

Elfde Jaargang, Nr. 3
Juni 1987

**** DE 6502 KENNERS ** — EEN CLUB VOOR 6XXXX GEBRUIKERS**

De vereniging heeft leden in Nederland, België, Duitsland, Frankrijk, Engeland, Denemarken, Zweden, Spanje, Portugal, Oostenrijk, Finland, Israël, Amerika. Het doel van de vereniging is: het bevorderen van de kennisuitwisseling tussen gebruikers van 6XXXX-computers, als COMMODORE-16/64/128, AMIGA, APPLE II/IIE/IIc/IIgs/III, MACINTOSH, ATARI 600/800XL/512/1024ST, QUANTUM LEAP, CHE-1, PEARCOM, AIM-65, SYM, PET, BBC, VIC-20, BASIS 108, PROTON-computers, IIT-2020, OSI, ACC 8000, ACORN ELECTRON, SYSTEM 65, PC-100, PALLAS, MINTA, FORMOSA, ORIC-1, STARLIGHT, CV-777, ESTATE III, SBC65/68, KIM, NCS, KEMPAC SYSTEM-4, Elektuur-computers (JUNIOR, EC65(K) alias OCTOPUS), LASER, dus ook 6800, 6809 en 68000-computers. De kennisuitwisseling wordt o.a. gerealiseerd door 6 maal per jaar DE 6502 KENNER te publiceren, door het houden van landelijke clubbijeenkomsten, door het instandhouden van een diskette-service en door het verlenen van paperware-service.

Verschijningsdata
DE 6502 KENNER 1985

derde zaterdag van
februari, april, juni,
augustus, oktober, december.

Redactie-adres en informa-
tie over paperware etc.

Willem L. van Pelt
Jacob Jordaensstraat 15
NL-2923 CK Krimpen/IJssel.
Tel.: 01807 - 19881

De vereniging is volledig onafhankelijk, is statutair opgericht en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305.

Voorzitter:
Rinus Vleesch Dubois
Fl. Nightingalestraat 212
NL-2037 NG Haarlem
Tel.: 023 - 330993

Bijeenkomsten van de club
derde zaterdag van
januari, maart, mei,
september, november.

Sekretaris:
Gert Klein
Diedenweg 119
NL-6706 CM Wageningen
Tel.: 08370 - 23646

Penningmeester:
John F. van Sprang
Tulp 71
NL-2925 EW Krimpen/IJssel.
Tel.: 01807 - 20589

Redactie DE 6502 KENNER:
Willem L. van Pelt
Jacob Jordaensstraat 15
NL-2923 CK Krimpen/IJssel.
Tel.: 01807 - 19881

Leden:
Adri Hankel (05490 - 51151) Hardware/software/DOS65
Erwin Visschedijk (05496 - 76764) Hardware/software/DOS65
Gert van Opbroek (01729 - 8636) 65802/65816/68000
Nico de Vries (010 - 4517154) Hard-/software/68000
Brevoorzitter: Siep de Vries
Ereleden : Mw. H. de Vries - Van der Winden
Anton Müller
Lidmaatschap : Hfl. 50,=
Subscription : Europe : Hfl. 59,50
(eurocheque = Hfl. 50,=)
Outside Europa:
Hfl. 114,50 (incl.transfers)
Advertenties : Tarieven op aanvraag bij de redactie.

**** DE 6502 KENNER ** — EEN BLAD VOOR 6XXXX GEBRUIKERS**

DE 6502 KENNER is een uitgave van de KIM Gebruikers Club Nederland. Het blad wordt verstrekt aan leden van de club. DE 6502 KENNER wordt van kopij voorzien door leden van de club, bij de opmaak van een publikatie bijgestaan door de redactie. De inzendingen van programma's dienen voorzien te zijn van commentaar in de listings en zo mogelijk door een inleiding voorafgegaan. Publikatie van een inzending bete-kent niet dat de redactie of het bestuur enige aansprake-lijkheid aanvaardt voor de toepassing ervan. De inzendingen kunnen geschieden in assembly-source-listings, in Basic, in Basicode, Forth, Focal, Comal, 'C', Pascal, Fortran, Cobol, Logo Elan, etc. etc. De leden schrijven ook artikelen over de door hen ontwikkelde hardware en/of aanpassingen daarop. Zij schrijven tevens artikelen van algemene aard of reageren op publikaties van andere inzenders.

DE 6502 KENNER IS EEN BLAD VAN EN DOOR DE LEDEN

Micro-ADE Assembler/Disassembler/Editor is een produkt van Micro Ware Ltd., geschreven door Peter Jennings en bestemd voor alle 6502-computers. De KIM Gebruikers Club Ned. heeft de copyrights verworven nadat ons lid Sebo Woldringh de 4 K KIM-1 versie uitbreidde tot 8 K KIM-1 versie. Adri Hankel paste deze aan voor de JUNIOR. Willem L. van Pelt stelde een nieuwe 8 K source-listing voor de JUNIOR samen. De implementatie op andere systemen dan de KIM-1 en JUNIOR kan eenvoudig gebeuren door het aanpassen van de I/O-adressen, welke in de source-listing gemakkelijk te vinden zijn FATE Format-listener/cond. Assembler/Tape-utilities/Editor is de door ons lid Rob Banen geschreven source-listing van een 12 K universeel systeem voor de JUNIOR-computer aan de hand van het universele disk operating system van de fa. Proton Electronics te Naarden. FATE wordt beschikbaar gesteld met toestemming van Proton. DOS65 V2.01 is the new system of our club, build with Elek-tor's CPU, VDU, RAM-cards and our own professional Floppy-Disk-Controllercard for SS, DS, 40 or 80 tracks and a max. of 720 Kbytes capacity. For use with 6502 or 65C02. For more information, write to E.J.M. Visschedijk Dillenaan 11, NL-7641 CX WIERDEN.

The new DOS65 V2.01 is hardware compatible with Elektor's OCTOPUS/EC65 computer, except the controllercard.

In de edities van DE 6502 KENNER worden regelmatig mededelingen gedaan over de door de club georganiseerde bijeenkomsten. Ook worden bestuurlijke mededelingen gedaan, naast informaties over hetgeen in de handel te koop is. Leden die iets te koop hebben of iets zoeken kunnen dit in de edities van DE 6502 KENNER bekend maken. Ook worden brieven aan de redactie gepubliceerd, evenals specifieke vragen van leden. De edities worden samengesteld op basis van een groot aantal prioriteiten, welke door een redactievergadering worden gehanteerd. Deze vergadering bestaat uit de vaste medewerkers zoals in de colofon vermeld. Het aantal inzendingen is groter dan in een enkele editie van minimaal 48 pagina's is te verwerken. Hierdoor kan het voorkomen dat een inzending eerst na enige tijd kan worden gepubliceerd.



De 6502 KENNER is een uitgave van de KIM gebruikers Club Nederland.

Adres voor het inzenden van en reacties op artikelen voor DE 6502 KENNER:
 Willem L. van Pelt
 Jacob Jordaensstraat 15
 2923 CK Krimpen a/IJssel
 Tel.: 01807 - 19881

Vaste medewerkers:
 Willem L. van Pelt
 Gerard van Roekel
 Frans Smeehuijzen
 Coen Boltjes

Freelance medewerkers:
 Fred Behringer (Germany)
 Andrew Gregory (England)
 Marc Lachaert (Belgium)
 Fernando Lopes (Portugal)
 Gert van Opbroek
 Leif Rasmussen (Denmark)
 Ruud Uphoff
 Frans Verberkt
 Herman Zondag (drawings)

Vertaalwerk:
 Fred Behringer (Germany)
 Willem van Asperen
 Frank Bens
 Albert v.d. Beukel
 F. Vandekerkhove (Belgium)
 Coen Kleipool (France)
 Maarten van Lieshout
 Antoine Megens
 Iddy Oort
 Mw. Elja van der Veer
 Piet K. de Vries
 en vele anderen.

Illustraties/cartoons:
 Antoine Megens
 Mw. Jitske de Vries
 Frank Vergoossen
 en anderen.

Gehele of gedeeltelijke overname van de inhoud van DE 6502 KENNER zonder toestemming van het bestuur is verboden. Toepassing van gepubliceerde programma's, hardware etc. is alleen toegestaan voor persoonlijk gebruik.

DE 6502 KENNER verschijnt 6 x per jaar en heeft een oplage van 500 exemplaren.

Copyright (C) 1987 KIM Gebruikers Club Nederland.

De voorpagina is de DOS65-controllerkaart. ontwikkeld door Ad Brouwer.
 CAD/CAM: E. Visschedijk.
 i.s.m. : A. Hankel
 Fotogr.: Fr. Visschedijk.

I.v.m. auteurswetgeving aanvaardt de redactie geen aansprakelijkheid voor inzendingen. Tenzij anders aangegeven, dient de inzending afkomstig te zijn van de inzender.

INHOUDSOPGAVE DE 6502 KENNER NR. 50 JUNI 1987

1. Van de redactie	2.
2. Self Modifying Code on Trial ... Leif Rasmussen, Denmark	2.
3. APPLE][DOS 3.3 FAST LOAD PATCH ... Ruud Uphoff, Holland	3.
4. FAST ALTERNATIVE FOR THE SLOW JUNIOR CASSETTE INTERFACE AND SOFTWARE ... Hans Christen, Holland	4.
5. FORMAT SPACES FOR ACORN ELECTRON AND BBC ... Ronald van Vugt, Holland	8.
6. VIDEO.H; Small C video control routines for IO65/DOS65 system ... Antoine Megens, Holland	9.
7. DOS65 V2.01 Zet printerlettermode commando voor de printer NMS-1431 van Philips ... Henk Quast, Holland	11.
8. RELOCATOR FOR DOS-JUNIOR AND OCTOPUS 65 ... Coen Boltjes, Holland	19.
9. RAM TEST PROGRAMM FOR THE OCTOPUS 65 ... Marc Lachaert, Belgium	21.
10. AMAZING MAZE v.1. COMAL; for MON/DOS65 systems ... Antoine Megens, Holland	23.
11. Basic: FUNCTION DISPLAY ... Ger van den Hoek, Holland	25.
12. FREQUENTIECOUNTER M.B.V. DE VIA 6522; voor de Motorola MC 6800 - 6802 - 6808 uP ... Frank Weijnen, Holland	27.
13. Forth: MEMORY-INPUT-HANDLER ... Klaus Pohlmeij, Germany	34.
14. COLUMNS PRINT FOR THE ACORN ATOM ... Piet K. de Vries, Holland	35.
15. HOLD + APPEND; Rapid APPLE version ... Frans Verberkt, Holland	37.
16. PROGRAM TIMEDATE ... Peter Roessingh, Holland	41.
17. TIME FOR THE BBC AND ELECTRON ... Ronald van Vugt, Holland	47.
18. Ingangsprotectie LOGIC ANALYSER voor DOS65 V2.01 ... Henk Quast, Holland	48.
19. Diversen: DATBAS voor OCTOPUS; een paar foutjes? 7. SOLVED PROBLEMS OCTOPUS; vdu-card, monitor-eprom 8. COPY PROBLEMS DOS65 8. FOR SALE/TE KOOP; QUESTION/VRAAG 8,18,34,36. DOS65 Hardware Bug 18. Changes made in COMAL V2.1 for DOS65 ... Antoine Megens 18. CALCULATOR ... Frans Verberkt 18. PAPERWARE SERVICE EPROM PROGRAMMER FOR JUNIOR ... Andrew Gregory 36. REFLECTED IMAGE ... Gerard van Roekel 36. BOEKINFORMATIE 20,36. PRIJSWINNAARS 7.	

WORDPROCESSOR V3.0 FULL SCREEN EDITOR BY LEIF RASMUSSEN ALLOWS YOU 'CRUISING' AROUND FROM TOP TO BOTTOM OF THE FILE (OR EVEN MORE THAN ONE FILE AT A TIME). ARE YOU INTERESTED? ASK THE EDITORIAL OFFICE FOR THE PRICE OF THE DISKETTE WITH HANDY MANUAL. SOURCE-LISTING ALSO AVAILABLE IN PAPERWARE SERVICE.

ON PAGE 49 THE NEW ASSEMBLER FOR EC65, DEVELOPED BY MARC LACHAERT AND BASED ON THE IDEA OF ROB BANEN'S 'FATE' ASSEMBLER FOR THE JUNIOR. NOW WITH SEVERAL NEW FEATURES AND, OF COURSE, ON DOS-LEVEL. ARE YOU INTERESTED? ASK THE EDITORIAL OFFICE FOR THE PRICE OF THE DISKETTE WITH VOLUMINOUS MANUAL. SOURCE-LISTING ALSO AVAILABLE IN PAPERWARE SERVICE.

Van de redactie

Een aantal veranderingen markeren het bestaan van onze club, althans volgens mijn waarneming. Deze veranderingen tekenen gewoonlijk de reactie op de behoeften onder de leden.

Opricht in januari 1977 als KIM Gebruikers Club Nederland volstond het om de gebruikers van de eerste zelfbouw-computer, de KIM-1, van dienst te zijn met het uitwisselen van kennis omtrent deze computer.

Toen enige jaren later de club haar poorten had geopend voor Elektuur's zelfbouw JUNIOR computer, in een poging van het bestuur om ook die gebruikers van dienst te zijn, werd het accent op een specifieke machine verschoven naar een specifieke processor: de 6502. Het is vooral de JUNIOR geweest die zichtbaar maakte dat onze club van het begin af onderdak bood aan gebruikers van de 6502, ongeacht de herkomst van de gebruiker, m.a.w. onze club bood van oudsher hobbygenoegen aan aan gebruikers in om het even welk land. Door de aard van het clubblad, dat werd uitgegeven in de nederlandse taal, bleef het ledenbestand beperkt tot gebruikers die de nederlandse taal machtig waren. Het ligt voor de hand daarbij aan Belgen te denken maar evengoed kwamen daaronder gebruikers uit Duitsland tegen. Om een en ander hanteerbaar te maken, spreek ik hier in het algemeen over nederlandsestaligen, dus over leden in alle landen die de nederlandse taal konden lezen en/of schrijven.

Het verschuiven van het accent van één machine naar één processor heeft het ledenbestand niet alleen uitgebreid naar meer nederlandsestaligen, maar vooral naar meer machines. Waar het zo was, dat AIM-65, OSI e.d. al in onze club vertegenwoordigd waren, de komst na de PET van de C-64, BBC en vele andere computers bracht ons een nog meer gevarieerd ledenbestand. Het is een absoluut misverstand om nog te spreken van een club van alleen zelfbouwers, al wekt de enorme activiteit onder deze leden soms een ander beeld op. Het is geen korrekte weergave van de werkelijkheid om van een club van zelfbouwers te spreken. Die tijd mag er rond KIM-1, en JUNIOR zo hebben uitgezien en lijkt er nog zo uit te zien, het is niettemin geen eeuwigheids-verschijnsel.

Met de komst van de Elektuur EC65 alias OCTOPUS computer wordt een verdere nadruk gelegd op het gebruik van meerdere processoren in één computer. De APPLE-compatible van de club bevat al een Z80 processor, de EC65-gebruikers verlangen er eveneens naar zo'n processorkaart toe te passen in combinatie met de 6xxxx-processor. Toch doet het niet af aan het feit dat andere computers in onze club de laatste tijd steeds meer van hun aanwezigheid getuigen. Zie daarvoor ook de publikaties van de gebruikers van APPLE, Acorn Atom, Acorn Electron, Commodore, en - niet te vergeten - de ATARI ST. Het leidt mijns inziens tot ongekend zelfbedrog om niet in te zien dat onze club niet meer vast kan houden aan een nostalgische maar kennisverengende eigengereidheid. De club verdient het om voortdurend de veranderende behoeften onder de computergebruikers waar te nemen om vandaaruit vast te stellen hoe verder te zullen gaan. Het krampachtig de naam KIM Gebruikers Club Nederland handhaven en tegelijkertijd de clubnaam DE 6502 KENNERS blijven gebruiken is een aanduiding dat het incorporeren van de toekomst in de opstelling van de club een niet eenvoudige zaak is, maar het mag nimmer een hinderpaal worden.

Hetzelfde geldt voor de discussie over onze taalkundige identiteit. De leden van onze club hebben, blijkens de uitkomst van een vorig jaar gehouden enquête en naar mijn vaste overtuiging, blijkend uit de ongelofelijke hoeveelheid communicatie die er vanuit het redaktiekamertje met de leden bestaat, zich in overgrote meerderheid in algemene zin allang vereenzelvigd met het gebruik van de engelse taal in ons blad, het logische gevolg van de acceptatie van niet-nederlandstaligen als lid, met als bijkomend voordeel dat kennis vanuit het buitenland onze leden evengoed bevruchten kan. Er vindt geen massale toetreding van niet-nederlandstaligen plaats, maar de animo is groeiend. Op 31.12.86 bedroeg het aandeel van hen zo'n 13,4 %. Tussen 1.1.87 en 30.4.87 zijn er 23 nieuwe leden toetrededen. Hieronder 11 Nederlanders, 2 Belgen en maar liefst 10 niet-nederlandstaligen! Het aandeel engelstaligen in de club steeg naar zo'n 17,5 %. In het bestuur heeft de penningmeester ooit eens gewag gemaakt van een zogenaamd strategisch minimum, dat is het aantal leden waaronder niet gedaald mag worden. De straf daarvoor zou zijn het wegvallen van het bestaansrecht. Als men een beleid zou voeren waarin slechts rekening gehouden zou worden met het verlangen van een enkeling om uitsluitend in de nederlandse taal te publiceren, al konstateer ik gelukkig dat DOS65 nu door de distributeurs ook voorzien wordt van engelse manuals voor de gebruikers in het buitenland, dan zou men voorbij gaan aan het feit dat die 17,5% engelstaligen in onze club bij hun afwezigheid een daling tot onder het strategisch minimum kunnen veroorzaken. Uw redaktieman worstelt met het probleem als er bekend is wat de leden willen er toch steeds discussie ontstaat. Het feit dat de colofon van ons blad weer in het nederlands is gesteld, is een opdracht van het bestuur. Het is de uitkomst van een discussie die ik zie als een tijdelijkheid, een stuipreiking van de weerstand die elke verandering met zich brengt.

van Pelt

SELF-MODIFYING CODE
ON TRIAL

Scene: It is a dark and gloomy night. The wind is howling through the desolate, rain-whipped streets. One lonesome light is flickering behind a window. An engrossed 'kenner is tampering with his Junior, trying his new, outstanding algorithm.

Defensor: "What's wrong with SMC?"

Prosecutor: "SMC is un-necessary, un-maintainable, hard to understand and can not be used in prom's. It should be forbidden!"

Defensor: "But SMC can sometimes shorten a program with several bytes."

Prosecutor: "When the algorithm is carefully planned, it is hardly never the case, where a few bytes more or less is a crucial obstacle."

Defensor: "SMC can make the routine faster."

Prosecutor: "As I have said before, if the algorithm is carefully planned, then don't even think of the extra instructions, that are run only once. The gain, obtained by screwing up the program, is infinitesimal."

Defensor: "SMC can be allowed, if the documentation is very extensive."

Prosecutor: "One might think, his comments are adequate, while in fact others find them incomprehensible, and if the source is not at hand, then the code is 'chinese. Even if documentation is all right, then SMC is very difficult to maintain."

Defensor: "Not all programs will be used in prom's."

Prosecutor: "Well, that's right. But who can tell that beforehand?"

"SMC is not necessary. See the last issues of 'de kenner'. There has been some bad examples of SMC, f.ex. mr Rasmussen's 'Screen-dump' in nr. 49 and mr. Tietsch's 'Copier' in nr 43-44. Take the latter f.ex. Why is he using SMC in line 4060, 4970? He could just as easily have used a flag outside the program, f.ex. along with the other temporarays. In mr. Rasmussen's case it's even worse. Here whole parts of the algorithm is shifted out with several different instructions, each going their own way!"

Defensor: "Ok, but what about the human brain. It uses SMC extensively all the time?"

Prosecutor: "Yeah, but here we are dealing with simple serial machines. Maybe in future technology things will change in that direction."

Judge: "The verdict is: SMC is forbidden, unless it is the absolutely only way to solve a problem, and then it should be heavy documented."

What is your opinion...??

Leif Rasmussen, Denmark.

```

0010 0010 *****
0020 0020 * APPLE II DOS 3.3 FAST LOAD PATCH *
0030 0030 * By: Ruud Uphoff 1987 *
0040 0040 *****
0050
0060 ;--Change definition of label "RAM" below into 16,32 or 48 according to the
0070 ; the space of memory available in your APPLE II
0080
0090 ;--Change definition of label "MASK" below according to the the type files
0100 ; that may be "fast loaded" by setting the bit for that type.
0110
0120 0120 RAM DEF 16
0130 0130 MASK DEF %00000111
0140 0140 IGN. | | | | | | | | Allow fast load of I(nteger) basic file.
0150 0150 IGN. | | | | | | | | Allow fast load of A(pplesoft) basic file.
0160 0160 IGN. | | | | | | | | Allow fast load of B(inary) file.
0170 0170 IGN. | | | | | | | | Ignore fast load of S-type file.
0180 0180 IGN. | | | | | | | | Ignore fast load of R-type file.
0190 0190 IGN. | | | | | | | | Ignore fast load of new A-type file.
0200 0200 IGN. | | | | | | | | Ignore fast load of new B-type file.
0210 0210 IGN. | | | | | | | | Always set "lock bit" to zero.
0220
0230 0230 SIZE DEF RAM*1024-$4000 ;Assembler offset according MASTER/SLAVE version
0240 0240 FREEMEM DEF $36B3+SIZE ;We have a lot of space in the BOOT 2 page
0250
0260 0260 ORI FREEMEM
0270 0270 OUT "SPUP" FREEMEM
0280 0280 STO
0290
0300 ;Parameters in the DOS file manager parameter list:
0310
0320 0320 RNLLEN DEF $35C1+SIZE ;Length of range to be read
0330 0330 MEMTGT DEF $35C3+SIZE ;Memory target address
0340 0340 DATBUF DEF $35CB+SIZE ;Address of sector data buffer
0350
0360 ;Variables in the DOS file manager work area:
0370
0380 0380 RELSEC DEF $35E4+SIZE ;Relative sector number in the file
0390 0390 BYTOFF DEF $35E6+SIZE ;Byte offset in current sector
0400 0400 FILTYP DEF $35F6+SIZE ;Current file type
0410
0420 ;File manager routines
0430
0440 0440 DECLEN DEF $31B5+SIZE ;Decrement length of file
0450 0450 RDNSECT DEF $30B6+SIZE ;Reload data sector buffer
0460
0470 ;Patch to speed up file manager action: "read a range of bytes"
0480
0490 0490 PATCH LDA FILTYP ;Get current file type.
0500 AND #MASK ;If not allowed on current type,
0510 BEQ EXIT ;then ignore patch.
0520 LDA RNLLEN+1 ;If length of file less than 1 sector,
0530 BEQ EXIT ;then also ignore patch.
0540 LDA BYTOFF ;If byte offset in sector not zero,
0550 BNE EXIT ;then no patch for first sector yet.
0560 LDA DATBUF ;Save address of data buffer.
0570 PHA
0580 LDA DATBUF+1
0590 PHA
0600 0600 NXTSECT LDA MEMTGT ;Use memory target,
0610 STA DATBUF ;as data buffer.
0620 LDA MEMTGT+1
0630 STA DATBUF+1
0640 JSR RDNSECT ;Read next sector in memory.
0650 INC RELSEC ;Increment sector count.
0660 BNE +=5
0670 INC RELSEC+1
0680 INC MEMTGT+1 ;Add 256 to memory target.
0690 DEC RNLLEN+1 ;Subtract 256 from length.
0700 BNE NXTSECT ;If more full sectors go repeat.
0710 PLA ;Else, restore buffer address.
0720 STA DATBUF+1
0730 PLA
0740 STA DATBUF
0750 0750 EXIT JMP DECLEN ;Exit patch for read of partial sector.

```

0760
0770
0780
0790
0800
0810
0820
0830
0840
0850
0860
0870
0880
0890
0900
0910
0920
0930
0940
0950
0960
0970
0980
0990
1000
1010
1020
1030
1040
1050
1060
1070
1080
1090
1100
1110
1120
1130
1140
1150
1160
1170
1180
1190
1200

PAGE

Speed increment is the same as when using "QUICK DOS". The following test may show the increase of speed:

Type: BSAVE LONGFILE,A#1000,L#7FFF

Reload by: BLOAD LONGFILE

DOS 3.3 (no patch) 33 seconds.

DOS 3.3 (patched) 7.5 seconds.

QUICK DOS Load error without message, so drop QDOS !

COMPATIBILITY

No restrictions. You are still using DOS 3.3. It just LOAD's faster.

HOW TO MAKE THE PATCH

Assemble the patch or type in the hex code and save this code on disk. Now use COPYA (on your DOS 3.3 master disk) to get a backup of your master disk. There are two method's to work in:

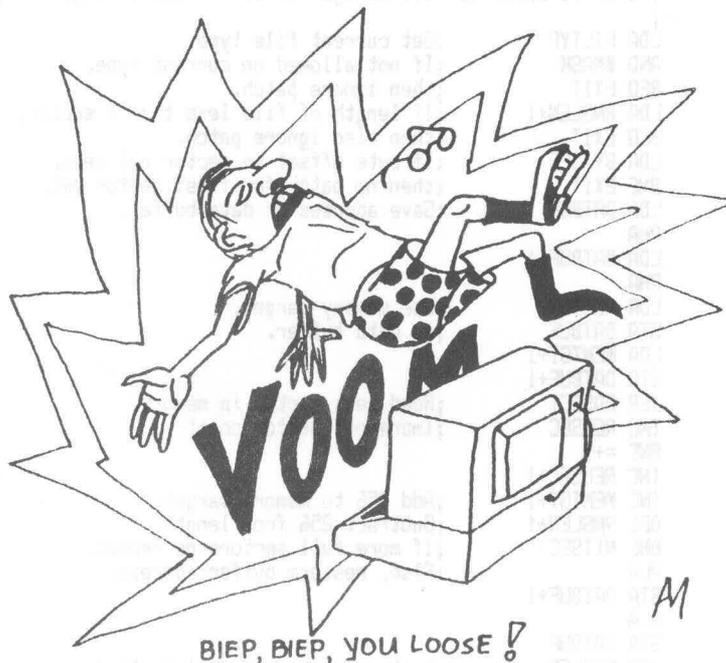
1. USING A DISK MONITOR

- a. Load track 0, sector 0 and type in the code at byte \$B3 and up.
- b. Rewrite track 0, sector 0.
- c. Load track 1, sector B and change the JSR at byte \$96 into a call of the patch: 20 B3 36 (JSR \$36B3)
- d. Rewrite track 1, sector B.

2. USING MASTER CREATE (See: "BENEATH APPLE DOS" by: Don Worth and Peter lechner. 1981)

- a. Insert the backup and type BLOAD MASTER CREATE. Don't remove the diskette.
- b. Type CALL-151 to get into the monitor.
- c. Type 80D: 4C N 8006. MASTER CREATE will load the MASTER IMAGE, and then return to the monitor.
- d. Type in the code for the patch at \$12B3 and up.
- e. Type 2D96: 20 B3 36 to call the patch.
- f. Continue execution of MASTER CREATE by typing 8D26. The program will ask you for the name of the greeting program as usual. Type in that name and re-master your copy into a new (patched) DOS 3.3 master disk.

36F6 36F6 00



JUNIOR'S MICRO-ADE ASSEMBLER FILE: TOTAPE 050187

 * FAST ALTERNATIVE FOR THE SLOW JUNIOR *
 * CASSETTE INTERFACE AND SOFTWARE *

By: Hans Christen, The Netherlands.
 Translated by: Frank Vandekerckhove, Belgium.

Reflections:

1. Waiting for more than 5 min. to read a Basic program (ca. 16K) from tape is getting to long for me.
2. Hardware adjustment is critical and changeable and that is why I am used to read tapes with a screwdriver in my hand, sometimes it takes me more than 10 min. to get that program.
3. I have no money to buy a diskdrive, even not in the future.

Therefore I developed a read/write program based on the hard- and software of "Basicode-2" as described in Elektor Oct. 1983 (Dutch edition).

In spite of all (eg. Elektor's) considerations about speed and reliability of using a low-cost cassette recorder and a normal tape (TDK.D) you can get a lot more out of it. The results are better than my expectations: 6300 bits/s (or 8K in 15 s.). Faster would be possible but, as an extra control, 8 bits are followed by a paritybit and, if necessary, a parity error. A checksum at the end is added in case several biterrors would give a right paritybit, altogether it makes 4350 databits/s.

All is working perfectly but it is to early to know much about the influence of old tapes; therefore it is always good to keep a copy aside.

The subroutines are easy to incorporate by changing the jumps into the monitorsoftware: \$0B02 → \$905F and \$090F → \$9000.

Hardware modifications:

- replace R 9 (10K) by a wire
- I use CA2 instead of CA1

Last important note: My microprocessor is running at 2 MHz. Reduce frequencies when your micro is running at 1 MHz; the baudrate will be less. You can read more about this at the beginning of the subroutine "READ".

```

0740: 2000      TOTAPE ORG      $2000
0750:
0760:          VIA-ADDRESSES
0770:
0790:          04 18  TOCL  *      $1804  TIMER 1 COUNTER
0800:          05 18  TOCH  *      $1805
0810:          06 18  TOLL  *      $1806  TIMER 1 LATCH
0820:          07 18  TOLH  *      $1807
0830:          0B 18  ACR   *      $180B
0840:          0C 18  PCR   *      $180C
0850:          0D 18  IFR   *      $180D
0860:          0E 18  IER   *      $180E
0870:
0880:          PIA-ADDRESSES
0890:
0900:          80 1A  PAD   *      $1A80
0910:          81 1A  PADD  *      $1A81
0920:          82 1A  PBD   *      $1A82
0930:          83 1A  PBDD  *      $1A83
0940:          F4 1A  CNTA  *      $1AF4  CNT 1T
0950:
0960:          POINTERS & TEMPS
0970:
0980:          FA 00  POINTL *      $00FA
0990:          FB 00  POINTH *      $00FB
1000:          6A 1A  BTSUML *      $1A6A  BYTE CNT
1010:          6B 1A  BTSUMH *      $1A6B
1020:          6C 1A  NULL   *      $1A6C  PERIODTIME
1030:          6E 1A  CHKSUM *      $1A6E
1040:          6F 1A  PARITY *      $1A6F
1050:          70 1A  SAL    *      $1A70  STARTADDRESS
1060:          71 1A  SAH    *      $1A71
1070:          72 1A  EAL    *      $1A72  ENDADDRESS
1080:          73 1A  EAH    *      $1A73
1090:          74 1A  HDRCTH *      $1A74  CNT 7200 Hz
1100:          79 1A  ID     *      $1A79  IDENTIFICATION
1110:
1120:          03 F0  PRCHA  *      $F003  PRINT CHAR ROUT
1130:          8F 12  PRBYT  *      $128F  PRINT BYTE
1140:          EB 11  CRLF   *      $11EB  CR & LF
1150:
1160:          *****
1170:          * WRITE *
1180:          *****
1190:
1200: 2000 48      WRITE PHA      SAVE A, Y & X
1210: 2001 98      TYA
    
```

```

1220: 2002 48
1230: 2003 8A
1240: 2004 48
1250: 2005 A9 47
1260: 2007 8D 82 1A
1270: 200A A9 FF
1280: 200C 8D 83 1A
1290: 200F A9 00
1300: 2011 8D 6E 1A
1310: 2014 AD 70 1A
1320: 2017 85 FA
1330: 2019 AD 71 1A
1340: 201C 85 FB
1350: 201E A9 7F
1360: 2020 8D 0E 18
1370: 2023 A9 C0
1380: 2025 8D 0B 18
1390: 2028 A9 01
1400: 202A 8D 05 18
1410: 202D 20 BF 20
1420: 2030 20 CD 20
1430: 2033 A9 77
1440: 2035 20 97 20
1450: 2038 AD 79 1A
1460: 203B 20 97 20
1470: 203E AD 70 1A
1480: 2041 20 97 20
1490: 2044 AD 71 1A
1500: 2047 20 97 20
1510: 204A AD 72 1A
1520: 204D 20 97 20
1530: 2050 AD 73 1A
1540: 2053 20 97 20
1550: 2056 A0 00
1560: 2058 B1 FA
1570: 205A 48
1580: 205B 4D 6E 1A
1590: 205E 8D 6E 1A
1600: 2061 68
1610: 2062 20 97 20
1620: 2065 E6 FA
1630: 2067 D0 02
1640: 2069 E6 FB
1650: 206B A5 FB
1660: 206D CD 73 1A
1670: 2070 D0 E4
1680: 2072 A5 FA
1690: 2074 CD 72 1A
1700: 2077 D0 DD
1710: 2079 AD 6E 1A
1720: 207C 20 97 20
1730: 207F 20 DA 20
1740: 2082 A9 1E
1750: 2084 8D 83 1A
1760: 2087 A9 00
1770: 2089 8D 0B 18
1780: 208C A0 13
1790: 208E 20 4B 22
1800: 2091 68
1810: 2092 AA
1820: 2093 68
1830: 2094 A8
1840: 2095 68
1850: 2096 60
1860: 2097 A0 00
1870: 2099 8C 6F 1A
1880: 209C A0 08
1890: 209E 4A
1900: 209F 48
1910: 20A0 B0 05
1920: 20A2 20 CD 20
1930: 20A5 70 06
1940: 20A7 20 DA 20
1950: 20AA EE 6F 1A
1960: 20AD 68
1970: 20AE 88
1980: 20AF D0 ED
1990: 20B1 4E 6F 1A
2000: 20B4 B0 05
2010: 20B6 20 CD 20
2020: 20B9 70 03
2030: 20BB 20 DA 20
2040: 20BE 60
2050: 20BF A2 70
2060: 20C1 A0 45
2070: 20C3 20 DA 20
2080: 20C6 CA
2090: 20C7 D0 FA
2100: 20C9 88
2110: 20CA D0 F7
2120: 20CC 60
2130:
2140: 20CD A9 58
2150: 20CF 8D 06 18
2160: 20D2 A9 00
2170: 20D4 8D 07 18
2180: 20D7 4C E4 20
2190: 20DA A9 45
    
```

```

PHA
TXA
PHA
LDAIM $47      SWITCH RELAIS ON
STA PBD        RED LED ON
LDAIM $FF      PIA PB OUTPUT
STA PBDD
LDAIM $00      RESET CHECKSUM
STA CHKSUM
LDA SAL        SET POINTER
STAZ POINTL
LDA SAH
STAZ POINTH
LDAIM $7F      DISABLE INTERRUPT
STA IER
LDAIM $C0      VIA PB7 SQUARE
STA ACR        OUTPUT
LDAIM $01      START TIMER 1
STA TOCH
JSR HEADER     7200 Hz REF.
JSR ZERO      STARTBIT
LDAIM $77      STARTBYTE
JSR OUTBYT
LDA ID         ID TO TAPE
JSR OUTBYT
LDA SAL        STARTADDRESS TO
JSR OUTBYT    TAPE
LDA SAH
JSR OUTBYT
LDA EAL        ENDADDRESS TO
JSR OUTBYT    TAPE
LDA EAH
JSR OUTBYT
DATATR LDYIM $00  DATA TO TAPE
LDAIY POINTL
PHA
EOR CHKSUM    ADJUST CHECKSUM
STA CHKSUM
PLA
JSR OUTBYT
INCZ POINTL   NEXT ADDRESS
BNE TEST
INCZ POINTH
LDAZ POINTH   END ?
CMP EAH
BNE DATATR
LDAZ POINTL
CMP EAL
BNE DATATR
LDA CHKSUM    CHECKSUM TO TAPE
JSR OUTBYT
JSR ONE       ENDBIT
LDAIM $1E     RED LED & MOTOR
STA PBDD     OFF
LDAIM $00     RESTORE VIA PB7
STA ACR      MODE
LDYIM $13     PRINT MESSAGE
JSR MESSY
PLA
RESTORE X,Y & A
TAX
PLA
TAY
PLA
RTS
OUTBYT LDYIM $00  RESET PARITY
STY PARITY
LDYIM $08     Y-REG IS BITCNT
OUTBIT LSR      SHIFT BIT IN CARRY
PHA          SAVE NEW ACCU
BCS HIGH     IF C=1 : 7200 Hz
JSR ZERO     IF C=0 : 5600 Hz
BVS NEXT    BRANCH ALWAYS
JSR ONE      1 PERIOD OF 7200
INC PARITY   INCREASE PARITY
PLA          RESTORE ACCU
DEY          DECREASE BITCNT
BNE OUTBIT  IF BITCNT ==0
THEN NEXT BIT
SHIFT BIT 0 IN CARRY
IF C=0 THEN WRITE 0
BRANCH ALWAYS
IF C=1 THEN WRITE 1
START 7200 Hz
BCS ONEVEN  IF C=0 THEN
ZERO        WRITE 0
BVS RTN     BRANCH ALWAYS
JSR ONE     IF C=1 THEN
RTS         WRITE 1
LDXIM $70   START 7200 Hz
LDYIM $45
JSR ONE
DEX
BNE HDR
DEY
BNE HDR
RTS
ZERO LDAIM $58  SET TIMER 1 TO
STA TOLL     88 MICROSEC
LDAIM $00
STA TOLH
JMP PER.O
ONE LDAIM $45  SET TIMER 1 TO
    
```

2200: 20DC 8D 06 18	STA	TOLL	69 MICROSEC.	3170: 21A7 DO DC	BNE	RDDATA	
2210: 20DF A9 00	LDAIM	\$00		3180: 21A9 AD 73 1A	LDA	EAH	
2220: 20E1 8D 07 18	STA	TOLH		3190: 21AC C5 FB	CMP	POINTH	
2230: 20E4 AD 04 18	PERIO	LDA	TOCL	3200: 21AE DO D5	BNE	RDDATA	
2240: 20E7 2C OD 18	HLFPP1	BIT	IFR	3210: 21B0 20 FA 21	JSR	RBYT	GET CHECKSUM
2250: 20EA 50 FB	BVC	HLFPP1		3220: 21B3 CD 6E 1A	CMP	CHKSUM	FROM TAPE
2260: 20EC AD 04 18	LDA	TOCL	RESET TIMER 1	3230: 21B6 FO 08	BEQ	NERROR	
			INTERRUPTFLAG	3240: 21B8 A0 00	ERROR	LDYIM	\$00
			1ST HALF PERIOD	3250: 21BA 20 4B 22	JSR	MESSY	DISPLAY CHECKSUM
2270: 20EF 2C OD 18	HLFPP2	BIT	IFR	3260: 21BD 4C D6 21	JMP	RETURN	ERROR
2280: 20F2 50 FB	BVC	HLFPP2		3270: 21C0 A0 22	NERROR	LDYIM	\$22
2290: 20F4 60	RTS		RESET TIMER 1				DISPLAY DATA
2295:			INTERRUPTFLAG				LOADED
2300:	*****		2ND HALF PERIOD	3280: 21C2 20 4B 22	JSR	MESSY	DISPLAY NUMBER
2310:	* READ *						OF LOADED BYTES
2320:	*****			3290: 21C5 AD 6B 1A	LDA	BTSUMH	
2330:				3300: 21C8 20 8F 12	JSR	PRBYT	
2340:				3310: 21CB AD 6A 1A	LDA	BTSUML	
2350:				3320: 21CE 20 8F 12	JSR	PRBYT	
2360:				3330: 21D1 A0 49	LDYIM	\$49	
2370:				3340: 21D3 20 4B 22	JSR	MESSY	
2380:				3350: 21D6 A9 06	RETURN	LDAIM	\$06
2390:				3360: 21D8 8D 82 1A	STA	PBD	GREEN LED &
2420: 20F5 48	READ	PHA	SAVE A,Y & X	3370: 21DB A9 1E	LDAIM	\$1E	MOTOR OFF
2430: 20F6 98		TYA		3380: 21DD 8D 83 1A	STA	PBDD	
2440: 20F7 48		PHA		3390: 21E0 68	PLA		RESTORE X,Y & A
2450: 20F8 8A		TXA		3400: 21E1 AA	TAX		
2460: 20F9 48		PHA		3410: 21E2 68	PLA		
2470: 20FA A9 7E		LDAIM	\$7E	3420: 21E3 A8	TAY		
2480: 20FC 8D 83 1A		STA	PBDD	3430: 21E4 68	PLA		
2490: 20FF A9 32		LDAIM	\$32	3440: 21E5 60	RTS		
2500: 2101 8D 82 1A		STA	PBD	3450:			
2510: 2104 A9 7F		LDAIM	\$7F	3460: 21E6 A9 01	PERIOD	LDAIM	\$01
2520: 2106 8D 81 1A		STA	PADD	3470: 21E8 2C OD 18	HLF	BIT	IFR
2530: 2109 8D 0E 18		STA	IER	3480: 21EB FO FB	BEQ	HLF	
2540: 210C A9 00		LDAIM	\$00	3490: 21ED 8D OD 18	STA	IFR	
2550: 210E 8D 0C 18		STA	PCR	3500: 21F0 A9 FF	LDAIM	\$FF	
2560: 2111 8D 0E 1A		STA	CHKSUM	3510: 21F2 AA	TAX		
2570: 2114 8D 6A 1A		STA	BTSUML	3520: 21F3 4D F4 1A	EOR	CNTA	THE TIME BETWEEN
2580: 2117 8D 6B 1A		STA	BTSUMH				2 EDGES IN ACCU
2590: 211A A9 0A	LEADER	LDAIM	\$0A	3530: 21F6 8E F4 1A	STX	CNTA	RESET TIMER
2600: 211C 8D 74 1A		STA	HDRCTH	3540: 21F9 60	RTS		
2610: 211F A9 36		LDAIM	\$36	3550:			
2620: 2121 20 3F 22		JSR	NOSYNC	3560: 21FA A9 55	RBYT	LDAIM	\$55
2630: 2124 20 E6 21	LDR	JSR	PERIOD	3570: 21FC 20 3F 22	JSR	NOSYNC	DISPLAY READ
2640: 2127 C9 80		CMPIM	\$80	3580: 21FF A9 00	LDAIM	\$00	BYTE SITUATION
2650: 2129 90 EF		BCC	LEADER	3590: 2201 8D 6F 1A	STA	PARITY	RESET PARITY
2660: 212B C9 9D		CMPIM	\$9D	3600: 2204 A0 08	LDYIM	\$08	\$08 IN BITCNT
2670: 212D B0 EB		BCS	LEADER	3610: 2206 48	PHA		SAVE ACCU
2680: 212F C8		INY		3620: 2207 20 E6 21	JSR	PERIOD	GET PERIODTIME
2690: 2130 DO F2		BNE	LDR	3630: 220A CD 6C 1A	CMP	NULL	5600 Hz ?
2700: 2132 20 3D 22		JSR	SYNC	3640: 220D B0 06	BCS	FNDZRO	THEN BIT=0
2710: 2135 CE 74 1A		DEC	HDRCTH	3650: 220F EE 6F 1A	INC	PARITY	IF BIT=1 THEN
2720: 2138 DO EA		BNE	LDR				INCREASE PARITY
2730: 213A A9 9D		LDAIM	\$9D	3660: 2212 38	SEC		SET CARRY
2740: 213C 8D 6C 1A		STA	NULL	3670: 2213 B0 01	BCS	SHIFT	BRANCH ALWAYS
				3680: 2215 18	FNDZRO	CLC	CLEAR CARRY
				3690: 2216 68	SHIFT	PLA	GET ACCU
2750: 213F 20 3D 22	START	JSR	SYNC	3700: 2217 6A	RORA		SHIFT CARRY IN
2760: 2142 20 E6 21		JSR	PERIOD	3710: 2218 88	DEY		DECREASE BITCNT
2770: 2145 CD 6C 1A		CMP	NUL	3720: 2219 DO EB	BNE	RB	IF BIT=0 THEN
2780: 2148 90 F5		BCC	START	3730: 221B 48	PHA		NEXT BIT
2790: 214A 20 FA 21		JSR	RBYT	3740: 221C 20 E6 21	JSR	PERIOD	GET PARITY BIT
2800: 214D C9 77		CMPIM	\$77	3750: 221F CD 6C 1A	CMP	NULL	IF PARITY BIT=0
2810: 214F DO C9		BNE	LEADER	3760: 2222 B0 07	BCS	EVEN	TEST ON EVEN
				3770: 2224 4E 6F 1A	LSR	PARITY	SHIFT PARITY IN
							CARRY
2820: 2151 20 FA 21		JSR	RBYT	3780: 2227 90 09	BCC	ERR	IF BIT=0 THEN
2830: 2154 CD 79 1A		CMP	ID				PARITY ERROR
2840: 2157 FO 10		BEQ	FND	3790: 2229 68	PLA		GET BYTE
2850: 2159 48		PHA		3800: 222A 60	RTS		
2860: 215A A0 43		LDYIM	\$43	3810: 222B 4E 6F 1A	EVEN	LSR	PARITY
2870: 215C 20 4B 22		JSR	MESSY				SHIFT PARITY IN
2880: 215F 68		PLA		3820: 222E B0 02	BCS	ERR	IF BIT=1 THEN
2890: 2160 20 8F 12		JSR	PRBYT				PARITY ERROR
2900: 2163 20 E8 11		JSR	CRLF	3830: 2230 68	PLA		GET BYTE
2910: 2166 4C 1A 21		JMP	LEADER	3840: 2231 60	RTS		
2920: 2169 20 FA 21	FND	JSR	RBYT	3850: 2232 A0 32	ERR	LDYIM	\$32
2930: 216C 8D 70 1A		STA	SAL	3860: 2234 20 4B 22	JSR	MESSY	DISPLAY PARITY
2940: 216F 85 FA		STA	POINTL	3870: 2237 68	PLA		ERROR
2950: 2171 20 FA 21		JSR	RBYT	3880: 2238 68	PLA		RESTORE STACK-
2960: 2174 8D 71 1A		JSR	SAH	3890: 2239 68	PLA		POINTER
2970: 2177 85 FB		STA	POINTH	3900: 223A 4C 1A 21	JMP	LEADER	
2980: 2179 20 FA 21		JSR	RBYT	3910:			
2990: 217C 8D 72 1A		STA	EAL	3920: 223D A9 69	SYNC	LDAIM	\$69
3000: 217F 20 FA 21		JSR	RBYT	3930: 223F 8D 80 1A	NOSYNC	STA	PAD
3010: 2182 8D 73 1A		STA	EAH	3940: 2242 AD 82 1A	LDA	PBD	DISPLAY NOSYNC
3020: 2185 20 FA 21	RDDATA	JSR	RBYT	3950: 2245 49 02	EORIM	\$02	CHANGE DISPLAY
3030: 2188 48		PHA		3960: 2247 8D 82 1A	STA	PBD	
3040: 2189 4D 6E 1A		EOR	CHKSUM	3970: 224A 60	RTS		
3050: 218C 8D 6E 1A		STA	CHKSUM	3980:			
3060: 218F 68		PLA		3990: 224B B9 5A 22	MESSY	LDAAY	MESS
3070: 2190 EE 6A 1A		INC	BTSUML	4000: 224E C9 03	CMPIM	\$03	GET FROM TABLE
3080: 2193 DO 03		BNE	STORE	4010: 2250 FO 07	BEQ	MESEND	ETX ?
3090: 2195 EE 6B 1A		INC	BTSUMH	4020: 2252 20 03 FO	JSR	PRCHA	THE END
3100: 2198 A0 00	STORE	LDYIM	\$00	4030: 2255 C8	INY		PRINT CHARACTER
3110: 219A 91 FA		STAYI	POINTL	4040: 2256 4C 4B 22	JMP	MESSY	INCREASE INDEX
3120: 219C E6 FA		INCZ	POINTL	4050: 2259 60	MESEND	RTS	NEXT CHARACTER
3130: 219E DO 02		BNE	END?	4060:			
3140: 21A0 E6 FB		INCZ	POINTH	4070: 225A OD	MESS	=	\$0D
3150: 21A2 AD 72 1A	END?	LDA	EAL	4080: 225B OA	=	\$0A	Y=\$00
3160: 21A5 C5 FA		CMP	POINTL				

```

4090: 225C 43 = 'C
4100: 225D 48 = 'H
4110: 225E 45 = 'E
4120: 225F 43 = 'C
4130: 2260 4B = 'K
4140: 2261 53 = 'S
4150: 2262 55 = 'U
4160: 2263 4D = 'M
4170: 2264 20 = 'I
4180: 2265 45 = 'E
4190: 2266 52 = 'R
4200: 2267 52 = 'R
4210: 2268 4F = 'O
4220: 2269 52 = 'R
4230: 226A 0A = $OA
4240: 226B 0D = $OD
4250: 226C 03 = $O3
4260: 226D 0A = $OA
4270: 226E 0D = $OD
4280: 226F 44 = 'D Y=13
4290: 2270 41 = 'A
4300: 2271 54 = 'T
4310: 2272 41 = 'A
4320: 2273 20 = 'I
4330: 2274 53 = 'S
4340: 2275 41 = 'A
4350: 2276 56 = 'V
4360: 2277 45 = 'E
4370: 2278 44 = 'D
4380: 2279 0A = $OA
4390: 227A 0D = $OD
4400: 227B 03 = $O3
4410: 227C 0D = $OD
4420: 227D 0A = $OA
4430: 227E 44 = 'D Y=22
4440: 227F 41 = 'A
4450: 2280 54 = 'T
4460: 2281 41 = 'A
4470: 2282 20 = 'I
4480: 2283 4C = 'L
    
```

```

4490: 2284 4F = 'O
4500: 2285 41 = 'A
4510: 2286 44 = 'D
4520: 2287 45 = 'E
4530: 2288 44 = 'D
4540: 2289 3A = ':'
4550: 228A 20 = 'I
4560: 228B 03 = $O3
4570: 228C 0D = $OD
4580: 228D 0A = $OA
4590: 228E 50 = 'P Y=32
4600: 228F 41 = 'A
4610: 2290 52 = 'R
4620: 2291 49 = 'I
4630: 2292 54 = 'T
4640: 2293 59 = 'Y
4650: 2294 20 = 'I
4660: 2295 45 = 'E
4670: 2296 52 = 'R
4680: 2297 52 = 'R
4690: 2298 4F = 'O
4700: 2299 52 = 'R
4710: 229A 0D = $OD
4720: 229B 0A = $OA
4730: 229C 03 = $O3
4740: 229D 0D = $OD Y=43
4750: 229E 0A = $OA
4760: 229F 49 = 'I
4770: 22A0 44 = 'D
4780: 22A1 3D = '='
4790: 22A2 03 = $O3
4800: 22A3 20 = 'I Y=49
4810: 22A4 42 = 'B
4820: 22A5 59 = 'Y
4830: 22A6 54 = 'T
4840: 22A7 45 = 'E
4850: 22A8 53 = 'S
4860: 22A9 0D = $OD
4870: 22AA 0A = $OA
4880: 22AB 03 = $O3
    
```

Het nieuwe Poly-Automatiserings Zakboekje is uit.

PNBA's nieuwe en geheel herziene Poly-Automatiserings Zakboekje telt nu 1760 pagina's waarin alle informatie in overeenstemming is gebracht met de nieuwste ontwikkelingen. Korte en bondige definities, omschrijvingen en formules, compact gepresenteerd met behulp van schema's, tekeningen, tabellen en voorbeelden. Met een ca. 6000 woorden tellend trefwoordenregister en handige zoektabs vindt u snel de weg. De prijs bedraagt f 96,50. Het Poly-Automatiserings Zakboekje is verkrijgbaar in elke boekhandel, eventueel door te bestellen met opgave van nummer ISBN 90 6228 086 2.

Who has information on the Commodore 1515 printer?

(VIC20, C16, C64)

I was recently able to get one for a very small amount of money. However, where do I get hardware information to satisfy my curiosity?

Fred Behringer, Strassbergerstr. 9c/519, D-8000 München 40

DATBAS VOOR OCTOPUS

Door: Albert v.d. Beukel, Nederland.

Ik meen dat er een paar foutjes zitten in het programma DATBAS, en wel in de regels 1570 en 1980 van het hoofdprogramma. In regel 1570 staat X=26:D\$="S":GOSUB enz., dat moet zijn X=26:D\$="SP":GOSUB enz. Zonder die P werkt de printer NIET. In regel 1980 staat PRINT#DV,R\$:RT=RT+1:IF RT=AR AND DV enz., dit moet zijn PRINT#DV,R\$:RT=RT+1:IF RT=AR AND DV enz. Dit zorgt ervoor dat bij het uitlezen van een file op het scherm de computer wacht op een ENTER voordat het volgende scherm verschijnt. Zonder deze verandering vliegen de gegevens over het scherm zonder dat je tijd hebt het te lezen.

PRIJZSWINNAARS

In de NOS-Basicode-wedstrijd hebben twee leden van onze club een prijs in de wacht gesleept:

7e prijs: Cor W. Verhagen in 't Harde voor zijn tapeteller programma voor video- en cassetterecorders die geen tijdaanduiding geven. Hij won een Star-printer.

8e prijs: Frans Verberkt uit Nijmegen (bekend van artikelen in ons blad) die een voorstel deed en een programma inleverde om Basicode-2 te laten tekenen. Hij won een Canon X-07 met bijpassende batterij printer in tas.

Beide programma's zullen t.z.t. in de editie worden gepubliceerd.



68020 Single Board Computer

A single board computer using the state-of-the-art MC68020 and an optional MC68881 floating-point co-processor, the Micro-20 offers a powerful, compact, 32-bit computing system with mainframe performance on 22x15cm, including 2 MB of high-speed DRAM, 4 serial ports, an 8-bit parallel port, a floppy disk controller, a SASI/SCSI peripheral interface for intelligent hard disk controllers and a time-of-day clock with battery backup. A 16-bit I/O expansion connector allows the use of off-the-shelf or custom interfaces to extend the board's I/O capabilities. The board supports up to 256KB of EPROM, and can be mounted directly on a standard 5 1/4" disk drive. It requires regulated +5 and +12Vdc supplies, and uses the same power connector as a 5 1/4" disk drive. Information:

1. Windrusk Micro Systems Ltd, Worstead Labs, North Walsham, Norfolk, NR28 9SA, U.K., (0692)404086
2. GMX Inc, 1337 W 37th Place, Chicago, IL 60609, USA, (312)9275510.

 * FORMAT SPACES FOR ACORN ELECTRON AND BBC *

By: Ronald van Vugt, The Netherlands.

With this program you are able to add spaces in an easy manner into an assembler program on all lines without preceding labels. The result is a better readable assembler-listing (like in the listing following this text). Start the routine by entering *SPC <number of spaces> (1-9). BE CAREFUL: the '[' and ']' characters must reside in front of the line.

Met dit programma kunt u in een assemblerprogramma op een makkelijke manier spaties toevoegen bij alle regels waar geen label voorstaat, zodat de assemblerlisting beter te lezen is (zoals in deze listing). U start de routine door *SPC <aantal spaties> (1-9) in te tikken. LET WEL: de '[' en de ']'-tekens moeten vooraan in de programmaregel staan.

```

120 osc=&70:cmp=&72:vec=&74
130 adres=&76:flag=&78:aantal_sp=&79
140 len=&80:top=&81
150 FOR pass%=0 TO 3STEP3
160 P%=&C000
161 REM Dit adres kan gewijzigd worden. Als u geen
162 REM 'TUBE' hebt, is &900 een prima plaats.
163 REM
164 REM This address may be changed. If you haven't
165 REM a 'TUBE', &900 will be fine.
170 [OPT pass%
180
190     \verander oscli vector
     \change oscli_vector
200
210     LDA &208:STA vec
220     LDA &209:STA vec+1
230     LDA #com MOD256:STA &208
240     LDA #com DIV256:STA &209
250     RTS
260
270     *SPC ?
280
290 .com STX osc:STY osc+1
300     LDA #str MOD256:STA cmp
310     LDA #str DIV256:STA cmp+1
320     LDY #1
330 .lsl LDA (osc),Y:AND #223:CMP (cmp),Y
340     BNE fls:INY:CPY #4:BNE lsl:BEQ bgn
350
360     \geen *spc
     \no *spc
370
380 .fls LDX osc:LDY osc+1:JMP (vec)
390
400     \bepaal aantal spaties
     \determine number of spaces
410
420 .bgn LDA (osc),Y:INY:CMP #&20:BEQ bgn
430     SEC:SBC #48:STA aantal_sp
440
450     \plaats PAGE in adres en reset flag
     \put PAGE into address and reset flag
460
470     LDA &1D:STA adres+1
480     LDA #0:STA adres:STA flag
490
500     \bepaal lengte van de regel en kijk of einde
     \determine length of line and check on end
510
520 .ls2 LDY #1
530     LDA (adres),Y:CMP #&FF:BNE sel:RTS
540 .sel INY:INY:LDA (adres),Y:TAX
550
560     \bepaal eerste karakter van regel na spaties
     \determine first char of line after spaces
570
580 .ls3 INY:LDA (adres),Y:CMP #ASC" ":BEQ ls3
590
600     \als flag is gezet =" de regel is een assem.
     \regel
     \if flag is set =" the line is a assem.line
610
620     LDY flag:BNE spc
630
640     \kijk of een assemblerstuk start
     \check whether an assemblerpiece starts
650
660     CMP #ASC"[" :BNE ntr
670     STA flag
680
690     \ga naar volgende regel
     \goto next line
700
710 .ntr JSR inc:JMP ls2
720
730     \bepaal of spaties toegevoegd moeten worden
    
```

```

740     \determine whether spaces have to be added
750 .spc CMP #ASC" ":BEQ ntr
760     CMP #ASC"[" :BNE vgt
770     LDA #0:STA flag:BEQ ntr
780
790     \verhoog 'lengte regel' in het programma
     \increment 'length of line' in program
800
810 .vgt LDA aantal_sp:LDY #3
820     CLC:ADC (adres),Y:STA (adres),Y:STA len
830
840     \plaats TOP in top
     \put TOP into top
850
860     LDA &12:STA top
870     LDA &13:STA top+1
880
890     \verhoog TOP
     \increment TOP
900
910     LDA aantal_sp:CLC:ADC &12:STA &12
920     LDA #0:ADC &13:STA &13
930
940     \schuifroutine om programma te verschuiven
     \shiftroutine to shift program
950
960     LDX #3:JSR inc
970 .ls4 LDY #0:LDA (top),Y
980     LDY aantal_sp:STA (top),Y
990     DEC top:LDA top:CMP #&FF:BNE esc
1000    DEC top+1
1010 .esc LDA top:CMP adres:BNE ls4
1020    LDA top+1:CMP adres+1:BNE ls4
1030    LDX #1:JSR inc
1040    DEC len:DEC len:DEC len:DEC len
1050
1060     \voeg gewenste aantal spaties toe
     \add desired number of spaces
1070
1080     LDX len:LDY #0:LDA #ASC" "
1090 .ls5 STA (adres),Y:INY
1100    CPY aantal_sp:BNE ls5:BEQ ntr
1110
1120     \verhoog adres met getal in X-reg
     \increment address with contents X-reg
1130
1140 .inc TXA:CLC:ADC adres:STA adres
1150    LDA #0:ADC adres+1:STA adres+1
1160    RTS
1170
1180     \commando om routine op te starten
     \command to start the routine
1190
1200 .str EQU$ " *SPC"
1210
1220 ]
1230 NEXT pass%
    
```

=====
SOLVED PROBLEMS ELEKTOR'S OCTOPUS/EC65 COMPUTER

By: Frank Vandekerkhove, Belgium.

- VDU-card: Some 74LS04 are unable to produce a good 16 MHz frequency. I selected a good one out of three 74LS04.
- Monitor Eprom: a. The cursor movement (in Basic) ctl-K is going wrong. Change \$EE into \$CE at \$F2ED in the Monitor Eprom (INC into DEC of \$E7D7).
 b. I changed \$16 into \$F9 at \$F7C9 in the Monitor Eprom to improve data transport to my slow and little Tandy plotter.

PROBLEMS DOS65 COMPUTER

By: Andrew Gregory, England

I think I have discovered a bug in the DOS65 copy utility. If you type: ASN U=@
 COPY -W FILENAME 0:
 it works perfectly. The problem comes when copying from a subdirectory: COPY -W F/FILENAME 0:
 ↑
 any sybdirectory
 The file is copied correctly, but then it copies track 0 sector 1 from source to destination disc. The destination disc becomes corrupt. A DIR command shows that the destination disc has assumed the name of the source disc. I could not find the bug in the source listing.

=====
FOR SALE:

ELEKTERMINAL/elekterminal, 4 pages and power supply.
 2500 BEF. Frank Vandekerkhove, Durletstraat 15, B-2018 Antwerp, Belgium. Phone: 03 - 2389509.

```

list video.h
/*****
/*
/* File: video.h
/*-----
/* Small C video control routines for IO65/DOS65 system
/* written by A.Megens spring 1987
/*
/*
*****/

cls()
{ printf("\014"); } /* formfeed */

invers()
{ printf("\033i"); } /* ESC i */

normal()
{ printf("\033n"); } /* ESC n */

grafon()
{ printf("\033F"); } /* ESC F */

grafoff()
{ printf("\033G"); } /* ESC G */

bell()
{ printf("\007"); } /* BELL */

home()
{ printf("\034"); } /* HOME */

crlf()
{ printf("\n\r"); } /* CRLF */

xcrLf(n) /* multiple CRLF */
int n;
{ while (n--) printf("\n\r"); }

/*****
/*
/* gotoxy(x,y) - cursor to coordinate x,y. The function is aborted with
/* return value 0 when x or y are out of range.
/*
/*
*****/

gotoxy(x,y)
char x,y;
{ if ((x<0)||(x>79)||(y<0)||(y>25)) return(0); /* if error return 0, abort */

/*-----
/* due to a bug in the direct cursor addressing mechanism with ^T (#14) */
/* (error in DOS65 when X or Y is 13 (CR)) here assembly code is used,
/* instead of the much simpler:
/* printf("\024%c%c",x,y);
/*-----

#asm
/* 1 set i to 0
ldc.u 0 ;get first argument (small C macro)
pha ;save on stack
ldc.u 2 ;get second argument
tay ;in Y-reg.
pla ;get stack-value
tax ;in X-reg.
jsr $F024 ;IO65 routine (goto screen coordinate X,Y)
#endasm

return(1); /* no errors, return 1 */
}

```



```
***** DOS - 65 V2.01 *****
*
* Zet printerlettermode commando voor de printer
* NMS-1431 van PHILIPS.
*
*****
```

H. A. J. Quast
 Dekemastate 15
 1275 CM Huizen N.H.
 tel. 02152-54905

Beschrijving van het printerinstelprogramma "SETPRINT"

Dit programma biedt de mogelijkheid om de printer NMS-1431 van PHILIPS een bepaalde voorinstelling te geven wat betreft:

- Letterkeuze.
- Kantlijn.
- Papierlengte.
- Opgaven printfile.

Door met dit instelprogramma te werken is het niet nodig om d.m.v. een escape-code boven in de te printen tekst aan te geven in wat voor een lettertype er geprint moet worden.

De commandoprocedure is opgezet volgens de DOS-65 stijl.

Command: SETPRINT [-HECPDLIK +m,n,o] filename

- Pica 10 cpi (default)
- H Help
- E Elite 12 cpi
- C Condensed 17 cpi
- P Proportional
- D Dubbele breedte
- L Letter Quality
- I curcief
- K spring over perforatie
- +m,n,o m = Zet linker kantlijn (default m=7)
- n = Pagina lengte in inches (default n=12)
- o = Aantal lijnen bij springen over perforatie (default o=12)
- FILE Filenaam voor de printfile

Letterkeuze.

Een aantal van de opties kunnen gecombineerd worden, welke dat zij staat in de tabel hieronder.

Lettersoort	pica	elite	condensed	double
lettertype	standaard	standaard	standaard	standaard
	prop.	prop.		prop.
	L.Q.	L.Q.		L.Q.
	Curcief	Curcief	Curcief	Curcief

Wanneer de letterkeuze niet opgegeven wordt dan is de defaultwaarde 'PICA' in de standaardmode.

Met de optie -K kan opgegeven worden of er over de perforatie van het papier heen geprint moet worden of niet. De defaultwaarde is: over de

perforatie heen springen. Het aantal lege regels wat verder gesprongen wordt is default 12. Wanneer optie -K wel gekozen wordt, kan men in de derde parameter aangeven hoeveel lege regels er gesprongen moet worden. De defaultwaarde is 0.

Kantlijn instelling.

- m Met de eerste parameter kan de linkerkantlijn worden ingesteld. De defaultwaarde voor de kantlijn is 7. Deze waarde is aan te passen door in de printercommando TABEL1 het getal (L007) aan te passen.
- n De tweede parameter stelt de bladzijdelengte in. Default is deze instelling 12 insch. De opgave moet in insches geschieden.
- o De derde parameter geeft zoals gezegd aan hoeveel lege regels er gesprongen moet worden over een perforatie of bij een nieuwe bladzijde.

Filenaam.

Als laatste kan opgegeven worden of er een file van disk uitgeprint moet worden met de huidige printerinstellingen. Wordt deze filenaam niet opgegeven dan wordt de printer wel geïntialiseerd zodat deze ingestelde waarde gebruikt kan worden voor een ander LIST programma. De file die geprint wordt mag escape-codes en printercontrolcodes bevatten. Wanneer het programma een CR uit de file leest dan genereerd deze zelf een linefeed, zodat dus voorkomen moet worden dat er in de file al LF staan. Bij mijn weten gaat het bijna altijd goed omdat de EDITOR alleen maar een CR in de file opneemt evenals in de outputfile die ontstaat bij het commando:

```
'> outputfile.lis AS filename.mac'
```

Wanneer er tekst uitgeprint moet worden met speciale printkarakters dan kan eerst het programma NMSVERTAAL gebruikt worden gevolgt door het programma SETPRINT.

voorbeeld:

```
EDIT tekst.doc
  invoeren van tekst
EX tekst.doc
NMSVERTAAL tekst.doc tekst.pri
SETPRINT -E +10 tekst.pri
```

Wanneer de optie -H wordt meegegeven met het commando geeft de computer een helpscherm weer. Hierna keert de computer terug in de commandomode op DOS niveau.

Alle printercommando-karakters staan als label gedeclareerd, zodat het gemakkelijker is om deze routine aan te passen voor een andere printer. In TABEL1 in de source staat de printer reset karakters. De resetprocedure bestaat uit de volgende handelingen:

```
ESC,APP - Zet de printer terug in de standaard-instelling.
BELL    - Geef een piepje dat de printer klaar is.
CAN     - Reset het printerbuffer.
ESC,'L007' - Zet de linker kantlijn op de positie 7.
```

```
***** D O S - 6 5 - V2.01 *****
*
* Set printercharactermode command for the printer
* NMS-1431 of PHILIPS.
*
*****
```

```
H.A.J. Quast
Dekemastate 15
1275 CM Huizen N.H.
tel. 02152-54905
```

Command: SETPRINT [-HECPDLIK +m,n,o] FILE

Page zero address

00A0	ORG	\$00A0	
OA0	OPTMASK RES	1	Option mask
OA1	POINTER RES	2	Commandbuffer pointer
OA3	LEFTMAR RES	2	Pointer for left margin
OA5	PAGELN RES	2	Pointer for page length
OA7	JUMPPER RES	2	Number of jump by skip
OA9	TEMPX RES	1	Save address X-reg.
OAA	FILEIN RES	1	Inputfilenumber

System subroutine's

F006	OUTDEV EQU	\$F006	Outputdevice
F00C	INITDEV EQU	\$F00C	Init. outputdevice
C03B	PRINT1 EQU	\$C03B	Print text on screen
C056	REDRIN EQU	\$C056	Redirect input
C068	SOPT1 EQU	\$C068	Get option
C06B	SPAR1 EQU	\$C06B	Get parameter
D003	SREAD1 EQU	\$D003	Single read out file
D0B7	ERMES1 EQU	\$D0B7	Error messages

Mask for option-code

0080	HELP EQU	%10000000	
0040	ELITE EQU	%01000000	
0020	CONDE EQU	%00100000	
0010	PROPO EQU	%00010000	
0008	DOUBL EQU	%00001000	
0004	LETQU EQU	%00000100	
0002	ITALI EQU	%00000010	
0001	PERFO EQU	%00000001	
0045	C. ELITE EQU	'E	Printercode elite caractertype
0051	C. CONDE EQU	'Q	Printercode condensed caractertype
0050	C. PROPO EQU	'P	Printercode proportional caractertype
000E	C. DOUBL EQU	\$0E	Printercode double caractertype
0021	C. LETQU EQU	'!	Printercode letter Quality
0043	C. ITAL1 EQU	'C	Printercode1 italic
0049	C. ITAL2 EQU	'I	Printercode2 italic
0002	PRINTER EQU	\$02	centronics printerdevices
0007	BELL EQU	\$07	Bell-code
001B	ESC EQU	\$1B	ESC-code
0040	APP EQU	\$40	@-code reset printer
0018	CAN EQU	\$18	Clears printerbuffer
004C	LMARGIN EQU	'L	Left margin-code
004F	PAGINC1 EQU	'O	Printcode1 page length
0049	PAGINC2 EQU	'I	Printcode2 page length
004F	SKIPC1 EQU	'O	Skip-code 1
0053	SKIPC2 EQU	'S	Skip-code 2

```

-----
Mainprogram
-----
1000      ORG      $1000

: Initialisatie

1000 20 68C0  SETPRINT JSR      SOPT1      Get option
1003 48      FCC      'H'          Help
1004 45      FCC      'E'          Elite 12 cpi
1005 43      FCC      'C'          Condensed 17 cpi
1006 50      FCC      'P'          Proportional
1007 44      FCC      'D'          Double
1008 4C      FCC      'L'          Letter Quality
1009 49      FCC      'I'          Italic
100A 4B      FCC      'K'          Perforation skip
100B 00      FCB      $0           Option tabel end
100C 90 03   BCC      1.f          Branch if option are correct
100E 4C FA10 JMP      ERROR

1011 86 A0   1      STX      OPTMASK      Save the option
1013 A2 40   LDX      #$40          dec. mode
1015 20 6BC0 JSR      SPAR1          Load parameters
1018 A3      FCB      #LEFTMAR
1019 A5      FCB      #PAGELEN
101A A7      FCB      #JUMPPER
101B 00      FCB      $00          page zero address for parameter
101C 90 03   BCC      2.f          Branch if par. are correct
101E 4C FA10 JMP      ERROR

1021 84 A1   2      STY      POINTER      Save filepointer address
1023 85 A2   STA      POINTER+1

: Test the help command

1025 A5 A0   TSHELP  LDA      OPTMASK      Load option masker
1027 29 80   AND      #HELP          Test help command
1029 F0 03   BEQ      INITPR
102B 4C 6211 JMP      INFO

: Initialisation printerdevice

102E A2 02   INITPR  LDX      #PRINTER      Load device number
1030 20 0CF0 JSR      INITDEV          Init. device
1033 90 19   BCC      RESET          Branch if printer ready
1035 20 3BC0 JSR      PRINT1
1038 075072696E FCC      7,'Printer not ready.\r',0
104D 60      RTS

: Reset printer
: Clears printerbuffer.
: adjust default printer mode

104E A0 00   RESET  LDY      #0
1050 B9 5911  2      LDA      TABEL1,Y      Load printer-code
1053 C0 09   CPY      #(TABEL2-TABEL1) Test end table 1
1055 F0 08   BEQ      TSCOND
1057 A2 02   LDX      #PRINTER      Load device number
1059 20 06F0 JSR      OUTDEV          Print code
105C C8      INY
105D 10 F1   BPL      2.b

: Test condensed charactertype

105F A5 A0   TSCOND LDA      OPTMASK      Load option masker
1061 29 20   AND      #CONDE          Test condensed charactertype
1063 F0 0B   BEQ      TSELITE
1065 A9 51   LDA      #C.CONDE      Load charactercode

```

Tijd 8-Apr-87 22:29 === file setprint.mac === pagina 3

1067 20 2311 JSR SETPRIN Set printer-code
 106A 20 4811 JSR TSOPT1 Test the charactertype-option
 106D 4C 8C10 JMP TSPERF

Test elite charactertype

1070 A5 A0 TSELITE LDA OPTMASK Load option masker
 1072 29 40 AND #ELITE Test elite charactertype
 1074 F0 08 BEQ TSDOUBL
 1076 A9 45 LDA #C.ELITE Load charactercode
 1078 20 2311 JSR SETPRIN Set printer
 107B 4C 8910 JMP TSLMOPT

Test double charactertype

107E A5 A0 TSDOUBL LDA OPTMASK Load option masker
 1080 29 08 AND #DOUBL Test double charactertype
 1082 F0 05 BEQ TSLMOPT
 1084 A9 0E LDA #C.DOUBL Load charactercode
 1086 20 2C11 JSR SETP1 Ascii data to printer

Test lettermode option

1089 20 3211 TSLMOPT JSR TSOPT Test the charactertype-option

Test perforation skip

108C A5 A0 TSPERF LDA OPTMASK Load option mask
 108E 29 01 AND #PERFO Test perforation-option
 1090 F0 0C BEQ TSMARTG
 1092 A9 4F LDA #SKIPC1 Load skip-code 1
 1094 20 2311 JSR SETPRIN Set printer
 1097 A9 53 LDA #SKIPC2 Load skip-code 2
 1099 A4 A7 LDY JUMPPER Load number of line by skip
 109B 20 FD10 JSR SETDATA Set printer with data

Test margin setting

109E A5 A3 TSMARTG LDA LEFTMAR Test if margin setting
 10A0 F0 0C BEQ TSLENGT
 10A2 A9 4C LDA #LMARGIN load margin-code
 10A4 20 2311 JSR SETPRIN Set printer
 10A7 A9 30 LDA #'0
 10A9 A4 A3 LDY LEFTMAR load margin setting
 10AB 20 FD10 JSR SETDATA Set printer with data

Test page length

10AE A5 A5 TSLENGT LDA PAGELEN Test if page length set
 10B0 F0 18 BEQ TSFILE
 10B2 A9 4F LDA #PAGINC1 load page length-code1
 10B4 20 2311 JSR SETPRIN Set printer
 10B7 A9 49 LDA #PAGINC2 load page length-code2
 10B9 A4 A5 LDY PAGELEN load page length setting
 10BB 20 FD10 JSR SETDATA Set printer with data
 10BE A9 4F LDA #SKIPC1 Load skip-code 1
 10C0 20 2311 JSR SETPRIN Set printer
 10C3 A9 53 LDA #SKIPC2 Load skip-code 2
 10C5 A4 A7 LDY JUMPPER Load number of line by skip
 10C7 20 FD10 JSR SETDATA Set printer with data

Test if filename is give

10CA A4 A1 TSFILE LDY POINTER Load the filepointer address
 10CC A5 A2 LDA POINTER+1
 10CE A2 81 LDX #81 file mode
 10D0 20 56C0 JSR REDRIN Redirect inputfile
 10D3 B0 1E BCS NONAME
 10D5 86 AA STX FILEIN Save inputfilenumber
 10D7 A6 AA READ LDX FILEIN Load the inputfilenumber
 10D9 20 03D0 JSR SREAD1 Single read out file

Tijd 8-Apr-87 22:29 === file setprint.mac === pagina 4

```

10DC B0 19          BCS      EXIT      Test if end of file
10DE 48             PHA          Save filedata
10DF A2 02          LDX      #PRINTER Load device number
10E1 20 06F0       JSR      OUTDEV   Print filedata
10E4 68             PLA          Restore filedata for test CR
10E5 C9 0D          CMP      #$0D
10E7 D0 EE          BNE      READ      branch if not CR
10E9 A9 0A          LDA      #$0A
10EB A2 02          LDX      #PRINTER Load device number
10ED 20 06F0       JSR      OUTDEV   print a linefeed
10F0 4C D710       JMP      READ

10F3 C9 12          NONAME CMP      #$12      Test no filename
10F5 D0 03          BNE      ERROR
10F7 60             EXIT      RTS

10F8 A9 16          ERERIT LDA      #$16      Errormessage
10FA 4C B7D0       ERROR  JMP      ERMES1    print error

;
; end main program
;
; Routine **SETDATA**
;
; This routine convert the decimal
; data into ascii data and send it
; to the printer.
; Start: Accu - data to printer
;        Y-reg. dec/ascii data
10FD 20 2C11       SETDATA JSR      SETP1    Ascii data to printer
1100 98             TYA
1101 20 0F11       JSR      DECASC    dec. to ascii
1104 86 A9          STX      TEMPX    Save ascii low nibble
1106 20 2C11       JSR      SETP1    Ascii data to printer
1109 A5 A9          LDA      TEMPX    Restore ascii low nibble
110B 20 2C11       JSR      SETP1    Ascii data to printer
110E 60             RTS

;
; Routine **DECASC**
;
; Convert decimal to ascii data
; Start: Accu decimal data
; Stop : X-reg lowbyte ascii
;        Accu highbyte ascii
110F 48             DECASC  PHA          Save databyte
1110 29 0F          AND      #$0F      Mask right nibble
1112 20 1B11       JSR      NASCII   Convert to ascii-value
1115 AA            TAX
1116 68             PLA          Restore databyte
1117 4A            LSRA      Shift right 4
1118 4A            LSRA
1119 4A            LSRA
111A 4A            LSRA
111B C9 0A          NASCII CMP      #$0A      Test op errordata
111D 10 D9          BPL      ERERIT
111F 18            CLC
1120 69 30          ADC      #'0'      Make a character of it
1122 60             RTS

;
; Routine **SETPRIN**
;
; This routine sends a esc-code
; followed bij a printer-code to
; the printer.
; Start: Accu printcode
1123 48             SETPRIN PHA          Save code

```

ljd 8-Apr-87 22:29 === file setprint.mac === pagina 5

```

124 A2 02          LDX      #PRINTER      Load device number
126 A9 1B          LDA      #ESC
128 20 06F0       JSR      OUTDEV      Print-code
12B 68            PLA
12C A2 02          SETP1  LDX      #PRINTER      Load device number
12E 20 06F0       JSR      OUTDEV      Print-code
131 60            RTS

```

;; Routine **TSOPT**

;; This routine test the option
;; for the differend charactertype

```

32 A5 A0          TSOPT  LDA      OPTMASK      Load option masker
34 29 04          AND      #LETQU      Test if letter quality
36 F0 05          BEQ      TSPROP
38 A9 21          LDA      #C.LETQU      Load charactercode
3A 20 2311       JSR      SETPRIN      Set printer-code
3D A5 A0          TSPROP LDA      OPTMASK      Load option masker
3F 29 10          AND      #PROPO      Test if propartional printen
41 F0 05          BEQ      TSOPT1
43 A9 50          LDA      #C.PROPO      Load charactercode
45 20 2311       JSR      SETPRIN      Set printer-code
48 A5 A0          TSOPT1 LDA      OPTMASK      Load option masker
4A 29 02          AND      #ITALI      Test if italic
4C F0 0A          BEQ      1.f
4E A9 43          LDA      #C.ITAL1      Load charactercode
50 20 2311       JSR      SETPRIN      Set printer-code
53 A9 49          LDA      #C.ITAL2
55 20 2C11       JSR      SETP1        Ascii data to printer
58 60            1      RTS

```

;; Printer command labels

```

59 1B4007181B    LABEL1  FCC      ESC,APP,BELL,CAN,ESC,'L007'
62              LABEL2

```

;; Screentext

```

62 20 3BC0       INFO    JSR      PRINT1      Print help-info
65 0C           FCC      '\f'
66 1B69205320   FCC      '\Ei SET - PRINTER - MODE \En\r\r'
68 4974206973   FCC      'It is possible whit this program to set the printer\r'
69 C1 616E642074 FCC      'and the character-mode.\r'
70 D9 0D         FCC      '\r'
71 DA 436F6D6D61 FCC      'Command: SETPRINT [-HECPDLIK +m,n,o] FILE\r\r'
72 05 1B69204F70 FCC      '\Ei Option label \En\r\r'
73 19 2020202020 FCC      'Pica 10 cpi (default)\r'
74 37 202D482020 FCC      ' -H Help\r'
75 44 202D452020 FCC      ' -E Elite 12 cpi\r'
76 59 202D432020 FCC      ' -C Condensed 17 cpi\r'
77 72 202D502020 FCC      ' -P Proportional\r'
78 87 202D442020 FCC      ' -D Double width\r'
79 9C 202D4C2020 FCC      ' -L Letter Quality\r'
80 B3 202D492020 FCC      ' -I Italic\r'
81 C2 202D4B2020 FCC      ' -K Perforation skip (default jump)\r'
82 EA 202B6D2C6E  FCC      ' +m,n,o m = Set left margin (default m=7)\r'
83 14 2020202020  FCC      ' n = Page length in inches (default n=12)\r'
84 46 2020202020  FCC      ' o = Number of lines by a jump (default o=12)\r'
85 7B 2046494C45  FCC      ' FILE\r'
86 9D 0D00        FCC      '\r',0
87 9F 60          RTS

```

1000 END SETPRINT

locator for DOS-Junior and Octopus 65

thor :Coen Boltjes
 anslator :Elja van der Veer

sides advantages the 6502 machine-language has over other languages, such as speed and the possibility of interrupts, it also has some drawbacks. For instance, it is impossible to write larger programmes independent of the location. By using subroutines it will be necessary to use the absolute addressing mode. If the source program is available, then there is no problem to find the programme in another location, but if there is only the object-code, then there may be some difficulties. The fact is that absolute references to locations in the programme may occur. If a programme is written between \$8000-\$8800 a JSR \$8500 may occur in it. Thus, if such a programme is loaded from \$A000 onwards, then these references must change into JSR \$A500. The same holds for all other absolute references to locations in the programme.

Of course, the programme can be run through with a disassembler, if necessary, adjusting the references by hand. But what do we have a computer for? Perhaps now we think: "I do have a relocater in the OSI Extended Relocator for this, don't I"? Yes and No. In the above mentioned case you can use the OSI relocater, but there are many instances in which a relocater is needed and for which the OSI relocater is insufficient.

Suppose you have used a few external subroutines in the programme, for example an I/O-routine. If this external routine is moved to another location, instead of the programme itself, for instance because it has been extended and needs more room, the OSI-relocater can not be used.

The relocater described below can relocate the programme as well, because external references can be updated too. Before initiating the programme, the following information is required:

```

;6502 RELOCATOR 170586 BY C.J. BOLTJES
;LAST REV. 050986
2D23= GETADR=$2D23 ;READS ADRES FROM BUF
30FE= VEC =$FE ;PLACE OF THE ADRES
2D73= STROUT=$2D73 ;PRINT ROUTINE
2CE5= BUFOFF=$2CE5 ;OFFSET INPUT BUFFER
2C9B= OSINP =$2C9B ;INPUT ROUTINE
2D5B= CKEQL =$2D5B ;TEST ON ','
30C0= CURAD =$C0
3D00= COUNT =$3D00 ;LENGTH OF PROGRAM
3D02= BEGAD =COUNT+2 ;BEGIN OF RELOC AREA
3D04= ENDAD =BEGAD+2 ;END OF RELOC AREA
3D06= RELOFF=ENDAD+2 ;RELOCATE OFFSET
3D08= SHOLD =RELOFF+2 ;TEMP STACKPOINTER
;
3A79 *=$3A79
3A79 7E3A .WORD $3A7E
3A7B 7E3C .WORD $3C7E
3A7D 01 .BYTE $01
;
3A7E A92E RELOC LDA #$2E
3A80 85E2 STA $E2
3A82 A91E LDA #$1E
3A84 85E1 STA $E1 ;SET BUFFER POINTER
3A86 AD4F2A LDA $2A4F
3A89 48 PHA
3A8A AD502A LDA $2A50
3A8D 48 PHA ;SAVE ERROR ADDRESS
3A8E BA TSX
3A8F 8E083D STX SHOLD ;SAVE STACK POINTER
3A92 20732D JSR STROUT
3A95 0A .BYTE $0A,$0A,$0A,$0D,'RELOCATOR'
3AA2 0A .BYTE $0A,$0D,$00
3AA5 20F93B JSR SETERA ;SET RETURN ADRES
3AA8 AE083D LDX SHOLD
3AAB 9A TXS ;RESTORE STACK
    
```

- the location of the area of the programme which is to be adjusted;
- the location of the area which is moved;
- the destination of the startinglocation which is to be moved.

(For the first two parameters you have to give the begin and end addresses separated by a colon. The last parameter only consist of the first address.) Then, the relocater runs through the programme which is to be changed, and examines the length of the instructions with the routine "LENAC" (Junior book 2). If this is 3, it is examined whether the operand is situated in the location to be moved, and if necessary, it is adjusted.

When using the relocater it will become clear that the adjusted programmes will not always function perfectly. This is caused by the fact that:

- data such as text, command- and jumptables can also be adjusted, because they are regarded as instructions;
- the running address may point at an operand, after the data in the location, instead of an instruction. After the data this may result in some wrong instructions. In practice these errors are not really serious and when they occur they can easily be traced back.

A larger problem is formed by placing error and interruptvectors.

This is usually done by successively loading and storing the MSB and LSM of the starting address of the routine. The address is determined by two immediate-load instructions. Hence, they will NEVER be adjusted by the relocater.

An example of placing a vector is the routine which place the DOS-errorvector (\$2A4F). The MSB is loaded in the Y register and the LSB in the accumulator, and then the routine at \$2A7D is executed. This routine can be used for properly following a programme after a DOS-error. As a code which can not be relocated can not be used in a relocater, a new routine has been devised to locate the error vector.

The routine is called just before the instruction to which the vector must be directed. By pulling the returnaddress from the stack the vector can be determined. Naturally, the address is pushed back on the stack so that a JSR can be concluded with a RTS as it should be.

```

370 3AAC 20732D RELOC1 JSR STROUT
380 3AAF 0A .BYTE $0A,$0D
390 3AB1 50 .BYTE 'PROGRAM AREA BEG,END: '
400 3ACA 00 .BYTE 00
410 3ACB 209B2C JSR OSINP ;INPUT PARAMETERS
420 3ACE A200 LDX #$00
430 3AD0 8EE52C STX BUFOFF ;RESET INPUT BUFFER
440 3AD3 20232D JSR GETADR ;READ BEGIN ADRES
450 3AD6 A5FE LDA VEC ;LOAD LSB
460 3AD8 85C0 STA CURAD ;SAVE IT
470 3ADA A5FF LDA VEC+1 ;LOAD MSB
480 3ADC 85C1 STA CURAD+1 ;
490 3ADE 205B2D JSR CKEQL ;TEST ON ','
500 3AE1 20232D JSR GETADR ;GET ENDADR
510 3AE4 38 SEC
520 3AE5 ASC0 LDA CURAD
530 3AE7 E5FE SBC VEC
540 3AE9 8D003D STA COUNT
550 3AEC A5C1 LDA CURAD+1
560 3AEE E5FF SBC VEC+1
570 3AF0 8D013D STA COUNT+1 ;IN COUNT LENGTH
580 3AF3 B0B7 BCS RELOC1 ;>ENDAD<BEGAD
590 3AF5 20F93B JSR SETERA ;SET ERROR ADRES
600 3AF8 AE083D LDX SHOLD
610 3AFB 9A TXS ;RESTORE STACK
620 3AFC 20732D RELOC2 JSR STROUT
630 3AFF 0A .BYTE $0A,$0D
640 3B01 52 .BYTE 'RELOCATED AREA BEG,END: '
650 3B1A 00 .BYTE $00
660 3B1B 209B2C JSR OSINP ;INPUT PARAMETERS
670 3B1E A200 LDX #$00
680 3B20 8EE52C STX BUFOFF ;RESET INPUTBUFFER
690 3B23 20232D JSR GETADR ;READ ADRES
700 3B26 A5FE LDA VEC
    
```

```

710 3B28 8D023D STA BEGAD ;BEGIN RELOCATED AREA
720 3B2B A5FF LDA VEC+1
730 3B2D 8D033D STA BEGAD+1
740 3B30 205B2D JSR CKEQL ;TEST ON ','
750 3B33 20232D JSR GETADR ;READ END ADRES
760 3B36 38 SEC
770 3B37 A5FE LDA VEC
780 3B39 8D043D STA ENDAD
790 3B3C EDO23D SBC BEGAD
800 3B3F A5FF LDA VEC+1
810 3B41 8D053D STA ENDAD+1 ;STORE END ADRES
820 3B44 EDO33D SBC BEGAD+1
830 3B47 90B3 BCC RELOC2
840 3B49 20F93B JSR SETERA ;SET ERROR ADRES
850 3B4C AE083D LDX SHOLD
860 3B4F 9A TXS ;RESTORE STACK
870 3B50 20732D RELOC3 JSR STROUT
880 3B53 0A .BYTE $0A,$0D
890 3B55 20 .BYTE ' TO: '
900 3B6E 00 .BYTE $00
910 3B6F 209B2C JSR OSINP ;READ PARAMETER
920 3B72 A200 LDX #00
930 3B74 8EE52C STX BUFOFF ;RESET INPUT BUFFER
940 3B77 20232D JSR GETADR ;READ RELOCATE ADRES
950 3B7A 38 SEC
960 3B7B A5FE LDA VEC
970 3B7D EDO23D SBC BEGAD
980 3B80 8D063D STA RELOFF ;COMPUTE OFFSET
990 3B83 A5FF LDA VEC+1
1000 3B85 EDO33D SBC BEGAD+1
1010 3B88 8D073D STA RELOFF+1 ;
1020 ;HERE STARTS THE RELOCATE PROCEDURE
1030 3B8B A200 REL LDX #00 ;INIT LOAD POINTER
1040 3B8D A1C0 LDA (CURAD,X) ;LOAD OPCODE
1050 3B8F 200C3C JSR LENAC ;COMPUTE LENGTH
1060 3B92 C003 CPY #03
1070 3B94 D034 BNE ADJUST ;=>LENGTH <3
1080 3B96 A001 RELTST LDY #01 ;INIT LOAD POINTER
1090 3B98 38 SEC
1100 3B99 B1C0 LDA (CURAD),Y ;LOAD LSB
1110 3B9B EDO23D SBC BEGAD
1120 3B9E C8 INY
1130 3B9F B1C0 LDA (CURAD),Y
1140 3BA1 EDO33D SBC BEGAD+1
1150 3BA4 9022 BCC RELEX ;=>(CURAD)<BEGAD
1160 3BA6 A001 LDY #01
1170 3BA8 38 SEC
1180 3BA9 B1C0 LDA (CURAD),Y
1190 3BAB EDO43D SBC ENDAD
1200 3BAE C8 INY
1210 3BAF B1C0 LDA (CURAD),Y
1220 3BB1 EDO53D SBC ENDAD+1
1230 3BB4 B012 BCS RELEX ;=>(CURAD)>=ENDAD
1240 3BB6 A001 LDY #01
1250 3BB8 18 CLC
1260 3BB9 B1C0 LDA (CURAD),Y ;LOAD LSB
1270 3BBB 6D063D ADC RELOFF ;ADD OFFSET
1280 3BBE 91C0 STA (CURAD),Y
1290 3BC0 C8 INY
1300 3BC1 B1C0 LDA (CURAD),Y
1310 3BC3 6D073D ADC RELOFF+1

1320 3BC6 91C0 STA (CURAD),Y
1330 3BC8 A003 RELEX LDY #03
1340 3BCA E6C0 ADJUST INC CURAD
1350 3BCC D002 BNE ADJ1 ;=>NO PAGE BOUNDARY
1360 3BCE E6C1 INC CURAD+1
1370 3BD0 EE003D ADJ1 INC COUNT
1380 3BD3 D005 BNE ADJ3
1390 3BD5 EE013D INC COUNT+1
1400 3BD8 F005 BEQ EXIT ;=>COUNT = 0000
1410 3BDA 88 ADJ3 DEY
1420 3BDB DOED BNE ADJUST ;=>NOT NEXT CODE
1430 3BDD FOAC BEQ REL
1440 3BDF 20732D EXIT JSR STROUT
1450 3BE2 0A .BYTE $0A,$0D,'RELOCATED',$0A,$0D,$00
1460 3BF0 68 PLA
1470 3BF1 8D502A STA $2A50 ;RESTORE ERROR ADRES
1480 3BF4 68 PLA
1490 3BF5 8D492A STA $2A49
1500 3BF8 60 RTS
1510 ;
1520 3BF9 68 SETERA PLA ;LOW RETURNADRES
1530 3BFA AA TAX
1540 3BFB 68 PLA
1550 3BFC 48 PHA ;HIGH ADRES BACK
1560 3BFD 8D502A STA $2A50
1570 3C00 8A TXA
1580 3C01 48 PHA ;LOW ADRES BACK
1590 3C02 E8 INX
1600 3C03 8E4F2A STX $2A4F
1610 3C06 D003 BNE SETERB
1620 3C08 EE502A INC $2A50
1630 3C0B 60 SETERB RTS
1640 ;
1650 ;LENAC COMPUTES THE INSTRUCTIONLENGTH
1660 ;IT LEAVES THE ROUTINE WITH THE LENGTH OF THE
1670 ;INSTRUCTION IN Y. A AND X ARE AFFECTED.
1680 ;
1690 3C0C A001 LENAC LDY #01 ;1 BYTE INSTRUCTIONS
1700 3C0E C900 CMP #00 ;IS IT A BREAK
1710 3C10 F01A BEQ LENEND ;=>BREAK
1720 3C12 C940 CMP #40
1730 3C14 F016 BEQ LENEND ;=>RTI
1740 3C16 C960 CMP #60
1750 3C18 F012 BEQ LENEND ;=>RTS
1760 3C1A A003 LDY #03 ;3 BYTE INSTRUCTIONS
1770 3C1C C920 CMP #20
1780 3C1E F00C BEQ LENEND ;=>JSR
1790 3C20 291F AND #1F ;NOW REGULAR
1800 3C22 C919 CMP #19
1810 3C24 F006 BEQ LENEND ;=>ANNY ABS INSTRUCTION
1820 3C26 290F AND #0F ;USE ONLY 4 BITS
1830 3C28 AA TAX ;USE X AS AN INDEX
1840 3C29 BC2D3C LDY LEN,X ;LOAD THE LENGTH
1850 3C2C 60 LENEND RTS
1860 ;
1870 3C2D 02 LEN .BYTE $02,$02,$02,$01
1880 3C31 02 .BYTE $02,$02,$02,$01
1890 3C35 01 .BYTE $01,$02,$01,$01
1900 3C39 03 .BYTE $03,$03,$03,$03
1910 ;
1920 END

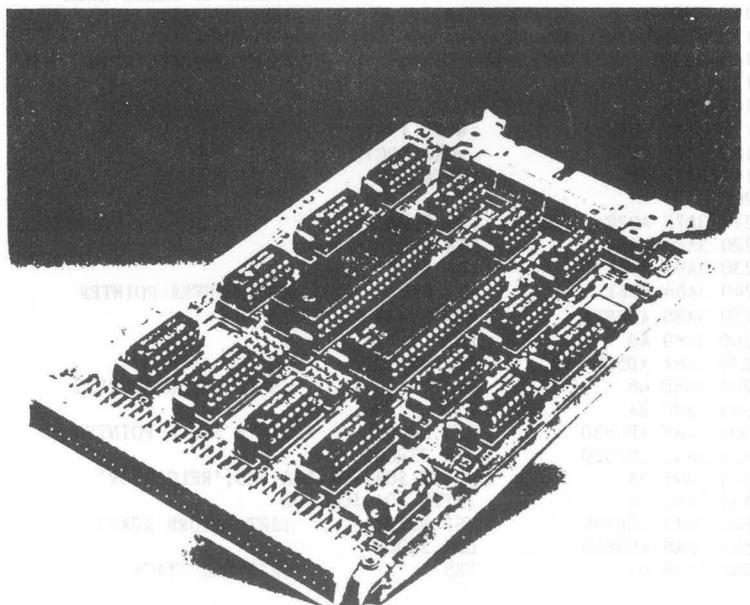
```

BOEKEN

NIEUW
DE 68000 FAMILIE. Thomas L. Harman en Barbara Lawson.
Vertaling: Ingenieurs- en Adviesbureau ACIDO GRID.
Uitg.: Academic Service bv./ca 600 pag./Hfl. 85,-.
ISBN 90 6233 246 3

NIEUW
UNIX; het standaard operating system. G.J.M. Austen en
H.J. Thomassen. 1985, 2e herziene druk 1986/343 pag./Hfl.
68,-.
Uitg.: Academic Service bv.
ISBN 90 6233 217 X

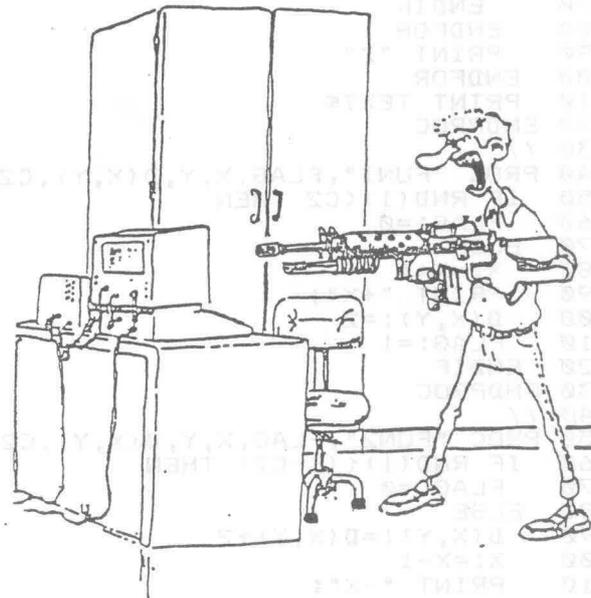
NIEUW
LOGISCH LOGO. Auke Sikma. 1986/226 pag./Hfl. 35,-
Uitg.: Academic Service bv.
ISBN 90 6233 203 X




```

1000 // *****
1010 // * AMAZING MAZE V.1.COMAL *
1020 // *-----*
1030 // * A.Megens febr.87 *
1040 // * for MON/DOS65 systems *
1050 // *****
1060 //
1070 B:=26
1080 H:=10
1090 DIM D(B,H),B(3),M(2*B+1,2*H+1),V(2*B+1,2*H+1)
1100 C1:=1/(B+H)
1110 C2:=.7
1120 C3:=.8
1130 C4:=.5
1140 E:=B*H
1150 I:=INT(((RND(1)+.5)*B)/2)
1160 T:=1
1170 A#:"400140240124304134324132"
1180 GRAF#:=CHR$(27)+"F"
1190 TEXT#:=CHR$(27)+"G"
1200 FOR X:=0 TO B
1210   FOR Y:=0 TO H
1220     D(X,Y):=0
1230   ENDFOR
1240 ENDFOR
1250 X:=I
1260 Y:=0
1270 D(X,Y):=1
1280 X:=X-1
1290 B:=B-1
1300 H:=H-1
1310 A:=0
1320 P:=0
1330 // ***** MAIN LOOP MAZE GENERATOR *****
1340 REPEAT
1350   PRINT T,
1360   REPEAT
1370     IF D(X,Y)=0 OR (A+P)=0 THEN
1380       REPEAT
1390         X:=X+1
1400         IF X>B THEN
1410           X:=0
1420           Y:=Y+1
1430           IF Y>H THEN
1440             Y:=0
1450           ENDFOR
1460         ENDFOR
1470         UNTIL D(X,Y)<>0
1480       ENDFOR
1490       A:=0
1500       P:=0
1510       IF X<B THEN
1520         IF D(X+1,Y)=0 THEN
1530           P:=1
1540         ENDFOR
1550       ENDFOR
1560       IF X>0 THEN
1570         IF D(X-1,Y)=0 THEN
1580           P:=P+2
1590         ENDFOR
1600       ENDFOR
1610       IF Y<H THEN
1620         IF D(X,Y+1)=0 THEN
1630           P:=P+4
1640         ENDFOR
1650       ENDFOR
1660       IF Y>0 THEN
1670         IF D(X,Y-1)=0 THEN
1680           A:=1
1690         ENDFOR
1700       ENDFOR
1710       IF P>0 THEN
1720         A:=A+1
1730         IF P>2 THEN
1740           A:=A+1
1750         ENDFOR
1760       ENDFOR
1770     UNTIL (A+P)<>0

```



Waddaya mean, user error!?

```

1780 FLAG:=0
1790 REPEAT
1800   REPEAT
1810     Q:=3*P+INT(RND(1)*A+1)
1820     UNTIL MID$(A$,Q,1)<>"0"
1830     C$="FUN"+MID$(A$,Q,1)
1840     EXEC: C$,FLAG,X,Y,D(X,Y),C2
1850   UNTIL FLAG<>0
1860   PRINT
1870   T:=T+1
1880   IF RND(1)<C1 THEN
1890     X:=INT(RND(1)*B)
1900     Y:=INT(RND(1)*H)
1910   ENDIF
1920 UNTIL T>=E
1930 CLS
1940 D(B-I,H):=D(B-I,H)+4
1950 EXEC: "MAZE"
1960 END.
1970 //
1980 PROC "MAZE"
1990   PRINT GRAF$;
2000   Y:=0
2010   FOR X:=0 TO B
2020     IF X=I THEN
2030       PRINT "Z ";
2040     ELSE
2050       PRINT "ZXX";
2060     ENDIF
2070   ENDFOR
2080   PRINT "Z"
2090   FOR Y:=0 TO H
2100     FOR X:=0 TO B
2110       IN:=D(X,Y)
2120       EXEC: "BINARY",IN,B(1),B(2)
2130       IF B(1)=1 THEN
2140         PRINT " ";
2150       ELSE
2160         PRINT "Y ";
2170       ENDIF
2180     ENDFOR
2190     PRINT "Y"
2200     FOR X:=0 TO B
2210       IN:=D(X,Y)
2220       EXEC: "BINARY",IN,B(1),B(2)
2230       IF B(2)=1 THEN
2240         PRINT "Z ";
2250       ELSE
2260         PRINT "ZXX";
2270       ENDIF
2280     ENDFOR
2290     PRINT "Z"
2300   ENDFOR
2310   PRINT TEXT$
2320 ENDPROC
2330 //
2340 PROC "FUN1",FLAG,X,Y,D(X,Y),C2
2350   IF RND(1)<C2 THEN
2360     FLAG:=0
2370   ELSE
2380     X:=X+1
2390     PRINT "+X";
2400     D(X,Y):=2
2410     FLAG:=1
2420   ENDIF
2430 ENDPROC
2440 //
2450 PROC "FUN2",FLAG,X,Y,D(X,Y),C2
2460   IF RND(1)<(1-C2) THEN
2470     FLAG:=0
2480   ELSE
2490     D(X,Y):=D(X,Y)+2
2500     X:=X-1
2510     PRINT "-X";
2520     D(X,Y):=1
2530     FLAG:=1
2540   ENDIF
2550 ENDPROC
2560 //
2570 PROC "FUN3",FLAG,X,Y,D(X,Y),C2

```

```

2580   IF RND(1)<C3 THEN
2590     FLAG:=0
2600   ELSE
2610     D(X,Y):=D(X,Y)+4
2620     Y:=Y+1
2630     PRINT "+Y";
2640     D(X,Y):=1
2650     FLAG:=1
2660   ENDIF
2670 ENDPROC
2680 //
2690 PROC "FUN4",FLAG,X,Y,D(X,Y),C2
2700   Y:=Y-1
2710   PRINT "-Y";
2720   D(X,Y):=4
2730   FLAG:=1
2740   IF RND(1)<C4 THEN
2750     C2:=1-C2
2760   ENDIF
2770 ENDPROC
2780 //
2790 PROC "BINARY",IN,B(1),B(2)
2800   NUM:=IN
2810   FOR I:=3 TO 0 STEP -1
2820     IF NUM-2^I>=0 THEN
2830       B(I):=1
2840       NUM:=NUM-2^I
2850     ELSE
2860       B(I):=0
2870     ENDIF
2880   ENDFOR
2890 ENDPROC
2900 //

```



```

1 REM          FUNCTION DISPLAY
2 REM
3 REM
10 REM        This program shows a function given by the
20 REM        user on line 950 and 960.
30 REM        It scales the function automatically and
40 REM        plots the x and y axes when necessary.
50 REM
60 REM        The program has been written for an Octopus 65
70 REM        (EC65,SAMSON) extended with a grafic card
80 REM        (described in Elektor Holland, September 1985).
90 REM
100 REM*****
110 REM
120 REM        System dependent parameters
130 REM
140 hn=512      :REM topright corner of the picture
150 vn=256      :REM hn=horizontal, vn=vertical coordinate
160 bh=0        :REM left under corner of the total picture
170 bv=0        :REM bh=horizontal line, bv=vertical row
190 GOTO1050    :REM start the program
200 REM*****
210 REM
220 REM        System dependent routines
230 REM
240 REM init of the grafic card in text mode
250 DISK!"GO C000":DISK!"IO ,03":RETURN
260 REM change to grafic mode
270 PRINTCHR$(18);:RETURN
280 REM change to text mode
290 PRINTCHR$(17);:RETURN
300 REM clearing of the screen
310 PRINTCHR$(12);:RETURN
320 REM moving the pen to absolute location
330 PRINT"M"x","y":RETURN
340 REM plotting of a point on (x,y), origin is left-under
350 PRINT"N0,"x","y":RETURN
360 REM plotting of a line from current position to (x,y)
370 PRINT"D"x","y":RETURN
380 REM printing of text in a$ starting in (x,y)
390 GOSUB330:PRINT"P"a$:RETURN
400 REM change to vertical print direction
410 PRINT"Q2":RETURN
420 REM change to horizontal print direction
430 PRINT"Q0":RETURN
900 REM*****
910 REM
920 REM        The function y must be given as a function
930 REM        of x
940 REM
950 y=.3849*(1-2*x/3+2*x*x/27)*EXP(-x/3)
960 :
970 RETURN
980 REM explanation of the axes
990 xs$="coordinate":ys$="ordinate":RETURN
1000 REM*****
1010 REM
1020 REM        MAINPROGRAM
1030 REM
1040 REM*****
1050 GOSUB250:GOSUB980
1060 LIST920-960:LIST980-990
1070 INPUT"Is this the right function";an$
1080 IFan$="n"ORan$="N"THENPRINT"Please change it":END
1090 INPUT"What is the starting value of x";fx
1100 INPUT"and the ending value of x";lx
1110 IFfx>lxTHENbu=fx:fx=lx:lx=bu
1200 REM take a certain number of functionvalues
1210 ht=40:vt=24:REM space needed for axes plus comment

```

```

1220 np=hn-ht-bh-10:REM number of function values
1230 PRINT:PRINT"Function value evaluation!!!"
1240 DIMfu(np-1)
1250 x=fx:GOSUB950:hy=y:ly=y:fu(0)=y
1260 st=(lx-fx)/(np-1)
1270 FORn=1TONp-1
1280 x=st*n+fx:GOSUB950:fu(n)=y
1290 IFy<lyTHENly=y
1300 IFy>hyTHENhy=y
1310 NEXT
1400 REM setting up the screen
1410 GOSUB250:GOSUB270
1420 x=ht+bh:y=vt+bv:GOSUB330
1430 y=vn-2:GOSUB370:x=hn-7:GOSUB370:y=vt+bv:GOSUB370
1440 x=ht+bh:GOSUB370
1500 REM scale on the x-axis
1510 xi=ABS(lx-fx):xo=0
1520 IFxi>30THENxo=xo+1:xi=xi/10:GOTO1520
1530 IFxi<3THENxo=xo-1:xi=xi*10:GOTO1530
1540 ad=0:IFxi>=10THENad=1
1550 ba=INT(fx/10↑xo-.5)
1560 IF(ba+.01)*10↑xo<fxTHENba=ba+1:GOTO1560
1570 IFad=1THENIFABS(INT(ba-2*INT(ba/2)))=1THENba=ba+1
1580 cl=(np-1)/(lx-fx):c2=ht+np+bh+1-cl*lx
1590 cl=cl*10↑xo:n=0
1600 cn=ba+n:IFad=1THENCn=ba+2*n
1610 co=INT(cn*cl+c2+.5)
1620 IFco>ht+np+bh+2THEN1690
1630 x=co:y=vt+bv:GOSUB330:y=vt+bv-4:GOSUB370
1640 IFcn=0THENy=vn-3:GOSUB370
1650 a$=STR$(cn):le=LEN(a$):IFcn>=0THENA$=RIGHT$(a$,le-1):le=le-1
1660 x=co-le*3+1:y=vt+bv-14
1670 IFx+6*le-2<hnTHENGOSUB390
1680 n=n+1:GOTO1600
1690 a$=xs$:IFxo<>0THENA$=xs$+" X10↑"+STR$(xo)
1695 x=INT((np-6*LEN(a$))/2)+ht+bh:y=vt+bv-24:GOSUB390
1700 REM scale on the y-axis
1710 yi=ABS(hy-ly):yo=0
1720 IFyi>30THENyo=yo+1:yi=yi/10:GOTO1720
1730 IFyi<3THENyo=yo-1:yi=yi*10:GOTO1730
1740 ad=0:IFyi>=10THENad=1
1750 ba=INT(ly/10↑yo-.5)
1760 IF(ba+.01)*10↑yo<lyTHENba=ba+1:GOTO1760
1770 IFad=1THENIFABS(INT(ba-2*INT(ba/2)))=1THENba=ba+1
1780 cl=(vn-vt-bv-6)/(hy-ly):c2=vn-4-cl*hy
1790 cl=cl*10↑yo:n=0
1800 cn=ba+n:IFad=1THENCn=ba+2*n
1810 co=INT(cn*cl+c2+.5)
1820 IFco>vn-2THEN1890
1830 y=co:x=ht+bh:GOSUB330:x=ht+bh-4:GOSUB370
1840 IFcn=0THENx=hn-8:GOSUB370
1850 a$=STR$(cn):le=LEN(a$):IFcn>=0THENA$=RIGHT$(a$,le-1):le=le-1
1860 y=co-4:x=ht+bh-le*6-5
1870 IFco<vn-3THENGOSUB390
1880 n=n+1:GOTO1800
1890 a$=ys$:IFYo<>0THENA$=ys$+" X10↑"+STR$(yo)
1895 y=INT((vn-vt-bv-6*LEN(a$))/2)+vt+bv:x=bh+8
1900 GOSUB410:GOSUB390:GOSUB430
2000 REM plotting the function
2010 FORn=0TONp-1
2020 x=ht+bh+2+n:y=INT(fu(n)/10↑yo*cl+c2+.5):GOSUB350
2030 NEXT:END
3000 REM*****
3010 REM
3020 REM      Writtten by
3030 REM      Ger van den Hoek
3040 REM      Vossendijk 129-8
3050 REM      6534 TL Nijmegen
3060 REM
3070 REM*****

```

```

:ASS COUNTER TUSSENTEXT
0000 0010 ;*****
0000 0020 ;*
0000 0030 ;* FREQUENTIECOUNTER M.B.V. DE VIA *
0000 0040 ;* 6522 *
0000 0050 ;*
0000 0060 ;* Men kan keuze maken uit INTerne of *
0000 0070 ;* EXTerne triggering. *
0000 0080 ;*
0000 0090 ;*****
0000 0100 ;
0000 0110 ;Voor de motorola MC 6800 - 6802 - 6808 uP
0000 0120 ;
0000 0130 ;Copyright : Frank Weijnen
0000 0140 ;klas : VTI-a
0000 0150 ;datum : 1-5-1985
0000 0160 ;versie : 2.5
0000 0170 ;
0000 0180 ;
A200 0190 ROM : EQU $A200 ;
A100 0200 RAM : EQU $A100 ;
A790 0210 STACKTOP : EQU $A790 ;top of stack
0000 0220 ;
0010 0230 VIAUS : EQU $0010 ;via 6522
8010 0240 PIASYS : EQU $8010 ;systeem pia
0000 0250 ;
0000 0260 ;
0000 0270 ;*****
0000 0280 ;* I/O DECLARATIE *
0000 0290 ;*****
0000 0300 ;
0000 0310 ;
0010 0320 DRBVIA : EQU VIAUS ;data register B van VIA
0012 0330 DDRBVIA : EQU VIAUS + 2 ;data direction reg. B
0018 0340 TIMER2L : EQU VIAUS + 8 ;T2 latch/counter (LSB)
0019 0350 TIMER2H : EQU VIAUS + 9 ;T2 counter (MSB)
0000 0360 ;
001B 0370 ACRVIA : EQU VIAUS + 11 ;aux. cont. register
001C 0380 PCRVIA : EQU VIAUS + 12 ;periph. cont. register
001D 0390 IFRVIA : EQU VIAUS + 13 ;interrupt flag register
001E 0400 IERVIA : EQU VIAUS + 14 ;interrupt enable reg.
0000 0410 ;
8010 0420 PDRASYS : EQU PIASYS ;systeem pia
8010 0430 DDRASYS : EQU PIASYS ;A-zijde
8011 0440 CRASYS : EQU PIASYS + 1 ;
0000 0450 ;
8012 0460 PDRBSYS : EQU PIASYS + 2 ;systeem pia
8012 0470 DDRBSYS : EQU PIASYS + 2 ;B-zijde
8013 0480 CRBSYS : EQU PIASYS + 3 ;
0000 0490 ;
0000 0500 ;*****
0000 0510 ;* BUFFER DECLARATIE *
0000 0520 ;*****
0000 0530 ;
0000 0540 ;
A100 0550 ; ORG RAM ;
A100 0560 ;
A100 0570 FREQUENTIE : RMB 1 ; ,, ,, 'F'
A101 0580 RMB 1 ; ,, ,, '='
A102 0590 RMB 1 ; ,, ,, space
A103 0600 TIENDZND : RMB 1 ;tienduizendtallen
A104 0610 DUIZEND : RMB 1 ;duizendtallen
A105 0620 HONDERD : RMB 1 ;honderdtallen
A106 0630 TIEN : RMB 1 ;tientallen
A107 0640 EEN : RMB 1 ;eenheden
A108 0650 ;
A108 0660 ;
A108 0670 REKENBUF : RMB 6 ;reken buffer hex-dec
A10E 0680 SAVEX : RMB 2 ;tijdelijke opslag Xreg.
A110 0690 OMREKENX : RMB 2 ;opslag X voor bereken
A112 0700 SAVESTACK : RMB 2 ;tijdelijke opslag stack
A114 0710 TEXTBUFF : RMB 2 ;pointer voor disp.text
A116 0720 FREQBUF : RMB 2 ;tijdelijke opsl. freq.
A118 0730 ;
A118 0740 ;
A118 0750 ;*****
A118 0760 ;* JUMP TABEL *
A118 0770 ;*****
A118 0780 ;
A118 0790 ;
A200 0800 ; ORG ROM ;
A200 0810 ;

```

```

A200 7E A393 0820 JUMPTABEL : JMP START ;begin hoofdprogramma
A203 7E A243 0830 JMP INITVIA ;initieer VIA
A206 7E A24F 0840 JMP INITSYS ;initieer systeem PIA
A209 7E A262 0850 JMP INITBUFF ;initieer (text) buffer
A20C 7E A278 0860 JMP DISPLAY ;display inhoud textbuf.
A20F 7E A296 0870 JMP CONVERT ;zet code naar dispcode
A212 7E A2B2 0880 JMP TELLAMP ;lamp aan tijdens tellen
A215 7E A2BB 0890 JMP DELAY ;tijdbasis van 1 sec
A218 7E A2C9 0900 JMP SETCOUNT ;set begin waarde freq.
A21B 7E A2CF 0910 JMP TRIGPULS ;wacht op triggerpuls
A21E 7E A2E0 0920 JMP BEOVERF ;kijk of freq. te groot
A221 7E A2F6 0930 JMP CONVFREQ ;echte waarde freq.
A224 7E A307 0940 JMP OMVORM ;subroutine van HEXDEC
A227 7E A31A 0950 JMP UNPACK ;pak packt BCD uit
A22A 7E A324 0960 JMP PUTCODE ;zet code intabel
A22D 7E A341 0970 JMP HEXDEC ;hex --) dec
A230 7E A37C 0980 JMP FREQCONV ;tabelinh. --) dispcodes
A233 0990 ;
A233 1000 ;
A233 1010 ;*****
A233 1020 ;* CONSTANTEN DECLARATIE *
A233 1030 ;*****
A233 1040 ;
A233 1050 ;
A233 40 1060 ERRORR : FCB $40 ;code voor '-'
A234 79 1070 FCB $79 ; ,, ,, 'E'
A235 50 1080 FCB $50 ; ,, ,, 'r'
A236 50 1090 FCB $50 ; ,, ,, 'r'
A237 5C 1100 FCB $5C ; ,, ,, 'o'
A238 50 1110 FCB $50 ; ,, ,, 'r'
A239 40 1120 FCB $40 ; ,, ,, '-'
A23A 40 1130 FCB $40 ; ,, ,, '-'
A23B 1140 ;
A23B 71 1150 MASKER : FCB $71 ;code voor F
A23C 48 1160 FCB $48 ; =
A23D 00 1170 FCB 0,0,0 ; space
A23E 00
A23F 00
A240 77 1180 FCB $77, $77 ; A A
A241 77
A242 37 1190 EINDMASK : FCB $37 ; N
A243 1200 ;
A243 1210 ;*****
A243 1220 ;* SUBROUTINE INITVIA *
A243 1230 ;*****
A243 1240 ;
A243 1250 ;
A243 1260 ; INPUT : -
A243 1270 ; OUTPUT : -
A243 1280 ;DESTRUCTIEREG. INHOUD : X en A
A243 1290 ;GEBRUIKTEGEHEUGENADR : 0
A243 1300 ;
A243 1310 ;
A243 CE 2010 1320 INITVIA : LDX #$2010 ;timer2 puls count. PBE
A246 DF 1B 1330 STX ACRVIA ;CB1 pos. flank voor EXT
A248 86 00 1340 LDA A #$00 ;geen inter. genereren
A24A 97 1E 1350 STA A IERVIA ;doorgeven
A24C 97 12 1360 STA A DDRBVIA ;B-zijde voor input
A24E 39 1370 RTS ;
A24F 1380 ;
A24F 1390 ;
A24F 1400 ;*****
A24F 1410 ;* SUBROUTINE INITSYS *
A24F 1420 ;*****
A24F 1430 ;
A24F 1440 ;
A24F 1450 ; INPUT : -
A24F 1460 ; OUTPUT : -
A24F 1470 ;DESTRUCTIEREG. INHOUD : X
A24F 1480 ;GEBRUIKTEGEHEUGENADR : 0
A24F 1490 ;
A24F 7F 8011 1500 INITSYS : CLR CRASYS ;DDRA beschikbaar
A252 7F 8013 1510 CLR CRBSYS ;DDR beschikbaar
A255 CE FF04 1520 LDX #$FF04 ;A-zijde output
A258 FF 8010 1530 STX DDRASYS ;
A25B CE 0F34 1540 LDX #$0F34 ;B-zijde in-/output
A25E FF 8012 1550 STX DDRBSYS ;7-4 input en 3-0 output
A261 39 1560 RTS ;en CB2 output
A262 1570 ;
A262 1580 ;

```

```

A262      1590 ;*****
A262      1600 ;*      SUBROUTINE INITBUFF      *
A262      1610 ;*****
A262      1620 ;
A262      1630 ;
A262      1640 ;           INPUT : -
A262      1650 ;           OUTPUT : -
A262      1660 ;DESTRUCTIEREG. INHOUD : X --> SOURCE ADRES DATA
A262      1670 ;           SP --> DESTINATION ADRES DATA
A262      1680 ;GEBRUIKTEGEHEUGENADR : 8   BEREIK DATA VERPLAATSING
A262      1690 ;
A262 BF A112 1700 INITBUFF : STS      SAVESTACK ;red stackpointer
A265 8E A107 1710          LDS      #REKENBUF-1 ;destenation van data
A268 CE A242 1720          LD      #EINDMASK ;source adres van data
A26B      1730          REPEAT
A26B A6 00   1740          LDA A    0,X      ;haal data op
A26D 3E     1750          PSH A          ;zet data weg in tabel
A26E 09     1760          DEX          ;volgende data
A26F 8C A23A 1770          CPX      #MASKER-1 ;alles gehad ?
A272 2E F7 A26B 1780          UNTIL EQ ;nee, dan doorgaan
A274 BE A112 1790          LDS      SAVESTACK ;ja, dan restore
A277 39     1800          RTS          ;stackpointer
A278      1810 ;
A278      1820 ;
A278      1830 ;*****
A278      1840 ;*      SUBROUTINE DISPLAY      *
A278      1850 ;*****
A278      1860 ;
A278      1870 ;           INPUT : -
A278      1880 ;           OUTPUT : -
A278      1890 ;DESTRUCTIEREG. INHOUD : A en X
A278      1900 ;GEBRUIKTEGEHEUGENADR : POINTER VAN DISPLAY TEXT
A278      1910 ;
A278 B6 8012 1920 DISPLAY : LDA A  PDRBSYS ;clear interrupt bit
A27B FE A114 1930          LD      TEXTBUFF ;load text buffer
A27E 8E 07   1940          LDA A  #07 ;doorloop loop 8 keer
A280      1950          REPEAT
A280 B7 8012 1960          STA A  PDRBSYS ;zet pia op 1e maal
A283 3E     1970          PSH A          ;save counter
A284 A6 00   1980          LDA A  0,X      ;haal karakter op
A28E B7 8010 1990          STA A  PDRASYS ;display karakter
A289 4F     2000          CLR A          ;
A28A      2010          REPEAT
A28A 4A     2020          DEC A          ;delay voor scherp
A28B 2E FD A28A 2030          UNTIL EQ ;karakter
A28D 08     2040          INX          ;volgende disp. karakt.
A28E 7F 8010 2050          CLR  PDRASYS ;maak display schoon
A291 32     2060          PUL A          ;haal counter op
A292 4A     2070          DEC A          ;scan next row
A293 2A EB A280 2080          UNTIL MI ;laatste row ?
A295 39     2090          RTS          ;ja
A296      2100 ;
A296      2110 ;
A296      2120 ;*****
A296      2130 ;*      SUBROUTINE CONVERT      *
A296      2140 ;*****
A296      2150 ;
A296      2160 ;
A296      2170 ;           INPUT : A --> CONVERTEER WAARDE
A296      2180 ;           OUTPUT : A --> NIEUWE WAARDE
A296      2190 ;DESTRUCTIEREG. INHOUD : A en X
A296      2200 ;GEBRUIKTEGEHEUGENADR : 2   REKENADRES VAN NIEUWE CODE
A296      2210 ;
A296 CE A2A8 2220 CONVERT : LD      #CONVTAB ;begin converteer tabel
A299 FF A110 2230          ST      OMREKENX ;
A29C BB A111 2240          ADD A  OMREKENX + 1 ;bepaal plaats van code
A29F B7 A111 2250          STA A  OMREKENX + 1 ;
A2A2 FE A110 2260          LD      OMREKENX ;X := code pointer
A2A5 A6 00   2270          LDA A  0,X      ;A := code
A2A7 39     2280          RTS          ;
A2A8      2290 ;
A2A8      2300 ;
A2A8 3F     2310 CONVTAB : FCB $3F ; 0
A2A9 0E     2320          FCB $0E ; 1
A2AA 5B     2330          FCB $5B ; 2
A2AB 4F     2340          FCB $4F ; 3
A2AC 6E     2350          FCB $6E ; 4
A2AD 6D     2360          FCB $6D ; 5
A2AE 7D     2370          FCB $7D ; 6
A2AF 07     2380          FCB $07 ; 7
A2B0 7F     2390          FCB $7F ; 8
A2B1 6F     2400          FCB $6F ; 9

```

```

A2B2      2410 ;
A2B2      2420 ;
A2B2      2430 ;
A2B2      2440 ;*      SUBROUTINE INVERTEER CB2      *
A2B2      2450 ;*
A2B2      2460 ;
A2B2      2470 ;
A2B2      2480 ;
A2B2      2490 ;      INPUT : -
A2B2      2500 ;      OUTPUT : -
A2B2      2510 ;DESTRUCTIEREG. INHOUD : A --) INHOUD CRBSYS REGISTER
A2B2      2520 ;GEBRUIKTEGEHEUGENADR : 0
A2B2      2530 TELLAMP      : LDA A      CRBSYS      ;bij iedere aanroep
A2B5      88 08      2540      EOR A      #$08      ;klapt CB2 uitgang van
A2B7      B7 8013    2550      STA A      CRBSYS      ;polariteit om
A2BA      39        2560      RTS
A2BB      2570 ;
A2BB      2580 ;
A2BB      2590 ;*
A2BB      2600 ;*      SUBROUTINE DELAY      *
A2BB      2610 ;*
A2BB      2620 ;
A2BB      2630 ;
A2BB      2640 ;      INPUT : X --) TEXT POINTER
A2BB      2650 ;      OUTPUT : -
A2BB      2660 ;DESTRUCTIEREG. INHOUD : B --) TIJDS EENHEID
A2BB      2670 ;GEBRUIKTEGEHEUGENADR : 0
A2BB      2680 ;
A2BB      CE 4F      2690 DELAY      : LDA B      #$4F      ;disp. 79 * voor vert.
A2BD      2700      REPEAT
A2BD      8D B9 A278 2710      CALL      DISPLAY      ;disp text in textbuff
A2BF      01        2720      NOP
A2C0      01        2730      NOP
A2C1      01        2740      NOP
A2C2      01        2750      NOP
A2C3      01        2760      NOP
A2C4      01        2770      NOP
A2C5      5A        2780      DEC B
A2C6      26 F5 A2BD 2790      UNTIL EQ
A2C8      39        2800      RTS
A2C9      2810 ;
A2C9      2820 ;
A2C9      2830 ;*
A2C9      2840 ;*      SUBROUTINE SETCOUNT      *
A2C9      2850 ;*
A2C9      2860 ;
A2C9      2870 ;
A2C9      2880 ;      INPUT : -
A2C9      2890 ;      OUTPUT : -
A2C9      2900 ;DESTRUCTIEREG. INHOUD : X --) BEGINWAARDE TELLER
A2C9      2910 ;GEBRUIKTEGEHEUGENADR : 0
A2C9      2920 ;
A2C9      CE 4FC3    2930 SETCOUNT : LDX      #$4FC3      ;imax freq. is 49.999 HZ
A2CC      DF 18      2940      STX      TIMER2L      ;eerst LSB dan MSB
A2CE      39        2950      RTS
A2CF      2960 ;
A2CF      2970 ;
A2CF      2980 ;*
A2CF      2990 ;*      SUBROUTINE TRIGPULS      *
A2CF      3000 ;*
A2CF      3010 ;GEBRUIKTEGEHEUGENADR : 0
A2CF      3020 ;
A2CF      3030 ;
A2CF      3040 ;      INPUT : -
A2CF      3050 ;      OUTPUT : -
A2CF      3060 ;DESTRUCTIEREG. INHOUD : A --) CB1 flag
A2CF      3070 ;GEBRUIKTEGEHEUGENADR : 0
A2CF      3080 ;
A2CF      3090 TRIGPULS : REPEAT
A2CF      8D A7 A278 3100      CALL      DISPLAY      ;display buffer
A2D1      96 1D      3110      LDA A      IFRVIA      ;tijdens wachtlus
A2D3      84 10      3120      AND A      #$10      ;kijk of er een
A2D5      26 04 A2DB 3130      IF EQ THEN      ;trigger puls is
A2D7      96 10      3140      LDA A      DRBVIA      ;is het Intern trig
A2D9      84 01      3150      AND A      #$01      ;Ja da verlaat rout.
A2DB      3160      FI
A2DB      27 F2 A2CF 3170      UNTIL NE
A2DD      96 10      3180      LDA A      DRBVIA      ;nee, dan blijf kijken
A2DF      3190      ;dummy read voor.
A2DF      39        3200      RTS      ;reset CB1 flag

```

```

A2E0      3210 ;
A2E0      3220 ;
A2E0      3230 ;*****
A2E0      3240 ;*      SUBROUTINE BEP OVERFLOW      *
A2E0      3250 ;*****
A2E0      3260 ;
A2E0      3270 ;
A2E0      3280 ;          INPUT : X PSEUDE WAARDE FREQ.
A2E0      3290 ;          A OVERFLOW CONDITIE
A2E0      3300 ;
A2E0      3310 ;          OUTPUT : -
A2E0      3320 ;DESTRUCTIEREG. INHOUD : X --) NIEUWE WAARDE TEXTBUFFER
A2E0      3330 ;GEBRUIKTEGEHEUGENADR : 2  BUFFER VOOR DISPLAY
A2E0      3340 ;
A2E0 FF A11E 3340 BEPOVERF : STX      FREQBUF      ;zet pseudo freq weg
A2E3 84 20   3350          AND A      #$20      ;overflow van timer2
A2E5 26 08 A2EF 3360          IF EQ      THEN      ;geen o.v. dan
A2E7 CE A100 3370          LDX      #FREQUENTIE ;textpointer op
A2EA FF A114 3380          STX      TEXTBUFF    ;frequentie
A2ED 20 0E A2F5 3390          ELSE      ;anders
A2EF CE A233 3400          LDX      #ERROR      ;textpointer op
A2F2 FF A114 3410          STX      TEXTBUFF    ;ERROR
A2F5      3420          FI
A2F5 39     3430          RTS
A2F6      3440 ;
A2F6      3450 ;
A2F6      3460 ;*****
A2F6      3470 ;*      SUBROUTINE CONVERT FREQ.      *
A2F6      3480 ;*****
A2F6      3490 ;
A2F6      3500 ;
A2F6      3510 ;          INPUT : -
A2F6      3520 ;          OUTPUT : -
A2F6      3530 ;DESTRUCTIEREG. INHOUD : A en B --) AFTREK WAARDE
A2F6      3540 ;GEBRUIKTEGEHEUGENADR : 2  BUFFER ECHE (HEX) FREQ WAARDE
A2F6      3550 ;
A2F6 86 4F   3560 CONVNFREQ : LDA A      #$4F      ;LSB - $4F
A2F8 C6 C3   3570          LDA B      #C3      ;MSB - $C3
A2FA B0 A11E 3580          SUB A      FREQBUF    ;bereken echte
A2FD F2 A117 3590          SBC B      FREQBUF + 1 ;frequentie waarde
A300 B7 A11E 3600          STA A      FREQBUF    ;restore echte
A303 F7 A117 3610          STA B      FREQBUF + 1 ;frequentie
A306 39     3620          RTS
A307      3630 ;
A307      3640 ;
A307      3650 ;*****
A307      3660 ;*      SUBROUTINE OMVORM      *
A307      3670 ;*****
A307      3680 ;
A307      3690 ;
A307      3700 ;          INPUT : A
A307      3710 ;          OUTPUT : B
A307      3720 ;DESTRUCTIEREG. INHOUD : A en B
A307      3730 ;GEBRUIKTEGEHEUGENADR : 0
A307      3740 ;
A307 16     3750 OMVORM : TAB
A308 84 0F   3760          AND A      #0F      ;B := A
A30A 80 05   3770          SUB A      #05      ;ls nibbel
A30C 2B 02 A310 3780          BMI      MSNIBBEL ;neg. getal ?
A30E CB 03   3790          ADD B      #03      ;nee, dan B + 3
A310 17     3800          TBA      ;ja, dan A := B
A311 84 F0   3810          AND A      #F0      ;ms nibbel
A313 80 50   3820          SUB A      #50      ;ms nibbel - 5
A315 2B 02 A319 3830          BMI      VOORBIJ ;neg. getal ?
A317 CB 30   3840          ADD B      #30      ;nee, dan B + 3(0)
A319 39     3850          RTS      ;ja, einde routine
A31A      3860 ;
A31A      3870 ;
A31A      3880 ;*****
A31A      3890 ;*      SUBROUTINE UNPACK      *
A31A      3900 ;*****
A31A      3910 ;
A31A      3920 ;
A31A      3930 ;          INPUT : A --) UNPACK WAARDE
A31A      3940 ;          OUTPUT : A en B --) MS en LS NIBBEL
A31A      3950 ;DESTRUCTIEREG. INHOUD : B
A31A      3960 ;GEBRUIKTEGEHEUGENADR : 0
A31A      3970 ;
A31A 16     3980 UNPACK : TAB
A31B 84 F0   3990          AND A      #F0      ;B := A
A31D C4 0F   4000          AND B      #0F      ;ms nibbel
A31F 44     4010          LSR A      ;ls nibbel
A320 44     4020          LSR A      ;bit 3-0 := 7-4
A321 44     4030          LSR A      ;
A322 44     4040          LSR A      ;
A323 39     4050          RTS
A324      4060 ;
A324      4070 ;

```

```

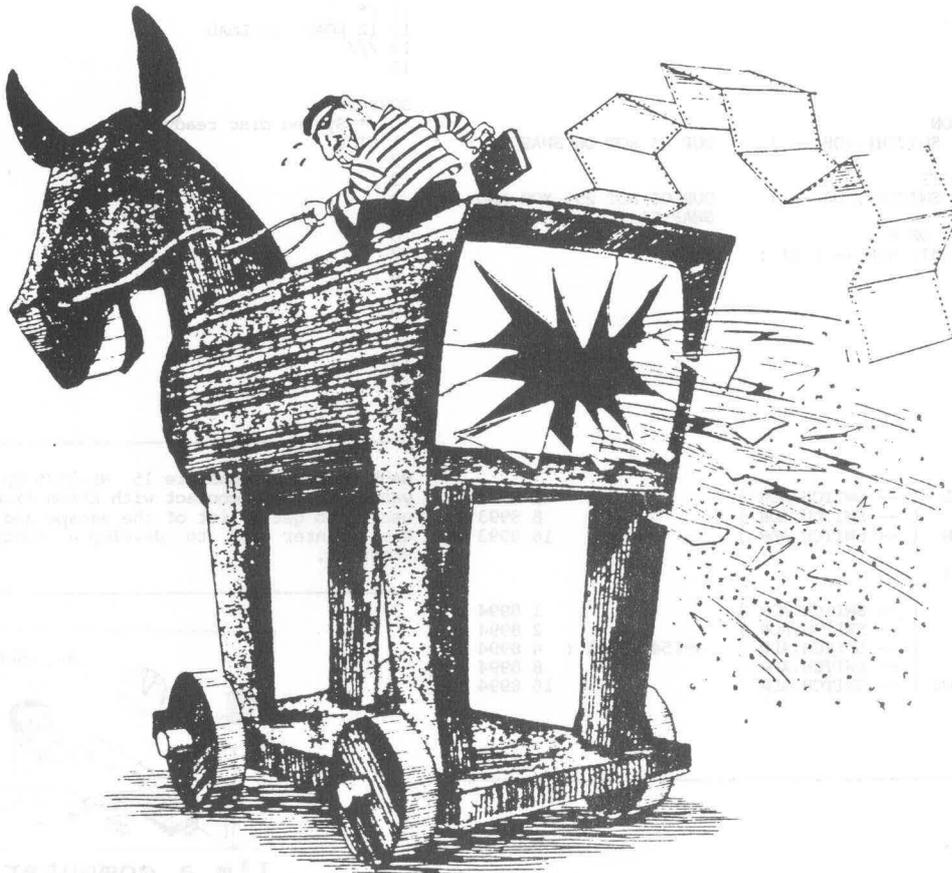
A324 4080 ;*****
A324 4090 ;*   SUBROUTINE ZET CODE IN TAB.   *
A324 4100 ;*****
A324 4110 ;
A324 4120 ;
A324 4130 ;           INPUT : -
A324 4140 ;           OUTPUT : -
A324 4150 ;DESTRUCTIEREG. INHOUD : A en B
A324 4160 ;GEBRUIKTEGEHEUGENADR : 5 PLAATS DECIMALE FREQ. WAARDE
A324 4170 ;
A324 CE A108 4180 PUTCODE      : LDX      #REKENBUF      ;pointer op rekenbuffer
A327 A6 01 4190 LDA A          1,X          ;haal tienduizendtallen
A329 B7 A103 4200 STA A          TIENDZND      ;iop en zet ze weg
A32C A6 02 4210 LDA A          2,X          ;haal 1000 en 100 tallen
A32E 8D EA A31A 4220 CALL          UNPACK        ;iop, rafel ze uit elkaar
A330 B7 A104 4230 STA A          DUIZEND      ;ien zet ze op duizend
A333 F7 A105 4240 STA B          HONDERD     ;ien honderd
A336 A6 03 4250 LDA A          3,X          ;idem met 10 en 1 heden
A338 8D E0 A31A 4260 CALL          UNPACK        ;zet ze weg op
A33A B7 A106 4270 STA A          TIEN        ;tien en een
A33D F7 A107 4280 STA B          EEN         ;
A340 39 4290 RTS              ;
A341 4300 ;
A341 4310 ;
A341 4320 ;*****
A341 4330 ;*   SUBROUTINE HEX -> DEC   *
A341 4340 ;*****
A341 4350 ;
A341 4360 ;
A341 4370 ;           INPUT : -
A341 4380 ;           OUTPUT : -
A341 4390 ;DESTRUCTIEREG. INHOUD : A en B A --) MSB VAN 16 BITS GETAL
A341 4400 ;GEBRUIKTEGEHEUGENADR : 6          VOOR HEX NAAR DEC OMREKENING
A341 4410 ;                               B --) LSB VAN 16 BITS GETAL
A341 4420 ;
A341 B6 A117 4430 HEXDEC      : LDA A      FREQBUF + 1 ;MSB
A344 F6 A116 4440 LDA B      FREQBUF      ;LSB
A347 CE A108 4450 LDX      #REKENBUF      ;
A34A 6F 01 4460 CLR       1,X          ;rekenbuf. 0 maken
A34C 6F 02 4470 CLR       2,X          ;
A34E 6F 03 4480 CLR       3,X          ;
A350 A7 04 4490 STA A      4,X          ;HEX waarde in
A352 E7 05 4500 STA B      5,X          ;reken buffer zetten
A354 86 10 4510 LDA A      ##10       ;routine 16* doorlopen
A356 A7 00 4520 STA A      0,X          ;
A358 7E A3E0 4530 JMP      SHIFT        ;
A35B 4540 REPEAT        ;
A35B A6 03 4550 LDA A      3,X          ;
A35D 8D A8 A307 4560 CALL     OMVORM        ;corrigeer nieuw getal
A35F E7 03 4570 STA B      3,X          ;
A361 A6 02 4580 LDA A      2,X          ;
A363 8D A2 A307 4590 CALL     OMVORM        ; idem
A365 E7 02 4600 STA B      2,X          ;
A367 A6 01 4610 LDA A      1,X          ;
A369 8D 9C A307 4620 CALL     OMVORM        ; idem
A36B E7 01 4630 STA B      1,X          ;
A36D 68 05 4640 SHIFT      : ASL       5,X          ;ispart *2, no carry
A36F 69 04 4650 ROL       4,X          ; *2, met carry
A371 69 03 4660 ROL       3,X          ; *2, met carry
A373 69 02 4670 ROL       2,X          ; *2, met carry
A375 69 01 4680 ROL       1,X          ; ispart *2, met carry
A377 6A 00 4690 DEC       0,X          ;
A379 26 E0 A35B 4700 UNTIL EQ ;
A37B 39 4710 RTS              ;
A37C 4720 ;
A37C 4730 ;
A37C 4740 ;*****
A37C 4750 ;*   SUBROUTINE CONVERT FREQ   *
A37C 4760 ;*****
A37C 4770 ;
A37C 4780 ;
A37C 4790 ;           INPUT : -
A37C 4800 ;           OUTPUT : -
A37C 4810 ;DESTRUCTIEREG. INHOUD : X en A X --) TABEL POINTER
A37C 4820 ;GEBRUIKTEGEHEUGENADR : 7 PLAATS WAAR FREQ. STAAT
A37C 4830 ;                               A --) CONVERTEER WAARDE
A37C 4840 ;
A37C CE A103 4850 FREQCONV    : LDX      #TIENDZND      ;begin om zetten tabel
A37F A6 00 4860 REPEAT        ;zet alle waarde om
A37F A6 00 4870 LDA A      0,X          ;naar display code's
A381 FF A10E 4880 STX      SAVEX      ;
A384 8D A296 4890 CALL     CONVERT      ;alleen tienduizend
A387 FE A10E 4900 LDX      SAVEX      ;
A38A A7 00 4910 STA A      0,X          ;tot en met
A38C 08 4920 INX              ;
A38D 8C A108 4930 CPX      #REKENBUF     ;een
A390 26 ED A37F 4940 UNTIL EQ ;
A392 39 4950 RTS              ;

```

```

A393      4960 ;
A393      4970 ;
A393      4980 ;*****
A393      4990 ;*      HOOFD      PROGRAMMA      *
A393      5000 ;*****
A393      5010 ;
A393      5020 ;
A393 8E A790 5030 START      : LDS      #STACKTOP ;init stackpointer
A396 CE A100 5040      : LDX      #FREQUENTIE ;text pointer op
A399 FF A114 5050      : STX      TEXTBUFF ;frequentie
A39C BD A243 5060      : CALL     INITVIA ;initieer de VIA
A39F BD A24F 5070      : CALL     INITSYS ;initieer systeem pia
A3A2 BD A262 5080      : CALL     INITBUFF ;juiste waarde in buf
A3A5 96 10 5090 OPNIEUW : LDA A   DRBVIA ;keuze IN- of Externe
A3A7 84 01 5100      : AND A   #$01 ;triggering
A3A9 26 03 A3AE 5110      : IF EQ   THEN ;
A3AB BD A2CF 5120      : CALL     TRIGPULS ;EXTERN
A3AE      5130      : FI ;INTERN
A3AE BD A2B2 5140      : CALL     TELLAMP ;tellamp aan/uit
A3B1 BD A2C9 5150      : CALL     SETCOUNT ;set frequentie
A3B4 BD A2BB 5160      : CALL     DELAY ;tijdbasis is 1 sec
A3B7 96 1D 5170      : LDA A   IFRVIA ;haal o.v. bit op timer2
A3B9 DE 18 5180      : LDX     TIMER2L ;get pseudo frequentie
A3BB BD A2E0 5190      : CALL     BEOVERF ;freq. overflow ?
A3BE BD A2B2 5200      : CALL     TELLAMP ;tellamp aan/uit
A3C1 BD A2F6 5210      : CALL     CONVREQ ;bepaal juiste freq.
A3C4 BD A341 5220      : CALL     HEXDEC ;bep. dec waarde freq.
A3C7 BD A324 5230      : CALL     PUTCODE ;zet de dec waarde weg
A3CA BD B0 A37C 5240      : CALL     FREQCONV ;zet de dec waarde om
A3CC 20 D7 A3A5 5250      : BRA     OPNIEUW ;in codes voor
A3CE      5260      : ;te displayen
A3CE      5270      : END ;

```



I'm a computer
hobbyist. You fool!

```

SCR# 12
0
1 *****
2 * Memory-Input-Handler *
3 * ===== *
4 * by Klaus Pohlmeij *
5 * 01/87 *
6 *****
7 Hardware : EC65 with 64k RAM, VDU card, and Graphic
8 card (Colorator)
9
10 The following commands should be used to set the I/O
11 tables of the OHIO DOS operating system.
12
13 e.g.: PRINTER ON
14 CRT OFF
15 KEYBOARD ONLY
    
```

```

SCR# 13
0 Up to the present, setting up a FORTH system disc with
1 instruction words from different discs has needed quite
2 a bit of effort and was bound to be full of errors.
3 Also it is not an easy task to rearrange LOAD instruc-
4 tions within one screen while having to change the sys-
5 tem disc. Using the present Memory Input Handler, the
6 rearrangement can be fixed on one disc where it
7 remains available for further use.
8
9 The running program will be set up screen by screen and
10 is started by
11
12 SCRnumber READSTART
13
14 All words will be executed as if typed in directly via
15 the keyboard.
    
```

```

SCR# 14
0 Available words:
1
2 /* The text that follows will be emitted on the CRT
3
4 PAUSE The program stops in order, e.g., to change the
5 disc.
6 /// Continue with the next screen.
7
8 STOP End of program. Control will be taken over by
9 the keyboard.
10
11
12
13
14
15
    
```

```

SCR# 15
0 ( SWITCH ON )
1 : ON ( SWITCH#,ADR -- ) DUP C$ ROT OR SWAP C! ;
2
3 ( SWITCH OFF )
4 : OFF ( SWITCH#,ADR -- ) DUP C$ ROT 255 XOR AND
5 SWAP C! ;
6 ( SWITCH 1 OF 8 )
7 : ONLY ( BIT,ADR -- ) C! ;
8 --/
9
10
11
12
13
14
15
    
```

```

SCR# 16
0 ( INPUT )
1
2 : KEYBOARD ( -- SWITCH,ADR ) 1 8993 ;
3 : NULL ( -- SWITCH,ADR ) 8 8993 ;
4 : MEMORYIN ( -- SWITCH,ADR ) 16 8993 ;
5
6 ( OUTPUT )
7
8 : CRT ( -- SWITCH,ADR ) 1 8994 ;
9 : USEROUT ( -- SWITCH,ADR ) 2 8994 ;
10 : GRAPHIC ( -- SWITCH,ADR ) 49154 8981 ! 4 8994 ;
11 : PRINTER ( -- SWITCH,ADR ) 8 8994 ;
12 : MEMORYOUT ( -- SWITCH,ADR ) 16 8994 ;
13
14 --/
15
    
```

```

SCR# 17
0 ( Memory Input Handler )
1
2 0 VARIABLE ACTSCR
3
4 : MEMORYREAD ( STARTADR -- ) 9098 ! KEYBOARD OFF
5 MEMORYIN ON ;
6 : SCRLOAD ( SCR# -- MEMADR ) CRT OFF GRAPHIC OFF
7 LIST CRT ON PREV $ 2+ FIRST 1024 - DUP /R
8 1024 CMOVE R/ ;
9
10 : INSERTCR ( STARTADR -- ) DUP DUP DUP 1083 + SWAP 63
11 + DO I 13 SWAP C! 64 +LOOP ;
12 --/
13
14
15
    
```

```

SCR# 18
0 ( Memory Input Handler )
1
2 : READSTART ( SCR# -- ) DUP ACTSCR ! SCRLOAD
3 DUP INSERTCR CRT ON MEMORYREAD ;
4
5 : /// EMIT EMPTY-BUFFERS ACTSCR $ 1+ READSTART ;
6
7 : STOP KEYBOARD ONLY QUIT ;
8
9 : PAUSE KEYBOARD ONLY 13 EMIT
10 ." Continue with any Key..." KEY DROP
11 MEMORYIN ONLY ;
12
13 : /* ( Ignore rest of line )
14 IN $ C/L / 1+ C/L * IN ! ; IMMEDIATE
15 ;S
    
```

```

SCR# 19
0 /* Example program for Memory Input Handler
1 /* =====
2 /*
3 /* Insert disc # 1 in drive A !
4 PAUSE
5 /* Loading ....
6 /*
7 10 LOAD 11 LOAD 12 LOAD 34 LOAD
8 40 LOAD 43 LOAD 8 LOAD
9 /* Insert disc # 2 in drive A !
10 PAUSE
11 /* Loading ....
12 /*
13 12 LOAD 23 LOAD
14 ///
15
    
```

```

SCR# 20
0 /* System disc ready !
1 STOP
2
3
4
5
6
7
8
9
10
11
12
13
14
15
    
```

Henk Quast, Dekemastate 15, NL-1275 CM Huizen, 02152-54905 wants to have contact with EPSON FX-85 or EPSON FX-105 users to get a list of the escape and other codes to set the printer and to develop a routine for the DOS65 computer.



I'm a computer hobbyist. You too?

**** COLUMNS PRINT FOR THE ACORN ATOM ****

By: Piet K. de Vries, The Netherlands

With help of: Frank Vergoossen, The Netherlands

INTRODUCTION

This paper is based on an article written by Leif Rasmussen in DE 6502 KENNER 10(1986)6(DEC)44. I adapted his program for the ACORN ATOM and extended it a little. Some features are:

- * possibility to make an empty column
- * possibility to have different columnlengths
- * after the end of textcode (\$03) a form-feed is automatically generated.
- * no output to the screen during printing

HOW IT WORKS

After loading the program, the only thing you have to do is to set the pointer at 208,209 to this routine. No pre-initialising has to be done while the routine checks this itself. After setting the pointer, all characters are normally send to the screen. When 'B' is received, the routine starts putting all characters in a buffer until a form-feed is detected in the input. When this has occurred, the printer starts printing all the characters of the buffer in the first column and all the characters it receives now in the second column. This is done until a second form-feed is detected, which means that a new page is starting which has to be put in the buffer. Printing is stopped after sending \$0C followed by \$03 and all characters are send to the screen again. See the assembler listing for more details.

CONSTANTS AND ADDRESSES OF THE ASSEMBLERCODE

bufpp is an address in the zeropage used for indirect indexed addressing of the buffer

bufposl is the lowbyte of the beginaddress of the buffer

bufposh is the highbyte

maxnum max number of characters per column
 the name says enough
 value normally about 70 (decimal)
 Warning: it's the programmer's responsibility that a line doesn't exceed maxnum characters !!

ASSEMBLY SOURCE LISTING COLUMNS PRINT

```

PHP          ;save PSW
BIT PRFLAG  ;is this character meant for the
             ;printer?
BVC SETOPR  ;yes, so jump
CMP @#2     ;no, then do we have to switch the
             ;printer on?
BEQ SWPRON  ;yes
PLP         ;no, it is a normal character, so
JMP #FE55  ;we send it to the screen
  
```

I SWITCH PRINTER ON I

```

SWPRON STA PRFLAG ;set flag indicating printer is on
PLP     ;restore PSW
JMP #FEFB ;switch it on
  
```

I SEND TO PRINTER I

```

SETOPR PHA          ;save A
        STX XHOLD   ;save X
        STY YHOLD   ;save Y

        LDY @#0
        CPY INITFL  ;do we have to initialise this
                     ;routine?
        BEQ START   ;no, so start
        STY BUFLAG  ;yes, set all flags
        STY INITFL  ;
        JSR RSTBFP  ;and reset the pointer

START  LDY BUFLAG   ;are we storing in the buffer or
        BNE PRCLMN ;printing?
        ;we are printing the right column

        CMP @#3     ;end of text?
        BEQ EXIT1   ;switch printer off and initialise
        STA (BUFFP),Y ;store character in buffer
        JSR INCBFP  ;increment bufferpointer
        CMP @#C     ;end of page?
        BNE EXIT2   ;no then exit and wait for next char
        DEC BUFLAG  ;yes, change flag
        JSR RSTBFP  ;reset to the beginning of buffer
        JMP PRLINE  ;print first line of left column
  
```

I PRINT COLUMN I

```

PRCLMN INY          ;Y := 0
        PHA          ;save character for later
        CMP @#3     ;end of text? (no right column)
        BEQ EMPBF2   ;yes: then empty buffer and switch
                     ;printer off
        CMP @#C     ;end of right column?
        BEQ EMPBF2   ;yes: empty buffer and generate FF
        JSR PROUT    ;no: send character to the printer
        PLA          ;get character back
        CMP @#D     ;did we print a new line?
        BNE EXIT2   ;no: then exit and wait for the next
                     ;character
  
```

I PRINT LINE I

```

PRLINE LDX @#0      ;yes: then print a new line of left
                     ;column
        LDA (BUFFP),Y ;get character from buffer
        CMP @#C     ;end of left column?
        BEQ GENTAB  ;yes: go to the right column
        JSR PRLBUF  ;no: then print a line of the buffer
  
```

I GENERATE TAB I

```

GENTAB LDA @#20     ;load the code for a space
        CPX MAXNUM  ;are we at end of the left column?
        BEQ EXIT2   ;yes: then wait for a character for
                     ;the right column
        JSR PROUT   ;no: then print a space
        INX         ;increment lettercount
        BNE GENTAB+2 ;and continue until we are ready
                     ;(always taken!)
  
```

I PRINTER OUTPUT I

```

PROUT  JMP #FEFB    ;send to printer, not to the screen
  
```

I INCREMENT BUFFERPOINTER I

```

INCBFP INC BUFFP    ;
        BNE READY1  ;
        INC BUFFP+1 ;
READY1 RTS          ;

EXIT1  JSR PROUT    ;switch printer off
        LDA @#FF    ;set all
        STA PRFLAG  ; flags
        LDY @#0     ; back again
        STY BUFLAG  ;
        JSR RSTBFP  ;set pointer to beginning of buffer
EXIT2  LDY YHOLD    ;restore
        LDX XHOLD   ; all
        PLA         ; old
        PLP         ; values
        RTS        ;return to caller
  
```

I EMPTY BUFFER I

```

EMPBF2 LDA @#D      ;generate
        JSR PROUT   ; new
        LDA @#A     ; line
        JSR PROUT   ;
EMPBF1 LDA (BUFFP),Y ;and send the
        CMP @#C     ; rest of the
        BEQ GENFF   ; buffer to the printer
        JSR PROUT   ;
        JSR INCBFP  ;
        BNE EMPBF1  ;always taken
  
```

I GENERATE FORMFEED I

```

GENFF  JSR PROUT    ;send a formfeed to the printer
        PLA         ;get character from the stack
        CMP @#3     ;was it end of text?
        BNE INIT    ;no, then it was end of page
                     ;so initialise and exit
        BEQ EXIT1   ;yes: switch printer off and exit
  
```

I PRINT LINE OF BUFFER I

```

PRLBUF LDA (BUFFP),Y ;get a character from the buffer
CMP @#A ;end of line?
BEQ SKCRLF ;yes: then skip because we have to
;print a right column too
CMP @#C ;end of text?
BEQ READY1 ;yes: then return to sender
INX ;increment letetrcount (for tab)
JSR PROUT ;send the character to the printer
JSR INCBFP ;set pointer to the next character
;in the buffer
BNE PRLBUF ;continue printing characters of
;the buffer
    
```

I SKIP CR AND LF I

```

SKCRLF JSR INCBFP ;skip LF
JMP INCBFP ;skip CR
    
```

I RESET BUFFERP I

```

RSTBFP LDY BUFPOSH ;
STY BUFP+1 ;
LDY BUFPOSL ;
STY BUFP ;
RTS ;
    
```

```

PRLFLAG NOP ;these
INITFL NOP ; are variables
BUFLAG NOP ; which are initialised
XHOLD NOP ; at $EA
YHOLD NOP ;
    
```

FIG-FORTH ON THE C-16

I've implemented FIG-FORTH on the C-16 on tape up to a certain point: the disk and/or tape links still missing. Because of lack of time, I answered to a general call of our new member Claus Vogt from Berlin for cooperation with respect to FORTH on the C-16, his work now being in progress. Anyone interested in joining us?
 Fred Behringer, Strassbergerstrasse 9c/519, D-8000 München 40.

REFLECTED IMAGE

By : Gerard van Roekel, The Netherlands
 EGAMI DETCELFER NI LLA SETIRW EMMARGORP TROHS YREV SIHT

No, this is not a mistake or my texteditor on tilt. Type the following mini-programme, RUN it, and enter your text. Press <CR> or <RETURN>-key and your full text appears on the screen in reflected image.

```
1 INPUTA$:FORA=LEN(A)TOLSTEP-1:PRINTMID$(A$,A,1);:NEXTA
```

BOEKINFORMATIE

Ontvangen van W. van Asperen, Maassluis.

PROGRAMMEREN IN PASCAL
 door: Peter Grogono (vert. uit het Engels)
 uitg: Addison-Wesley Nederland b.v.
 ISBN 90 6789 044 8

- samenvatting uit de inhoud:
- * grondbegrippen van programmeren
 - * gegevens, expressies en toekenningen
 - * beslissing enherhaling (if-while-repeat-for)
 - * procedures en functies
 - * typen variabelen
 - * arrays en records
 - * bestanden
 - * dynamische gegevensstructuren
 - * gecompliceerde onderwerpen
 - * programma ontwerp

Het is een duidelijk boek, vlot leesbaar, met vele geteste programma's als voorbeeld. Het beschrijft de taal volledig en wat erg handig is, het geeft ook syntax diagrammen voor de verschillende instructies in Pascal. Het beschrijft ook een aantal programmeermethoden die gebruikt worden voor de oplossing van de gestelde problemen. Kortom, een aanbeveling waard voor de serieuze Pascal programmeur.

PAPERWARE SERVICE

HARD- AND SOFTWARE FOR A 6522 CONTROLLED EPROM PROGRAMMER.

By: Andrew Gregory, England

Through the paperware service plans and software for an eprom programmer are now available. It can work with any computer which has a spare 6522 VIA. The prototype connects to the Junior VIA as an alternative to a printer.

Specifications:

Hardware:
 Devices programmed : 2716/32/64/128/256/512
 Programming voltages : 0, 5, 12.5, 21, 25 /30mA
 Power supply voltages : 0, 5, 6 /200mA
 Voltage selection : Automatic, by software
 Device selection : Plug-in modules to define device pinning.

Software:
 Written for : Expanded 1MHz Junior with 80 column terminal. (Easily adapted for other 6502 machines.)
 Written on : DOS65 using AS65
 Memory : Code and workspace \$2000 - \$2C00.
 Remaining ram is a buffer area for copying eproms to and from.

Functions : Device select, Program Read, Buffer, Change memory, Hex/Ascii memory dump, Verify, Set buffer address, Check device empty. Address parameters allowed.
 Program speed : Up to 1K per 10 seconds.

Brief description

Data is transferred to the eprom and the 'registers' of the programmer through port A of the VIA while port B controls the mode of port A and the PGM, OE and CS connections to the eprom. There are registers for setting the volts, and the upper and lower bytes of the eprom address. Each of these is formed from a pair of 74HC173 latches. The remaining digital circuit consists of a decoder (74HC138), some buffers and gates. Particular care has been taken to ensure that the latches are not triggered by noise. The eprom voltages are generated with 723 regulators with VFETs to change the voltage by switching in different resistors. The voltages can be switched in 300 microseconds allowing interactive programming techniques.

A full description including circuits (drawn by Herman Zondag) and an assembler listing is available from the paperware service. The program is available on DOS65 disc.

PRICES

a. Paperware of the mentioned article
 Europe : Hfl. 82,00 Outside Europe : Hfl. 99,50
 Members: Hfl. 32,00 Members: Hfl. 49,50
 Eurocheque : subtract Hfl. 9,50.

b. DOS65 disc with object code of the programme
 Europe : Hfl. 72,50 Outside Europe : Hfl. 90,00
 Members: Hfl. 22,50 Members: Hfl. 40,00
 Eurocheque: subtract Hfl. 9,50.

Send your order to the editorial office and mention what you need/what format of the DOS65 disc.

All prices including packages and postages etc. We accept no responsibility for damages etc. during transports.



Tiger.

 * HOLD + APPEND 28 E/N 870327 *

Rapid APPLE version

By: Frans Verberkt
 Hillekensacker 12 - 10
 NL-6546 KG NIJMEGEN
 Phone: 080 - 779555

Translated by: Nico Verberkt

According to the rules of Pieter de Visser from Veldhoven, presented in DE 6502 KENNER nr. 40 page 6, I was able to make this program for APPLE. Since it operates on Microsoft, I expect that the program can also be run on different computers. So I have used the same terminology as the description about this subject of Ruud Uphoff. I suggest that you read his article once more and adjust the routines to your Microsoft-addresses (DE 6502 KENNER nr. 28).

HOW DOES THE PROGRAM OPERATE

The program commences with a start by which the ampersand vector is being set. Namely APPLE knows an instruction: & (return); this is as it were a user-vector to start a machine-language-program (MLX). Is there someone who can explain whether something to that effect is arranged in other computers?

Then the program is not allowed to be destroyed by Basic-programs. So we will secure it setting by HIMEM. After that a statement how you arrive in the "warm" entry. Naturally, using other computers, a CALL or SYS-instruction or something like it is able to serve as message. The "warm" entry sets HIMEM once more, because HIMEM is brought back in the original position by an interim visitation of RESET (Microsoft).

(APPLE RESET = *(CTRL-B) , *E000G)

HOLD as well as APPEND commence with searching the real conclusion of the Basic-program. The reason of this is that it is possible to conceal MLX after a Basic-program, at least with APPLE. So you are not allowed to put together this kind of programs (without sufficient knowledge).

DIRECTIONS FOR USE

You have to prepare your programs !!!

The first program is compelled to have LOWER linenumbers than the second. If this is not the case then this utility can be supplied by a RENUMBER. Who wants to deal with this topic in DE 6502 KENNER ???

YOU HAVE TWO PROGRAMS ON TAPE OR DISC WITH DIFFERENT LINENUMBERS.

- HOLD + APPEND is activated with the "cold" start.
- First load the program with the LOW linenumbers.
- You may list, run or modify this program.
- Start the "warm" entrance of HOLD + APPEND.
- With the instruction (H)old the program is brought beyond the Basic-area. So it cannot be listed, run or something like it.
- Now load the program with the HIGH linenumbers.
- This program may also be listed, run or modified.
- Go again to the "warm" entrance of HOLD + APPEND.
- Choose (A)ppend.
- A list informs you that both programs are linked together.

Note: Do you want to put together more than two programs, then you may use HOLD as many times as you wish.

So: HOLD transports the program from "Basic" to "Hold", this being the case together with a program already present in "Hold". APPEND transports the program from "Hold" to "Basic", this being the case together with a program already present in "Basic".

Finally the total program is compelled to have successive and exclusive linenumbers !!!

 * HOLD + APPEND 28 E/N 870327 *

Snelle APPLE versie

Door: Frans Verberkt
 Hillekensacker 12 - 10
 NL-6546 KG NIJMEGEN
 Phone: 080 - 779555

Vertaald door: Nico Verberkt

Volgens de regels van Pieter de Visser uit Veldhoven, gepresenteerd in DE 6502 KENNER nr. 40 blz. 6, heb ik voor APPLE dit programma kunnen maken. Angezien dit op Microsoft werkt, verwacht ik dat de programmatuur ook te runnen is op andere computers en heb dus dezelfde terminologie aangehouden als de beschrijving hierover door Ruud Uphoff; lees het betreffende artikel dus nog eens over, en pas de routines aan Uw Microsoft adressen aan. (DE 6502 KENNER nr. 28)

HOE WERKT HET PROGRAMMA?

Het programma begint met een start waarbij de ampersand vector gezet wordt. APPLE kent namelijk een opdracht & (return), dit is als het ware een gebruikersvector om een machinetaalprogramma te starten. Kan iemand eens uitleggen of iets dergelijks ook in andere computers geregeld is?

Vervolgens mag dit programma niet kapot gemaakt worden door Basic-programma's, dus we gaan dit beveiligen met het zetten van HIMEM. Dan een mededeling hoe in de "warme" ingang te komen. Voor andere computers kan natuurlijk een CALL of SYS opdracht of iets dergelijks als boodschap dienen. De warme ingang zet nogmaals HIMEM, omdat bij een tussentijds bezoek van RESET (Microsoft) HIMEM weer in de originele stand gebracht wordt.

(APPLE RESET = *(CTRL-B) , *E000G)

Zowel HOLD als APPEND beginnen met het werkelijke einde van het Basicprogramma te zoeken. Dit, omdat het (bij APPLE althans) mogelijk is machinetaal te verstopten na een Basic programma: dit soort programma's moet u dus niet (zonder kennis) aan elkaar voegen.

GEBRUIKSAANWIJZING

U moet Uw programma's wel voorbereiden !!!

Het eerste programma moet LAGERE regelnummers voeren dan het tweede programma. Als dit niet het geval is, dan kan deze utility geleverd worden door een RENUMBER programma. Wie wil dit eens een keer behandelen voor DE 6502 KENNER?

U HEEFT TWEE PROGRAMMA'S OP TAPE OF DISK MET VERSCHILLENDE REGELNUMMERS.

- HOLD + APPEND wordt geactiveerd met de "koude" ingang.
- Laadt eerst het programma met de LAGE regelnummers.
- U mag dit programma listen, runnen en eventueel wijzigen
- Start de "warme" ingang van HOLD + APPEND.
- Met de opdracht (H)old wordt het programma buiten het Basic gebied gebracht en is dus niet meer te listen, runnen en dergelijke.
- Laadt nu het programma met de HOGE regelnummers.
- Ook dit programma is eventueel te listen, runnen en wijzigen.
- Opnieuw naar de "warme" ingang van HOLD + APPEND.
- Kies (A)ppend.
- Een list leert U nu dat beide programma's aan elkaar geplakt zijn.

Noot: Heeft U meer programma's aan elkaar te voegen, dan mag U HOLD net zo vaak gebruiken als U lief is.

Dus: HOLD brengt een programma van "Basic" naar "Hold" eventueel tezamen met een programma dat al in "Hold" aanwezig is. APPEND haalt het programma van "Hold" naar "Basic" eventueel tezamen met een programma dat al in "Basic" aanwezig is.

Het totale programma moet uiteindelijk opeenvolgende en van elkaar afwijkende regelnummers bezitten !!!

```

1930 ; ##### PAGE ZERO #####
1940
1950 TEMP      .DE $18                ;TEMPORARY MEMORY
1960
1990 ; ##### BASIC POINTERS PAGE ZERO #####
2000
2010 BEGP      .DE PTRSTRT            ;BEGIN POINTER
2020 ENDP1     .DE PTRSTRT+$02        ;END POINTER
2030 HIM1      .DE PTRSTRT+$08        ;(HIMEM)
2040 HIM2      .DE PTRSTRT+$0C        ;(HIMEM) END OF MEMORY
2050
2080 ; ##### APPLESOFT PAGE ZERO #####
2090
2100 ; other computers: ENDP2 .DE ENDP1
2110
2120 ENDP2     .DE $AF                ;END POINTER COPY
2130
2160 ; ##### PAGE $03 APPLE #####
2170
2180 AMPERSAND .DE $3F5              ;& VECTOR
2190
2220 ; MICROSOFT BASIC
2230
2240 ; Other computers:
2250
2260 ; DE 6502 KENNER nr. 28 page 22 - 26
2270
2280 PTRSTRT   .DE $67                ;SERIE BASIC POINTERS AT
                                           ;PAGE ZERO
2290 TXTAREA   .DE $800              ;BEGIN MEMORY
2300 PRSTRYA   .DE $DB3A            ;PRINT STRING
2310 RESTART   .DE $E003            ;WARM ENTRY BASIC
2320
2350 ; ##### APPLE ROUTINES #####
2360
2380 CLSC      .DE $FC58              ;CLEAR SCREEN
2390 KEYIN     .DE $FDOC              ;GET KEY
2400 BEEP      .DE $FF3A
2410
2440 ; ##### HOLD + APPEND #####
2450
2460           .OS
2470           .BA $9000
2480
9000- 4C 06 90 2490 STARTCOLD JMP COLD
9003- 4C 39 90 2510 STARTWARM JMP WARM
2520
2550 ; ##### COLD ENTRY #####
2560
9006- 20 1C 90 2570 COLD      JSR SETAMP      ;SET & VECTOR
9009- 20 2C 90 2580          JSR SETHIM      ;SET HIMEM
900C- 20 58 FC 2590          JSR CLSC       ;CLEAR SCREEN
900F- A9 CA 2600          LDA #L, TXTCOLD
9011- A0 90 2610          LDY #H, TXTCOLD
9013- 20 3A DB 2620          JSR PRSTRYA     ;PRINT MESSAGE
9016- A2 FA 2630 BASIC     LDX #$FA      ;RESET STACK
9018- 9A 2640          TXS
9019- 4C 03 E0 2650          JMP RESTART    ;BACK TO BASIC
2660
2690 ; ##### SET ROUTINE'S (INIT) #####
2700
901C- A9 4C 2710 SETAMP     LDA #$4C      ;SET AMPERSANDVECTOR
901E- 8D F5 03 2720          STA AMPERSAND    ; WITH JUMP
9021- A9 03 2730          LDA #L, STARTWARM
9023- 8D F6 03 2740          STA AMPERSAND+1
9026- A9 90 2750          LDA #H, STARTWARM
9028- 8D F7 03 2760          STA AMPERSAND+2
902B- 60 2770          RTS
902C- A9 00 2790 SETHIM     LDA #L, STARTCOLD
902E- 85 6F 2800          STA *HIM1

```

```

9030- 85 73      2810      STA *HIM2
9032- A9 90      2830      LDA #H,STARTCOLD
9034- 85 70      2840      STA *HIM1+1
9036- 85 74      2850      STA *HIM2+1
9038- 60         2870      RTS
                2880
                2910 ; #### WARM ENTRY ####
                2920
9039- 20 2C 90   2930 WARM      JSR SETHIM          ;SET HIMEM
903C- A9 03      2950      LDA #L,TXTWARM
903E- A0 91      2960      LDY #H,TXTWARM
9040- 20 3A DB   2970      JSR PRSTRYA        ;PRINT MESSAGE
9043- 20 0C FD   2990 KEYLOOP  JSR KEYIN
9046- 29 7F      3000      AND #$7F           ;APPLE KEY > $80
9048- C9 48      3020 KEYHOLD  CMP #'H            ;HOLD?
904A- F0 04      3030      BEQ HOLDGO
904C- C9 68      3040      CMP #'h
904E- D0 0D      3050      BNE KEYAPP
9050- A9 19      3060 HOLDGO   LDA #L,TXTHOLD
9052- A0 91      3070      LDY #H,TXTHOLD
9054- 20 3A DB   3080      JSR PRSTRYA
9057- 20 78 90   3100      JSR HOLD
905A- 4C 16 90   3110      JMP BASIC
905D- C9 41      3130 KEYAPP   CMP #'A            ;APPEND?
905F- F0 04      3140      BEQ APPENDGO
9061- C9 61      3150      CMP #'a
9063- D0 0D      3160      BNE KEYERR
9065- A9 1F      3170 APPENDGO  LDA #L,TXTAPP
9067- A0 91      3180      LDY #H,TXTAPP
9069- 20 3A DB   3190      JSR PRSTRYA
906C- 20 89 90   3210      JSR APPEND
906F- 4C 16 90   3220      JMP BASIC
9072- 20 3A FF   3240 KEYERR   JSR BEEP           ;ERROR BEEP
9075- 4C 43 90   3250      JMP KEYLOOP
                3260
                3290 ; #### HOLD ####
                3300
9078- 20 95 90   3320 HOLD     JSR SEARCH        ;SEARCH END OF BASIC
907B- 38         3330      SEC              ;BEGINPOINTER = ENDPOINTER-2
907C- A5 69      3340      LDA *ENDP1
907E- E9 02      3350      SBC #$02
9080- 85 67      3360      STA *BEGP
9082- A5 6A      3380      LDA *ENDP1+1
9084- E9 00      3390      SBC #$00
9086- 85 68      3400      STA *BEGP+1
9088- 60         3420      RTS
                3430
                3460 ; #### APPEND ####
                3470
9089- 20 95 90   3480 APPEND   JSR SEARCH        ;SEARCH END OF BASIC
908C- A9 01      3500      LDA #L,TXTAREA+1 ;BASIC MEMORY START AT
908E- 85 67      3510      STA *BEGP        ; TXTAREA (MICROSOFT)
9090- A9 08      3520      LDA #H,TXTAREA+1
9092- 85 68      3530      STA *BEGP+1
9094- 60         3540      RTS
                3550
                3580 ; #### SUBROUTINE SEARCH ####
                3590
9095- A5 67      3600 SEARCH   LDA *BEGP        ;BEGIN ==> TEMP
9097- 85 18      3610      STA *TEMP
9099- A5 68      3620      LDA *BEGP+1
909B- 85 19      3630      STA *TEMP+1
909D- A0 00      3640 SEARCHLOOP LDY #$00
909F- B1 18      3650      LDA (TEMP),Y    ;GET LINKADDRESS(L)
90A1- D0 17      3660      BNE SEARCHGO   ;NO END OF BASIC
90A3- C8         3670      INY
90A4- B1 18      3680      LDA (TEMP),Y    ;GET LINKADDRESS(H)
90A6- D0 12      3690      BNE SEARCHGO   ;NO END OF BASIC
90A8- 18         3710      CLC              ;IF LINKADDRESS = $0000
90A9- A5 18      3720      LDA *TEMP      ; THEN END OF PROGRAM

```

```

90AB- 69 02      3730      ADC #\$02          ;ENDPOINTER + 2
90AD- 85 69      3740      STA *ENDP1
90AF- 85 AF      3750      STA *ENDP2
90B1- A5 19      3770      LDA *TEMP+1
90B3- 69 00      3780      ADC #\$00
90B5- 85 6A      3790      STA *ENDP1+1
90B7- 85 B0      3800      STA *ENDP2+1
90B9- 60         3810      RTS
90BA- A0 00      3840 SEARCHGO     LDY #\$00
90BC- B1 18      3850      LDA (TEMP),Y     ;LINKADDRESS (L)
90BE- 48         3860      PHA
90BF- C8         3870      INY
90C0- B1 18      3880      LDA (TEMP),Y     ;LINKADDRESS (H)
90C2- 85 19      3900      STA *TEMP+1      ;LINKADDRESS ==> TEMP
90C4- 68         3910      PLA
90C5- 85 18      3920      STA *TEMP
90C7- 4C 9D 90   3940      JMP SEARCHLOOP
3950
3980 ; ##### TEXT #####
3990
90CA- 49 46 20   4000 TXTCOLD     .BY 'IF YOU WANT HOLD + APPEND....' 13 13
90CD- 59 4F 55
90D0- 20 57 41
90D3- 4E 54 20
90D6- 48 4F 4C
90D9- 44 20 2B
90DC- 20 41 50
90DF- 50 45 4E
90E2- 44 2E 2E
90E5- 2E 2E 0D
90E8- 0D
90E9- 50 52 45   4010     .BY 'PRESS ..... & <RETURN>' 13 13
90EC- 53 53 20
90EF- 2E 2E 2E
90F2- 2E 2E 2E
90F5- 20 26 20
90F8- 3C 52 45
90FB- 54 55 52
90FE- 4E 3E 0D
9101- 0D
9102- 00         4020
9103- 28 48 29   4030 TXTWARM     .BY 0
                      .BY '(H)OLD OR (A)PPEND ? '
9106- 4F 4C 44
9109- 20 4F 52
910C- 20 28 41
910F- 29 50 50
9112- 45 4E 44
9115- 20 3F 20
9118- 00         4040     .BY 0
9119- 48 4F 4C   4050 TXTHOLD     .BY 'HOLD' 13
911C- 44 0D
911E- 00         4060     .BY 0
911F- 41 50 50   4070 TXTAPP      .BY 'APPEND' 13
9122- 45 4E 44
9125- 0D
9126- 00         4080     .BY 0
9127- 77         4110     .BY \$77          ;EOF SIGN
                      4130     .EN

```

SYSTEM: DOS65 2.01

PROGRAMMA TIMEDATE

(simpel alternatief voor een real time clock kaart)
 Dit programma vraagt de tijd en de datum af. Geldige invoer wordt
 in de monitor variabelen voor deze parameters gezet. Wanneer alleen
 een CR wordt ingegeven wordt niets veranderd. De routine kan in de
 LOGIN.COM file worden aangeroepen.

PROGRAM TIMEDATE

(Simple alternative for real time clock cart)
 This routine prompts for time and date. If valid data is entered
 the monitor variables for these parameters are updated. If CR is
 entered no changes are made. The routine can be called from the
 LOGIN.COM file.

Peter Roessingh, March 1986
 Changed to DOS65 2.01 December 1986

ORG \$2000

*** External addresses***

DATUPD	equ	\$E77E	; flag for date update
DAY	equ	\$E77F	; day register
MONTH	equ	\$E780	; month register
YEAR	equ	\$E781	; year register
HOURS	equ	\$E782	; hour register
MINUTES	equ	\$E783	; minute register
SECONDS	equ	\$E784	; seconds register

*** Temporary storage locations ***

SVSTCK	res	1	; temp for stackpointer
INDEX	res	1	; temp for bufferindex

*** Library files ***

lib	INOUT	; DOS65 I/O entry's
lib	GETPAR	; read decimal parameters
lib	BOUND8	; check boundaries, 8 bit

*** Main program starts here ***

TIMDAT	JSR	DATLOOP	; get date and update parameters
	JSR	TIMLOOP	; get time and update parameters
EXIT	JSR	CRLF	; print crlf
	JSR	CRLF	; and another one
	RTS		; end of program timedate

*** subroutine DATLOOP, get new date, loop until valid ***

DATLOOP	JSR	PRINT	; start loop: ask for new date
	fcc	\$0D,\$0A,\$0A,'Today\'s date: ',0	
	JSR	BUFIN	; get new date in buffer
	JSR	DODATE	; try to set date
	BCS	ERRDAT	; on error continue in loop


```

BNE      DODERR      ; error exit if not zero
;
STY      INDEX       ; save index
LDY      #$01        ; lower limit 1
LDX      #$0C        ; higher limit 12
JSR      BOUNDS8    ; check boundaries
BCS      DODERR      ; error exit

PHA      ; result on stack

*** get year

LDY      INDEX       ; pass index in Y index
LDA      #$0D        ; pass delimiter in A
JSR      GETPAR      ; get parameter
BCS      DODERR      ; error exit
CPX      #$00        ; test high byte
BNE      DODERR      ; error exit if not zero

STY      INDEX       ; save index
LDY      #$57        ; lower limit 1987
LDX      #$63        ; higher limit 1999
JSR      BOUNDS8    ; check boundaries
BCS      DODERR      ; error exit

PHA      ; result on stack

*** set the date

PLA      ; get year back
STA      YEAR        ; set year
PLA      ; get month back
STA      MONTH       ; set month
PLA      ; get day back
STA      DAY         ; set day
LDA      #$FF        ; get flag
STA      DATUPD      ; and set it

*** update the date in the statusline

BCC      DODEND      ; branch always
DODERR   LDX      SVSTCK ; get saved stackpointer
          TXS      ; restore stackpointer
          SEC      ; indicate error
DODEND   RTS         ; return from dodate

*****
Routine to get and set the time
DOTIME   JSR      BEGIN ; init bufferpointer, test on CR
          BEQ      DOTEND ; CR found, immediate exit

          TSX      ; get stackpointer
          STX      SVSTCK ; and save it

*** get hours

LDY      INDEX       ; pass index in Y index
LDA      #':'        ; pass delimiter in A
JSR      GETPAR      ; get parameter
BCS      DOTERR      ; error exit
CPX      #$00        ; test high byte
BNE      DOTERR      ; error exit if not zero

STY      INDEX       ; save index
LDY      #$00        ; lower limit 0
LDX      #$17        ; higher limit 23
JSR      BOUNDS8    ; check boundaries
BCS      DOTERR      ; error exit

```

```

PHA                                ; result on stack
;
;
*** get minutes
LDY    INDEX                       ; pass index in Y index
LDA    #$0D                         ; pass delimiter in A
JSR    GETPAR                       ; get parameter
BCS    DOTERR                       ; error exit
CPX    #$00                         ; test high byte
BNE    DOTERR                       ; error exit if not zero

STY    INDEX                       ; save index
LDY    #$00                         ; lower limit 00
LDX    #$3B                         ; higher limit 59
JSR    BOUND8                       ; check boundaries
BCS    DOTERR                       ; error exit

PHA                                ; result on stack

*** set time ***
PLA                                ; get minutes back
STA    MINUTES                      ; set minutes
PLA                                ; get hours back
STA    HOURS                        ; set hours
LDA    #$00                         ; seconds are zero
STA    SECONDS                      ; set seconds

BCC    DOTEND                       ; branch always

DOTERR LDX    SVSTCK                 ; get saved stackpointer
TXS                                ; restore stackpointer
SEC                                ; indicate error

DOTEND RTS                          ; return from dotime

*****
*** routine to init bufferpointer
BEGIN  LDA    #$00                   ; init index to buffer
STA    INDEX
;
TAY                                ; point to begin buffer
JSR    GETBUF                       ; get first char
BEQ    1.f                          ; CR found, immediate exit

LDA    #$FF                         ; signal normal exit
RTS

*****
END    TIMDAT                       ; end of program

```

```

*****
Library file INOUT.MAC  DOS65 i/o entry's
*****
IN      equ    $C020                 ; get char => A C=0
OUT     equ    $C023                 ; put char => A C=0
INECHO  equ    $C026                 ; get and put char
BUFIN   equ    $C029                 ; get line in inputbuffer
GETBUF  equ    $C02C                 ; get from buffer, Y A; check eol
CRLF    equ    $C02F                 ; print CRLF
HEXOUT  equ    $C038                 ; A hex out
PRINT   equ    $C03B                 ; print string after call
*****

```

Library file GETPAR.MAC Subroutine getpar

This routine gets a decimal parameter from the DOS65 inputbuffer and converts it to hex. The delimiter for the parameter should be in A, the startposition in the buffer in Y. The hex result is given back in A and X. For the algorithm, see Leventhal and Saville (1982).

The library file INOUT.MAC is expected to be present in the calling program.

Call: A delimiter
Y buffer index

Return A low byte result (hex)
X high byte result (hex)
Y buffer index
C error flag

*** temporarys ***

RETADR res 2 ; return address
DLMTR res 1 ; delimiter
ACCU M res 2 ; result (2 bytes)
NGFLAG res 1 ; flag for negative number

GETPAR STA DLMTR ; save delimiter
LDA #\$00
STA ACCUM ; init result
STA ACCUM+1 ; high byte also
STA NGFLAG ; default positive

PLA ; get return address
STA RETADR ; save it
PLA
STA RETADR+1

JSR GETBUF ; get character
CMP #'-' ; test on minus
BNE PLUS ; branch if not minus
LDA #\$FF ; get flagbyte
STA NGFLAG ; indicate negative number
INY ; skip past minus sign
JMP CNVERT ; start conversion

PLUS CMP #'+' ; test on plus sign
BNE CNVERT ; branch if not plus
INY ; skip past plus sign

CNVERT JSR GETBUF ; get character
CMP DLMTR ; test if delimiter
BEQ 2.f ; exit if found
CMP #'0' ; test if smaller than 0
BMI 4.f ; error, < 0 (not a digit)
CMP #'9'+1 ; test if greater than 9
BPL 4.f ; error, > 9 (not a digit)
PHA ; save digit on stack

*** Valid decimal digit, convert to hex ***

ASL ACCUM
ROL ACCUM+1 ; * 2
LDA ACCUM
LDX ACCUM+1 ; save result * 2
ASL ACCUM
ROL ACCUM+1
ASL ACCUM
ROL ACCUM+1 ; * 8
CLC

2-Apr-87 23:00 getpar.mac Page 2

```

ADC      ACCUM      ; sum with * 2
STA      ACCUM
TXA
ADC      ACCUM+1
STA      ACCUM+1    ; result := result * 10
;
next digit
PLA      ; get digit back
SEC
SBC      #'0'      ; convert digits to binary
CLC
ADC      ACCUM
STA      ACCUM
BCC      1.f       ; branch if no carry to high byte
INC      ACCUM+1   ; else increment high byte
;
1  INY      ; point to next character
   JMP      CNVERT ; continue convert
;
2  LDA      NGFLAG  ; get signindication
   BPL      3.f     ; branch if value was positive
   LDA      #$0     ; else replace result with neg result
   SEC
   SBC      ACCUM
   STA      ACCUM
   LDA      #$0
   SBC      ACCUM+1
   LDA      ACCUM+1
;
3  CLC      ; indicate no error
   BCC      5.f     ; and exit
   SEC      ; indicate error
;
*** exit ***
;
5  LDA      RETADR+1
   PHA
   LDA      RETADR
   PHA
;
   INY      ; bufferpointer passed in Y
   LDA      ACCUM  ; low byte is passed in A
   LDX      ACCUM+1 ; high byte is passed in X
;
   RTS
   END      GETPAR ; end of routine

```

2-Apr-87 22:53 bound8.mac Page 1

```

*****
Library fille BOUND8.MAC      Subroutine bound8
This routine tests if a 8 bit parameter falls between
two 8 bit boundaries. If this is true the routine returns
with the C flag clear, otherwise C=1
Call:   A parameter
        X higher limit
        Y lower limit
Return  A parameter
        C errorflag
Peter Roessingh December 1986
*****
temporary locations
PARAM  res      1
;
*** start of routine ***

```

```

;
; BOUNDS8 STA PARAM ; save parameter
;
; CPY PARAM ; test lower limit
; BEQ 1.f ; equal is o.k.
; BCS 2.f ; par too low, error
;
; 1 CPX PARAM ; test high limit
; BCC 2.f ; par too high, error
;
; CLC ; flag no error
; BCC 3.f ; branch always, normal exit
;
; 2 SEC ; flag error
; 3 RTS ; and return
;
; END BOUNDS8 ; end of routine
;*****

```

TIME FOR THE BBC AND ELECTRON

With this program it's possible to constantly print the time in the lefttop corner of the screen. Unfortunately, there are some limitations. The program will only work in one-colour modes (mode 0, 3, 4, 6). The program also doesn't work with a 'TUBE' or shadowram. To set the time, type: *TIME hhmmss. The program doesn't check whether you typed a valid time, to save memory. To disable the time, type *FX13,5. To start the time again, press [BREAK] and call the program again (CALL &900). You may read the time from your own machine-code programs by reading the addresses clock, clock+1, clock+2 (see listing). These addresses subsequently contain the seconds, minutes and hours in BCD. The zero-page addresses &71 until &76 are used by the program.

TIJD IN BEELD VOOR DE BBC EN ELECTRON

Met dit programma kunt u konstant de tijd in de linkerbovenhoek van het scherm zetten. Helaas zijn er enige beperkingen. Ten eerste werkt het programma alleen in een-kleur modes (mode 0, 3, 4, 6) en ten tweede werkt het niet met een 'TUBE' of met shadowram. Om de tijd goed te zetten tikt u in: *TIME uuumss. Er wordt niet getest of u een geldige tijd intikt om geheugen te sparen. U kunt de tijd uitzetten door *FX13,5 in te tikken. Om de tijd weer op te roepen, drukt u op [BREAK] en vervolgens roept u het programma veer aan (CALL &900). U kunt de tijd in uw eigen machinetaalprogramma's oproepen in de geheugenadressen clock, clock+1 en clock+2 (zie listing). Hier staan achtereenvolgens in BCD de seconden, minuten en uren. De zero-page adressen &71 t/m &76 worden gebruikt door het programma.

```

10 REM Clock for the Electron and BBC
20 REM By Ronald van Vugt (PA3EAH)
30 MODE 6
40 osbyte=&FFF4
50 osword=&FFF1
60 teller=&71
70 screen_low=&350:screen_high=&351
80 REM !!T!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
90 REM If you use the OS EPROM/ROM 0.10
100 REM in the BBC, then change screen_low
110 REM into &322 and and screen_high
120 REM into &323.
130 REM !!T!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
140 start%=&900
150 FOR pass%=0 TO 3 STEP3
160 Px=start%
170 OPT pass%
180 LDA &208:STA &73
190 LDA &209:STA &74
200 SEI
210 LDA #begin MOD256:STA &220
220 LDA #begin DIV256:STA &221
230 LDA #oscli MOD256:STA &208
240 LDA #oscli DIV256:STA &209
250 CLI
260 LDA #0:STA &72
270 JSR loadT:LDA #14:LDX #5:JMP osbyte
280 .begin LDA &72:BNE print
290 JSR loadT:LDX #0:SEI:SED
300 .lus1 LDA #0:SEC:ADC clock,X:STA clock,X
310 CMP #96:BNE rts:LDA #0:STA clock,X
320 INX:CPX #1:BEQ lus1
330 SEC:ADC clock+2:STA clock+2
340 CMP #36:BNE rts:LDA #0:STA clock+2
350 .rts CLD:CLI:INC &72:RTS
360 .print JSR loadT:LDX #2
370 LDA screen_low:STA pokes+1
380 LDA screen_high:STA pokes+2
390 .lus2 LDA #7:STA teller:LDA clock,X:PHA
400 LSR A:LSR A:LSR A:LSR A
410 JSR prnt1:LDA #7:STA teller:PLA:JSR prnt1
420 DEX:BMI rts1:CPX #0:BPL dubbl:BNI lus2
430 .dubbl LDA #7:STA teller:LDA #10:JSR prnt1:BMI lus2
440 .prnt1 AND #15:ASL A:ASL A:ASL A:TAY
450 .lus3 LDA &C080,Y
460 .pokes STA &FFFF:INC pokes+1
470 INY:DEC teller:BPL lus3:RTS
480 .loadT LDA #206:STA block:LDA #4
490 LDX #block MOD256:LDY #block DIV256:JMP osword
500 .rts1 DEC &72:RTS
510 .clock EQUW 0:EQUB 0
520 .block EQUW 0:EQUD &FFFFFFF
530 .oscli STX &75:STY &76:LDY #1
540 .oslus LDA (&75),Y:AND #223:CMPI cmd,Y
550 BNE false:INY:CPY #5:BNE oslus
560 LDX #2
570 .tilus INY:LDA #0:STA teller:LDA (&75),Y
580 SEC:SBC #ASC"0":ASL A:ASL A
590 ASL A:ASL A:STA teller
600 INY:LDA (&75),Y:SEC:SBC #ASC"0":ORA teller
610 STA clock,X:DEX:BPL tilus:RTS
620 .false LDX &75:LDY &76:JMP (&73)
630 .cmd EQUW " *TIME" \kan gewijzigd worden in *TIJD
640 ]
650 NEXT pass%
660 MODE6
670 CALL start%

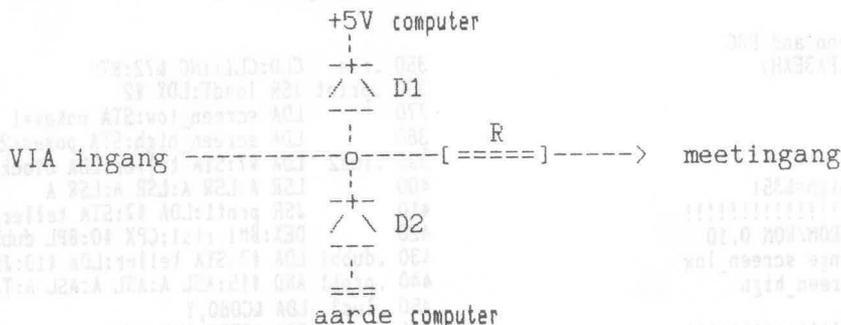
```

**** Ingangsprotectie LOCIG ANALYSER voor DOS-65 V2.01 ****

H. A. J. Quast
 Dekemastate 15
 1275CM Huizen
 tel. 02152-54905

Ik ben sinds korte tijd in het bezit van het programma "LOGICANALYSER" voor de DOS-65 computer. Dit fraaie programma biedt de mogelijkheid om 8 verschillende digitalen signalen plus een triggersignaal op het scherm zichtbaar te maken. Met dit programma kunnen dus allerlei tijdrelatie's van digitale signalen in een schakeling gemeten worden. Voor het meten van de 8 signalen wordt de VIA-2A gebruikt. Wanneer we nu in een ander apparaat dan de DOS-65 computer deze signalen willen meten en dit apparaat wordt niet vanuit de computer gevoed, dan kan het volgende probleem ontstaan. Het is mogelijk dat de nul van het te testen apparaat niet aan de aarde van de computer zit. In dat geval kan er soms een aanzienlijke spanning ontstaan tussen de aarde van de computer en de nul van het apparaat. Wanneer we nu bij het meten in het apparaat de aarde eerst aansluiten dan is dit probleem over omdat het te meten apparaat de aarde van de computer overneemt. Het wordt echter vervelend wanneer we de nul nog los hebben en we sluiten een van de ingangen van de VIA aan op het te meten apparaat. Stel dat de nul van het apparaat op +5V ligt t.o.v. de aarde van de computer, dan is dus een logic "hoog" niveau in het apparaat +10V t.o.v. de computeraarde. We zetten nu dus 10V op de ingang van de VIA. Het gevolg hiervan is dat we waarschijnlijk de ingang opblazen. Nog gemener zijn spanningspieken die enkele tientallen volts kunnen bedragen.

Om de ingangen te beschermen tegen deze overspanning kan voor elke ingang de volgende schakeling gebruikt worden.



In deze schakeling beschermt de diode D1 de ingang tegen spanningen die hoger zijn dan $\approx +5,6V$ omdat boven deze spanning de diode in geleiding komt.

Diode D2 beschermt de ingang tegen spanningen die lager zijn dan $\approx -0,6V$. Voor diode D2 zou eigenlijk een Schottky of een germanium diode gebruikt moeten worden aangezien deze een lagere doorlaat- spanning hebben. De serieweerstand dient als stroombegrensings- weerstand. Hoe groter de weerstand R is des te lager is de belasting op de te meten schakeling. Met één ding moet echter wel rekening worden gehouden: aangezien de ingang van de VIA en de diode een bepaalde capaciteit hebben zal er bij een signaal met een hoge frequentie vervorming van de flanken ontstaan.

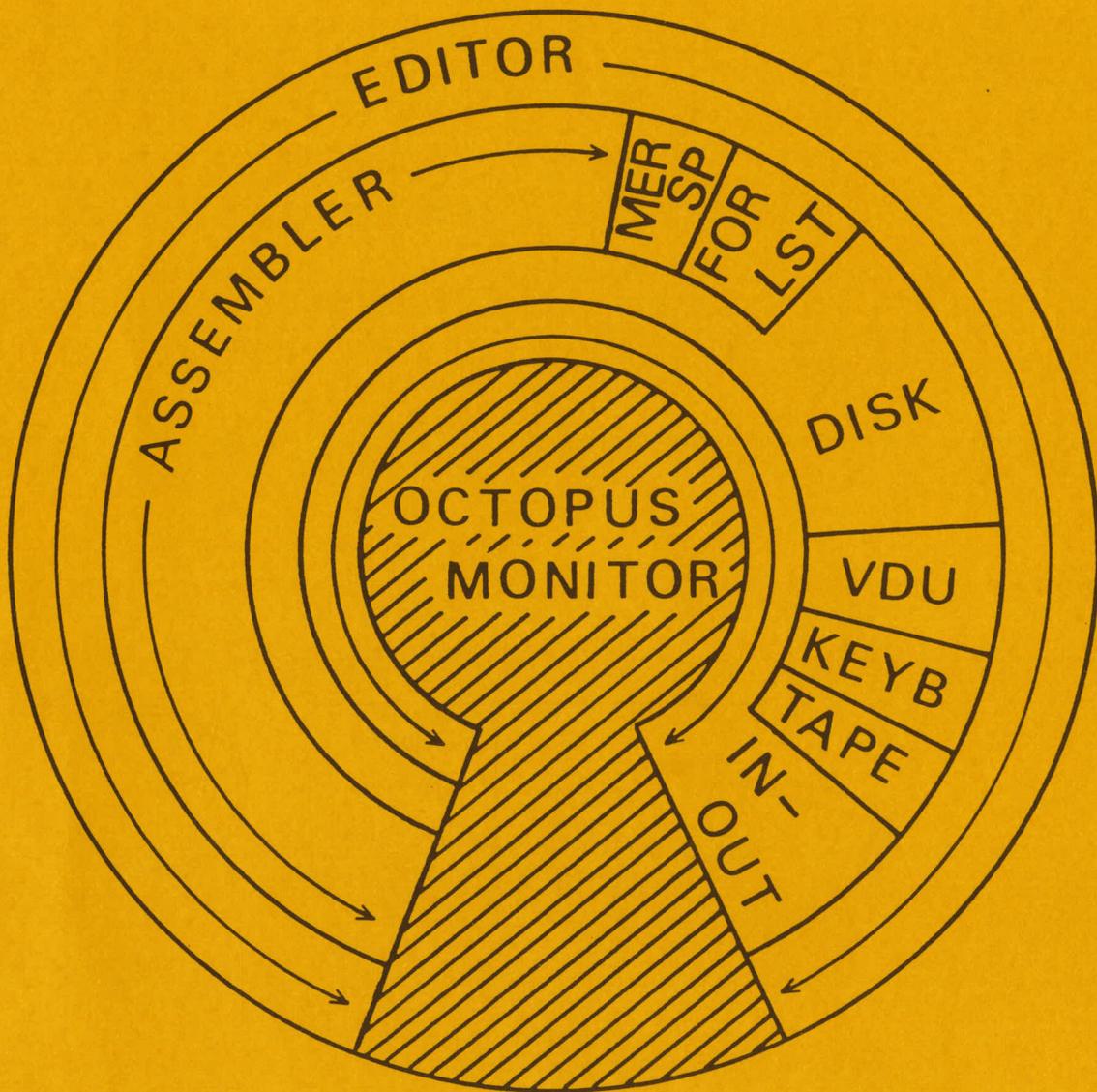
Een bruikbare waarde voor $R = 5\text{ k}\Omega$ tot $10\text{ k}\Omega$

Voor de diode D1 en D2 kan één van de onderstaande type gebruikt worden.

Diode D1	V_R (V)	Diode D2	V_R (V)
1N4148	75 V	BAT 82	50 V
BAW 62	75 V	BAT 83	60 V
BAV 20	150V	BAT 86	50 V

OCTOFATE v2-1

SYSTEM DIAGRAM



ML

Fabelachtig printen in kleur of zwart/wit



OKIMATE 20



Futuristische OKI-afdrukkers erkend de beste in Nederland. Nu ook voor iedereen betaalbaar geworden. Aansluitbaar op Commodore 64, BBC, MSX e.a. homecomputers.



OKI MICROLINE 182

OKImate 20 v.a. fl. 985.--
Microline 182 v.a. fl. 1170.--
excl. b.t.w. Bruto adviesprijs.

OKI printers, door computers voor computers.

Vraag de dealerlijst bij de officiële importeur

Technitron B.V.

Zwarteweg 110, Postbus 14, 1430 AA Aalsmeer

tel. 02977-22456 - telefax 02977-40968 - telex 13301