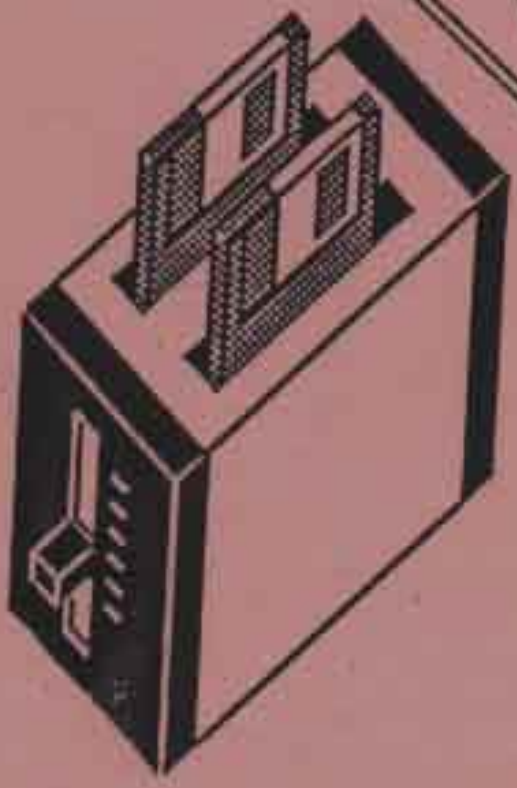
GEHEIMCODE SAVER

MAPPER

Enz.

Enz.

Enz.



NIEUWSBRIEF

DATA - SKIP

HET ADRES VOOR AL ÜW WENSEN OP GEBIED VAN:

SPECTRUM
DISCIPLE EN PLUS-D
ATARI
PC

WAAR SERVICE NOG EEN BEGRIP IS.

BEL ONS VOOR PRIJZEN;
01820 - 20581

OF KOM EENS LANGS;
OOSTHAVEN 58
2801 PE GOUDA

COLOFON

Redactie : Disciple Nieuwsbrief,
Peter Faas,
[redacted]

Techniek, Vormgeving en Lay-out: Carl Faas, [redacted] Gerard Nesselaar [redacted]

Copy : Tasword 2 files of programma's op 5 1/4 - 3.5 inch alle formaten. Of op tape met hardkopie.
Bewaar een kopie ingeval de PTT verstek laat gaan!
Stuur een gefrankeerde antwoord-envelope mee als u de schijf of cassette terug wilt hebben, en vergeet niet overal uw naam en adres op te vermelden.

Svp GEEN artikelen uit andere bladen overschrijven.

: HOE IS DE DISCIPLE NIEUWSBRIEF TE VERKRIJGEN??
- Op de SPECTRUM dagen in het HCC gebouw te HOUTEN
- Bij DATASKIP te Gouda.
- Door overschrijving per Bank of Giro, Giro [redacted] t.n.v. P.Faas, Red. Nwsbr. te [redacted]

>>>>>>> - VERMELD DUIDELIJK GEWENST NUMMER EN DISKFORMAAT DUS 3.5" of 5.25" en DOUBLE of SINGLE SIDED.....!!!!!!

Prijs : Afgehaald; (Houten of Dataskip)

-ZONDER	disk	Fl 5.-
-MET	disc	Fl 7.50

Per post (via Bank/Giro en incl.porto)

-ZONDER	disk	Fl 7.45
-MET	disk	Fl 9.95

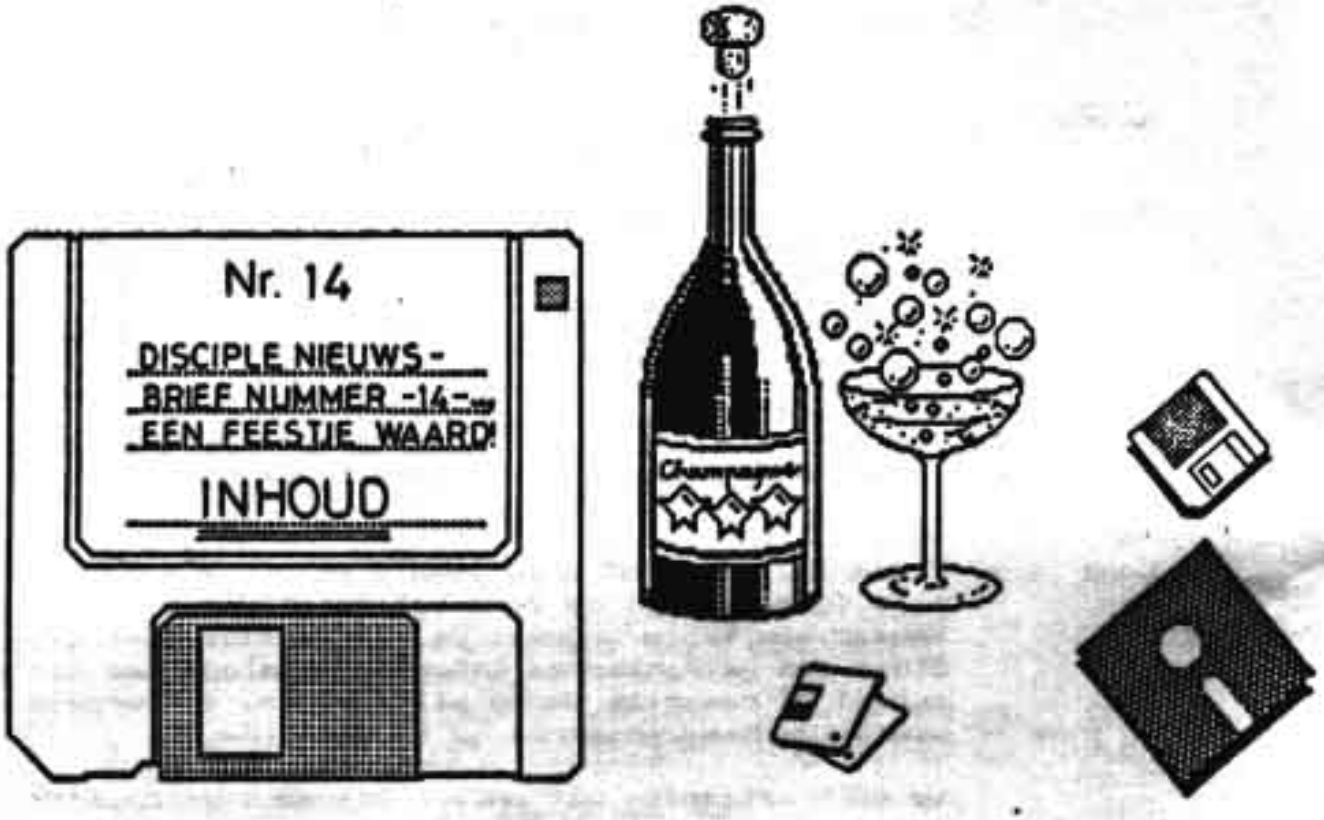
>>>>>>> : Toeslag voor 3.5" disk Fl 3.50
(Zie ook prijzenfestival in NB-7)

>>>>>>> : *ALGEMEEN INFO Nr. : [redacted], 19.00-21.00 uur*

Copyright: De redactie aanvaardt geen aansprakelijkheid voor de overtreding van COPYRIGHT door inzenders.
Inzenders worden geacht zich te houden aan de geldende regels met betrekking tot het COPYRIGHT van anderen.

Noot : Alle publicaties in de Disciple Nieuwsbrief zijn vrij van copyright.

Adverteren: Niet commerciele advertenties gratis binnen de beschikbare ruimte.



INHOUD

DISK INHOUD

```
* DISCIPLE .. DISC 1 DIRECTORY *

 1 LINE    .BS   6 BAS     10
 2 SAVER   .BS   1 BAS
 3 OPUSId  .BS   7 BAS   9000
 4 OPUSId1.MC   4 CDE 55430,1634
 5 OPUSId2.MC   1 CDE 62000,164
 6 OPUSCAT.BS   6 BAS   9000
 7 OPUSCAT.MC   1 CDE 60000,341
 8 FASTRAC.BS   1 BAS   9999
 9 FASTRAC.MC   1 CDE 60000,115
10 MAPPER .BS   5 BAS   1000
11 MAPPER .MC   1 CDE 61000,173
12 FRMAT84.BS   2 BAS
13 TOOLKIT.BS   1 BAS     30
14 TOOLKIT.MC  34 CDE 25000,17100
15 PINCODE.BS   2 BAS    200
```

HACK AUTOSTARTREGEL UIT (BEVEILIGDE) BASIC!

Bij het kraken van bepaalde BASIC programma's loop je tegen een beveiliging op die het onmogelijk maakt het programma te MERGEn. Omdat ook tegen BREAK beveiligd is kun je er dan moeilijk in komen. Ook als je de BASIC met "T>D2" naar DISCiPLE overzet blijft het probleem bestaan (bij "Tape>Disk" niet). Daarom vond ik het handig een programma te hebben dat de AUTOSTART-LINE uit BASIC files kan verwijderen.

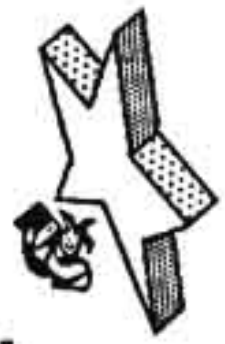
In Nieuwsbrief 7 staat beschreven hoe de directory in elkaar zit. Op adres 218 en 219 staat bij BASIC files het autostart nummer, is er geen LINE opgegeven dan staat er 65535. Dus dat is makkelijk, dacht ik, gewoon op deze adressen 255 POKEn en klaar!

Toen ik dit echter bij een BASIC file probeerde bleek het programma echter nog steeds vanzelf te starten. De oorzaak hiervan staat even verderop in hetzelfde artikel. Aan het begin van de file staat namelijk een copie van 9 bytes uit de directory. Hierbij is ook het autostart regelnummer, deze waren door mijn POKEs echter niet veranderd. Bij het laden van de file worden blijkbaar de waardes aan het begin van de file gebruikt en niet die uit de directory.

In het programma "LINE" worden uiteraard beide versies van het autostart regelnummer gewijzigd. Behalve het verwijderen van LINE kun je hiermee ook een niet vanzelf startend programma wel een LINE geven of je kunt LINE veranderen.

Loop maar eens even mee de listing door;

```
  10 GO TO 9500
 100 REM           Boodschap
 110 CLS : PRINT INVERSE 1;"  Autorun veranderaar voor de
  DISCiPLE en PLUS-D.",
 120 PRINT '''   Door Tonnie Stap  (R)1989."
 190 RETURN
 200 REM    Verander drive nummer
 210 LET drivenr=3-drivenr
 220 PRINT AT 14,13;drivenr
 230 IF INKEY$="1" THEN GO TO 230
 240 RETURN
 300 REM     Catalogus van disk
 310 CAT drivenr
 320 IF INKEY$<>"" THEN GO TO 320
 330 PRINT #1; INVERSE 1;"      Druk op een toets.",
 340 IF INKEY$="" THEN GO TO 340
 350 IF INKEY$<>"" THEN GO TO 350
 360 RETURN
1000 REM   Invoer programmanummer
1020 GO SUB 100
1030 INPUT "Programmanummer: ";prognr
1040 IF prognr<1 OR prognr<>INT prognr OR prognr>80 THEN RETURN
1050 LET dirtrack=INT ((prognr-1)/20)
1060 LET dirsector=INT ((prognr-20*dirtrack-1)/2)+1
1070 LET diradres=60000+256*(prognr/2=INT (prognr/2))
1080 LOAD *drivenr,dirtrack,dirsector,60000
1090 FOR x=1 TO 10
1100 LET n$(x)=CHR$ PEEK (diradres+x)
1110 NEXT x
```



```
1120 LET oudLINE=PEEK (diradres+218)+256*PEEK (diradres+219)
1130 IF PEEK diradres=1 THEN RETURN
1140 PRINT ''"***";n$;"*** is geen BASIC!"
1150 BEEP 1,0
1160 LET prognr=0
1170 LET n$="Geen file!"
1180 LET oudLINE=65535
1190 PAUSE 100
1200 RETURN
```

Hier vinden dus de veranderingen plaats.
Zowel de directory wordt gewijzigd als de header VOOR het file.

```
2000 REM    Verander LINE nummer
2005 IF prognr=0 THEN GO SUB 1000: IF prognr=0 THEN RETURN
2010 GO SUB 100
2020 PRINT ''"***";n$;"*** LINE: ";oudLINE
2030 INPUT "Nieuw LINE nummer (65535 voor   geen LINE): ";nieuwL
INE
2040 IF nieuwLINE<0 OR nieuwLINE<>INT nieuwLINE OR nieuwLINE>655
35 THEN LET nieuwLINE=65535: RETURN
```

PEEK het track en sectornummer van het eerste fileblok uit
de directory, zodat ook de header gewijzigd kan worden.

```
2041 LET eerstetr=PEEK (diradres+13)
2042 LET eerstesec=PEEK (diradres+14)
2043 LOAD $drivenr,eerstetr,eerstesec,60512
2050 LET laag=nieuwLINE-256*INT (nieuwLINE/256)
2055 LET hoog=INT (nieuwLINE/256)
2060 POKE diradres+218,laag
2065 POKE 60519,laag
2070 POKE diradres+219,hoog
2075 POKE 60520,hoog
```

Beide veranderingen moeten natuurlijk bewaard worden.

```
2080 SAVE $drivenr,eerstetr,eerstesec,60512
2090 SAVE $drivenr,dirtrack,dirsector,60000
2100 RETURN

8000 REM         Hoofdprogramma
8010 GO SUB 100
8020 PRINT ''''"1) Verander drivenummer."
8030 PRINT "2) Catalogus van disk."
8040 PRINT "3) Voer programmanummer in."
8050 PRINT "4) Verander LINE van basic."
8060 PRINT ''''"Drivenummer: ";drivenr''"Programma: ";prognr;":""
";n$;"""''"Oude LINE: ";oudLINE;TAB 19;"Nieuwe: ";nieuwLINE
8070 LET i$=INKEY$
8080 IF i$<"1" OR i$>"4" THEN GO TO 8070
8090 IF i$="1" THEN GO SUB 200: GO TO 8070
8100 IF i$="2" THEN GO SUB 300: GO TO 8000
8110 IF i$="3" THEN GO SUB 1000: GO TO 8000
8120 IF i$="4" THEN GO SUB 2000: GO TO 8000
8130 STOP
9000 REM         Save programma
9010 CLEAR
```

```
9020 GO SUB 100
9030 INPUT "Save op drive: ";drivenr
9040 IF drivenr<>1 AND drivenr<>2 THEN GO TO 9030
9050 SAVE ddrivenr"LINE" LINE 10
9060 VERIFY ddrivenr"LINE"
9070 STOP
9500 REM          Initialisatie
9505 CLEAR 59999
9510 LET drivenr=1
9520 LET prognr=0
9530 LET n$="Geen file!"
9540 LET oudLINE=65535
9550 LET nieuwLINE=65535
9560 GO TO 8000
```

Tonnie Stap.

Overigens een mooi voorbeeld van gemoduleerd programmeren in basic.

Erg overzichtelijk waardoor u de werking snel kunt doorvorsen en aldus eventueel onderdelen kunt gebruiken om uw eigen programmeer ideeen te verwezelijken.

(Red.)

$-$—$-$-$-$-$-$-$-$

DISCIPLE/PLUS-D VERSUS OPHUS-DISCOVERY

Dankzij Tonnie Stap kunt u nu Opus Discovery schijfjes van IEDER formaat inlezen met de Disciple en files op Discipleformaat kopieren. Bijna alle software, behalve die specifiek van de Opus DOS gebruik maken, wordt zo voor ons toegankelijk gemaakt. Komt u nu iemand tegen met Opus schijfjes dan is er geen belemmering meer om ze te bekijken en over te nemen. Ik heb het geprobeerd en het werkt perfect!!!

P.Faas

Hier volgt Tonnies eigen verhaal:

Bij mij in de buurt wonen enkele mensen die een OPUS diskinterface hebben, met hen wissel ik wel eens wat software uit. Dit moest echter via tape omdat de DISCiPLE geen OPUS files in kan lezen en omgekeerd ook niet. Met LOAD $ zijn echter wel de sectoren van een OPUS schijf te lezen.

Daarom leek het mij mogelijk een copieer programma te schrijven dat files leest van een OPUS schijf en die wegschrijft naar een DISCiPLE schijf. Als extra handigheid heb ik bovendien nog een programma geschreven dat de catalogus van de OPUS schijf laat zien.

Een OPUS schijf is (normaliter) geformateerd met 18 sectoren per spoor (256 bytes per sector), de sectoren zijn genummerd van 0 t/m 17. Een sector op een OPUS disk wordt een 'blok' genoemd, elk blok wordt aangegeven door een nummer, deze telling begint met -1 (track 0, spoor 0) en kan tot b.v. 2878 lopen (DS, 80 sporen).

Het blok -1 bevat onder andere de volgende informatie:

Byte 2: Aantal sporen op een kant.
3: Aantal sectoren per spoor.
4: In bits 0 en 1 een codering voor de sectorgrootte, en in bit 6 een 1 voor dubbelzijdig en een 0 voor enkel.

De codering voor sectorgrootte is: 00 = 128 bytes
01 = 256 bytes (standaard)
10 = 512 bytes
11 = 1024 bytes

In blok 0 begint de catalogus, per file zijn er 16 bytes gereserveerd, en wel als volgt:

Byte 0- 1 : Aantal bytes in laatste blok van file.
2- 3 : Nummer van eerste blok.
4- 5 : Nummer van laatste blok.
6-15 : De naam van de file.

In de catalogus komen geen 'gaten' voor, als er een file gewist wordt worden de gegevens van de files na de gewiste 16 bytes terug geschoven. Voor de eerste en na de laatste file in de catalogus staat nog iets:

Byte 0- 1 : Sectorlengte- 1
2- 3 : Nummer van eerste catalogus blok (voor ie info) of aantal vrije blokken op de schijf.
4- 5 : Nummer van laatste catalogus blok (voor ie info) of einde van catalogus indicator: 65535.
6-15 : Naam van de disk.

Standaard beslaat de catalogus 7 blokken maar dit kan veranderd worden evenals de sectorgrootte, aantal tracks (afh. van drive soes tot 86) en het aantal sectoren per spoor. Dit is dus een stuk flexibeler dan bij de DISCiPLE maar nu is het juist die flexibiliteit die voor problemen zorgt. We moeten namelijk met alle mogelijke gevallen rekening houden, hiervoor zou het dus handig zijn om de informatie in blok -1 te kunnen lezen, echter met LOAD * is deze sector (0,0) niet te laden.

Daarom heb ik een gedeelte van de 'load sector' routine uit de DISCiPLE ROM gecopieerd, en aangepast om het gewenste blok te laden (met dank aan Rudy Biesma). Deze routine 'OPUSld2.MC' zal ik hier verder niet uitleggen, het aansturen van de disk controller zal ik (als er belangstelling voor is) misschien in een toekomstig artikel bespreken. (Zeer graag, red.)

Nu zitten we nog met het 40/80 spoor probleem, de standaard OPUS diskdrive had 40 sporen en was enkelzijdig. Velen zullen inmiddels een drive met hogere capaciteit hebben dus zijn er vele formaten in omloop. Het lezen van een 80 spoors schijf in een 40 spoors drive is misschien mogelijk maar waarschijnlijk slechts de even sporen (niet zo nuttig dus). Maar omgekeerd moet het wel mogelijk zijn.

Als de kop van de drive naar een ander spoor moet wordt er een stap puls naar de drive gezonden. Op een 40 spoors disk liggen de sporen 2 maal zo ver van elkaar dan bij 80.

Je hoeft dus 'alleen maar' een extra stap puls naar je 80 spoors drive te sturen om genoemde schijven te kunnen lezen. Bij het laden van sectoren van de OPUS schijf houd ik daarom steeds bij op welk spoor er het laatst geladen is. Moet er daarna op een ander spoor geladen worden en is er sprake van het 40/80 probleem dan worden er step pulsen gegeven tot het gewenste track bereikt is.

Nu moeten we er nog achter komen of de schijf die we van een OPUS gebruiker krijgen van een 40 of 80 spoors drive komt. Dat lijkt makkelijk, in byte 2 van blok -1 staat namelijk het aantal sporen. Echter dit getal kan door de OPUS gebruiker veranderd zijn, op een 40 spoors drive gaan soms wel 43 sporen, een gebruiker zou kunnen beslissen van de 80 sporen maar 40 te gebruiken. Daarom pas ik de volgende truck toe: Eerst wordt de kop naar spoor 0 verplaatst met de 'RESTORE' commando code (deze werkt voor alle drive/schijf combinaties). Hierna krijgt de drive 2 stap pulsen en wordt een markering op de disk gelezen, byte 0 van deze markering is nu het spoor nummer.

Uit dit nummer zijn de volgende conclusies te trekken: Is het 1 dan hebben we een 40 schijf in 80 spoors drive, is het 2 dan zijn formaat van schijf en drive gelijk en is het 4 dan zit er een 80 schijf in een 40 spoors drive (ga maar na.) Elk ander getal duidt op echte problemen.

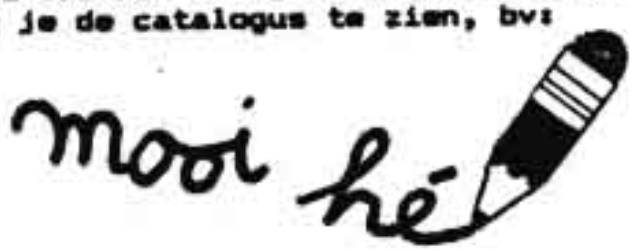
Omdat beide programma's in elkaars verlengde liggen zal ik nu ingaan op het programma 'OPUSCAT.BS'.

Dit programma voert een PEEK § uit (in MC) om te bepalen welke drive(s) je kunt gebruiken, is er meer dan 1 dan mag je kiezen. Hierna wordt alle informatie over de schijf opgehaald zoals hierboven beschreven en worden deze gegevens op het scherm afgedrukt. Door op SPACE te drukken kun je een nieuwe disk bekijken en na ENTER krijg je de catalogus te zien, bv:

```
    speldisk 1  0,7

1   arkan load  7,1
2   arkanoid    8,82

Free: 157.25 K
```



Dit betekent dat de disk 'speldisk 1' heet en de catalogus begint bij blok 0 en 7 blokken lang is. De eerste file heet 'arkan load' en beslaat alleen blok 7, de tweede file beslaat 82 blokken vanaf nr. 8. Tenslotte is er nog 157.25 kilobyte vrij.

Voordat ik nu inga op het OPUS copieer programma moet ik nog iets vertellen over de structuur van de files op een OPUS disk. Zoals uit de catalogus wel bleek moet er in de file nog wel iets staan over het type van de file, het laad adres etc. Het eerste blok van de file begint dan ook met de volgende 7 bytes:

0 : File type (1-4).
1-2: Lengte van file.
3-4: LINE van BASIC, start van CODE of variabele van ARRAY.
5-6: Eventueel BASIC lengte zonder variablen.

(Dit zijn ook de bytes die in de tape header staan.)

In tegenstelling tot de DISCiPLE gebruikt de OPUS geen bitmap, een file zal dus altijd opeenvolgende blokken gebruiken. Dit is bij het laden wel handig want alleen het bloknummer hoeft worden opgehoogd voor de volgende sector en een omreken routine houdt dan wel rekening met het aantal sectoren per spoor en sporen per kant.

Nu wat betreft het programma OPUS loader, net als bij OPUS CAT wordt eerst gekeken hoeveel drives er zijn en eventueel gekozen (voor bron en bestemming). Hierna wordt de informatie van de OPUS schijf gelezen en afgedrukt, met SPACE is er nu nog een mogelijkheid tot schijf verwisselen, met ENTER start het copieeren.

Een voor een worden de filenamen uit de directory opgehaald en afgedrukt. Er wordt dan gevraagd of de file gecopieerd moet worden en zoniet dan wordt de volgende opgehaald. Wil je de file wel copieren dan wordt hij geladen. De OPUS kent echter ook nog een soort OPENTYPE files maar deze kan het programma niet aan, dit geeft de foutmelding 'Wrong file TYPE'. Gaat het laden goed en past de file in het geheugen ('Out of memory.') dan wordt eventueel gevraagd om de schijf te verwisselen en wordt de file weggeschreven op de DISCiPLE schijf. Na (eventueel) weer een schijfwissel wordt de volgende filenaam afgedrukt etc.

Zijn er niet meer files op de schijf dan vraagt het programma of er nog een schijf te copieren is, dit moet wel een schijf van precies hetzelfde formaat zijn als de eerste!

In het geval van een DISCiPLE foutmelding wordt deze afgedrukt en kun je kiezen uit dezelfde mogelijkheden als bij mijn D>D copier, dit spreekt wel voor zich.

Dan hier nog een mededeling voor mensen die een hardware- matige stapverdubbelaar op hun drive hebben geinstalleerd: Bij het copieeren van een 40 spoors schijf mág de stapverdubbeling zowel aan als uit staan als je de stand maar niet verandert!

Voorlopig is er alleen maar een DISCiPLE versie van dit programma, binnenkort krijgt een kennis van mij echter een PLUS-D zodat ik na een paar aanpassingen en tests ook een PLUS-D versie zal kunnen publiceren.

Tonnie Stap

84 TRACK FORMAT & DIRECTORY SAVER

Van Jasper van de Bent, een onzer jongste inzenders ontvingen wij deze aardige bijdrage.

Jasper heeft ontdekt dat je de meeste moderne 3.5" drives kunt formatteren op dubbelzijdig 84 tracks. Deze 8 extra tracks zijn voor het normale systeem niet toegankelijk, maar het LOAD§ en SAVE§ commando heeft er geen moeite mee.

Nu had hij al eens een programma ingestuurd waarmee de directory van een schijf die "klaar" was kon worden gesaved naar de laatste 4 tracks op een schijf. Treedt er nu een sectorerror op in de directory dan haal je de gesavede versie weer op en alle programma's blijven toegankelijk.

Probleem was wel eens bij schijven vol Snapshots dat er geen 4 lege tracks over bleven op een schijf.

En jawel, u raad het al. Er is nuttig gebruik te maken van de 4 extra tracks door daar die directory naar toe te saven.

Hier volgt de listing: (IV=inverse video TV=true video in regels 10 en 50)

```
 10 CLS : PRINT "iv 84 tr FORMAT & directory SAVER tv"
 20 PRINT AT 5,5;"1. FORMAT  ( 84tr. DSDD )"
 30 PRINT AT 7,5;"2. SAVE directory"
 40 PRINT AT 9,5;"3. LOAD directory"
 50 PRINT AT 14,0;"iv JASPER vd BENT  Research Ltd tv"
100 PAUSE 0
110 IF INKEY$="1" THEN GO TO 2000
120 IF INKEY$="2" THEN GO TO 500
130 IF INKEY$="3" THEN GO TO 1000
140 GO TO 100
```

Save de directory naar track 80 ta 83

```
500 LET adr=40000
510 FOR a=0 TO 3
530 FOR b=1 TO 10
540 LOAD §1,a,b,adr
550 LET adr=adr+512
560 NEXT b
570 NEXT a
580 LET adr=40000
590 FOR a=80 TO 83
600 FOR b=1 TO 10
610 SAVE §1,a,b,adr
620 LET adr=adr+512
630 NEXT b
640 NEXT a
650 RUN
```

Laad directory vanuit tracks 80 ta 83 terug naar tracks 0 ta 3.

```
1000 LET adr=40000
1020 FOR a=80 TO 83
1030 FOR b=1 TO 10
1040 LOAD §1,a,b,adr
```

```
1050 LET adr=adr+512
1060 NEXT b
1070 NEXT a
1080 LET adr=40000
1090 FOR a=0 TO 3
1100 FOR b=1 TO 10
1110 SAVE #1,a,b,adr
1120 LET adr=adr+512
1130 NEXT b
1140 NEXT a
1150 RUN
```

De 84 tracks FORMAT routine.

```
2000 POKE #1,212
2010 FORMAT d1
2020 POKE #1,208
2030 RUN
9999 SAVE d$"FRMAT84.BS"
```

Waarschuwing: Schrijf geen oude directories terug indien u tussentijds wijzigingen hebt aangebracht. Een schijf moet echt AF zijn, voordat u de directory bewaard!

Vergissingen door de driectories van verschillende schijven met elkaar te verwisselen wordt door dit programma toch aardig voorkomen doordat de directory niet de schijf verlaat voor opslag op bv. een tweede "directory verzamelschijf".

Jasper vd Bent.

MASTERPUBLISHER - DE KLERK

Naar aanleiding van het in Nieuwsbrief 13 besproken desktop publishing programma "DE KLERK/MASTER PUBLISHER" van WILLEM LABEMAN (zie advertentie op het achterblad) hebben wij een uitbreiding van de gebruiksaanewijzingen mogen ontvangen.

Stap voor stap wordt de gebruiker hierin meegenomen naar een volledige paginaopmaak.

Hieraan wordt dit werkelijk schitterende programma nu razendsnel toegankelijk voor echt iedereen!!

Gezien de kwaliteit en de sympatieke prijsstelling en nu dus ook om de zeer komplete handleiding een echte aanrader voor iedereen die met de Spectrum Disciple/plus-D kombinatie op het gebied van DESKTOP PUBLISHING aktief wil zijn (of worden?)

Gezien de bloedarmoede die begint op te treden in de aanmaak en aanvoer van dit soort goede gebruiksprogramma's een pluim voor de auteur. En bedenk dat ook in dit geval het gezegde heel goed opgaat dat een programma dat het waard is om te gebruiken, het ook waard is om te kopen! (Red.)

PINCODE ENZ, GOED BEWAARD EN TOCH GEHEIM

Enige tijd geleden heb ik van Jack van der Marel een alleraardigst programma ontvangen waarin je al je geheime nummers, pincodes, Zwitserse bankrekeningnummer e.d. kunt opbergen zonder het risico dat iemand er achter kan komen of de code kan kraken!

Jack is niet de schrijver van het programma, maar ik ga er vanuit dat zowel hij als de maker er weinig bezwaar tegen zullen hebben dat ik u laat meedelen in dit toch handig hulpmiddel in de strijd tegen de vergeetachtigheid.

Want u weet als je je pincode vergeet is de hele pas ongeldig en kan je wachten op een nieuwe, terwijl ergens opschrijven ook niet ontbloot is van risico's.

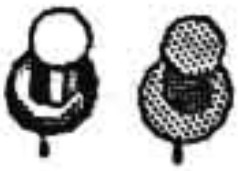
U moet de BASIC naar eigen behoefte uitbreiden en aanpassen. Is dat gebeurd, dan heeft u er geen omkijken meer naar.

Even iets over de werking

1. Haal het REM statement uit regel 300
2. Voeg regel 312 STOP toe
3. Bedenkt nu een geheime code in de vorm van een, voor u makkelijk te onthouden woord, bv de naam van uw minnares.
4. RUN het programma en geef uw geheime code in.

Op de printer wordt nu een lijst getallen afgedrukt, die UNIEK is en alleen bij uw codewoord past!
De getallen zijn oplopend genummerd van 1 te 200.
Hou dat lijstje bij de hand, verwijder regel 312 en voeg de REM weer toe in regel 300.

```
200 BORDER 5: PAPER 5: CLS : DIM k(200): LET b=0: POKE 23658,0
205 PRINT "  PINCODES  en andere nummers": INPUT "GEHEIME CODE
: ";w$: CLS : GO TO 230
210 PRINT "  PINCODES  en andere nummers"
220 INPUT "TYP GEHEIME CODE IN ->";w$: IF w$="" THEN LET w$="2"
230 FOR x=1 TO LEN w$: LET b=b+CODE w$(x): NEXT x
240 RANDOMIZE b
250 PRINT '''''"          Even geduld!"
280 FOR j=1 TO 200
290 LET k(j)=INT (RND*10)
300 REM LPRINT "   ";k(j);"    ";
310 NEXT j
```



Vanaf hier mag u zelf regels toevoegen die betrekking hebben op al uw geheim te houden getallen die u nooit mag kwijtraken, cq vergeten. U hebt de ruimte tot regel 1000.

De truuk is dat u met een PRINT statement aangeeft welk nummer het betreft en daarna de plaats van dat getal in de de string van 200 cijfers. Ieder getal komt meerdere malen voor in de string, steeds op een andere plaats, zodat u de vrije keuze hebt om naar eigen believen te kiezen.

Stel uw PINCODE= 2243

Uw pakt uw lijstje wat u op de printer gemaakt hebt en kiest ergens een 2.

U telt op welke plaats die staat, bv 24. Achter PRINT "PINCODE : "; zet u nu k(24);

U zoekt nog een 2 op, bij voorkeur op een andere plaats in de string om te voorkomen dat iemand ziet dat het twee dezelfde cijfers zijn, bv een 2 op plaats 133. Achter de eerste k(24); komt nu k(133); te staan en dit proces herhaalt zich voor ieder cijfer.

Zie de voorbeeld regels 320 en 330 die ik heb ingevoerd.

```
 315 CLS
 320 PRINT AT 3,0;"POSTBANK PINCODE P1 : ";k(44);k(75);k(79);k(i
57)
 330 PRINT "POSTBANK PINCODE P2 : ";K(4);K(196);K(142);K(33)
 340 REM volgende pincode
 350 REM  nog een pincode
```

Het kan zijn dat u echt de pincodes vergeten bent, maar hoe weet u dan dat het programma geen foutje heeft gemaakt en u met de een of andere rare code opzadelt?

Zelf los ik dat op door een referentiecijfer, ofwel een getal dat u absoluut niet kunt vergeten.

Dat kan een heel makkelijk cijfer zijn, bv uw eigen telefoonnummer of geboortedatum. Zolang u maar nog NIET gebruikte cijfers uit de reeks gebruikt heeft niemand er wat aan als dit bekend zou zijn, want de reeks is RANDOM met het geheime woord.

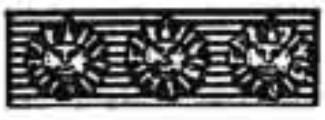
RUNT u later het programma dan ziet u aan het korrekte referentienummer dat de overige codes ook goed zijn. Klopt het referentienummer niet, dan heeft u waarschijnlijk een outje gemaakt bij het intypen van uw geheime code.

```
 360 PRINT AT 15,0;"REFERENTIENUMMER     : ";k(99);k(114);k(37);k
(3);k(174);k(48)
1000 INPUT "": PAUSE 0: RUN
9998 STOP
9999 CLEAR : SAVE d$"PINCODE.BS" LINE 200
```

Nu moet u het programma bewaren en het liefst op meer dan EEN schijf voor het geval er een verloren gaat door storingen. Zorg wel dat u dit doet dmv GO TO 9999 want er MOET vooraf een CLEAR staan omdat u anders uw geheime code meesaved vanuit het variabelen geheugen en uw codes direkt gekraakt worden als u een ander een kopietje geeft! (Een simpel PRINT w$ zou dan voldoende zijn om uw geheime code te onthullen!!!!)

Dank aan de maker van dit toch wel unieke, niet te kraken geheugensteuntje!

(Red.)



MAPPER - brengt uw disks in kaart!

De meesten van u zullen wel gehoord hebben dat de DISCiPLE op zijn disks een 'bitmap' gebruikt. Wat is nu zo'n bitmap en wat zijn de gevolgen van het gebruik ervan?

Het oppervlak van een diskette is bij elk diskinterface verdeeld in tracks (sporen). Dit zijn cirkels op het oppervlak van de schijf die te lezen zijn door de kop van de drive naar binnen of naar buiten te bewegen. Het aantal tracks hangt af van de gebruikte drive (40 of 80 tracks, enkel of dubbelzijdig). De tracks zijn genummerd van 0 (aan de buitenkant) tot bv. 79 in het midden.

Elke track is verdeeld in een aantal sectoren, deze worden aangebracht tijdens het formateren. Op de DISCiPLE zijn dit er altijd 10 waarin 256 (enkele dichtheid) of 512 bytes (dubbele dichtheid) passen. Op de OPUS is het aantal sectoren geheel instelbaar, de standaard is 18 sectoren van 256 bytes, maar ook 10 * 512 of 4 * 1024 is mogelijk. Andere interfaces of coeputers hebben nog veel meer verschillende indelingen.

Van nu af aan zal ik uit gaan van een 80 tracks, dubbelzijdige, dubbele dichtheid diskette. De tracks op de achterkant worden per afspraak 128 t/m 207 genummerd.

Als je nu informatie op de disk wil bewaren moet je bijhouden wat er op staat en welke sectoren er al in gebruik zijn, zodat je de informatie niet overschrijft. Hiervoor zijn bij de DISCiPLE de tracks 0 tot en met 3 gereserveerd. Een preciese beschrijving van de directory vindt u in Nieuwsbrief 7.

Nu komen we eindelijk aan de bitmap, om bij te houden welke sectoren al gebruikt zijn op de disk wordt er in de directory bij elke file vermeld welke sectoren de file beslaat. De schijf bevat op de voorkant 80 tracks waarvan 4 voor de directory en op de achterkant nog eens 80. Voor de files zijn dus 156 beschikbaar met elk 10 sectoren geeft 1560 sectoren. In de bitmap van een file wordt elke van deze sectoren voorgesteld door 1 bit. Is de bit 1 dan beslaat de betreffende file die sector.

Hoe is nu de indeling van de sectoren over de bitmap? Deze is zeer rechtuit, de eerst beschikbare sector heeft nummer 1 en staat op track 4. De hierbij horende bit is het laagste bit (bit 0) van de eerste byte van de bitmap. Track 4, sector 2 komt overeen met bit 1 van dezelfde byte, dit gaat zo door tot sector 8 van track 4. Omdat er 8 bits in een byte gaan is de byte nu vol, daarom wordt voor de volgende sector het laagste bit van de tweede byte gebruikt. In bit 1 komt dan track 4, sector 10 en in bit 2 komt track 5, sector 1 omdat er maar 10 sectoren op een track gaan.

Dit gaat zo door voor alle sectoren van de disk, dit kost 1560 / 8 = 195 bytes voor elke file in de directory. Niet erg efficient als je bedenkt dat alleen het aangeven van eerste en laatste gebruikte sector slechts 4 bytes kost. Maar dan moet de file wel gehheel in opeenvolgende sectoren geSAVEd worden en bij gebruik van een bitmap is dat niet het geval.

disk in kaart?
Nee dank u,
de wijnkaart
alstublieft.

Stel je nu eens voor dat de disk gevuld is met 80 files die elk 1 sector groot zijn, als we nu elke 2e file wissen krijgen we een disk met allemaal ongebruikte 'gaatjes'. Als je nu een lange file op de disk zet zal deze zich verdelen over deze gaatjes zodat de diskruimte optimaal benut wordt.

Het gevolg hiervan is dat op een veel gebruikte werkdisk de files soms zeer vreemd over de disk verdeeld liggen en om dit te laten zien heb ik het programma "Mapper" geschreven. Dit programma laadt de gehele directory en gaat dan voor iedere (niet gewiste) file na welke sectoren in beslag genomen worden. De uitvoer van de machinetaal (BASIC is hier veel te traag voor) is in de vorm van een tabel van 1560 bytes. In deze tabel staat het bij elke sector horende programmanummer (of 0 als er geen is). De volgorde is gelijk aan die van de bitmap. Met een eenvoudig BASIC programma wordt de tabel dan op het scherm of printer zichtbaar gemaakt.

Hier volgt de listing van de machine code om de tabel te maken. Deze staat ook in de file "Mapper.GNS".

```
        ORG  61000        ;Adres waarop de MC begint.
        LD   A,1          ;Drivenummer (POKEN op 60001,1 of 2)
        RST  8            ;Selecteert drive, nummer in A
        DEFB #21          ;
        LD   HL,TABEL     ;Eerste adres van de tabel.
        LD   DE,TABEL+1
        LD   BC,1559      ;Aantal bytes in de tabel-1.
        LD   (HL),0       ;Maak eerste byte leeg.
        LDIR              ;Maak hele tabel leeg.

        CALL LDDIR        ;Laadt de directory.
        LD   HL,DIRECT    ;Adres van de directory in HL.
        LD   B,1          ;Begin bij programma nummer 1.

LOOPFL  LD   A,(HL)       ;Haal directory description op.
        AND  A            ;Test of die nul is (betekent gewist).
        JR   Z,NEXTFL     ;Als dat zo is ga dan naar de volgende.

        PUSH HL           ;Bewaar HL voor volgende file.
        LD   DE,15        ;De bitmap begint 15 bytes hoger.
        ADD  HL,DE        ;Adres van bitmap in HL.

        LD   DE,TABEL     ;Adres van tabel in DE.
        LD   A,8          ;Aantal bits die nog niet bekeken zijn.
        LD   (BITCNT),A   ;Sla dat op in (BITCNT)
        LD   A,(HL)       ;Haal 1e byte van bitmap op.
        LD   (DIRBYT),A   ;En sla die op in (DIRBYT)
        LD   C,195        ;Er moeten 195 bytes bekeken worden.


LOOPTA  CALL GETBIT       ;Zet volgende bit van bitmap in CARRY.
        JR   NC,NOUSED    ;Bit is nul dus ongebruikte sector.

        LD   A,(DE)       ;Haal byte uit tabel voor deze sector op.
        AND  A            ;Staat er al wat in?
        JR   Z,NUSYET     ;Zo nee: OK.
```

even door-lezen nog!

```
        SET   7,A         ;Deze sector is ook al  door  een  andere
        JR    TAMOVE      ;file in gebruik! Zet daarom hoogste bit.
                          ;En laat alles verder zoals het was.

NUSYET  LD    A,B         ;Nog ongebruikt: Zet prog. no. in A.

TAMOVE  LD    (DE),A      ;Zet programmanummer in de tabel.

NOUSED  INC   DE          ;Ga naar volgende sector in de tabel.
        LD    A,C         ;Haal de teller van de bytes op.
        AND   A           ;Is die niet gelijk aan nul herhaal
        JR    NZ,LOOPTA   ;de lus dan nog eens.

        POP   HL          ;Haal begin adres van file weer op.
NEXTFL  INC   H           ;Volgende file zit 256 bytes hoger.

        INC   B           ;Hoog ook het programma nummer op.
        LD    A,B
        CP    81          ;Is dat kleiner dan 81?
        JR    C,LOOPFL    ;Zo ja, herhaal dan voor dit prog. no.

        RET               ;Keer anders terug naar BASIC.


GETBIT  LD    A,(BITCNT)  ;Haal bit teller op.
        AND   A           ;Is die nul?
        JR    NZ,NONBYT   ;Zo nee, sla dit dan over.

        INC   HL          ;Nu gaan we 1 byte verder in de bitmap.
        LD    A,(HL)      ;En halen die byte op.
        LD    (DIRBYT),A  ;Sla nieuwe byte ook op in (DIRBYT).
        LD    A,8         ;En maak bit teller weer 8.
        DEC   C           ;De bitmap byte teller wordt verlaagd.

NONBYT  DEC   A           ;Verlaag de bit teller.
        LD    (BITCNT),A  ;En sla die weer op.
        LD    A,(DIRBYT)  ;Haal bitmap byte op.
        RRCA              ;En roteer de bits naar rechts.
        LD    (DIRBYT),A  ;Sla dan de byte weer op.
        RET               ;En keer terug met bitmap bit in CARRY.

TABEL   EQU   30000       ;De tabel wordt op dit adres aangemaakt.
BITCNT  DEFS  1           ;Adres voor de bit teller
DIRBYT  DEFS  1           ;Sla hier de bitmap byte op
DIRECT  EQU   40000       ;Adres waarop de directory geladen wordt.
```

Dan volgt de directory lader waarvan u de beschrijving kunt vinden in Nieuwsbrief 9, blz. 19.

Deze lader haalt de komplete directory binnen vanaf adres 40000 en vanuit die directory wordt door bovenstaande routine de bitmap tabel samengesteld.

Tonnie Stap.

ASSEMBLERLISTINGEN IN DE NIEUWSBRIEF.

Soms hoor ik wel eens mensen verzuchten dat bepaalde artikelen in de Nieuwsbrief hun ver boven de pet gaan, doelend op de assemblerlistingen, die steeds vaker afgedrukt worden. Nu is het waar dat de Nieuwsbrief evolueert met zijn makers. Door voor u bezig te zijn vergaren we steeds meer kennis van zaken en dat komt terug in de artikelen. Pak ik nummer 1 nog eens, dan heb ik een inhoudelijk heel ander blad in handen dan nummer 13.

Wat we wel proberen is u mee te nemen in die vooruitgang en uw interesse te wekken voor o.a. het echt niet enge terrijn van de machinecode en het programmeren in assembler. Door een zo helder mogelijke uitleg bij de listingen proberen we de angst voor programmeren in assembler weg te nemen. De basiskennis moet u zich zelf eigen maken door een of meerdere naslagwerken of een cursus Z80 aan te schaffen. Dat zo'n cursus niet duur hoeft te zijn bewijst de HCC die het boek "Cursus Z-80" van R.Hutty (bestelnummer 6302120) voor slechts fl 36,— te koop aanbied. (Zelf heb ik het boek "Leren programmeren van de Z-80" van Rodney Zacks bij de bibliotheek geleend en dat kostte dus bijna niks.) Een ander prima boek is "Machinetaal voor de ZX-Spectrum" van William Tang en als naslagwerk het "zakboekje voor de ZX Spectrum. Voor het leren programmeren in basic heeft u van Sinclair toch ook een handboek bij uw Speccie gekregen!

Het kan zijn dat u geen interesse in assembler- of basic-programmeren hebt. Niet iedere chauffeur voelt de behoefte om ook een monteur te zijn, nietwaar? In dat geval geval bent u een passieve gebruiker van onze programma's en krijgt u die kant en klaar aangeleverd op een schijfje, tesamen met een uitgebreide beschrijving in de Nieuwsbrief. Het technische verslag en listing slaat u gewoon over. Uw handicap wordt pas merkbaar wanneer u met panne komt te staan, cq aanpassingen wilt maken, wilt experimenteren of delen van de code wilt inpassen in uw eigen programma's. Een minimale kennis van de materie is dan noodzakelijk. Geloof me dat niemand te oud is om te leren en dat interesse het enige is wat u nodig heeft om iets wijzer te worden en uw hobby aktiever te gaan beleven.

Zou u wel eens een poging willen wagen om wat kennis te vergaren over assembler, dan zal ik hier wat basisbegrippen proberen uit te leggen. Verwacht echter geen cursus van me, die zijn door anderen gemaakt en vele malen beter dan ik ooit zou kunnen.

Machinetaal.

Wilt u weten wat machinecode is? →
Type dan dit in: FOR f=0 TO 16383 : PRINT PEEK f;",";: NEXT f
Niet erg begrijpelijk al die getallen, maar uw Speccie kan er niet buiten. Op het scherm ziet u het programma wat in de ROM ingebakken is. U ziet de decimale vorm, maar de Z80 processor in uw Spec verwerkt deze getallen in binaire vorm. Een getal van 0 tot 255 kan dan worden voorgesteld door 8 bitjes.
0000 0000 = 0 en 1111 1111 = 255.

Stel nou eens een listing voor in dat formaat en u begrijpt dat het voor ons niet leesbaar is, ook niet in decimale of hexadecimale vorm. (Is een getal in hexadecimale vorm dan laten we het voorafgaan door het # teken! 10 is dus niet #10) Het hexadecimaal (16tallig) stelsel is vreselijk handig als we getallen in binaire (2-tallig) vorm willen omzetten. Een byte bestaat uit 8 bits. Die rangschikken we in 2 groepjes van 4 bits. Zo'n groepje van 4 bits noemen we een NIBBLE. Met 4 bits kan je maximaal de getallen 0 tm 15 vormen, wat ook de basisgetallen in het hexadecimale stelsel zijn.

Zien?	0000 = 0	0100 = 4	1000 = 8	1100 = C
	0001 = 1	0101 = 5	1001 = 9	1101 = D
	0010 = 2	0110 = 6	1010 = A	1110 = E
	0011 = 3	0111 = 7	1011 = B	1111 = F

Het getal 0 is in 8 bits vorm #00 en bin 0000 0000
We kunnen ook met 16 bits getallen werken met een Z80
Het getal 0 is in 16 bits vorm #0000 en bin 0000 0000 0000 0000

Het grootste 16 bits getal is 1111 1111 1111 1111 of #FFFF
(zie de tabel)
Dit is tevens de grens van direkt adresseerbare geheugengebied met een Z80 processor. (#FFFF is 65535, jawel 64 K)

Ieder willekeurig binair of hex-getal kunnen we moeiteloos met dit eenvoudige tabelletje omzetten. #37 = bin 0011 0111 en 0100 1111 1011 1100 = #4FBC , zonder enige rekenarij!

Wat ook belangrijk is van het hexadecimale stelsel is de mogelijkheid om het kleinste en het grootste direkt verwerkbare getal in 4 cijfers (4 nibbles) aan te geven. Decimaal heeft hiervoor 5 nodig en er is geen samenhang met de nibbles.

Zo dit was even een algemeen stukje tussendoor en u heeft dit in het geheel NIET nodig om in staat te zijn om onze assembler listings te kunnen lezen of begrijpen!
Dus niet door laten ontmoedigen, alhoewel ik van dit cijfer gegoochel wel pap lust.

REGISTERS.

Zoals in onze Spec 65536 (0 TM 65535) geheugenadressen zijn waar we per adres een 8 bits getal in kunnen stoppen, zo heeft de Z80 een stukje geheugenruimte om getallen in op te slaan voor intern gebruik. Die geheugenadressen zijn nu niet genummerd maar hebben een letter als aanduiding en we noemen ze REGISTERS. (Niks engs dus.) We kennen de registers A,B,C,D,E,F,H en L en een schaduwset A',B',C',D',E',F',H' en L'. Om 16-bits getallen te kunnen verwerken zijn bepaalde registers gekoppeld tot registerparen. Dit zijn de registerparen BC,DE,HL.

Daarbij zijn er nog wat speciale registers, de indexregisters IX en IY, de stackpointer SP, de programcounter PC en de interrupt en refreshregisters IR.
Voorlopig moet u die allemaal maar vergeten.

Van belang is alleen maar te weten dat een register/paar niets anders is dan een intern geheugenadres waar een 8- of 16 bits getal bewaard en bewerkt kan worden.

HET F of FLAGregister

Het F register is een buitenbeentje.
Dit 8 bits register wordt gebruikt voor signaleringen.
Van de 8 bitjes hebben er 6 een naampje gekregen en worden aangeduid als vlag. De vlag kan 1 of 0 zijn, ofwel set of reset.
De vlaggen zijn: S = Sign of teken vlag
Z = zero vlag
H = Half-carry vlag
P/V= Pariteit/overflow vlag
N = optel/aftrek vlag
C = carry vlag

Zo'n signalering is natuurlijk buitengewoon nuttig.
Bepaalde instructies hebben invloed op diverse vlaggen.
Daardoor worden (relatieve) sprongen in de code mogelijk.
U kent in basic IF a>b THEN GO TO c
In assembler wordt dit nu JP of JR met de vlagaanduiding.
(jump of jump relatif = spring of spring relatief)
JP NZ,45000 betekent spring naar adres 45000 als de zerovlag niet 0 (dus 1) is.

ASSEMBLER.

Krijgt de processor een getal of getallencombinatie toegevoerd dan wordt de daarbij passende instructie, die door de ontwerpers van de Z80 is uitgedacht, uitgevoerd.
Om de boel nu voor ons ook leesbaar te maken heeft men aan elke instructie een naampje gegeven, een zgn MNEMONIC.
Al die afkortingen of mnemonics vormen met de bijpassende getallen, of operands de eigenlijke assembler-taal.
Omdat iedere instructie direkt door een of meerdere getallen vervangen kan worden staat deze taal heel dicht bij wat de processor begrijpt en wordt daarom een lage programmeertaal genoemd, anders dan een hogere taal die door eerst door een interpreter of compiler moet worden bewerkt. (Dus trager)

Praten we over assembler of 'n assembler, dan zijn dat twee vrschillende dingen. Assembler is de taal en een assembler is een hulp-programma die het ons mogelijk maakt direkt in die taal te programmeren. Zoals we in basic een programma schrijven, editten en debuggen, zo verzorgt een assembler dit bij programmeren in assembler.

Zo'n programma gebruikt vaak nog wat eigen mnemonics om het schrijven te vergemakkelijken en om later het teksfile te kunnen coopileren naar de echte machinecode.

Pak de listing van MAPPER er eens bij.

Wat vooral direkt opvalt is het gebruik van LABELS. Die LABELS zijn razend handig. Ziet u CALL GETBIT staan?

CALL is een mnemonic en betekent spring naar subroutine .. In basic doet u dit met GO SUB regelnummer. Nu geeft de programmeur die subroutine een door hem verzonnen naam GETBIT en de assembler zoekt later wel uit waar die routine werkelijk staat. Geen omkijken naar.

De listing begint met ORG, ook een eigen mnemonic. Hiermee wordt bedoeld, maak de machine code aan vanaf adres.. De programmeur bepaald zelf waar zijn code komt te staan, maar hoeft daar zelf niets aan te doen. Onderaan staan nog wat eigen- of pseudo mnemonics die niets met de Z80 instructieset van doen hebben maar alleen bedoeld zijn het programmeren te vergemakkelijken.

```
EQU (of EQUals, is gelijk aan )
DIRECT EQU  40000  : met het label DIRECT bedoelen we dus het
                     adres 40000.
```

De programmeur schrijft dus niet "laad register HL met 40000" maar LD HL,DIRECT
Moet later dit adres veranderd worden dan hoeft alleen maar de EQU aangepast en alle adressen in de code worden automatisch aangepast als het tekstfile opnieuw gecompileerd wordt!

```
DEFS 1  - reserveer een string geheugenadressen van 1 lang
DEFB 0  - reserveer een 8 bits geheugenplaats
DEFW 0  - reserveer een 16 bits geheugenplaats
```

hé?

MAPPER assemblerlisting.

De listing van Mapper werd door Tonnie Stap voorzien van uitleg wat het behoort te doen.
Lezen in assembler gaat eigenlijk anders.
Om het verschil te laten zien zal ik de listing nu gedeeltelijk beschrijven vanuit assembler.

```
        ORG   61000       ;maak de code aan vanaf adres 61000
        LD    HL,TABEL    ;Laad registerpaar HL met TABEL (30000)
        LD    DE,TABEL+1   Laad DE met 30001
        LD    BC,1559     ;Laad BC met 1559
        LD    (HL),0      ;Laad het ADRES aangewezen door HL
                           (=30000) met het getal 0
        LDIR              ;Laad verhoog en herhaal. Het adres
                           aangewezen door DE wordt geladen met de
                           inhoud van het adres aangewezen door HL.
                           HL wordt dan HL+1, DE wordt DE+1 en BC
                           wordt BC-1. Herhaal to BC=0
        CALL LDDIR        ;Voer subroutine LDDIR uit en keer na
                           uitvoer (instr. RET) hier terug.
        LD    HL,DIRECT   ;Laad HL met DIRECT (zie EQU=40000)
        LD    B,1         ;Laad B met 1
LOOPFL  LD    A,(HL)      ;Laad A met de inhoud van adres
                           aangegeven door HL (inhoud van 40000)
        AND   A           ;Voer een logische AND bewerking uit
                           AND beinvloed o.a. de ZERO vlag.
        JR    Z,NEXTFL    ;Spring naar NEXTFL als de zero-vlag op 1
                           staat.
        PUSH  HL          ;Zet de waarde van HL op de stapel
                           (gereserveerde geheugenplaats aangegeven
                           door de stackpointer.)
        LD    DE,15       ;Laad DE met 15
        ADD   HL,DE       ;Tel DE op bij HL (HL=HL+15)
        LD    DE,TABEL    ;Laad DE met TABEL (=30000)
        LD    A,8         ;Laad A met 8
        LD    (BITCNT),A  ;Laad adres BITCNT met A (=8)
        LD    A,(HL)      ;Laad A met de inhoud van het adres
```

```
                        aangegeven door HL
       LD   (DIRBYT),A ;Laad adres DIRBYT met A
       LD   C,195      ;Laad telregister C met 195
LOOPTA CALL GETBIT     ;voer subroutine GETBIT uit en kom na de
                        RET hier terug
       JR   NC,NOUSED  ;Spring naar de routine NOUSED als de
                        carry vlag NIET geset is.
```

Enfin, dit moet u een idee geven hoe e.e.a. gelezen wordt. Zelf neem ik dan een vel papier en noteer stap voor stap wat er met welke register gebeurd en kan zo de loop van de code redelijk volgen en dus begrijpen waarom bepaalde dingen zo gedaan worden.

Vaak volsta ik het commentaar van de programmeur te lezen. Als hij schrijft "LDDIR" zorgt dat de directory geladen wordt vanaf adres 40000, dan ga ik niet exact uitpluizen hoe hij dat doet en neem gewoon aan dat die subroutine doet wat hij beweert.

Voor de rest is deze materie een kwestie van goed documenteren, veel lezen, beginnen met de meest eenvoudige routines en boeken hierover, veel interesse en wat doorzettingsvermogen.

Tot slot een kleine illustratie van het belang van machinecode.

```
10 LET a=40000
20 FOR f=16384 TO 23295
30 POKE a,PEEK f :LET a=a+1
40 NEXT f
50 CLS
60 LET a=16384
70 FOR f=40000 TO 46911
80 POKE a,PEEK f :LET a=a+1
90 NEXT f
```

Laad een SCREEN$ en type GO TO 10
Het SCREEN$ wordt naar adres 40000 gepoked en na de CLS weer op het schera opgebouwd in 2 minuten en 50 seconden!

En nu deze:

```
10 CLEAR 59999
20 FOR f=60000 TO 60011: READ g: POKE f,g: NEXT f
30 DATA 33,0,64,17,64,156,1,0,27,237,176,201
40 FOR f=60013 TO 60026: READ g: POKE f,g: NEXT f
50 DATA 33,64,156,17,0,64,1,0,27,237,176,201
```

Maak de machinecode aan door RUN en wis de basic met NEW
Laad nu een SCREEN$ en type RANDOMIZE USR 60000 : CLS en dan RANDOMIZE USR 60013

Begrijpt u nu wat ik bedoel?
Machinecode is niet voor elk doeleinde geschikt, maar soms kan je er niet buiten en wat kennis op dat gebied is dan nooit weg.

Wat staat er nu in assembler in het geheugen?

```
vanaf 60000 : LD   HL,16384  vanaf 60013: LD   HL,40000
              LD   DE,40000                LD   DE,16384
              LD   BC,6912                 LD   BC,6912
              LDIR                         LDIR
              RET                          RET
```

Heeft u vragen over dit onderwerp schroom dan niet om een briefje te sturen aan het redactieadres (met antwoordzegel s.v.p.) of u kunt me hierover aanspreken tijdens de overbekende Spectrumdag in Houten.

Succes!

P.Faas

SAVER, DE SUPER SECTOR REDDER

Als redacteur van de nieuwsbrief mag ik regelmatig schijven in ontvangst nemen met daarop uw bijdragen voor de nieuwsbrief, uiteraard tot mijn grote genoegen.

Helaas wil het een enkele keer voorkomen dat op zo'n schijfje een sectror error staat, waardoor de desbetreffende bijdrage niet meer in te lezen valt.

Nu is zo'n sector error snel te verhelpen door er een blanko sector overheen te schrijven mbv SAVE $1,t,s,adres waarbij echter alle data in de gerepareerde sector verloren gaat. Niet echt een elegante oplossing dus.

Op de laatste schijf die ik van Tonnie Step mocht ontvangen deed zich dit nare probleem ook voor en juist bij de eerste sector van een nieuw basic-programma, zodat die helemaal de mist in dreigde te gaan.

De enige oplossing leek mij de bestaande inhoud van de sector te lezen en die daarna weer terug te schrijven. Door de schrijfaktie wordt de sector gerepareerd en hopelijk zou dan niet alle informatie verloren zijn.

Bij een sector error valt vooraf nooit te voorspellen of er data verminkt is en zo ja in welke mate. Maar beter iets gered dan niets! Met LOAD * gaat het niet, vanwege de foutmelding Sector error, dus moeten we teruggrijpen naar de hookcodes in machinecode waarmee we foutmeldingen gewoon kunnen negeren!

Schrik niet, met 13 bytes is alles gepiept!

```
             10    ORG  62000   maak hier de code aan
16 tr        20    LD   D,6     tracknummer in reg. D
1E se        30    LD   E,9     sectornummer in reg. E
D5           40    PUSH DE      bewaren op de stapel
CF           50    RST  8       de Disciple/+d routine "RSAD" ofwel
3F           60    DEFB #3F     Read Sector At DE
```

De desbetreffende sector is nu in de DRAM geladen.

```
A7           70    AND  A       reset carryflag (0), door sector
                                  error wordt carryflag geset (1).
D1           80    POP  DE      haal track en sectornr. van stapel
CF           90    RST  8       Disciple/+d routine "WSAD" ofwel
3E          100    DEFB 3E      Write Sector At DE
```

De sector wordt vanuit de DRAM weer op schijf gezet.

```
A7          110     AND  A      reset carryflag, geen foutmelding
C9          120     RET         terug naar BASIC.
```

Regel 40 is nodig omdat de inhoud van het DE registerpaar door de RST routine tussentijds wordt veranderd. Omdat we DE nog nodig hebben voor de schrijfaktie PUSHen we DE dus op de stack en halen de inhoud terug vlak voor de schrijfaktie (regel 80).

Bij het laden wordt de CARRYFLAG 1 (set) vanwege de sectorfout. Eigenlijk kan regel 70 achterwege blijven omdat we de carryflag alleen maar op 0 hoeven te zetten voor we naar basic terugkeren. Dat op 0 zetten doen we met regel 110.

De carryflag moet op 0 gezet worden om de Disciple foutmelding: Invalid CODE te onderdrukken. Zo'n foutmelding zou een BASIC programma afbreken en dat is nu niet de bedoeling.

Bij deze paar bytes machinecode heb ik een klein stukje basic gemaakt.

De Disciple geeft bij een sectorerror altijd precies aan welke track en sector het betreft. Even onthouden dus en bij de twee INPUTs opgeven.

```
 10 PRINT "Red een sector...(of niet.)"
 20 INPUT "Tracknummer : ";t
 30 INPUT "Sectornummer : ";s
```

Even de machinecode aanmaken.

```
 120 FOR f=62000 TO 62012: READ g: POKE f,g: NEXT f
 130 DATA 22,t,30,s,213,207,63,167,209,207,62,167,201
```

Schijf met de fout moet nu natuurlijk wel in de drive zitten.

```
 150 RANDOMIZE USR 62000
 160 PRINT ''"Sector ";s;" van track ";t'"gelezen en weggeschrev
en."''"klaar...."
```

Wel goed opletten, de INPUT is niet beveiligd tegen onzinnige waarden waardoor de hele boel over zijn nek kan gaan.
Sector 0 bestaat bv niet en kan een crash veroorzaken!

Op Tonnies schijf was sector 9 van track 6 kapot en uiteraard heb ik dit programma er even op los gelaten. Groot was mijn vreugde toen bleek dat de fout hersteld was en er geen data verloren was. Het programma laadde nu feilloos.

Het blijft echter iedere keer een gok of data beschadigd is. Maar u heeft nu de mogelijkheid om een kapotte sector te repareren met behoud van data en zo dat gevoel van machteloosheid kwijt te raken.

P.Faas

******* STER PROGRAMMA *******

DISCIPLE ** TOOL **

Op de bijgaande diskette staat het programma 'tool'. Het bevat een aantal utilities om klusjes aan Disciple diskettes te klaren. Het programma bevat een wat aangepaste versie van de disk-naar-disk copier van Tonnie Stap alsmede ander materiaal Dat in de Nieuwsbrief gepubliceerd is plus wat aanvullingen gecombineerd tot een, erg gebruiksvriendelijk, programma. U bent vrij het programma op niet commerciele basis te gebruiken.

Na het opstarten met LOAD d$"tool" kan men een te bewerken diskette plaatsen. Daarna typt men ENTER. De directory wordt gelezen en het hoofdmenu verschijnt. Met de cursor toetsen of een joystick kan men nu files selecteren oa te worden bewerkt.

De volgende opties zijn ingebouwd:

N- Gebruik een andere diskette

H- Display de uitleg van de letters voor de 00orten bestanden

K- Kill selected files, razendsnel.

F- File test. Geeft een uitgebreide beschrijving van een bestand en checkt of alle sectoren van de map overeenkomen met de sectoren in het bestand.

U- Un-erase. Van de gewiste files wordt nagegaan of de sectoren nog overeenkomen met de map. Zo ja, dan wordt de file weer zichtbaar gemaakt. Met 'K' zijn ongewenste files gauw genoeg weer te wissen.

T- Test diskette. Alle sectoren vanaf track 4 worden gelezen. Bij een leesfout wordt het betreffende bitje in file 80 gezet, zodat de sector als gebruikt bekend staat. Als er slechte sectoren zijn, dan heet deze file "Sect_Error". Een al bestaande file 80 wordt elders in de directory geplaatst. "Sect_Error" is een SPECIAL file, zodat hij binnen dit programma onzichtbaar is.

R- Reset selectie

C- Copieer geselecteerde files. Alle bestanden behalve SPECIAL files kunnen gecopieerd worden. Er kunnen meerdere copieen na elkaar gemaakt worden. Omdat alle RAM voor het copieren gebruikt wordt zal de Spectrum zich na het copieren resetten.

Om un-erase te vergemakkelijken worden obk de andere file-types bij het lezen van de directory van een nummer in HD00, byte 211 voorzien. MD files, executes, snapshots etc. protesteren daar niet tegen.

W. Lageman.

N.B.
De test optie "T" opent nu de mogelijkheid om schijven die na herhaald formatteren sectorerrors blijven geven, toch volwaardig te gebruiken zonder het risico dat het systeemfile zal proberen om files op deze slechte sectors weg te schrijven. Al met al typisch een programma wat zichzelf snel onmisbaar zal maken bij menig enthousiaste Disciple gebruiker. (Red.)

DISKFILER

PROGRAMMA VOOR HET OPTIMAAL REORGANISEREN
EN BEHEERSEN VAN UW DISKETTE-BESTANDEN.

CAPACITEIT TOT LIEFST 2000 FILES!

**LEEST DIRECTORIES UIT, DOORZOEKT
SUPERSNEL BESTANDEN, VERANDERD,
SORTEERT EN BEWAARD.**

BULK ERASE OPTIE.

GESORTEERD LIJSTEN PRINTEN.

2 DRIVE OPTIE.

ZEER GEBRUIKSVRIENDELIJK.

TE BESTELLEN DOOR STORTING VAN FL 12,50 OP
GIRO [redacted], T.N.V. P. FAAS, RED.NWSBR, [redacted]

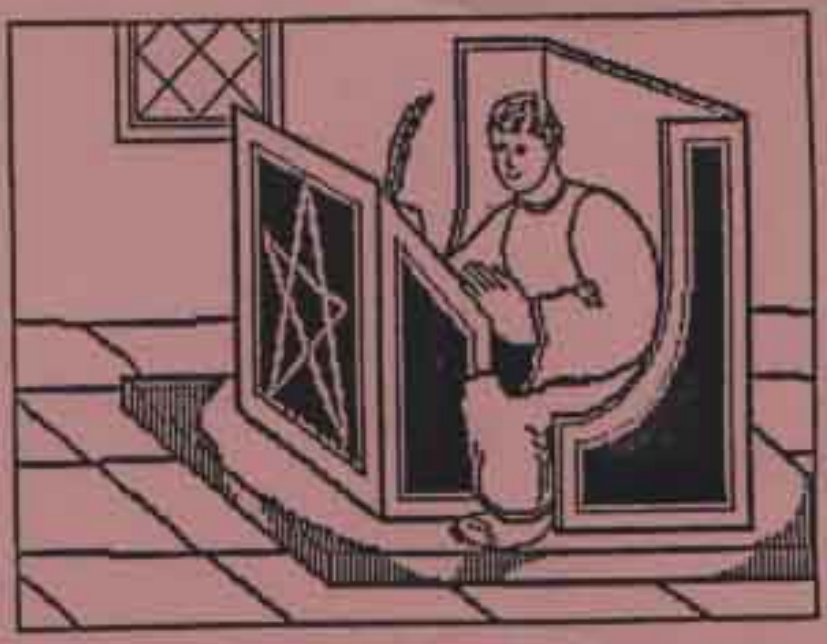
GOUD VAN OUD

THE BEST OF THE DISCIPLE NIEUWSBRIEF

**MET O.A. UN-ERASE-(HERSTELTOOL)
BAS-TAS (BASIC-NAAR TASWORD)
SNELLE RAMTEST-(DIAGNOSETOOL)
SPECWORD 48/+/128-(WORDPROCESSOR)
TASWORD II- SPECIAL (ZONDER CODEBLOK)
TRANS-D>T-(DISK NAAR TAPE KOPIER)
TRANS-T>D-(TAPE NAAR DISK KOPIER)
TRANS-OTYPE-(OPEN TYPEFILE KOPIER)
MULTIFACER-(MULTIFASE EMULATOR)
DISK-DOKTER-(REPARATIE TOOL)
SCREENMAKER-(VERVANG SNAPSHOT SCREENS)
SECTORPIEK-(MACHINECODE PEEKER & POKER)
AUTO JACK-(AUTO LOADER)
FIN A FILE-(UN-ERASE TOOL)
OKIE POKIE-(POKE DUMPER)
TRANSSNAP-(SNAPSHOT DISK>TAPE KOPIER)**

TE BESTELLEN DOOR STORTING VAN FL 12,50 OP
GIRO [redacted], T.N.V. P. FAAS, RED.NWSBR, [redacted]

De Klerk



Tekstbewerker voor de ZX Spectrum 48K/128K met Disciple Interface.

De Klerk is snel, kent letters met accenten, kan rekenen, heeft maximaal 234 helppagina's, de gebruiker kan eigen help-pagina's invoeren.

In- en uitvoer van Ascii, Tasword II, Tasword III, Devpac, Pascal en C-compiler bestanden tot een maximale grootte van 27K. Back up files kunnen automatisch worden aangemaakt.

De Klerk kan dit alles en nog meer voor een prijs van ƒ 59,—

Master Publisher

Desk Top Publishing pakket voor ZX Spectrum 48K/128K.

De Master Publisher heeft alles wat De Klerk ook heeft, maar kan de tekst afdrukken als professioneel drukwerk. Geen laser printer nodig. De Master Publisher werkt met een gewone Dot-Matrix printer.

Er zijn 5 standaard lettertypes, welke gewoon, cursief en/of vet gezet kunnen worden. Met FontEdit kun je je eigen letters maken.

Tekst kan proportioneel gezet worden, waarbij ook kernen mogelijk is. Regels kunnen links gericht, rechts gericht, gecentreerd en uitgelijnd gezet worden. Ook is horizontaal- en/of vertikaal vergroten mogelijk.

Men kan kolommen met een joystick aangeven. De Master Publisher zal zelf de tekst over de aangegeven kolommen verdelen.

Men kan lijnen en vlakken aangeven om helemaal zwart weer te geven. Met de snapshot kunnen in allerlei programma's SCREENS's gemaakt worden. De Master Publisher kan deze geheel of gedeeltelijk weergeven, zo nodig tot een keer of 10 vergroot.

Dit pakket is z'n geld meer dan waard voor ƒ 99,—

De Klerk en de Master Publisher zijn verkrijgbaar bij Data-Skip in Gouda. Ook kan men deze pakketten bij de auteur bestellen door het genoemde bedrag plus ƒ 5,— verzendkosten over te maken naar postgiro [redacted] t.n.v. Willem Lagerman,
[redacted] [redacted]

DISCIPLE HANDBOEK

NEDERLANDS TALIG

FL. 12.50

TE BESTELLEN DOOR STORTING
VAN FI 12.50 OP GIRO 3154455
TNV. P. FAAS, RED. NWSBR.
ZAANDAM