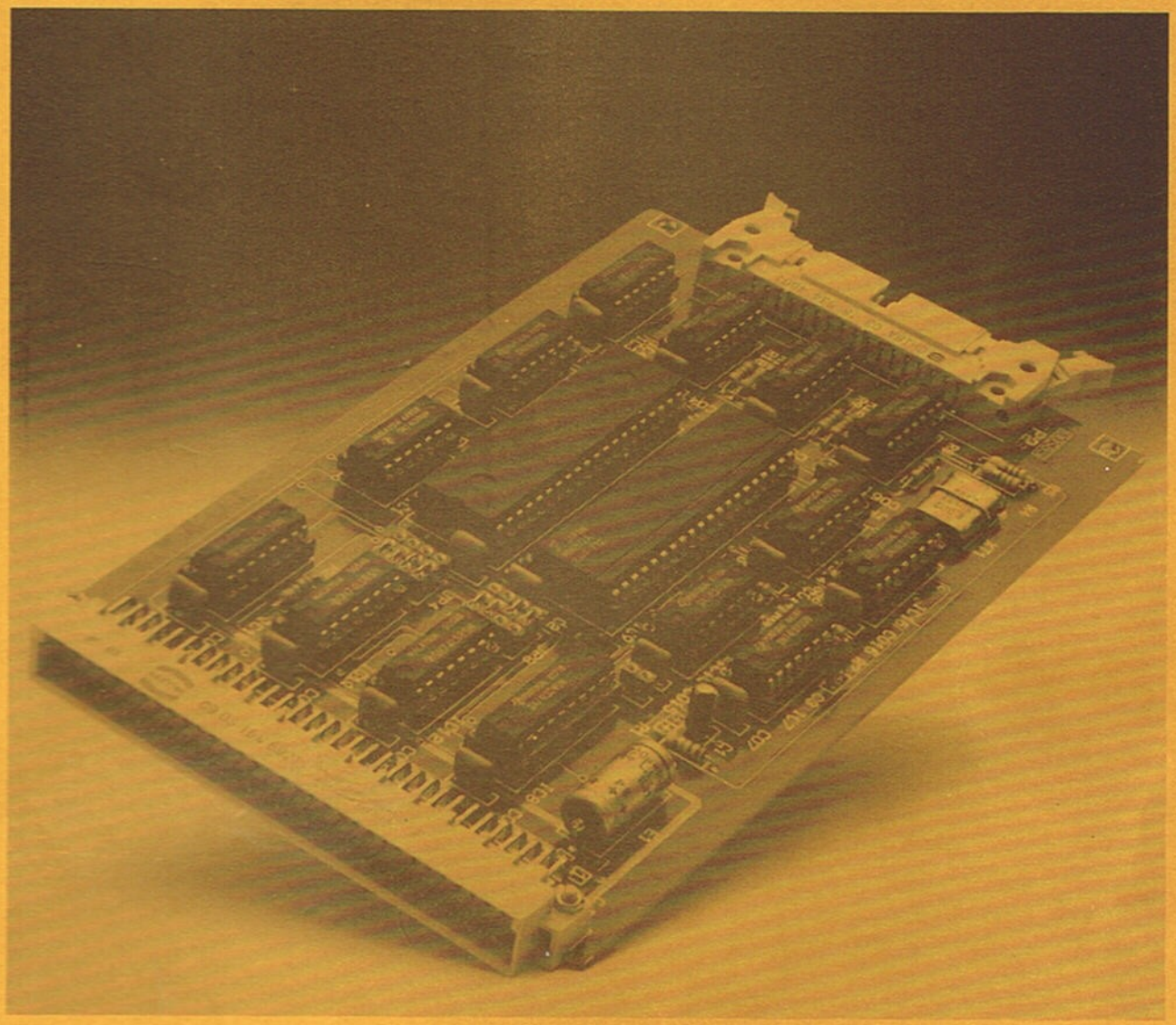
DE 6502 KENNER 46



Tiende Jaargang, Nr. 5
September 1986

DE 6502 KENNER

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

Adres voor het inzenden van en reakties 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:
Rob Banen
Fred Behringer, (Germany)
Fridus Jonkman
Gert Klein
Roger Langeveld
Marc Lachaert, (Belgium)
Fernando Lopes, (Portugal)
Frank Manshande
Gert van Opbroek
Leif Rasmussen, (Sweden)
Ruud Uphoff
Frans Verberkt
Herman Zondag
Vertaalwerk:
Fred Behringer (Germany)
Willem van Asperen
Frank Bens
Albert v.d. Beukel
Rene Hettfleisch
Jaap de Hoop
Coen Kleipool (France)
Maarten van Lieshout

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

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 redaktie geen aansprakelijkheid voor inzendingen. Tenzij anders aangegeven, dient de inzending afkomstig te zijn van de inzender.

CONTENTS OF DE 6502 KENNER NO. 46, OCTOBER 1986 VOL. 10/NO. 5

Redaktioneel:

De nu voorliggende editie kenmerkt zich door een enorme hoeveelheid korte berichten die hun plaats hebben kunnen vinden op lege plekken op de pagina's. Opvallend daarbij is het animo waarmee men antwoord geeft op de in de vorige editie geplaatste oproep op pagina 49. We hebben ook weer wat ruimte gevonden om de aandacht weer wat te richten op de programmeertaal FORTH, en hopen echt dat er leden zullen zijn die op dit gebied hun programma's in de copy-buffer van de DE 6502 KENNER zullen brengen.
Het bestuur heeft het voorstel van het dagelijks bestuur om geen stands meer in te richten op de HCC-dagen overgenomen. Zoals uit de vorige editie al kon worden gelezen, de HCC heeft gemeend de clubs, die altijd gratis de HCC-dagen konden bijwonen, thans forse bedragen te moeten vragen. Dat die dagen ons al enorm veel geld kostten, en nu ook zo'n Hfl. 1200,00 extra, heeft ertoe geleid dat we meenden op een onverantwoorde manier gelden van de leden te besteden, terwijl het konkrete resultaat aan nieuwe leden, hoewel niet precies meetbaar, niet opzienbarend is. Ik meen te mogen stellen dat nu op de schouders van de individuele leden een zware last wordt gelegd! het mee helpen zoeken naar nieuwe wegen om nieuwe leden te winnen.
We hebben het idee dat de OCTOPUS computer van Elektuur ons in de komende tijd nog heel wat stof tot publikatie gaat geven. Er is een redelijk vergelijk met de JUNIOR mogelijk. Elektuur heeft het aan de club overgelaten nog de nodige verbeteringen in de software aan te brengen. Wat dit betreft kan ik de verzekering geven dat op dit gebied nog de nodige artikelen zullen verschijnen. Marc Lachaert heeft zelfs Micro-ADE onder handen genomen, en zo is er nu Micro-ADE V2.0. Proficiat Marc ! Willem.

UITNODIGING BIJEENKOMST

Datum : zaterdag 15 november 1986
Lokatie : Kerkgebouw: "DE BRON", Tel.: 01807 - 16287
Hobbemalaan 1, 2923 XD Krimpen aan den IJssel

Route :

- per auto - komende uit de richting Utrecht
Volg autoweg Utrecht-Rotterdam-Dordrecht tot de afslag Capelle aan den IJssel voor de Brienenoordbrug. Afslag rechtsaf richting Cappelle aan den IJssel voert linksaf onder de weg naar de Brienenoord door. Houdt richting Cappelle aan den IJssel aan tot de rotonde Capelse plein, waarop (zie richtingaanwijzers) rechtsaf richting Schoonhoven en Krimpen aan den IJssel. Deze weg voert over de Algerabrug met de sluizen die het eerst gebouwde objekt zijn van de Deltawerken. Een stukje verder ontmoet U stoplichten. Ga hier linksaf de Nieuwe Tiendweg op en volg deze tot de volgende rotonde, waarop rechtsaf de Burg. Aalberslaan, dan de eerste zijstraat rechts, de Jan van Goyen-straat in, dan de eerste zijstraat rechts de Hobbemalaan in, tot het eind. Daar kunt U parkeren. Aan de rechterkant het vlakke kerkgebouw (zie pijl op kaartje bij no. 30).

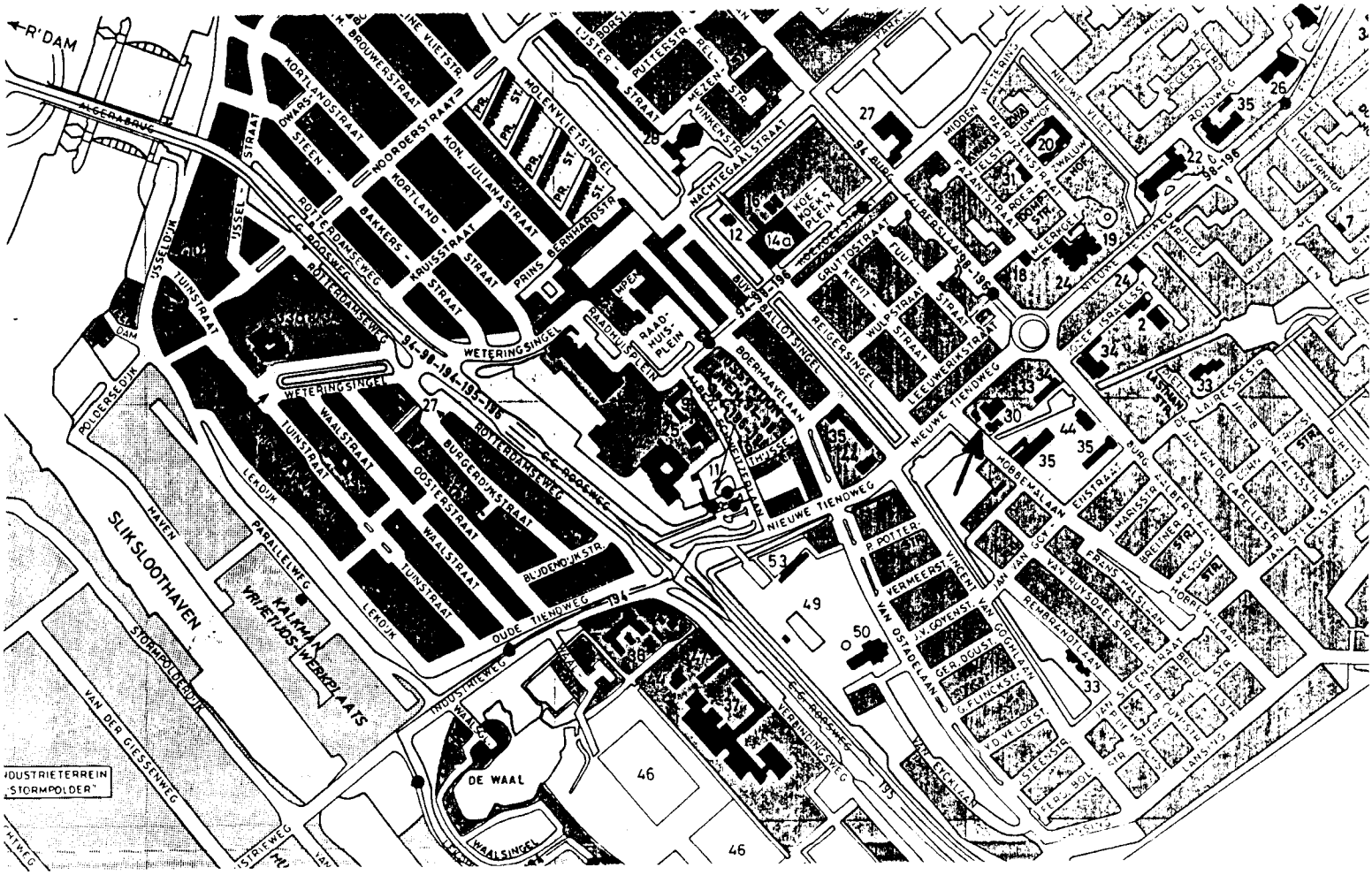
komende uit de richting Amsterdam
Volg A4 A'dam-Rotterdam. Knooppunt Leidschendam passeren. Verder richting Dordrecht via Brienenoordbrug aanhouden. En voor de Brienenoordbrug afslag rechts naar Cappelle aan den IJssel nemen. Vervolgen als hierboven.

komende uit de richting Rotterdam of Dordrecht
Richting Capelle aan den IJssel aanhouden en verder als bij eerste beschrijving.

PROGRAMMA

10.00 OPENING LEDENVERGADERING
10.15 CONCEPT- BEGROTING
VERKIEZING KASCONTROLECOMMISSIE.
VERKIEZING BESTUURSLEDEN:
aftredend en herkiesbaar: lid Gert Klein
lid Gert van Opbroek
lid Nico de Vries
(kandidaten kunnen schriftelijk worden aangemeld bij het sekretariaat of mondeling voor de aanvang van de vergadering).
RONDVRAAG en SLUITING.
12.00 LUNCH.
13.00 LEZING EN DEMO: Adri Hankel. De Laser Printer.
14.00 INFORMEEL GEDEELTE.
In het informeel deel wordt ieder in de gelegenheid gesteld kennis met elkaars systeem te maken.
BRENG DAAROM UW SYSTEEM MEE + VERLENGKABELS !!!
Heeft U hardware aan te bieden? Doe dat op een eigen tafel.
Heeft U software zelf ontwikkeld? Geef dat aan de redaktie mee.
Kopieren van software waarop enige vorm van auteursrechten rusten is binnen onze gelederen geen gebruik. Doe dat ook niet op de bijeenkomst.
17.00 Sluiting.

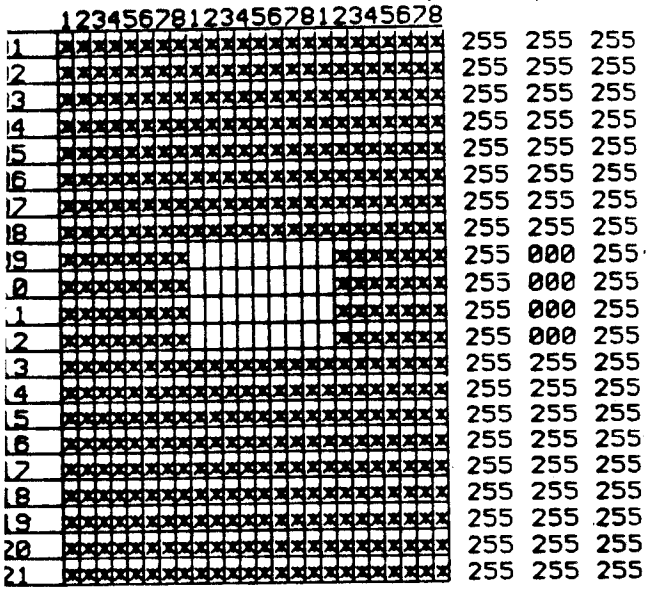
Consumpties tegen betaling



SPRITES OP DE COMMODORE 64

De opzet van dit programma is niet echt als lesmateriaal bedoeld, maar gewoon om wat inzicht te krijgen in normale en multicolor sprites. We beginnen met de opbouw van een sprite. Een sprite is opgebouwd uit 3 bytes horizontaal en 21 bytes vertikaal. Dus een sprite bestaat uit 3 maal 21 bytes = 63 bytes. Een byte is ingedeeld in de volgende getallenreeks:

```
bit 1  =  128
bit 2  =   64
bit 3  =   32
bit 4  =   16
bit 5  =    8
bit 6  =    4
bit 7  =    2
bit 8  =    1
          --- +
          255
```




```
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 000 255
255 000 255
255 000 255
255 000 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
```

TEKENING VAN EEN -1- KLEUREN SPRITE

Elk getal van 1 byte (8 vakjes) kan nooit groter zijn dan 255. Zoals U ziet in de -1- kleuren sprite tekening, zijn bijna alle vakjes gevuld, behalve in het midden, daar ziet U een gat (ook straks bij de meerkleurensprite).
Om een figuurtje te maken moet U vakjes invullen zodat de vereiste afbeelding ontstaat. De getallen van de ingevulde vakjes telt U per 8 bij elkaar op. Hierdoor ontstaan 63 verschillende getallen, welke in het geheugen ge'POKE'd moeten worden. De volgende tabel geeft de decimale adressen aan van sprite 1 t/m 4.

Beginadres sprite 0 is 11*64 = 704. Eindadres: 704+63= 767
Beginadres sprite 1 is 13*64 = 832. Eindadres: 832+63= 895
Beginadres sprite 2 is 14*64 = 896. Eindadres: 896+63= 959
Beginadres sprite 3 is 15*64 = 960. Eindadres: 960+63=1023

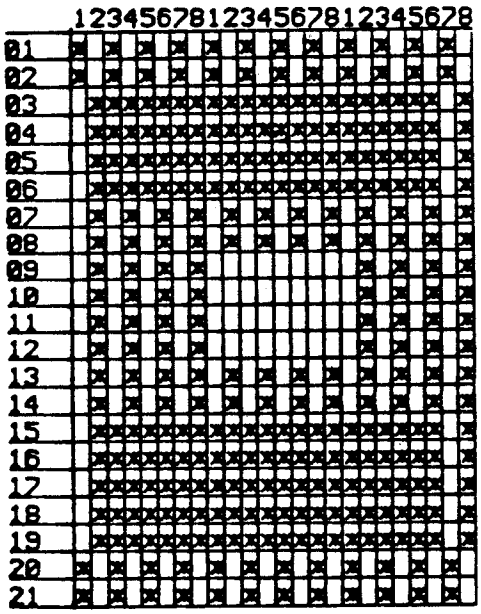
Het vormen van de sprites en het 'POKE'n op de juiste geheugenplaatsen is echter niet voldoende. De sprite moet ook aangezet worden. Dit gebeurt door een getal te 'POKE'n op adres 53269 en wel als volgt:

POKE53269,1 voor sprite 0
POKE53269,2 voor sprite 1
POKE53269,4 voor sprite 2
POKE53269,8 voor sprite 3

De getallen achter de komma kunt U ook optellen, waardoor meerdere sprites zichtbaar zijn. Bijvoorbeeld POKE53269,12 Dan staan sprite 2 en 3 aan. Doet U daarna weer POKE53269,8 dan is alleen sprite 3 aan. Als laatste moet de plaats van de sprite op het beeldscherm bepaald worden. Per sprite dienen er twee adressen ge'POKE'd te worden:

Sprite:	Horizontaal adres:	Vertikaal adres:
0	53248	53249
1	53250	53251
2	53252	53253
3	53254	53255

In ons programma gebruiken we sprite 0. Horizontaal heeft U van 0 t/m 320 plaatsen en vertikaal van 0 t/m 200. Horizontaal noemen we X en vertikaal Y. Als U de sprite in het midden van het beeldscherm wilt plaatsen, dan kunt U voor sprite 0 horizontaal POKE53248,120 geven en vertikaal POKE 53249,100. Om de sprite over het beeldscherm te laten bewegen, kan men gebruik maken van een subroutine. Dit geldt ook voor de joystick.




```
170 170 170
170 170 170
127 255 253
127 255 253
127 255 253
127 255 253
085 085 085
085 085 085
085 000 085
085 000 085
085 000 085
085 000 085
085 085 085
085 085 085
127 255 253
127 255 253
127 255 253
127 255 253
127 255 253
170 170 170
170 170 170
```

TEKENING VAN MULTI KLEUREN SPRITE

De tweede sprite tekening is een meerkleurensprite. Ik vertelde in het begin dat een getal of data is opgebouwd uit 8 vakjes. Bij de -1- kleurensprite moeten alle vakjes die op het scherm zichtbaar moeten zijn, ingevuld worden. Bij de meerkleuren gaat dat anders. Van de 8 vakjes horen er steeds 2 bij elkaar. Als 1 van de 2 vakjes, of beide, zijn ingevuld, dan worden altijd 2 vakjes zichtbaar gemaakt. De vakjes 1+2, 3+4, 5+6 en 7+8 horen bij elkaar. De computer kijkt namelijk welke vakjes zijn ingevuld en kiest daarmee zijn kleuren op de volgende manier:

Multicolor - 0 = POKE53285
Multicolor - 1 = POKE53286
Eigenkleur = POKE53287 t/m 53294

In deze drie 'POKE's kunnen dus de kleuren 0 - 15 gePOKEd worden. Even nog een keer herhalen. Als van de sprite de vakjes die bij elkaar horen de rechtse is ingevuld, dan worden beide vakjes zichtbaar met de kleur uit POKE53285. Als van de vakjes die bij elkaar horen de linkse is ingevuld, dan wordt voor sprite 0 de kleur uit POKE53287 gehaald. Om aan te geven dat de computer werkt met meerkleuren wordt sprite 0 - 7 in POKE53276 gezet. Hier geldt ook weer, dat de getallen bij elkaar opgeteld kunnen worden.

```
100 PRINTCHR$(147)
110 REM PROGRAMMA VOOR SPRITES
120 VARIABELEN VOOR KLEUREN
130 REM SNELHEID VAN SPRITE 0
140 REM STARTPLAATS X EN Y
150 A=0:B=1:C=2:X=145:Y=140:T=3
160 POKE53264,0
170 REM MULTICOLOR VAN SPRITE 1 AAN
180 POKE53276,1
190 REM INLEZEN VAN 63 DATA OP ADRES 704
200 FORI=0TO62
210 READQ
220 POKE704+I,Q
230 NEXT
240 DATA 170,170,170
250 DATA 170,170,170
260 DATA 127,255,253
270 DATA 127,255,253
280 DATA 127,255,253
290 DATA 127,255,253
300 DATA 127,255,253
310 DATA 085,085,085
320 DATA 085,085,085
330 DATA 085,000,085
340 DATA 085,000,085
350 DATA 085,000,085
360 DATA 085,085,085
370 DATA 085,085,085
380 DATA 127,255,253
390 DATA 127,255,253
400 DATA 127,255,253
410 DATA 127,255,253
420 DATA 127,255,253
430 DATA 170,170,170
440 DATA 170,170,170
450 REM SPRITE 0 WORDT OPGEHAALD
460 POKE2040,11
470 REM SPRITE 0 AAN
480 POKE53269,1
490 REM VERGROOT SPRITE 0 HORIZONTAAL
500 POKE53271,1
510 REM VERGROOT SPRITE 0 VERTIKAAL
520 POKE53277,1
530 REM KLEURENSPRITE
540 GOSUB 600
```

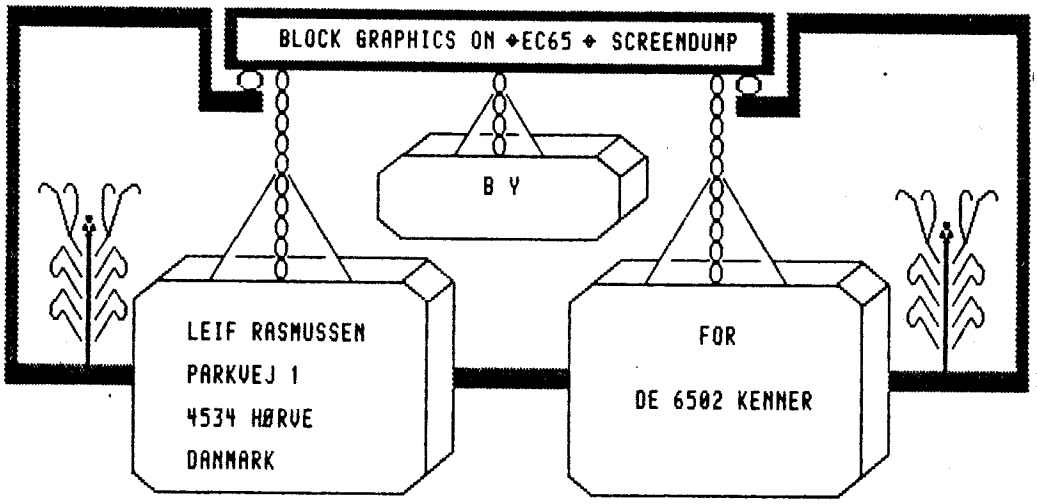

```
550 REM KONTROLE JOYSTICK
560 GOSUB740
570 REM POSITIE SPRITE
580 GOSUB910
590 GETA$:IFA$="S"THENGOSUB710:GOTO540
600 REM KLEUREN VERANDEREN VAN SPRITE 0
610 A=A+1:IFA=15THENA=0
620 B=B+1:IFB=15THENB=0
630 C=C+1:IFC=15THENC=0
640 REM MULTICOLOR 0
650 POKE53285,A
660 REM MULTICOLOR 1
670 POKE53286,B
680 REM SPRITE COLOR 0
690 POKE53287,C
700 RETURN
710 REM ROUTINE SPRITES UIT
720 POKE53269,0:LIST
730 RETURN
740 REM ROUTINE JOYSTICK
750 J=PEEK(56320:N=JAND1:Z=JAND2
760 W=JAND4:O=JAND8:VK=JAND16
770 NO=N+O:ZO=Z+O:ZW=Z+W:NW=N+W
780 IFZ=0ANDO=8ANDW=4THENY=Y+T
790 IFW=0ANDN=1ANDZ=2THENX=X-T
800 IFN=0ANDO=8ANDW=4THENY=Y-T
810 IFO=0ANDN=1ANDZ=2THENX=X+T
820 IFNO=0THENX=X+1:Y=Y-T
830 IFZO=0THENX=X+1:Y=Y+T
840 IFNW=0THENX=X-1:Y=Y-T
850 IFZW=0THENX=X-1:Y=Y+T
860 IFVK=0THENFORI=1560TO1547STEP-1
870 POKEI,81:PRINTCHR$(147):NEXT
880 RETURN
890 REM PLAATSBEPALING X EN Y VAN SPRITE 0
900 REM GRENS BOVEN
910 IFY(50THENY=50
920 REM GRENS ONDER
930 IFY)210THENY=210
940 POKE53249,Y
950 REM KONTROLE LINKER EN RECHTERZIJDE
960 P=PEEK(53264)AND1
970 IFP=0THENGOSUB1010
980 IFP=1THEN1060
990 RETURN
1000 REM GRENS LINKS
1010 IFX(20THENX=20
1020 IFX)252THENPOKE53264,1:X=0
1030 POKE53248,X
1040 RETURN
1050 REM GRENS RECHTS
1060 IFX)40THENX=40
1070 IFX(TTHENPOKE53264,0:X=255
1080 POKE53248,X
1090 RETURN
```

```
FOR APPLE : 10 A$="JACKZIP":FOR H = 1 TO 7;CALL -198:FOR S = 1 TO 16*(ASC(MID$)A$,H))-64:NEXT S,H

FOR APPLE : 10 PRINTCHR$(ASC(CHR$(ASC(CHR$(ASC("F")/(ASC("P")/8))))):GOTO 10

FOR APPLE : 10 HGR2:FORY=0TO191:POKE 228,C:C=C+1/9-256*(C=255)
            30 HPLOT 0,4 TO 279,4:NEXT Y:POKE2053,58:GOTO 10

FOR APPLE : 10 HGR2:FOR X=0 TO 255:POKE 228,X:HPLOT 0,0:CALL -3082:NEXT X
```



To be able make a screen dump of the characters in EC65's character generator (ESS523), it is necessary first to turn the 8*8 hex dump matrix "upside down". This is done with the following basicprg. Then save the new "printer" hex dump on disk and use it together with the following machine prg. in connection with fex. "GRAFIK" or "FULLSCR" in EC2.

```
1 REM This prg. convert a 8*8 matrix hex dump from VDU character-
2 REM generator, to printable chr.s, in Epson's graphics mode.
3 REM Store the character hex dump on adr.$B000-$BFFF (2 TRACKS)
4 REM The converted hex dump will placed on adr.$A000-$A800 (1 TRACK)
10 DIMA(7,7):FORE=0TO255*16STEP16:FORI=7TO0STEP-1:Y=PEEK(45063+E-I)
20 FOR J=7TO0 STEP-1:A(I,J)=0:IFY>=2^JTHENY=Y-2^J:A(I,J)=2^I
30 NEXTJ:NEXTI:FOR J=0TO7
40 X=0:FOR I=0TO7:X=X+A(I,J):NEXTI:POKE40967+(E/2)-J,X:NEXTJ:NEXTE
```

```
                                                Date: 13/04/86          Page: 0001

4720:
0010: 3A80                      ORG    #3A80
0020:                   *******SCREEN-DUMP ON EC65********
0030:
0040: 2322              OUTABL  EQU    #2322    OSI OUTPUT TABLE
0050: 2343              PRINT   EQU    #2343    PRINT CHR. IN A
0060: 25A9              SKIPOC  EQU    #25A9    TEST FOR #0C IN PRINT
0070: 2D73              STROUT  EQU    #2D73    PRINT CHR. STRING
0080: 00D0              SCRENP  EQU    #00D0    SCREEN POINTER LO
0090: 00D2              TABPNT  EQU    SCRENP   +02 TABLE POINTER
0100: 00D4              KARTEL  EQU    SCRENP   +04 CHR. COUNTER
0110: 00D5              SAVEY   EQU    SCRENP   +05 SAVE Y-REGISTER
0120: 00B0              TABHI   EQU    #B0      TABLE STORED IN #B000
0130: 0000              SCRELO  EQU    #00      SCREEN BEGIN AT #E800
0140: 00E8              SCREHI  EQU    #E8
0150:
0160:                   ------------------------SKIP LF'S---------------
0170: 3A80 A9 60                LDAIM #60
0180: 3A82 8D A9 25             STA    SKIPOC   WE DO'NT WANT EXTRA LF'S
0190:                   ------------------------PRINTER ON-------------
0200: 3A85 A9 08                LDAIM #08
0210: 3A87 8D 22 23             STA    OUTABL   PRINTER ON!!!
0220: 3A8A 20 73 2D             JSR    STROUT   LINE FEED=24/216",TAB15
0230: 3A8D 1B 33 18             HEX    1B33181B4C0E00
      3A90 1B 4C 0E
      3A93 00
0240:                   ------------------------INITIALISE--------------
0250: 3A94 A9 00                LDAIM SCRELO   BEGIN BY #E800
0260: 3A96 85 D0                STA    SCRENP
0270: 3A98 A9 E8                LDAIM SCREHI
0280: 3A9A 85 D1                STA    SCRENP   +01
0290:                   ------------------------START OF MAIN PRG.-----
0300: 3A9C A2 00        NEWLIN  LDXIM #00
0310: 3A9E 86 D4                STX    KARTEL   CHR. COUNTER=0
0320:                   ------------------------INIT. PRINTER----------
0330: 3AA0 BD AD 3A     IGEN    LDAX   INILIN   LF; GRAPH.SEL; 640 DOTS/LINE
0340: 3AA3 20 43 23             JSR    PRINT
0350: 3AA6 E8                   INX
0360: 3AA7 E0 07                CPXIM #07
0370: 3AA9 D0 F5                BNE    IGEN
0380: 3AAB F0 07                BEQ    NEXKAR
0390: 3AAD 0A 0D 1B     INILIN  HEX    0A0D1B2A01B002
      3AB0 2A 01 B0
      3AB3 02
0400:                   ------------------------GET ASCII--------------
0410: 3AB4 A0 00        NEXKAR  LDYIM #00
0420: 3AB6 B1 D0                LDAIY SCRENP   GET ASCII NR. FROM SCREEN FX.#31
0430: 3AB8 A8                   TAY
0440:                   ------------------------MULTIPLY 8; + TABHI-----
0450: 3AB9 0A                   ASLA
0460: 3ABA 0A                   ASLA
0470: 3ABB 0A                   ASLA
0480: 3ABC 85 D2                STA    TABPNT   TABEL POINTER FX. #31-->#88
0490: 3ABE 98                   TYA
0500: 3ABF 4A                   LSRA
0510: 3AC0 4A                   LSRA
0520: 3AC1 4A                   LSRA
0530: 3AC2 4A                   LSRA
```

```
                                                Date: 13/04/86          Page: 0002

0540: 3AC3 4A                   LSRA
0550: 3AC4 18                   CLC
0560: 3AC5 69 B0                ADCIM TABHI     TABEL BEGIN ADR. HIBYT
0570: 3AC7 85 D3                STA   TABPNT    +01 FX. $B188-BEGIN OF CHR.$31
0580:                   ------------------------PRINT ONE CHAR.--------
0590: 3AC9 A0 00                LDYIM $00
0600: 3ACB B1 D2        SKRIV   LDAIY TABPNT    GET ONE OF EIGTH FROM TABLE
0610: 3ACD 84 D5                STY   SAVEY
0620: 3ACF 20 43 23             JSR   PRINT     AND PRINT IT
0630: 3AD2 E6 D5                INC   SAVEY
0640: 3AD4 A4 D5                LDY   SAVEY
0650: 3AD6 C0 08                CPYIM $08       EIGTH BYTES FOR EACH CHR.
0660: 3AD8 D0 F1                BNE   SKRIV
0670:                   ------------------------NEXT CHAR.-------------
0680: 3ADA A5 D0                LDA   SCRENP
0690: 3ADC 18                   CLC
0700: 3ADD 69 01                ADCIM $01       INCREMENT SCREEN POINTER
0710: 3ADF 85 D0                STA   SCRENP
0720: 3AE1 A8                   TAY
0730: 3AE2 A5 D1                LDA   SCRENP    +01
0740: 3AE4 69 00                ADCIM $00
0750: 3AE6 85 D1                STA   SCRENP    +01
0760:                   ------------------------END OF SCREEN ?--------
0770: 3AE8 C0 80                CPYIM $80
0780: 3AEA D0 04                BNE   FULLIN
0790: 3AEC C9 EF                CMPIM $EF       END OF SCREEN=$EF80
0800: 3AEE F0 0A                BEQ   SLUT
0810:                   ------------------------80 CHAR. ?-------------
0820: 3AF0 E6 D4        FULLIN  INC   KARTEL    INCREMENT CHR. COUNTER
0830: 3AF2 A5 D4                LDA   KARTEL
0840: 3AF4 C9 50                CMPIM $50       HAS EIGTHY CHR.S BEEN PRINTED
0850: 3AF6 D0 BC                BNE   NEXKAR    IF NOT GET NEXT CHR.
0860: 3AF8 F0 A2                BEQ   NEWLIN    IF YES, SEND LF BEFORE NEXT CHR.
0870:                   ------------------------FINISH----------------
0880: 3AFA A9 48        SLUT    LDAIM $48
0890: 3AFC 8D A9 25             STA   SKIPOC    RESTORE DOS PRINT ROUTINE
0900: 3AFF A9 01                LDAIM $01
0910: 3B01 8D 22 23             STA   OUTABL    PRINTER OFF
0920: 3B04 60                   RTS             BACK TO BASIC
0930:                   ------------------------------------------------

>>> Error in 0000 statement(s)
>>> Op-Code: $B000 - $B084 / $3A80 - $3B04 / 0133 Bytes / 01 Page(s)
>>> Assembled by ASS114 / 3.4
```

##################################
#AN INTERRUPT DECODER FOR THE 6809#
##################################

Author: Andrew Gregory, England
System: Solascan Micro-Systems Ltd. system based on 8-bit Motorola MC 6809 (Elektor bus compatible)

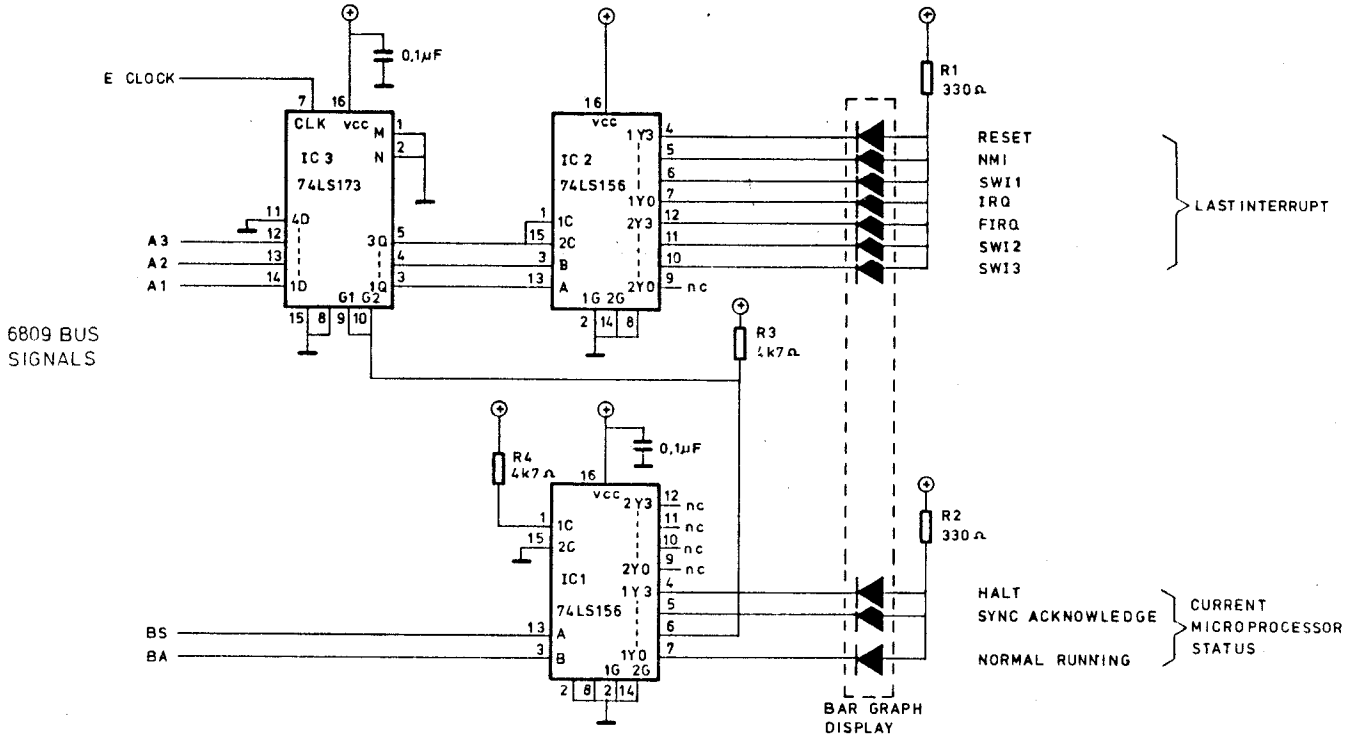
BA	BS	M.P.U.-state	COMMENTS
0	0	Normal running	
0	1	Interrupt Reset acknowledge	Occurs after a hard-ware or software in-terrupt
1	0	Sync acknowledge	Occurs while waiting for an interrupt during SYNCHRONISATION instruction
1	1	Halt on bus grant	Controls D.M.A.

The 6809 and 6809E microprocessors have two output control signals, the Bus Available (BA) and Bus Status (BS) signals. The table above shows what the signals indicate (this information comes from the Motorola data sheet) The idea behind this circuit is to use the interrupt acknowledge state to latch the address lines whenever an interrupt vector is fetched and display them on an LED-display therefore indicating which interrupt last occurred. To see this consider the interrupt timing:

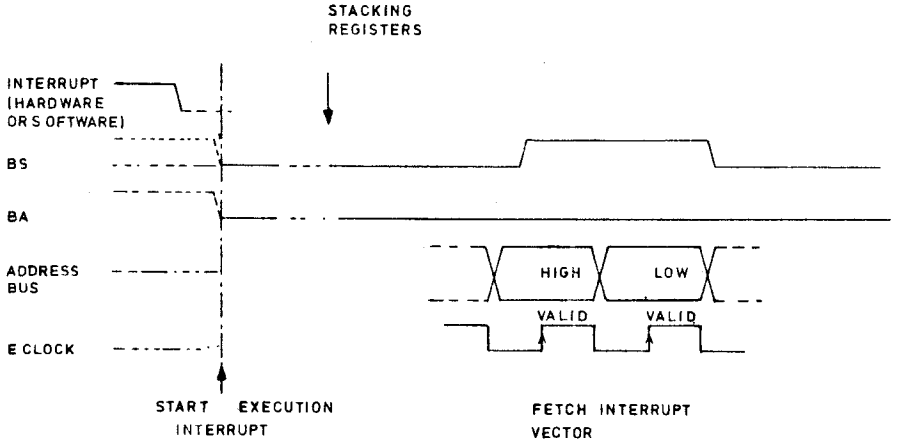
The four m.p.u states are decoded by IC1 in the circuit shown. After an interrupt pin 6 goes low for two cycles while the high and low bytes of the interrupt vector are fetched. This enables the latch IC3 which triggers on positive transitions of the E clock which occur when the address is valid (equivalent to the phase 2 clock of a 6502). Of the address A0 - A15 only A1, A2 and A3 are latched. The latch outputs are decoded by IC2 and displayed on a LED bar graph. The LED alight indicates the last interrupt as shown in the table. The remaining outputs of IC1 are also displayed giving the current m.p.u. state (Normal running, Sync acknowledge or Halt). The LED's can be driven directly because IC1 and IC3 have active low open collector outputs.

		A3	A2	A1
FFF0 + FFF1	Not used	0	0	0
FFF2 + FFF3	SWI3 (Software interrupt 3)	0	0	1
FFF4 + FFF5	SWI2 (Software interrupt 2)	0	1	0
FFF6 + FFF7	FIRQ (Fast interrupt request)	0	1	1
FFF8 + FFF9	IRQ (Interrupt request)	1	0	0
FFFA + FFFB	SWI1 (Software interrupt 1)	1	0	1
FFFC + FFFD	NMI (Non-maskable interrupt)	1	1	0
FFFE + FFFF	RESET	1	1	1

6809 INTERRUPT DECODER (ACTUAL CIRCUIT) (tested)



HERMAN ZONDAG



= PATCH ON Dr. TIETCH's COPIER PROGRAM =

Author : Marc Lachaert, Belgium

In 'DE 6502 KENNER' issues 43 and 44, a very fine adaptation of the well-known OHIO-diskette copier was published, written by Dr. Tietsch. This version works a lot more comfortable than the original OSI version, and I wish to congratulate Dr. Tietsch for his contribution to the Octopus software.
Unfortunately, I discovered some 'bugs' in the program, which could make a run very problematic.

In line 560 the address of the FLOPPY CONTROLLER is declared as to be $C000. This is false. In the Octopus the controller resides on address $E000 !

In lines 1430 through 1450, Dr. Tietsch writes down:

```
;ldaim $3E          ;is in the original code
;ldaim $43          ;is in the original code
;jsr EXCOM          ;set pointer for DOS command
```

It seems to me that Dr. Tietsch did not knew very wel the significance of those instructions, so for safety, he made 'comments' of them. In fact, these instructions are very important for the continuity of an error handling by 'copier' in case of a DOS (read/write...) error. These instructions are intended to fill the DOS error return vector ERRVEC (residing on $2A4F/2A50) with the appropriate users address to which DOS must return after an error. The user must load LSB/MSB of his own address into the A/Y register pair, and then perform a JSR to what Dr. Tietsch calls vaguely "execute a DOS command". In fact, this routine on $2A7D (I called it ERFILL) fills ERRVEC with the appropriate return vector. In this particular case, the vector must point on the label LAB13 in line 5080. So, change lines 1430 through 1450 into what follows:

```
ldaim LAB13         ;fill DOS error vector with LAB13
ldyim LAB13 /256    ;remark ---) ldY and not ldA !!!!
jsr EXCOM           ;is $2A7D
```

The three instructions will naturally no more be 'comments' but executable code!

In his version. Dr. Tietsch preferred a return to BASIC's BEXEC* after a succesful copy, rather than OSI's original solution af a total reboot of the system. This is in fact a much better idea. In front of the patch described, it will then also be necessary to reinstall the normal ERRVEC like it was before calling the copier. So, insert between lines 2370 and 2380 what follows:

```
ldaim $D7           ;fill ERRVEC with the return vector
ldyim $20           ;to the normal BASIC ERROR handler
jsr EXCOM           ;like it was before copier
```

Remark that the program in Machine Language has been elongated by 9 bytes, so the BASIC program bihind should be relocated too (use CHANGE option in BEXEC*) or you can either, while reassembling the M.L. source, shorten somewhere a text, in order to recuperate those 9 bytes...

A third patch concerns the amount of tracks to be copied. In his published version, the copier only works on drives with a max of 35 tracks. This is an unusual amount, and if you wish to adapt to another drive, you should have to change an other amount of lines:

```
- line 3160   cmpim $40 for 40 tr, cmpim $80 for 80 tr
- line 6490   ldaim $39 for 40 tr, ldaim $79 for 80 tr
- line 6570   ldaim $39 for 40 tr, ldaim $79 for 80 tr
```

and also the texts in lines 1640 and 710.

=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o
o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=

! 8 K RAM VOOR DE ATOM !

Autheur: Karel Doon, Nederland.

Voor mensen die geen schakelkaart hebben en toch een aaneengesloten geheugen willen hebben vanaf #0000 - #1FFF.

6264 (8 K CMOS) 28 pens

```
PIN  1 = N.C.          }  info
PIN  2 = A 12          }  info
PIN 27 = WE            }  info
PIN 28 = + 5 Volt      }  info

PIN 20 + 22 --) CS --) IC 8  PIN 11 = CS
PIN 27 -------)NWDS--) IC 49 PIN 11 = NWDS
```

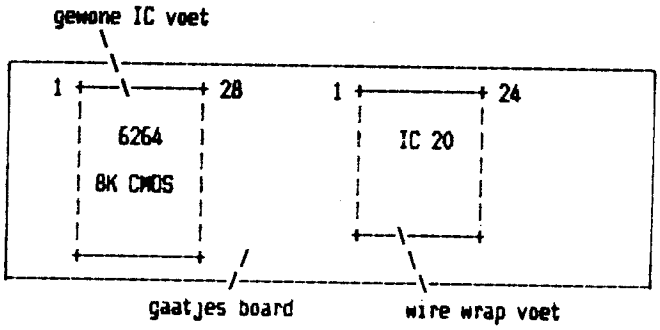
Alle andere pennen aansluiten op gelijknamige pennen op Atom board.

(IC 20)

Pen 2, 3 en 9 van IC 7 uit socket. Doorverbinden met pen 5 van IC 7. Is CS van 1 K naar 8 K.
Eventueel aanwezige Ram uit Atom verwijderen (zero page).

Opbouw:

Men hale IC 20 uit zijn voetje, en zette dit IC in de wire wrap voet.
In het voetje ernaast komt de 6264 8 K CMOS. En dit geheel komt dan in de IC 20 voet op het Atom board.
Vergeet de CS en de NWDS lijnen niet !
Gelijknamige pennen van de 6264 doorverbinden naar IC 20.
Succes !



MON/DOS65 CORNER

A MACROLOADER AND SAVER FOR ED

Author: Bram de Bruine, The Netherlands.
System: MON/DOS65, based on Elektor's buscompatible cards and the FDC controllercard 850328 of our club.

The screeneditor of DOS65 can handle macro's. To prevent to type everytime the same -frequent used- macro, you can use the program below. This program saves/loads the macrobuffer and its length, and the search/replace buffers with their lengths.

Program 1.a saves the buffers and their lengths on disk.
Program 1.b loads the files (which are saved with program 1.a) in memory. It is possible to load 4 files with one command, if you make a commandfile!

1A)
```
SAVE 'filenm_1.mcr' 37,37   !length macrobuffer;
SAVE 'filenm_2.mcr' 27F,2CE !macrobuffer;
SAVE 'filenm_3.mcr' 54,55   !length Search/Replace;
SAVE 'filenm_4.mcr' 210,22F !S/R buffers;
```

1B) ED (CR)
```
LOAD 'filenm_1.mcr'
LOAD 'filenm_2.mcr'
LOAD 'filenm_3.mcr'
LOAD 'filenm_4.mcr'
EX 'filenm_5.mcr'
```

Call it in the ED-commandmode with !@ 'filenm_5.mcr'
As you see it is much work to save/load a macro. The new release of DOS65 V2.01 makes it a lot easier, because parameter substitution in commandfiles is then possible. Warning: The macrobuffer is located at an other location as by DOS65 V1.01.

Program 2:

a)
```
;       filename : smacro
;       usage    : saves the last macro and find/replace buffer used in ED.
;       author   : Erwin Visschedijk
;       date     : 26th May, 1986.

;       This file makes a commandfile that can be used as a macro in ED.
;       This command has to be called in ED commandmode as
;                           !smacro 'macroname'
;       The macro with the name 'macroname' can in the editor's
;       commandmode be called as
;                           !lmacro 'macroname'

save -b &1.mcr 37,37     ; Save macrobuffer length
save -a &1.mcr 287,2d6   ; Save 80 macro characters
save -a &1.mcr 54,55     ; Save search and replace buffer lengths
save -a &1.mcr 210,22f   ; Save search and replace patterns
setmode -c &1.mcr
```

b)
```
;       filename  : lmacro
;       usage     : loads a specified macro in ED.
;       author    : Erwin Visschedijk
;       date      : 26th May, 1986

;       This commandfile loads a macro into the editor.
;       In the commandmode of the editor this command is called as
;                           !lmacro 'filename'

load &1.mcr
```

PS: This works only with release 2.01 !!!

LOWER-CASE COMMENT IN MOSER's ASSEMBLER

Function: It is not easy to switch the cap.lock key, everytime you insert comment in a file. This program changes capitals into lower cases, after the ";" semicolon. The utility can be called in the command-mode of the editor as: !COMMENT (With SETMODE -C is this program defined as a command)

```
;File COMMENT.MAC by B. de Bruine /E. Visschedijk

                9F00            org     $9f00

                0090    curpoi  equ     $90
                0092    endpoi  equ     $92

                0000    lome    equ     $00         Begin memory
                000A    ceme    equ     $0a         End of text pointer

9F00 A5 00      comment lda     lome
9F02 85 90              sta     curpoi
9F04 A5 01              lda     lome+1
9F06 85 91              sta     curpoi+1
9F08 A5 0A              lda     ceme
9F0A 85 92              sta     endpoi
9F0C A5 0B              lda     ceme+1
9F0E 85 93              sta     endpoi+1
9F10 A0 00              ldy     #$00
9F12 20 4C9F    next    jsr     incpoint    increment curpoi
9F15 B1 90              lda     [curpoi],y
9F17 48         compar  pha                 End of file ?
9F18 A5 91              lda     curpoi+1
9F1A C5 93              cmp     endpoi+1
9F1C D0 08              bne     xit
9F1E A5 90              lda     curpoi
9F20 C5 92              cmp     endpoi
9F22 D0 02              bne     xit
9F24 68                 pla
9F25 60                 rts                 Back to caller program

9F26 68         xit     pla
9F27 C9 3B              cmp     #';'        Found semicolon
9F29 D0 E7              bne     next
9F2B 20 4C9F            jsr     incpoint
9F2E B1 90              lda     [curpoi],y
9F30 C9 0D              cmp     #$0d        Skip over (CR)
9F32 F0 DE              beq     next
9F34 20 4C9F    lower   jsr     incpoint    First character remains
9F37 B1 90              lda     [curpoi],y  Upper case
9F39 C9 41              cmp     #'A'
9F3B 30 08              bmi     same
9F3D C9 5B              cmp     #'Z'
9F3F 10 04              bpl     same
9F41 09 20              ora     #$20
9F43 91 90              sta     [curpoi],y  Lower case ?
9F45 C9 0D      same    cmp     #$0d        End of line ?
9F47 F0 C9              beq     next        Search for next one
9F49 4C 349F            jmp     lower

9F4C            incpoint
9F4C E6 90              inc     curpoi
9F4E D0 02              bne     incit
9F50 E6 91              inc     curpoi+1
9F52 60         incit   rts

                9F00            end     comment

label table:

ceme        000A  comment   9F00  compar    9F17  curpoi    0090  endpoi    0092
incit       9F52  incpoint  9F4C  lome      0000  lower     9F34  next      9F12
same        9F45

Errors detected: 0
In DOS65 ED V2.01 this utility can also be released with a macro.
```

DOS65 AIM-BASIC ($7000-$9FFF) TIPS

1.) $738D fcc $4F. Terminalwidth and bufferwidth = 80. The extra bufferwidth destroys page zero variables from *$5a. Save this area before-, and restore it after input.
2.) $7389 fcc $18. Cancel: ^X replaces @ (This substitution is needed for basicode usage).
3.) Break with ^C:

```
 ;Basicbreak
 ;These patches comes instead of the BITtest Procedure in AIM65-Basic.
 ;It checks the inputport (keyboard) for ^C.
                 ;
          C111   KEYPORT equ     $C111
                 ;**
                 7640    org     $7640
7640 4C AC90             JMP     BRKTEST      ;Goto Patch
7643 EA                  NOP
7644 EA                  NOP
7645 60                  RTS
                 90AC    org     $90AC
                 ;
90AC 48          BRKTEST PHA
90AD AD 11C1             LDA     KEYPORT      ;Read inputport keyboard
90B0 C9 03               CMP     #$03         ;^C pressed ?
90B2 D0 04               BNE     CONT         ;If not, continue
90B4 68                  PLA
90B5 4C 4676             JMP     $7646        ;If ^C then goto Basic break-
90B8 68          CONT    PLA                  ;handler
90B9 60                  RTS
                         end

label table:     BRKTEST   90AC  CONT      90B8  KEYPORT    C111

Errors detected: 0
```

Back to DOS65 with SYS10*4096 (Cold) or ^] (Warm)
If you have any suggestions, corrections, extensions, a.s.o., please let me know.

B. de Bruine

***** BITPATROON VOOR C-64 *****

Het komt weleens voor dat er nieuwsgierige computerfanaten de verleiding niet kunnen weerstaan om van bepaalde data de bitwaarde te weten te komen. Het nu volgende programma zal na RUN vragen om een decimaal adres, waarna het de bitwaarde van de data van dit adres weergeeft en de decimale waarde van de data.

```
 10 REM BITPATROON PER ADRES
 20 REM DOOR W.BORSBOOM
 30 REM VLAARDINGEN
 40 DIMK(7)
 50 REM SCHOONMAKEN BEELDSCHERM
 60 INPUT"WELK DEC. ADRES WILT U BITSGEWIJS ZIEN:";A$:PRIN
    T:PRINT:PRINT
 70 B=LEN(A$):IFB)5GOTO50
 80 A=VAL(A$):B=PEEK(A)
 90 PRINT"DE DATA IS "B:PRINT:PRINT
100 FORX=7TOOSTEP-1
110 C=B-2^X
120 IFSGN(C)()-1THEN140
130 K(X)=0:GOTO150
140 K(X)=1:B=C
150 PRINT"BIT"X"-"K(X)"-"2^X
160 NEXTX:PRINT:PRINT:PRINT:PRINT
170 PRINT"ANDER ADRES OF STOPPEN A/S"
180 GETA$:IFA$="A"THEN50
190 IFA$()"S"THEN180
200 END
```

Na dit programma te hebben zien werken zult U zich misschien afvragen wat U hieraan heeft als U met een ander programma bezig bent. Daarom is er een handigheidje bedacht. Voordat U het programma intikt doet U het volgende:

```
POKE 44,64      + RETURN
POKE 46,64      + RETURN
POKE 48,64      + RETURN
POKE 16384,0    + RETURN
```

Hierna tikt U het bitpatroon programma in. Na de laatste regel (+ RETURN natuurlijk) te hebben ingetoetst doet U POKE 44,8 (RETURN) (geen reset). Nu kunt U een ander Basic programma(2) neerzetten op de oude plaats. Wilt U nu programma(1) draaien, dan POKE 44,8 en RETURN, en wilt U programma(2) draaien, dan POKE 44,64 en RETURN.
Bij reset wordt dat programma gereset waar U op dat moment mee bezig was. Programma(2) staat in dit geval nog op adres 16384 (hex 4000).

Wil men in DATA-regels een 0 plaatsen, dan kan dit ook weg gelaten worden. Het bespaart arbeid en geheugenruimte.

DATA 5,0,0,17,0,32 is gelijk aan DATA 5,,,17,,32

De NOS HOBBYSCOOP BASICODE-2 programma's lijken weer door iedereen zonder problemen te kunnen worden ontvangen. De redaktie wacht de door U ontvangen programma's graag in. Er is geen tijd zelf opnamen te maken, zodat we aangewezen zijn op Uw medewerking. Vooral het eerste halfjaar van 1986 is er weinig of niets van terecht gekomen. Als U uit die periode wel goede ontvangsten hebt gehad, stuur deze dan het redaktie-adres. We kunnen er veel ideeen mee opdoen zodat er werk aan de winkel blijft.
ZEND UW BASICODE PROGRAMMA'S DUS IN NAAR DE REDAKTIE.

Het volgende programma is een Basicprogramma geschreven voor een PRIME computer, en derhalve afwijkend van Microsoft Basic.
LIN(5) betekent dat op de printer of op het scherm 5 regels worden overgeslagen voordat de volgende regel wordt afgedrukt.

Zie ook DE 6502 KENNER nr. 43, April 1986.

De redaktie heeft behoefte aan Basicprogramma's ter publikatie. Stuur het naar het redaktie-adres van DE 6502 KENNER, Jacob Jordaensstraat 15, 2923 CK Krimpen a.d. IJssel.

```
10  REM AUTEURS: GERT KWETTERS/BART VAN PELT
15  REM        : 1e JAARS HOGERE ECONOMSCHE SCHOOL
16  REM        : AUTOATISCHE INFORMATIE VERWERKING (AIV)
17  REM        : Kralingse Zoom 91, 3063 ND Rotterdam
18  REM
20  PRINT CHAR(154); REM Clear Screen
30  PRINT "                                                  = = = = = = = = = = = = = = = ="
40  PRINT "                                                  =   DIT PROGRAMMA GEEFT U     ="
50  PRINT "                                                  = VAN 2- EN 3-LETTERWOORDEN   ="
60  PRINT "                                                  =   DE MOGELIJKE LETTER-      ="
70  PRINT "                                                  =        COMBINATIES          ="
80  PRINT "                                                  = = = = = = = = = = = = = = = ="
90  PRINT LIN(5)
100 PRINT "    DE REGELS: GEBRUIK HET $-TEKEN OM TE STOPPEN,"
110 PRINT "               OF ALS U GEEN WOORD HEEFT. HET ANT-"
120 PRINT "               WOORD NEE IS ALTIJD GELIJK AAN $. "
130 PRINT "               DRUK NA EEN WOORD OF EEN $ RETURN IN. "
140 PRINT LIN(3)
150 INPUT "WILT U VERDER GAAN ? ";A$
160 PRINT CHAR(154)
170 FOR A=1 WHILE A$()"$"
180     INPUT "TOETS NU EEN 2-LETTERWOORD IN:  ";B$
190     FOR B=1 WHILE B$()"$"
200         PRINT LIN(2)
210         PRINT "HIER VOLGEN DE MOGELIJKE COMBINATIES :"
220         FOR I=1 TO LEN(B$)
230             FOR J=1 TO LEN(B$)
240             PRINT SUB(B$,I)+ SUB(B$,J),
250             NEXT J
260         NEXT I
270         PRINT LIN(10)
280         INPUT "WILT U NOG EEN 2-LETTERWOORD LATEN BEWERKEN (ZO JA, WELK)?;B$
290     NEXT B
300     PRINT CHAR(154)
310     INPUT "TOETS NU EEN 3-LETTERWOORD IN :  ";C$
320     FOR C=1 WHILE C$()"$"
330         PRINT LIN(2)
340         PRINT "HIER VOLGEN DE MOGELIJKE COMBINATIES : "
350         FOR I=1 TO LEN(C$)
360             FOR J=1 TO LEN(C$)
370                 FOR K=1 TO TO LEN(C$)
380                 PRINT SUB(C$,I) + SUB(C$,J) + SUB(C$,K),
390                 NEXT K
400             NEXT J
410         NEXT I
420         PRINT LIN(5)
430         INPUT "WILT U NOG EEN 3-LETTERWOORD LATEN BEWERKEN (ZO JA, WELK)?;C$
440     NEXT C
450     PRINT LIN(3)
460     INPUT "WILT U VERDER NOG BEWERKINGEN LATEN UITVOEREN ? ";A$
470     PRINT CHAR(154)
480 NEXT A
490 PRINT LIN(10)
500 PRINT "                                                  = = = = = = = = = = = ="
510 PRINT "                                                  = DANK U EN TOT ZIENS ="
520 PRINT "                                                  = = = = = = = = = = = ="
530 PRINT LIN(10)
540 END
```

****
* JUNIOR BEKENT KLEUR *
****

Door : Phons Bloemen

Na de fantastisch mooie uitbreiding van de VDU-kaart door Janssen in DE 6502 KENNER tot een grafisch display van 640 * 200 pixels, nu een low-cost uitbreiding van de VDU-kaart tot kleurenkaart. Deze uitbreiding geeft de volgende mogelijkheden:
-low cost: er is slechts 1 RAM IC 6116 nodig in plaats van 8, en die dingen kosten fl. 30,==, als het niet meer is.
-2K karakter RAM en 2K kleuren RAM.
-8 kleuren: wit, zwart, rood, groen, blauw, geel, magenta, cyaan.
-voor elk karakter keus uit 2 van de 8 kleuren: voor- en achtergrond.
-direkt ook in de gehele video-RAM.
-graphics : 160*100 of 128*128 (afhankelijk van de schermindeling) met 8 kleuren; er kunnen echter niet teveel kleuren bij elkaar, omdat de pixels uit 'karakters' bestaan, en er zijn maar 2 kleuren per karakter mogelijk.
-karkaterset van 512 direkt vertoonbare karakters in 4K Eprom (deze zit al op de VDU-kaart). Dit is mogelijk, omdat de 9e, 10e lijn van een regel automatisch zwart gemaakt worden (onderscheid tussen de regels), en dit niet meer in de Eprom geprogrammeerd hoeft te worden. De karaktermatrix wordt nu 8*8, en niet 8*16.
-knipperende karakters
-twee kristallen mogelijk, bijvoorbeeld 12 MHz (64 kar/regel) en 15 MHz (80 kar/regel). Ze zijn softwarematig omschakelbaar via een poortlijn van de VIA.

Beschrijving van de schema's.

De schema's zijn gedeeltelijk kopieen van de Elektuurschema's van de VDU-kaart, met de uitbreiding erin getekend. Toegevoegde onderdelen dragen nummers 1XX, 'oude' onderdelen hun Elektuurnummer.

Twee oscillators (N101-102 en N17-18, omschakelbaar door VIA PA6, of een schakelaar, via N103-105) geven de dot-clock. IC 21 deelt hem door 8 tot karakterclock. Deze is de hartslag voor CRTC IC 11, die het hele zaakje stuurt. De karakteradressen die het IC uitgeeft gaan via multiplexers IC 12-14 naar beide RAM IC's 15 en 110. IC 15 geeft de karakternummers uit die door Eprom IC 19, met behulp van de door de CRTC uitgegeven rij-adressen RA0-RA2 tot karakters worden omgevormd. Het stippenpatroon komt op de uitgang van schuifregister IC 20 te staan. N34 mengt het met het cursorsignaal, terwijl N106-108 het knippersignaal eraan toevoegen. Het knippersignaal wordt door teller IC 107 van de vertikale synchpulsen afgeleid, en is alleen aanwezig als bit 3 van het kleurenbyte hoog is.
Intussen geeft de tweede RAM IC 110 de kleurinformatie uit, via 2 latches (IC 109-108). Bit 7 wordt afgebogen naar adreslijn A3 van de karakter-Eprom, om de 2e set te selekteren. Bit 3 stuurt het knippersignaal.
De kleureninformatie (bit 0-1-2 voor voorgrond, bit 4-5-6 voor achtergrond) komt op de ingangen van een rij AND-poorten terecht (N117, N124, N119-122). Het uiteindelijke stippenpatroon, beschikbaar op N112 en N113 bepaalt welke informatie wordt doorgelaten. Wanneer de lijnen DEN of RA3 (bij regels met meer dan 8 lijnen) hoog zijn, wordt via N115-116, N118 en N123 het uitgangssignaal alsnog onderdrukt. R104-106 zorgen voor verschillende grijstinten.
Wie dit verhaal nog uitgebreider wil lezen, verwijs ik naar het artikel in Elektuur september 1983.

De bouw.

De VDU-kaart moet grondig worden verbouwd. De Eelektuur-Eprom kan blijven zitten of opnieuw geprogrammeerd worden. Verder moet er flink gekrast worden, en een spinneweb van nieuwe verbindingen aangelegd worden. Ikzelf heb een uitbreidingsprintje gemaakt en dat op de VDU-kaart geschroefd. Misschien is het wel beter om de zaak helemaal opnieuw op te zetten. Wat er allemaal nodig is en wat er gedaan moet worden staat op bijgaande bouwlijsten.

De karaktergenerator Eprom.

De oorspronkelijke karakter Eprom van Elektuur werd slechts 'voor de helft gebruikt', omdat er ruimte tussen de regels moest zijn. De matrix was derhalve

8*16. Nu kan die matrix 8*8 zijn, en zijn er 512 karakters mogelijk. Laat je fantasie dus maar de vrije baan! Grieks, sierletters, letters aan elkaar, allerhande poppetjes, space invaders.
De 'graphics' komen ook uit de Eprom. Zij worden gevormd door blokjes in een 2*4 matrix, op de volgende manier:

```
..  x.  .x  xx  ..  x.  .x  xx  ..  x.  .x  xx  ..  x.  .x  xx  ..  x.
..  ..  ..  ..  x.  x.  x.  x.  .x  .x  .x  .x  xx  xx  xx  xx  ..  ..
..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  x.  x.
..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F  10  11
```

enzovoort, tot nummer 7F (127). En wat zien we nu: nummer 80 (128) is de inverse van 7F, 81 van 7E etc.. FF van 00. Door nu voor- en achtergrond te verwisselen wordt dus de 2e helft van de set verkregen, en zijn er 128 karakters over die ergens anders voor gebruikt kunnen worden.
NB! De programmeervolgorde van de karakters is anders dan je zou denken.
De Eprom-adressen:

nr. 0	0000-0007	nr. 251	0FB0-0FB7	nr. 256	0008-000F	nr. 507	0FB8-0FBF
nr. 1	0010-0017	nr. 252	0FC0-0FC7	nr. 257	0018-001F	nr. 508	0FC8-0FCF
nr. 2	0020-0027	nr. 253	0FD0-0FD7	nr. 258	0028-002F	nr. 509	0FD8-0FDF
nr. 3	0030-0037	nr. 254	0FE0-0FE7	nr. 259	0038-003F	nr. 510	0FE8-0FEF
nr. 4	0040-0047	nr. 255	0FF0-0FF7	nr. 260	0048-004F	nr. 511	0FF8-0FFF

Let hierop, anders zit er ineens een c op de plaats waar de een b wilt!

Programmeermodel kleurenkaart.

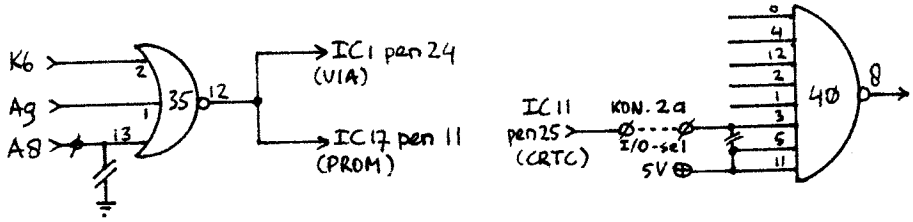
bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
kar. adr. 0	kar. adr. 1	kar. adr. 2	kar. adr. 3	kar. adr. 4	kar. adr. 5	kar. adr. 6	kar. adr. 7

Karakterram byte (bij mij $E000-$E7FF).

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
ROOD voor	GROEN voor	BLAUW voor	knip-oeren	ROOD achter	GROEN achter	BLAUW achter	kar. adr. 8

Kleurenram byte ($E800-$EFFF). Beide blokken achter elkaar decoderen.

Veranderen aan interfacekaart: de CRTC wordt op $1900 gedecodeerd.



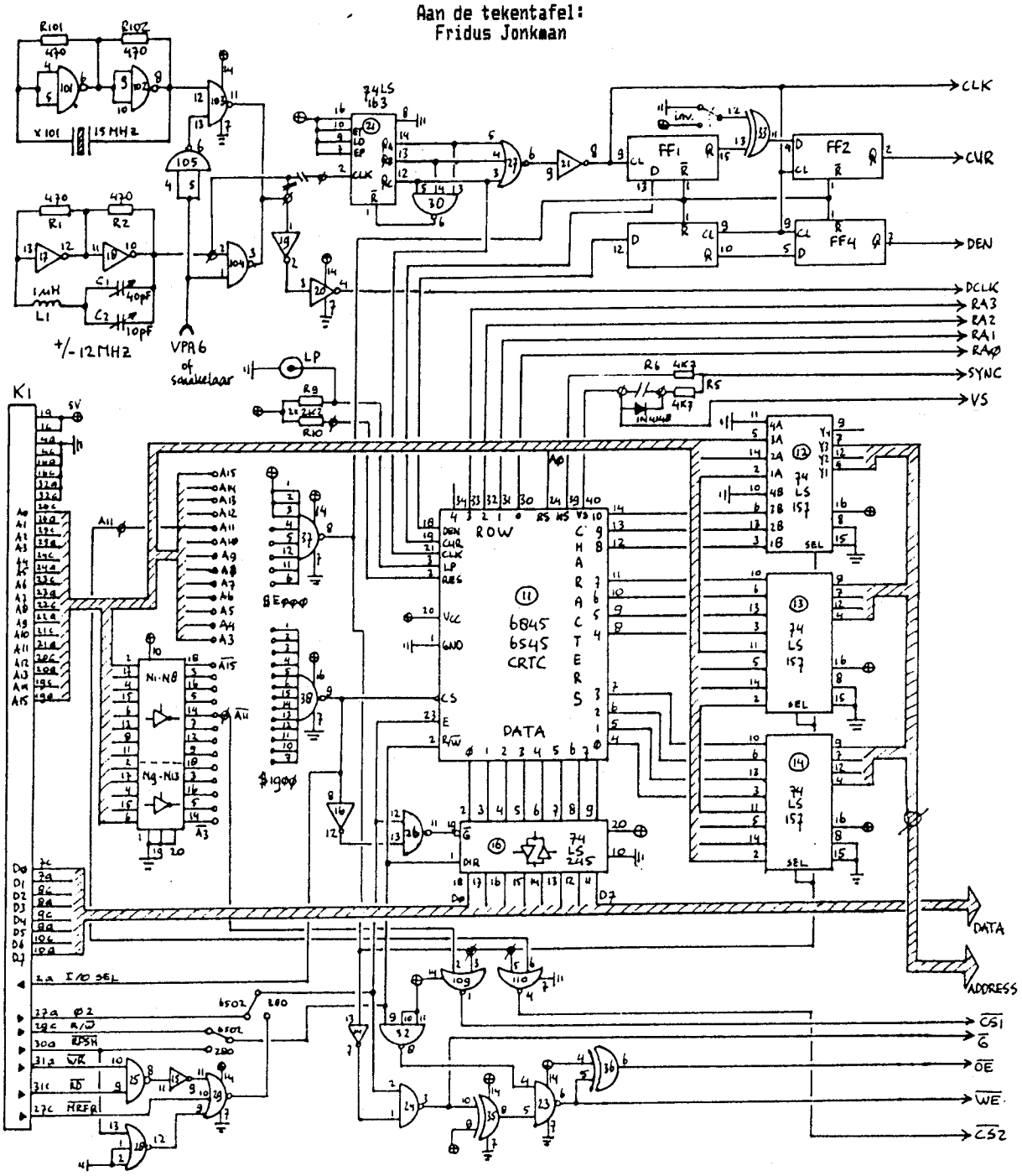
Benodigd: 1 x 6116 1 x 74LS02 1 x 10 uF 2 x 470 Ohm
2 x 74LS273 1 x 4024 1 x 33 Ohm 1 x 10 kOhm
3 x 74LS00 1 x 74LS245 1 x 68 Ohm 1 x 1N4148
2 x 74LS08 9 x 100 nF 1 x 120 Ohm
1 x kristal 12 of 15 MHz

Ombouwschema.

Aan componentenzijde van de VDU-kaart niets doorkrassen.
Aan soldeerzijde volgende verbindingen doorkrassen:

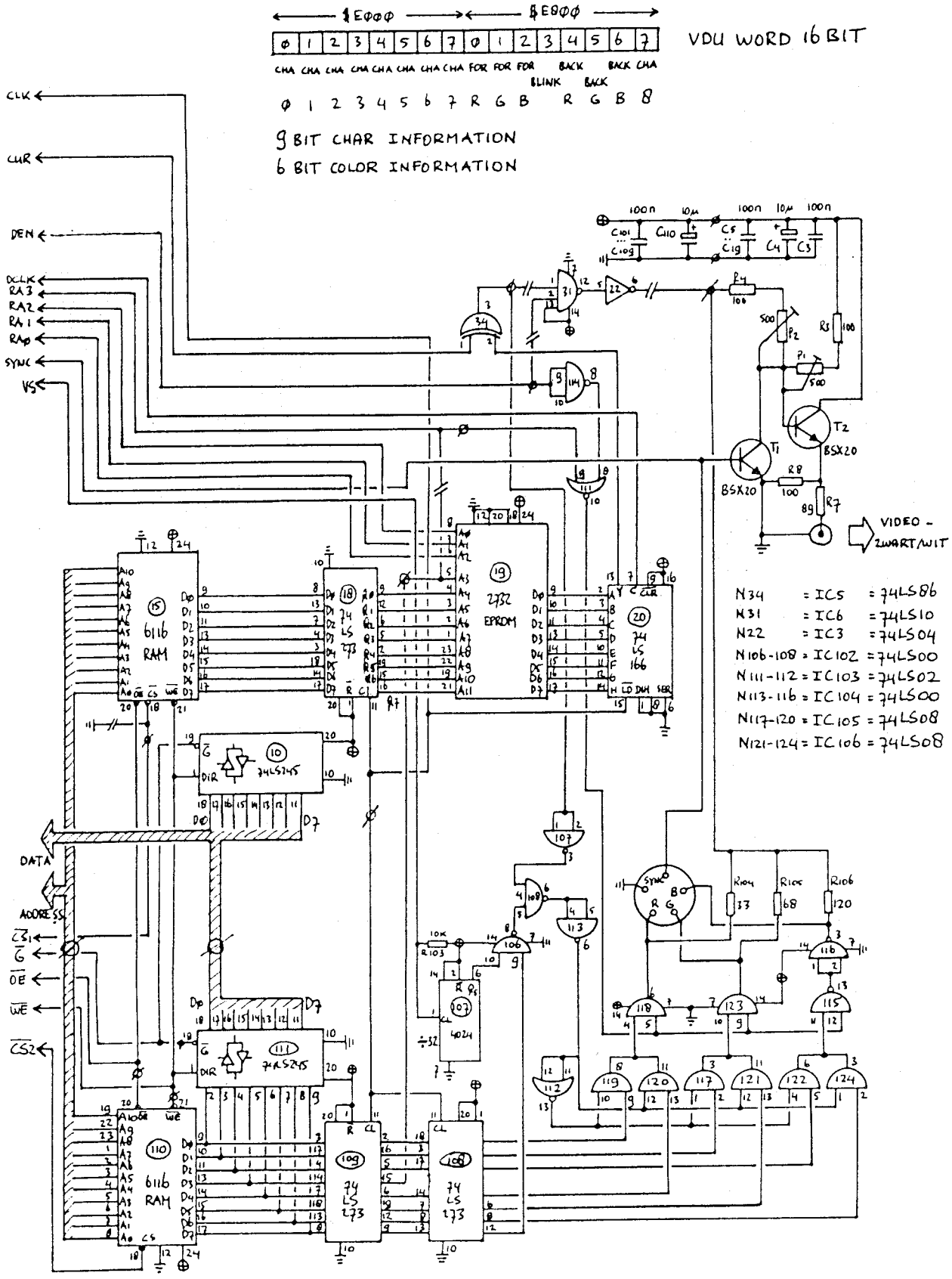
IC 3 pen 10 - IC 21 pen 2	Weerstand R2 - IC 3 pen 1
IC 5 pen 3 - IC 6 pen 1	IC 17 pen 7 - IC 6 pen 2
IC 3 pen 6 - Weerstand R4	IC 11 pen 35 - IC 19 pen 5
IC 15 pen 18 - IC 15 pen 12	IC 11 pen 40 - Weerstand R6

Aan de tekentafel:
Fridus Jonkman



N1-N8 = IC1 = 74LS240
N9-N16 = IC2 = 74LS240
N17-N21 = IC3 = 74LS04
N23-N26 = IC4 = 74LS00
N27-N29 = IC7 = 74LS27
N30-N32 = IC6 = 74LS10
N33-N36 = IC5 = 74LS86
N101-N104 = IC101 = 74LS00
N105 = IC102 = 74LS00
N109-N110 = IC103 = 74LS02
N37 = IC8 = 74LS30
N38 = IC9 = 74S133
FF1-FF4 = IC17 = 74LS175

COLOR VDU BOARD
FOR JUNIOR COMPUTER
based on Elektuur VDU-board
83082 september 1983.
parts with no's 1XX are added.


$E000
$E800
CHA CHA CHA CHA CHA CHA CHA CHA FOR FOR FOR BLINK BACK BACK CHA
0 1 2 3 4 5 6 7 R G B R G B 8
VDU WORD 16 BIT
9 BIT CHAR INFORMATION
6 BIT COLOR INFORMATION
CLK
CHR
DEN
DCLK
RA3
RA2
RA1
RA0
SYNC
VS
VIDEO - ZWART/WIT
BSX20
DATA
ADDRESS
N34 = IC5 = 74LS86
N31 = IC6 = 74LS10
N22 = IC3 = 74LS04
N106-108 = IC102 = 74LS00
N111-112 = IC103 = 74LS02
N113-116 = IC104 = 74LS00
N117-120 = IC105 = 74LS08
N121-124 = IC106 = 74LS08

Nieuwe verbindingen.

```
Op VDU-kaart zelf: IC  3 pen 10 - Weesrtand R2
Diode 101        : IC 11 pen 40 (anode) - Weerstand R6 (kathode)
```

Tussen VDU-kaart en nieuwe uitbreidingsprint:

```
IC      3 pen 10 - IC 101 pen  2    (X MHz) IC 21 pen  2 - IC 101 pen  3 (DCL)
IC 10/16 pen 11 - IC 111 pen 11     (D7)    IC 12 pen  4 - IC 110 pen 19 (BA10)
IC 10/16 pen 12 - IC 111 pen 12     (D6)    IC 12 pen 12 - IC 110 pen 22 (BA9)
IC 10/16 pen 13 - IC 111 pen 13     (D5)    IC 12 pen  7 - IC 110 pen 23 (BA8)
IC 10/16 pen 14 - IC 111 pen 14     (D4)    IC 13 pen  9 - IC 110 pen  1 (BA7)
IC 10/16 pen 15 - IC 111 pen 15     (D3)    IC 13 pen  7 - IC 110 pen  2 (BA6)
IC 10/16 pen 16 - IC 111 pen 16     (D2)    IC 13 pen 12 - IC 110 pen  3 (BA5)
IC 10/16 pen 17 - IC 111 pen 17     (D1)    IC 13 pen  4 - IC 110 pen  4 (BA4)
IC 10/16 pen 18 - IC 111 pen 18     (D0)    IC 14 pen  9 - IC 110 pen  5 (BA3)
IC      1 pen 14 - IC 103 pen  2    (A11)    IC 14 pen  7 - IC 110 pen 6 (BA2)
IC     12 pen  1 - IC 103 pen 3/5   (VE)   IC 14 pen 12 - IC 110 pen  7 (BA1)
IC     15 pen 18 - IC 103 pen  1    (CSI)   IC 14 pen  4 - IC 110 pen  8 (BA0)
IC     11 pen 35 - ic 103 pen  9    (RA3)   IC 19 pen  5 - IC 109 pen 15 (CH8)
IC     17 pen  7 - IC 104 pen 9/10  (DEN)   IC 17 pen  9 - IC 109 pen 11 (CLK)
IC      5 pen  3 - IC 102 pen 1/2   (CUR)   IC 11 pen 40 - IC 107 pen  1 (VS)
Weerstand R4 - Weerstand 104/105/106        Konnektor 21a- IC103 pen  6 (A11)
IC  5 pen 10 / IC 4 pen 3 / IC 10 pen 19  - IC 111 pen 19 (C)
IC 15 pen 21 / IC 5 pen 5 / IC  4 pen  6  - IC 110 pen 21 (WE)
           IC 15 pen 20 / IC 5 pen 6 - IC 110 pen 20 (OE)
```

```
+=======================+
! LOOK AT REAL CONTENTS !
!   FOR ATARI 600 XL    !
+=======================+
```

Author: H. Speksnijder, The Netherlands.

This is a small program to look at the real contents of the Ram or the Rom. After RUN you type the decimal address of the first location. The program then shows this and the next 19 addresses.
The data is shown as a number and as the Atari character. Some numbers cause unpleasant effects in the text and therefor line 50, 60 and 70 are added.
If you press (SELECT) then you get the next 20 addresses, and if you press (START) then you can enter a new begin-address.

```
10   ?"BEGIN";:INPUT B
20   FOR I=0 TO 19
30   C=PEEK(B+I)
40   ? B+I,C,
50   IF C=28 OR C=29 THEN 90
60   IF C=125 OR C=155 THEN 90
70   IF C=156 OR C=157 THEN 90
80   ? CHR$(C):GOTO 100
90   ?
100  NEXT I
110  IF PEEK(53279))6 THEN 110
120  IF PEEK(53279)=6 THEN 10
130  B=B+20
140  GOTO 20
```

It is interesting to look how the Atari puts his program in memory. When you calculate :
PEEK(136)+256*PEEK(137)
it will give the first address where the Basic program is in memory. If you add a line like:
1 REM BASIC
and then run this Basic program. As the beginaddress use the result of the calculation. The text after the REM is exact in the program, but it is a lot of work to find out how the rest is done.

==
o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=
==

```
;*******************************************************;
; PROGRAM CURBACK  Reposition cursor of CRTC UM 6845    ;
;                                                       ;
; The UM 6845 is different from other CRTC chips.       ;
; In the MON/DOS65 system this results in a cursor one  ;
; position ahead of its proper place. This effect can be;
; corrected using the bits in the higher datanibble of  ;
; the interlace control register. If bit 8 is set (as it;
; is in the default situation) the cursor position is   ;
; wrong. Changing the bit to 0 moves the cursor back to ;
; where it should be.                                   ;
;                                                       ;
; This program can be made into a command placed in the ;
; LOGIN.COM file.                                       ;
;                                                       ;
;*******************************************************;
```

By : Peter Roessingh, The Netherlands.
System: MON/DOS 65

```
CURBACK ORG $800    ; Startaddress

; *** Variables and definitions ***

CRTCAR EQU $C140             ; crtc address register
CRTCRF EQU $C141             ; crtc register file

; *** Start of program ***

 LDA #$08                    ; get registeraddress
 STA CRTCAR                  ; and store it in crtc

 LDA #$00                    ; get new value
 STA CRTCRF                  ; and reset bit 8 in reg.

 RTS                         ; end of program
```

==
o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=o=
==

MODIFICATION VDU CARD OCTOPUS 65 (By: Albert v.d. Beukel)
When I boot up the system disk I have interference on my monitor like stripes and other things. You can solve it with a little modification as follows: disconnect pin 1 of IC 8 (74LS30) and connect it with pin 27-a of the 64-pins card connector. If done, the interference is gone.

CHANGING THE 6502 BY A 65C02 PROCESSOR
IN ELEKTOR'S JUNIOR COMPUTER.

By: Jan Vernimmen, The Netherlands.

Problems occured (measures made with Rockwell proc.):
1 : clocksignals phase 1 and phase 2.
2 : the fan-out.
3 : Ram R/W signal.

1. The input/output of the pins 37 and 39 (X-tal is between them) of the C-mos are not compatible with the old H-mos version. The voltages switching low to high and reverse are different. The cristal is giving something like a triangle-wave form. The result was an instable timing sequence. Phase 2 high: 630 nSec, low 370 nSec. This is far more than the published tolerances of the Rockwell 65C02. To solve this problem, put a resistor from pin 37 to earth. For my system I needed 68 KOhm. The new phase 2 low and phase 2 high were: 480/520 nSec. But now the oscillator refused to start mostly. This was solved by putting a resistor of 1,5 MOhm between the 2 pins of the X-tal, the new phase 2 low/phase 2 high became 482/518 nSec.
   Everything was done with help of an oscilloscope. If you don't have one: build an extern oscillator on a small print and give the processor a nice blackwave.
2. The fan-out of the C-mos version is 1; that of the normal (H-mos version is 2. Be sure that all TTL, getting signals direct from the CPU are LS-TTL, than this problem is solved (Eprom and PIA take far less then 1 TTL pro pin; they are Mos-IC's).
3. Check resistor R5 on the main print. This has to be 470 Ohm or less, down to 330 Ohm. This gives a better shape of the Ram R/W signal.

To improve the quality of the control signals you can give the earth points of the processor, Eprom and PIA an extra connection to earth: to pin 16 a-c of the connector. These pins 16 a-c were not in use. Check their connection to earth first.

From our member K. Brouwer, Dwingelo, we received comments on the article "Van 80 naar 40 tracks" in Elektor's magazine in Holland, July/Aug., 1986, p. 69, in which they give a schematic diagram to make it possible to read 40 tracks diskettes on your 80 tracks drive. In this diagram they wrote: N1...N2 = IC2 = 74LS23. Read for 74LS23 now 74LS33 and you may read your 40 tracks diskettes.

DISKDRIVE TIP VOOR C-64

Wanneer er op een schijf een programma staat, dat niet netjes is afgesloten, er staan dan bijvoorbeeld:
(PROGRAMMA-NAAM * PRG),
dan is de eerste reaktie om dat programma eventjes te verwijderen met een SCRATCH. Dit kan bijvoorbeeld als volgt worden ingevoerd:

```
(OPEN15,8,15, "S0:PROGRAMMA-NAAM":CLOSE15).
```

Deze methode werkt wel.

DE CBM-64 ALS TYPEMACHINE

Onderstaand programma laat U de CBM-64 als typemachine benutten. Laadt het programma en zet de printer aan. Als het programma gestart wordt, verschijnt er een vragteken. Nu kunt U tot max. 76 karakters per regel invoeren en op (RETURN) drukken. De ingevoerde regel wordt direkt geprint. Als U klaar bent, geef dan 'XXX'.

```
100 POKE59468,12
110 OPEN7,4,7:PRINT#7:CLOSE7
120 OPEN4,4
130 INPUT A$
140 IF A$ = "XXX" THEN PRINT#4:CLOSE4:END
150 PRINT#4,A$
160 A$ = ""
170 GOTO130
```

PLEASE NOTE THAT YOUR 1987 SUBSCRIPTION MUST BE PAYD IN THE MONTH OF DECEMBER 1986
SEND CHEQUE OF HFL. 59,50 TO W.L. VAN PELT (EUROCHEQUE 50,00)

```
 5          PUT  CHESS1.S
)1    ********************************
)2    * IN EDITIE 20 VAN DE 6502 KENNER VAN FEBRUARI 1982 WERD
)3    * EEN HERPUBLIKATIE GEPLAATST VAN HET KIM-SCHAAKPROGRAM-
)4    * MA VAN TH. KORTEKAAS UIT DE KIM-KENNER NR. 14, DEC.'80
)5    * DIT PROGRAMMA TROK DE AANDACHT VAN VELE LEZERS. EDITIE
)6    * 20 WERD VOLLEDIG UITVERKOCHT.
)7    *
)8    * IN DIT PROGRAMMA IS HET  PROGRAMMA VAN KORTEKAAS INTE-
)9    * GRAAL OPGENOMEN, AANGEPAST VOOR DE JUNIOR-COMPUTER.
)10   * HET GEINTEGREERDE DEEL BEVAT GEEN KOMMENTAARREGELS. D-
)11   * TOEVOEGING DAARVAN BLEEK NIET MEER MOGELIJK.
)12   *
)13   * MET DANK AAN THEO KORTEKAAS.
)14   ********************************
)15                                ;
)16                                ; ZERO PAGE LOKATIES
)17                                ;
)18   ZP        EQU  $0000
)19                                ;
)20                                ; MONITOR ADRESSEN
)21                                ;
)22   MONI      EQU  $1C00
)23                                ;
)24                                ; NMI-VECTOR

)26                                ;
```

```
                    )27   VECT     EQU  $1A7A
                    )28
                    )29   * SUBROUTINE CALC *
                    )30
                    )31
2808: E6 0C         )32   CALC     INC  NIVO
280A: A9 C0         )33            LDA  #$C0
280C: 85 08         )34            STA  ZWRD
280E: 45 15         )35            EOR  CKAZ
2810: 85 15         )36            STA  CKAZ
2812: 98            )37            TYA
2813: 85 06         )38            STA  NZET
2815: 38            )39            SEC
2816: E5 0B         )40            SBC  WRDE
2818: 85 0B         )41            STA  WRDE
281A: A9 3F         )42            LDA  #$3F
281C: 85 05         )43            STA  VAN
281E: A6 05         )44   BOZ      LDX  VAN
2820: B5 16         )45            LDA  BORD,X
2822: F0 6C         )46            BEQ  BOM
2824: 45 15         )47            EOR  CKAZ
2826: 29 40         )48            AND  #$40
2828: D0 66         )49            BNE  BOM
282A: B5 16         )50            LDA  BORD,X
282C: 29 07         )51            AND  #7
282E: AA            )52            TAX
282F: B5 56         )53            LDA  PTZ,X
2831: 85 07         )54            STA  ITZ
2833: A6 05         )55   BOA      LDX  VAN
2835: 86 04         )56            STX  NAAR
2837: 46 04         )57   BOC      LSR  NAAR
2839: 46 04         )58            LSR  NAAR
283B: 46 04         )59            LSR  NAAR
283D: 8A            )60            TXA
283E: 29 07         )61            AND  #7
2840: AA            )62            TAX
2841: A4 07         )63            LDY  ITZ
2843: B9 64 00      )64            LDA  TZET,Y
2846: A0 02         )65            LDY  #2
2848: 0A            )66   BOD      ASLA
2849: B0 0E         )67            BCS  BOG
284B: 0A            )68            ASLA
284C: 90 01         )69            BCC  BOE
284E: E8            )70            INX
284F: 0A            )71   BOE      ASLA
2850: 90 01         )72            BCC  BOF
2852: E8            )73            INX
2853: E0 08         )74   BOF      CPX  #8
2855: 90 0A         )75            BCC  BOZZ
2857: B0 2F         )76            BCS  BOL
2859: 0A            )77   BOG      ASLA
285A: 90 01         )78            BCC  BOH
285C: CA            )79            DEX
285D: 0A            )80   BOH      ASLA
285E: CA            )81            DEX
285F: 30 27         )82            BMI  BOL
2861: 88            )83   BOZZ     DEY
2862: F0 38         )84            BEQ  BOO
2864: 86 14         )85            STX  HULP
2866: A6 04         )86            LDX  NAAR
2868: 10 DE         )87            BPL  BOD
286A: 20 8E 2A      )88   BOJ      JSR  MOVE
286D: 68            )89   BOK      PLA
286E: 85 13         )90            STA  EPS
2870: 24 0E         )91            BIT  CZO
2872: 70 25         )92            BVS  BON
2874: A5 08         )93            LDA  ZWRD
2876: C9 41         )94            CMP  #$41
2878: F0 1F         )95            BEQ  BON
287A: A6 05         )96            LDX  VAN
287C: B5 16         )97            LDA  BORD,X
```

;
;
;

LETTER TO THE EDITOR

Object: Troubles with Micro-ADE and Elektor's OCTOPUS.
(Question in the June '86 Club Magazine)

Red. : In some cases, the Micro-ADE Assembler does not work correctly and produces a continious loop of error messages after starting up. It was already known that the removing of jumpers on PL 3 looses the problem, but not in case if a serial printer is connected.

M. Lachaert, Belgium.

If I well understood the kind of the trouble, it seems to be the interaction of a harware error in the system and the rather unusual way of 'Error handling' in Micro-ADE.

An error is treated by Micro-ADE as a 'BRK' state. At the initialisation of the progrma, the IRQ-vector is deviated to the error handling routine of Micro-ADE. As known, the IRQ-vector treats as well as hardware interrupts (INTERRUPT REQUESTs caused by the 'pulling low' of CPU's pin 4) as software interrupts when the program encounters a 'BRK' instruction (=$00). These techniques were already treated extensively and very clearly by Ruud Uphoff in the issue number 40 of October 1985.

During running Micro-ADE, a HARDWARE BREAK should in no way occur! The program should interpret this as an error in the program, and return automatically to it's error handler. If during this, pin 4 is still (too...) low, the program will go into a loop and will produce error messages to eternity.

I had the same problem while installing Micro-ADE for the first time, simply because of an accidental short-circuit in the FCU-card. The IRQ-entry of the CPU was connected via....via with it's SYNC, that produced nice pulses, pulling IRQ down at the misfortunate moment... That short-circuit was there since months, but I had never earlier problems. No other program indeed used BRK instructions... I can assure that I spent many evenings to find out what really happened!

It is naturally hard to determine from here where the problem resides in any individual system. It can in a simple way be located when the signal on pin 4 of the CPU is displayed on a scope. This pin must ALWAYS REMAIN HIGH. If not, one must urgently find out which composant 'pulls the pin low'. In a normal Octopus configuration, there is no element that does so, thus there must be cabling error, shortened print tracks, or an unfortunately choosen extension... A first diagnose trick can be to lift pin 4 out of the socket, and to connect it with a 10K resistor with the + 5V. Normally all must then work well.

==========

LETTER TO THE EDITOR

Solution of Micro-ADE problem of continuous errors.

By: Ronald Hermens, The Netherlands.

The problem is in the routine: FNDND = $2496. This subroutine will search from source begin address for the following pattern: $0D,**,**,$40 (** = don't care). If this pattern is not found, the subroutine will continue searching on successor addresses, until $FFFF and will go on from $0000. Eventually a PIA timer address will be read, and the timer is set, on a time-out an IRQ will occur, the

```
287E: 29 04     )99             AND   #4
2880: F0 06     )100            BEQ   BOL
2882: A6 04     )101            LDX   NAAR
2884: B5 16     )102            LDA   BORD, X
2886: F0 AF     )103            BEQ   BOC
2888: E6 07     )104  BOL       INC   ITZ
288A: A6 07     )105            LDX   ITZ
288C: B5 64     )106            LDA   TZET, X
288E: D0 A3     )107            BNE   BOA
2890: C6 05     )108  BOM       DEC   VAN
2892: 10 8A     )109            BPL   BOZ
2894: A0 00     )110            LDY   #0
2896: 4C 42 29  )111            JMP   BN
2899: 4C D6 29  )112  BON       JMP   BBD
289C: 85 10     )113  BOO       STA   CZET
289E: 8A        )114            TXA
289F: 0A        )115            ASLA
28A0: 0A        )116            ASLA
28A1: 0A        )117            ASLA
28A2: 05 14     )118            ORA   HULP
28A4: 85 04     )119            STA   NAAR
28A6: A5 13     )120            LDA   EPS
28A8: 48        )121            PHA
28A9: 84 03     )122            STY   PROM
28AB: 24 10     )123            BIT   CZET
28AD: 30 02     )124            BMI   BIIZ
28AF: 50 12     )125            BVC   BA
28B1: 84 13     )126  BIIZ      STY   EPS
28B3: 10 4C     )127            BPL   BG
28B5: 50 B3     )128            BVC   BOJ
28B7: A5 04     )129            LDA   NAAR
28B9: 24 15     )130            BIT   CKAZ
28BB: C9 20     )131            CMP   #$20
28BD: 50 35     )132            BVC   BE
28BF: 90 AC     )133            BCC   BOK
28C1: B0 33     )134            BCS   BF
                )135                                  ;
                )136  * PION SCHUIN SLAAN *
                )137                                  ;
28C3: A6 04     )138  BA        LDX   NAAR
28C5: B5 16     )139            LDA   BORD, X
28C7: F0 04     )140            BEQ   BB
28C9: 84 13     )141            STY   EPS
28CB: D0 3A     )142            BNE   BH
                )143                                  ;
                )144  * PION EN PASSANT SLAAN *
                )145                                  ;
28CD: E4 13     )146  BB        CPX   EPS
28CF: D0 6B     )147            BNE   BL
28D1: A0 08     )148            LDY   #8
28D3: 24 15     )149            BIT   CKAZ
28D5: E8        )150  BC        INX
28D6: 70 02     )151            BVS   BD
28D8: CA        )152            DEX
28D9: CA        )153            DEX
28DA: 88        )154  BD        DEY
28DB: D0 F8     )155            BNE   BC
28DD: 84 13     )156            STY   EPS
28DF: B5 16     )157            LDA   BORD, X
28E1: 94 16     )158            STY   BORD, X
28E3: 48        )159            PHA
28E4: 8A        )160            TXA
28E5: 48        )161            PHA
28E6: E6 0B     )162            INC   WRDE
28E8: 20 8E 2A  )163            JSR   MOVE
28EB: C6 0B     )164            DEC   WRDE
28ED: 68        )165            PLA
28EE: AA        )166            TAX
28EF: 68        )167            PLA
28F0: 95 16     )168            STA   BORD, X
28F2: D0 48     )169            BNE   BL
                )170                                  ;
```

interrupt mask bit was not set in Micro-ADE, so the break error routine will be executed, followed by a warm start of Micro-ADE (which clears the interrupt disable flag). Because the timer of the PIA is still giving interrupts, the break routine will be executed again after a few miliseconds. This is followed by a warm start, which enables interrupts etc.

FNDND calls the subroutine LOAD = $24FB the subroutine increases the address-pointer (BLO-BHI) and loads the contents of this address in the accumulator, if the address-pointer is beyond the end of the source-area the N-flag will be set, this new routine will test this flag.

NEW ERROR:
**********(0E) = Format of the source file is not correct, the last correct line number was printed just before this error. If line number = 0000 there is no valid line found in source aria.

```
                PATCH:
        15 00   NLO     *       $0015   LINE NUMBER
        16 00   NHI     *       NLO     +01  IN SOURCE AREA
        FB 24   LOAD    *       $24FB   GET NEXT CHAR FROM WS
        EB 2D   HEXSP   *       $2DEB   PRINT A & X, HEX-VALUE

3FEC            PATCH   ORG     $3FEC   ANY ADDRESS WILL DO
3FEC 20 FB 24           JSR     LOAD    GET NEXT VALUE
3FEF 10 08              BPL     PATOK   BRANCH IF OK
3FF1 A5 16              LDAZ    NHI     GET LAST
3FF3 A6 15              LDXZ    NLO     VALID LINE #
3FF5 20 EB 2D           JSR     HEXSP   PRINT IT
3FF8 00                 BRK             ERROR # 0E
3FF9 60         PATOK   RTS             CHAR IN ACCU, RETURN
3FFA EA                 NOP             NEXT FREE ADDRESS

                ADDRESS IN FNDND ROUTINE:
2499                    ORG     $2499   WAS JSR LOAD FETCH CHAR
2499 20 EC 3F           JSR     PATCH   NEW ROUTINE WITH CHECK
```

VRAAG EN AANBOD

Wim Schimmel, Calslaan 28 B - 62, 7522 MC Enschede,
Telf.: 053 - 895040
Ik ga het MON/DOS65 systeem bouwen, Ik bezit echter geen ASCII parallel toetsenbord, maar een IBM (compatible) toetsenbord. Dit heeft een ander programma in Eprom en een ACIA als interface. Wie wil mij helpen of informatie bezorgen om er een ASCII parallel toetsenbord van te maken?

LETTER TO THE EDITOR

Ronald Hermens, The Netherlands.

Several times I have noticed the malfunction of the display of my JUNIOR-computer. When I slightly touched the CPU-card it would function normally. It has happened that the connection with the Elekterminal broke down, and while reading a tape, first the green LED and later on, the red LED lighted up. I found that this was caused by sticking half out of the socket, of the PIA on the CPU-card.
The processor and the Eprom bungled down not so well either, according to me this is caused by: using no top-quality sockets, and the continuous little movements of the CPU-card, (because of the keyboard) in co-operation with the gravitation. Solution: check this regularly, or take an isolated (!) copper wire with a massive nucleus, put this under the socket and wrap both ends on top of the IC round each-other tight. This can only be done when there is sufficient space between socket and pcb.

```
                    )172  * PION TWEE VELDEN VOORUIT *
                    )173                              ;
28F4: B0 46         )174  BE        BCS  BL
28F6: 18            )175  BF        CLC
28F7: 65 05         )176            ADC  VAN
28F9: 4A            )177            LSRA
28FA: AA            )178            TAX
28FB: B5 16         )179            LDA  BORD,X
28FD: D0 3D         )180            BNE  BL
28FF: 86 13         )181            STX  EPS
2901: A6 04         )182  BG        LDX  NAAR
2903: B5 16         )183            LDA  BORD,X           ;
2905: D0 35         )184            BNE  BL
                    )185                              ;
                    )186  * PION PROMOTIE *
                    )187                              ;
2907: E0 08         )188  BH        CPX  #8
2909: 90 04         )189            BCC  BI
290B: E0 38         )190            CPX  #$38
290D: 90 30         )191            BCC  BM
290F: A6 05         )192  BI        LDX  VAN
2911: B5 16         )193            LDA  BORD,X
2913: 48            )194            PHA
2914: 09 03         )195            ORA  #3
2916: 85 03         )196            STA  PROM
2918: A5 03         )197  BJZZ      LDA  PROM
291A: A6 05         )198            LDX  VAN
291C: 95 16         )199            STA  BORD,X
291E: 29 07         )200            AND  #7
2920: C9 07         )201            CMP  #7
2922: F0 15         )202            BEQ  BK
2924: AA            )203            TAX
2925: A5 0B         )204            LDA  WRDE
2927: 48            )205            PHA
2928: 18            )206            CLC
2929: 75 5D         )207            ADC  TSW,X
292B: 85 0B         )208            STA  WRDE
292D: C6 0B         )209            DEC  WRDE
292F: 20 8E 2A      )210            JSR  MOVE
2932: 68            )211            PLA
2933: 85 0B         )212            STA  WRDE
2935: E6 03         )213            INC  PROM
2937: D0 DF         )214            BNE  BJZZ
2939: 68            )215  BK        PLA
293A: 95 16         )216            STA  BORD,X
293C: 4C 6D 28      )217  BL        JMP  BOK
293F: 4C 6A 28      )218  BM        JMP  BOJ
                    )219                              ;
                    )220  * TEST OP SCHAAK *
                    )221                              ;
2942: A5 13         )222  BN        LDA  EPS
2944: 48            )223            PHA
2945: 84 13         )224            STY  EPS
2947: A9 40         )225            LDA  #$40
2949: 85 05         )226            STA  VAN
294B: 85 04         )227            STA  NAAR
294D: 06 06         )228            ASL  NZET
294F: 20 8E 2A      )229            JSR  MOVE
2952: A2 BF         )230            LDX  #$BF
2954: A5 06         )231            LDA  NZET
2956: F0 79         )232            BEQ  BBB
2958: 46 06         )233            LSR  NZET
295A: 90 77         )234            BCC  BBC
                    )235                              ;
                    )236  * KORTE ROCHADE *
                    )237                              ;
295C: 20 F8 29      )238            JSR  RC
295F: B0 30         )239            BCS  BONZ
2961: E8            )240            INX
2962: B5 16         )241            LDA  BORD,X
2964: 86 04         )242            STX  NAAR
2966: E8            )243            INX
2967: 15 16         )244            ORA  BORD,X
```

DIRECTORY DISK 1 'SYSTEM LOYS DISKETTE'
for Elektor's OCTOPUS computers

By : Fernando Lopez, Portugal

TRACK	SEC	PAG	MEMORY	NAME	DESCRIPTION
00 -	1 -	8	2200,29FF	V3.3/1	OS-65D V3.3 part 1
01 -	1 -	8	2A00,31FF	V3.3/2	OS-65D V3.3 part 2
02 -	1 -	8	0200,09FF	BAS/1	BASIC part 1
03 -	1 -	8	0A00,11FF	BAS/2	BASIC part 2
04 -	1 -	8	1200,19FF	BAS/3	BASIC part 3
05 -	1 -	8	1A00,21FF	BAS/4	BASIC part 4
06 -	1 -	1	2200,22FF	B/5V/3	BASIC part 5
	2 -	1	3200,32FF		OS-65D V3.3 part 3
	3 -	1	0000,00FF		OS-65D V3.3 part 0
	4 -	4	E400,E7FF		BAS/ASM/WP HANDLER DOS EXTENSIONS SHORTHAND BASIC
07 -	1 -	8	0200,09FF	ASM/1	
08 -	1 -	8	0A00,11FF	ASM/2	
09 -	1 -	8	1200,19FF	ASM/3	
10 -	1 -	8	2200,29FF	TOV3.1	OS-65D V3.2 part 1
11 -	1 -	8	2A00,31FF	T1V3.1	OS-65D V3.2 part 2
12 -	1 -	1	2E79,2F78	DIRECT	DIRECTORY part 1
	2 -	1	2E79,2F78		DIRECTORY part 2
	3 -	1	20C4,21C3		BASIC OVERLAYS
	4 -	1	2E79,2F78		PUT/GET OVERLAYS (BASIC)
	5 -	2	DD00,DEFF		FULL SCREEN EDITOR for BASIC
13 -	1 -	8	3274,3A73	V3.3/4	OS-65D V3.3 part 4
14 -	1 -	8		BEXEC*	UTILITIES (DISK/)
15 -	1 -	8			"
16 -	1 -	8			"
17/18	1 -	8		COPIER	COPIER UTILITY
19 -	1 -	8		CHANGE	CHANGE PARAM. UTIL
20 -	1 -	8			"
21/23	1 -	8		GARBAGE	GARB.COLLECTOR AND UTILITIES
24 -	1 -	8		DIR	DIRECTORY UTILITY
25 -	1 -	8		MERGE	FILE MERGE UTILITY
26 -	1 -	1		COPY**	
	2 -	1	5C00,5CFF		BEXEC* CREATE UTIL
	3 -	1	5D00,5DFF		"
	4 -	1	5E00,5EFF		"
	5 -	1	5F00,5FFF		"
	6 -	1	5C00,5CFF		"
	7 -	1	25A0,269F		"
27 -	1 -	8		RENAME	RENAME FILE UTIL
28/30	1 -	8	0200,19FF	WP1.1	WORDPROCESSOR
31 -	1 -	8		TRACE	TRACE (BASIC PROG. UTILITY
32 -	1 -	8		NUMCON	NUMBER CONVERSION
33/34	1 -	8		FORCE2	VT-52 COMPATIBLE TERMINAL
35 -	1 -	8		ATNENB	ATN vs PRINT EXTENSIONS
36 -	1 -	5		COP/TO	
	2 -	1	DC00,DCFF		FILE-MERGE UTIL BASIC-TEXT
37 -	1 -	8		PARSER	CALC. PARALLEL AND SERIAL RESISTOR
38 -	1 -	8		DUMP*	DECIMAL DUMP UTIL
39 -	1 -	5	0200,06FF	COM/TO	COMPAR / TRACK ZERO
	2 -	2	2000,21FF		"

```
2969: D0 26                 BNE  BONZ
296B: 06 06                 ASL  NZET
296D: 20 8E 2A              JSR  MOVE
2970: 46 06                 LSR  NZET
2972: 90 1D                 BCC  BONZ
2974: A6 05                 LDX  VAN
2976: E8                    INX
2977: E8                    INX
2978: 86 04                 STX  NAAR
297A: E8                    INX
297B: B5 16                 LDA  BORD,X
297D: 94 16                 STY  BORD,X
297F: CA                    DEX
2980: CA                    DEX
2981: 95 16                 STA  BORD,X
2983: 20 66 2B              JSR  MVRO
2986: A6 05                 LDX  VAN
2988: E8                    INX
2989: B5 16                 LDA  BORD,X
298B: 94 16                 STY  BORD,X
298D: E8                    INX
298E: E8                    INX
298F: 95 16                 STA  BORD,X

               * LANGE ROCHADE *

2991: 20 F8 29     BONZ     JSR  RC
2994: 0A                    ASLA
2995: B0 36                 BCS  BBA
2997: CA                    DEX
2998: 86 04                 STX  NAAR
299A: B5 16                 LDA  BORD,X
299C: CA                    DEX
299D: 15 16                 ORA  BORD,X
299F: CA                    DEX
29A0: 15 16                 ORA  BORD,X
29A2: D0 29                 BNE  BBA
29A4: 06 06                 ASL  NZET
29A6: 20 8E 2A              JSR  MOVE
29A9: 46 06                 LSR  NZET
29AB: 90 20                 BCC  BBA
29AD: A6 05                 LDX  VAN
29AF: CA                    DEX
29B0: CA                    DEX
29B1: 86 04                 STX  NAAR
29B3: CA                    DEX
29B4: CA                    DEX
29B5: B5 16                 LDA  BORD,X
29B7: 94 16                 STY  BORD,X
29B9: E8                    INX
29BA: E8                    INX
29BB: E8                    INX
29BC: 95 16                 STA  BORD,X
29BE: 20 66 2B              JSR  MVRO
29C1: A6 05                 LDX  VAN
29C3: CA                    DEX
29C4: B5 16                 LDA  BORD,X
29C6: 94 16                 STY  BORD,X
29C8: CA                    DEX
29C9: CA                    DEX
29CA: CA                    DEX
29CB: 95 16                 STA  BORD,X

               * EINDROUTINE CALC *

29CD: A6 06        BBA      LDX  NZET
29CF: D0 02                 BNE  BBC
29D1: 86 08        BBB      STX  ZWRD
29D3: 68           BBC      PLA
29D4: 85 13                 STA  EPS
29D6: 98           BBD      TYA
29D7: 38                    SEC
29D8: E5 08                 SBC  ZWRD
```

POSVAL SCHAAKMONITOR
==================

Wijzigingen in gepubliceerde listings.

Door : Frans Raaijmakers.

Ik heb de gepubliceerde listings nog eens nagekeken. Ondanks alle eerdere kontroles, zijn er een paar foutjes ingeslopen, twee in de programmalisting zelf en een grote in de informatie over de beginadressen van de verschillende routines op pag. 27 van editie 45. De wijzigingen die nodig zijn in de thans gepubliceerde listing konden niet meer verwerkt worden en worden eveneens hierbij gegeven.

Meteen in het begin van het programma op regel 16, editie 45, staat JSR $2000. Dat moet zijn JSR ADD (20 03 20). Op pagina 27 staan de adressen van de schaakmonitor en POSVAL welke als volgt moeten worden gewijzigd:

590 = 2B98	5F3 = 2BFB	629 = 2C31	683 = 2C8B
5A5 = 2BAD	5FC = 2C04	64F = 2C57	68D = 2C95

De IRQ en NMI adressen moeten nog aangepast worden. Zie de listing in deze editie. Wijzig deze als volgt:
2C9E = A2 7F   2CA3 = A2 2B   2CA8 = A2 2C   2CAD = A2 2A

===
=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0
===

```
 10;BELL-routine for OCTOPUS/EC65
 20;with BASICODE interface-card
 30;By Coen Boltjes 015-136812
 40;   VIDIBUS       400029830
 50;The routine can be placed after
 60;the PRINT&(X,Y) routine of the
 70;ELECTOR COMPUTING SPECIAL 2.
 80;and produces an "Bleep" on the
 90;output 8 of IC4, which can be
100;connected with an amplifier in
110;for example your monitor.
120;You have to adjust the VIDEO-
130;Commandinterpreter in the EPROM.
140;$F398 $18 -) $1A  END OF TABLE
150;$F3B9 $FF -) $00  NO ESCAPE
160;$F3BA $FF -) $07  ASCII BELL
170;$F3D9 $FF -) LSB of BELL
180;$F3DA $FF -) MSB of BELL
190;Parameters for 2 MHz
200;
210OUT    =$E280      ;BASE I/O ADDRESS
220;
230BELL    LDA #$80
240        LDY #$FF   ;DEFINE LENGTH
250BELLA   STA OUT    ;
260        LDX #$40   ;DEFINE HEIGHT
270BELLB   DEX
280        BNE BELLB  ;=)LOOP
290        EOR #$80   ;TOGGLE BIT 7
300        DEY
310        BNE BELLA  ;=)NOT READY
320        JMP $F00F  ;EXIT VIA VIDEND
330.END
340;You now can implement the BASICODE
350;subroutine 250
360;    250 PRINT CHR$(7);:RETURN
```

```
29DA: 85 08     )320            STA  ZWRD
29DC: 98        )321            TYA
29DD: 38        )322            SEC
29DE: E6 0B     )323            INC  WRDE
29E0: 85 0B     )324            STA  WRDE
29E2: A5 15     )325            LDA  CKAZ
29E4: 49 FF     )326            EOR  #$FF
29E6: 85 15     )327            STA  CKAZ
29E8: C6 0C     )328            DEC  NIVO
29EA: A6 0C     )329            LDX  NIVO
29EC: E0 02     )330            CPX  #2
29EE: D0 04     )331            BNE  EXCL
29F0: A5 06     )332            LDA  NZET
29F2: 85 91     )333            STA  OZET
29F4: 60        )334  EXCL      RTS
                )335                         ;
29F5: EA        )336            NOP
29F6: EA        )337            NOP
29F7: EA        )338            NOP
                )339                         ;
                )340  * INSPEKTIE ROCHADE  - CODE ROCO *
                )341                         ;
29F8: A2 04     )342  RC        LDX  #4
29FA: A5 0A     )343            LDA  ROCO
29FC: 24 15     )344            BIT  CKAZ
29FE: 50 04     )345            BVC  ROI
2A00: A2 3C     )346            LDX  #$3C
2A02: 0A        )347            ASLA
2A03: 0A        )348            ASLA
2A04: 86 05     )349  ROI       STX  VAN
2A06: 0A        )350            ASLA
2A07: 60        )351            RTS
                )352                         ;
                )353  * INPUT *
                )354                         ;
2A08: A0 00     )355  HFD       LDY  #0
2A0A: A2 01     )356            LDX  #1
2A0C: B5 FA     )357  MI        LDA  DISP2,X
2A0E: 18        )358            CLC
2A0F: 69 5F     )359            ADC  #$5F
2A11: 95 FA     )360            STA  DISP2,X
2A13: 4A        )361            LSRA
2A14: 4A        )362            LSRA
2A15: 4A        )363            LSRA
2A16: 4A        )364            LSRA
2A17: 85 14     )365            STA  HULP
2A19: B5 FA     )366            LDA  DISP2,X
2A1B: 29 07     )367            AND  #7
2A1D: 0A        )368            ASLA
2A1E: 0A        )369            ASLA
2A1F: 0A        )370            ASLA
2A20: 05 14     )371            ORA  HULP
2A22: 95 01     )372            STA  ZETII,X
2A24: CA        )373            DEX
2A25: 10 E5     )374            BPL  MI
2A27: 86 90     )375            STX  WIS
2A29: 4C 6E 2A  )376            JMP  MV
                )377                         ;
                )378  * HOOFROUTINE DEEL BRAKE *
                )379                         ;
2A2C: 84 00     )380  BREAK     STY  ZETI
2A2E: A6 90     )381            LDX  WIS
2A30: F0 56     )382            BEQ  DISP
2A32: 9A        )383            TXS
2A33: E8        )384            INX
2A34: 86 0D     )385            STX  CZA
2A36: 86 90     )386            STX  WIS
2A38: 86 0F     )387            STX  PZET
2A3A: 86 0C     )388            STX  NIVO
2A3C: A2 01     )389            LDX  #1
2A3E: 86 11     )390            STX  MAXI

2A40: A2 03     )392            LDX  #3
```


Author: Ronald Hermens, The Netherlands.
System: ELEKTOR's JUNIOR-computer

ERROR MESSAGES MICRO-ADE, ADAPTED FOR JUNIOR

NR.	ADDRESS OF BRK. INSTR.	REASON
07	$2905	INSTRUCTION SYNTAX ERROR
14	$3812	SK COMMAND WHILE K-FLAG = OFF
1C	$211A	INSERTION OVERFLOW ONLY (=9 LINES ARE ALLOWED
23	$2921	ILLEGAL ADDRESS MODE
26	$2224	ATTEMPT TO MOVE BEYOND THE END OF FILE CAN'T FIND 'FROM' LINE OR 'TO' LINE
2A	$3828	SK: FIRST LINE NOT FOUND
3B	$2439	SOURCE FILE SPACE IS FULL
49	$3147	ERROR IN 'V'-COMMAND
4A	$3B48	VK: K-FLAG NOT SET
52	$2E50	TOO MANY 'BACKSPACES' TYPED, CAN'T GO BEFORE BEGINNING OF LINE
68	$2366	COMMAND SYNTAX ERROR
6F	$286D	SYMBOL ALREADY EXISTS
74	$3372	NO ARGUMENT SUPPLIED
7B	$3379	NO ARGUMENT NEEDED
8C	$388A	SA:)= 2 PARAMETERS GIVEN
9E	$209C	COMMAND PARAMETER SYNTAX ERROR PARAMETERS SHOULD BE 0 TO 2 NUMBERS
A4	$28A2	SYMBOL TABLE OVERFLOW
A8	$29A6	UNDEFINED SYMBOL
B6	$3CB4	CHECK ZERO-PAGE WITH ABS IS FALSE SEE $3C9C MEANING = ??
B9	$31B7	ERROR IN 'V'-COMMAND ??????
C1	$3CBF	SPACES IN + - ARITHMETIC ADJUSTMENTS
D4	$24D2	ATTEMPT TO MOVE BEGINNING OF FILE ERROR IN "SPACE" $24B2, BLO,BHI NOT IN SOURCE AREA, YOUR SOURCE IS ALREADY MUTILATED
DF	$37DD	K: # PARAMETERS MUST BE 0: RESET K-FLAG, OR 2: SET K-RANGE
E2	$28E0	ADDRESS MODE SYNTAX ERROR
F6	$2AF4	BRANCH OUT OF RANGE

*********(9C) FOLLOWED BY
*********(E7) AND THIS REPEATED ENDLESSLY
CAUSE: ERROR IN FNDND $2496
THIS ROUTINE LOOKS FOR END OF SOURCE SYMBOL, EVEN OUTSIDE SOURCE AREA. WHEN IT READS ADDRESSES ON PIA, CPU, IT WILL SET OFF THE TIMER, WHICH GENERATES INTERRUPTS UNTIL YOU HIT RESET, OR RE-INITIALISE THIS PIA.

```
2A42: 86 12     )393            STX  MAXII
2A44: 20 08 28  )394            JSR  CALC       ; HIER PATCH VOOR OPENINGEN
2A47: A5 08     )395            LDA  $08
2A49: 85 F9     )396            STA  DISP1
2A4B: C4 06     )397            CPY  $06
2A4D: F0 31     )398            BEQ  CODE
2A4F: A2 01     )399            LDX  #1
2A51: B5 01     )400  MIII      LDA  ZETII,X
2A53: 4A        )401            LSRA
2A54: 4A        )402            LSRA
2A55: 4A        )403            LSRA
2A56: 85 14     )404            STA  HULP
2A58: B5 01     )405            LDA  ZETII,X
2A5A: 29 07     )406            AND  #7
2A5C: 0A        )407            ASLA
2A5D: 0A        )408            ASLA
2A5E: 0A        )409            ASLA
2A5F: 0A        )410            ASLA
2A60: 05 14     )411            ORA  HULP
2A62: 18        )412            CLC
2A63: 69 A1     )413            ADC  #$A1
2A65: 95 FA     )414            STA  DISP2,X
2A67: CA        )415            DEX
2A68: 10 E7     )416            BPL  MIII
2A6A: A5 00     )417            LDA  ZETI
2A6C: 85 F9     )418            STA  DISP1
                )419                            ;
                )420  * OUTPUT *                ;
                )421                            ;
2A6E: 86 0D     )422  MV        STX  CZA
2A70: 9A        )423            TXS
2A71: E8        )424            INX
2A72: 86 0C     )425            STX  NIVO
2A74: E8        )426            INX
2A75: 86 11     )427            STX  MAXI
2A77: 86 12     )428            STX  MAXII
2A79: 20 08 28  )429            JSR  CALC
2A7C: A9 FF     )430  FOUT      LDA  #$FF
2A7E: 85 F9     )431            STA  DISP1
2A80: A9 C0     )432  CODE      LDA  #$C0
2A82: 85 FB     )433            STA  DISP3
2A84: A9 DE     )434            LDA  #$DE
2A86: 85 FA     )435            STA  DISP2
2A88: 4C AD 2B  )436  DISP      JMP  MMI
2A8B: 4C 88 2A  )437            JMP  DISP
                )438                            ;
                )439  * SUBROUTINE MOVE ROCHADE *
                )440                            ;
2A8E: 84 0E     )441  MOVE      STY  CZO
2A90: A6 04     )442            LDX  NAAR
2A92: B5 16     )443            LDA  BORD,X
2A94: 85 09     )444            STA  STUK
2A96: D0 15     )445            BNE  BIO
2A98: C4 06     )446            CPY  NZET
2A9A: F0 22     )447            BEQ  BO
2A9C: A5 11     )448            LDA  MAXI
2A9E: C5 0C     )449            CMP  NIVO
2AA0: B0 1C     )450            BCS  BO
2AA2: E6 06     )451            INC  NZET
2AA4: A5 0B     )452            LDA  WRDE
2AA6: C5 08     )453            CMP  ZWRD
2AA8: 30 02     )454            BMI  BJ
2AAA: 85 08     )455            STA  ZWRD
2AAC: 60        )456  BJ        RTS
2AAD: 45 15     )457  BIO       EOR  CKAZ
2AAF: 29 40     )458            AND  #$40
2AB1: F0 0A     )459            BEQ  EXM
2AB3: B5 16     )460            LDA  BORD,X
2AB5: 29 07     )461            AND  #7
2AB7: C9 02     )462            CMP  #2
2AB9: D0 03     )463            BNE  BO
```

CORRECTION ON THE 68000 ARTICLE 6502-KENNER 4 1986.

In rereading my article about the 68000, I found two more or less significant errors:

- The example in section 3 should be:

```
MOVEA.L #$00000000,A0
MOVEA.W #$FFFF,A0      ; A0 contains now $FFFFFFFF
CMPA.L  #$FFFF,A0
BEQ.B   LABEL
```

When we use D0 in stead of A0, D0 contains $FFFF.

- The Addressing modes: Absolute Register Indirect Addressing with Index and Program Counter with Index are incorrect described. The right description is: The contents of an address or data register is added to another address register. An 8 bit signed displacement is added to this result giving the address of the operand. So the indirection is always in an address register, never in memory.

This means that indexed addressing ZERO and ABSOLUTE X and Y on the 6502 is also present on the 68000 and indexed indirect and indirect indexed are not present on the 68000.

```
10  REM
        EI-WEG

20  REM   De volgende subroutine kan bij APPLE van p
    as komen als je een invoer
30  REM   wilt plegen met komma's en/of dubbele punt
    .
40  REM
50  REM   Je krijgt wel een EXTRA IGNORED melding, m
    aar deze heeft geen invloed
60  REM   op de samenstelling van IN$.
70  REM
80  REM          Frans Verberkt, Nijmegen
90  REM
1000  REM
        INVOER MET , EN :

1010  INPUT IN$: REM .............Haal een invoerreg
    el
1020 LE =  LEN (IN$): REM ........Bepaal lengte hier
    van
1030 IB = 512: REM ...............$0200 = begin Inpu
    tBuffer voor APPLE
1040 CH =  PEEK (IB + LE): REM ...Haal karakter
1050  IF CH = 0 THEN 1090: REM ...Einde input = $00
1060 IN$ = IN$ +  CHR$ (CH): REM .Zet dit karakter n
    a IN$
1070 LE = LE + 1: REM ............Verhoog Lengtetell
    er
1080  GOTO 1040: REM .............en kijk naar volge
    nd karakter
1090  RETURN
```

```
2ABB: C6 0E     )465          DEC  CZO
2ABD: 60        )466  EXM     RTS
2ABE: A2 08     )467  BO      LDX  #8
2AC0: B5 03     )468  BOI     LDA  PROM,X
2AC2: 48        )469          PHA
2AC3: CA        )470          DEX
2AC4: 10 FA     )471          BPL  BOI
2AC6: A0 06     )472          LDY  #6
2AC8: A5 0A     )473          LDA  ROCO
2ACA: B6 83     )474  BIA     LDX  $83,Y
2ACC: E4 05     )475          CPX  VAN
2ACE: F0 04     )476          BEQ  BIB
2AD0: E4 04     )477          CPX  NAAR
2AD2: D0 03     )478          BNE  BIC
2AD4: 19 89 00  )479  BIB     ORA  $89,Y
2AD7: 88        )480  BIC     DEY
2AD8: D0 F0     )481          BNE  BIA
2ADA: 85 0A     )482          STA  ROCO
2ADC: A6 05     )483          LDX  VAN
2ADE: B5 16     )484          LDA  BORD,X
2AE0: 94 16     )485          STY  BORD,X
2AE2: A6 04     )486          LDX  NAAR
2AE4: 95 16     )487          STA  BORD,X
2AE6: A5 09     )488          LDA  STUK
2AE8: F0 0A     )489          BEQ  BID
2AEA: 29 07     )490          AND  #7
2AEC: AA        )491          TAX
2AED: A5 0B     )492          LDA  WRDE
2AEF: 18        )493          CLC
2AF0: 75 5D     )494          ADC  TSW,X
2AF2: 85 0B     )495          STA  WRDE
2AF4: A5 0B     )496  BID     LDA  WRDE
2AF6: 85 08     )497          STA  ZWRD
2AF8: A5 12     )498          LDA  MAXII
2AFA: C5 0C     )499          CMP  NIVO
2AFC: 90 03     )500          BCC  BRA
2AFE: 20 08 28  )501          JSR  CALC
2B01: A2 00     )502  BRA     LDX  #0
2B03: 68        )503  BIE     PLA
2B04: 95 03     )504          STA  PROM,X
2B06: E8        )505          INX
2B07: E0 05     )506          CPX  #5
2B09: D0 F8     )507          BNE  BIE
2B0B: 24 0E     )508          BIT  CZO
2B0D: E6 0E     )509          INC  CZO
2B0F: 68        )510          PLA
2B10: 70 3E     )511          BVS  BIK
2B12: E6 06     )512          INC  NZET
2B14: 24 05     )513          BIT  VAN
2B16: 70 38     )514          BVS  BIK
2B18: A6 0C     )515          LDX  NIVO
2B1A: E0 01     )516          CPX  #1
2B1C: D0 2E     )517          BNE  BIJ
2B1E: 24 0D     )518          BIT  CZA
2B20: 70 1E     )519          BVS  BIH
2B22: C5 08     )520          CMP  ZWRD
2B24: D0 09     )521          BNE  BIF
2B26: 4C 8A 27  )522          JMP  STUUR    ; JUMP POSVAL
2B29: EA        )523          NOP
2B2A: D0 03     )524          BNE  BIF
2B2C: CD F4 1A  )525          CMP  TIMER
2B2F: 10 1F     )526  BIF     BPL  BIK      ; RETOURADRES POSVAL
2B31: A6 91     )527          LDX  OZET
2B33: 86 0F     )528          STX  PZET
2B35: A2 02     )529          LDX  #2
2B37: B5 03     )530  BIG     LDA  PROM,X
2B39: 95 00     )531          STA  ZETI,X
2B3B: CA        )532          DEX
2B3C: 10 F9     )533          BPL  BIG
2B3E: 30 12     )534          BMI  BIL
                )535  *
2B40: A2 02     )536  BIH     LDX  #2
2B42: B5 03     )537  BII     LDA  PROM,X
```

F.J.M. Sweehuijzen, The Netherlands

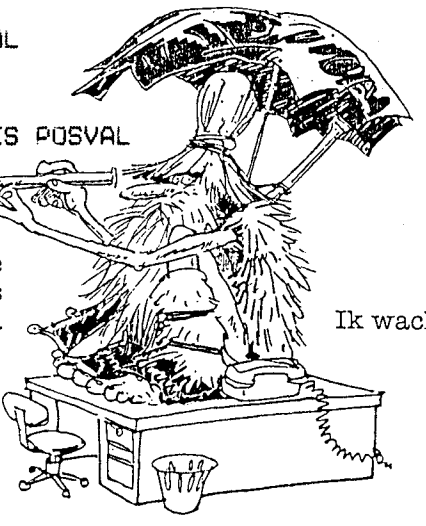
When I had the intention to connect a printer with a serial interface on 9600 BPS by using ready/busy handshake on pin 5 (CTS) to the ACIA of Elektor's CPU card, it did not work at all. Because I,m in favourable circumstances having a datascoop, I was enabled to examine the ACIA's output. To my astonishment there was nothing coming out at all in spite of changing all data formats of the datascoop so it had to be anything else.
I checked the straps on the CPU card over and over again, corresponding to the German 'Sonderheft 19'. There was no mistake found on any strap so the next step was looking into the source-listing of Paperware 3. Here I discovered the mistake there has been made. In the 'Sonderheft 19' the command register is programmed by PL4 and the control register is programmed by PL3 but in Paperware 3 it is done in reverse.
So there were two solutions: reversing PL3 and PL4 straps or changing in the routine 'CTLCMD' the saving of ACICMD and ACICTL.

==========

We expect for Elektor's OCTOPUS-computer: METAFORTH (OCTOPUSFORTH 1.2) on two 80 tracks single side diskettes, containing the new FORTH-compiler and all source-screens, developed by our member Fricus Jonkman, The Netherlands, completed with handy manual.
What is META? META is a program to generate a new FORTH from source-screens, that describe a FORTH. META can be used to
- relocate your FORTH
- make a headerless FORTH
- make a ROM-able code
- make changes anywhere in FORTH
- make dedicated FORTH-systems with a minimum kernel

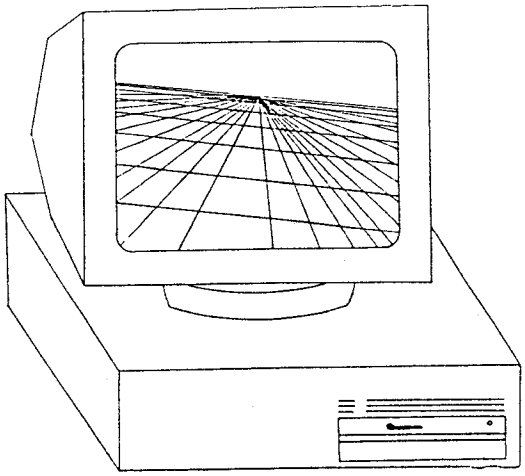
The new FORTH will start at the value of the variable METASTART, which can be changed. META is a vocabulary within the ASSEMBLER-vocabulary.
Ask for more news at the editorial office.

==========



Maandag voel ik me altijd als Robinson Crusoë.

Ik wacht op Vrijda



```
2B44: D5 00     )539             CMP  ZETI,X
2B46: D0 04     )540             BNE  BIJ
2B48: CA        )541             DEX
2B49: 10 F7     )542             BPL  BII
2B4B: 00        )543             BRK
                )544                                  ;
                )545  BIJ
2B4C: C5 08     )546             CMP  ZWRD
2B4E: 30 02     )547             BMI  BIL
2B50: 85 08     )548  BIK        STA  ZWRD
2B52: A6 04     )549  BIL        LDX  NAAR
2B54: B4 16     )550             LDY  BORD,X
2B56: 68        )551             PLA
2B57: 95 16     )552             STA  BORD,X
2B59: A6 05     )553             LDX  VAN
2B5B: 94 16     )554             STY  BORD,X
2B5D: 68        )555             PLA
2B5E: 85 0A     )556             STA  ROCO
2B60: 68        )557             PLA
2B61: 85 0B     )558             STA  WRDE
2B63: A0 00     )559             LDY  #0
2B65: 60        )560             RTS
2B66: A9 01     )561  MVRO       LDA  #1
2B68: C5 0C     )562             CMP  NIVO
2B6A: D0 02     )563             BNE  MVY
2B6C: 84 0F     )564             STY  PZET
2B6E: 20 8E 2A  )565  MVY        JSR  MOVE
2B71: 60        )566             RTS
2B72: A4 05     )567  PTCH       LDY  VAN
2B74: C0 03     )568             CPY  #3
2B76: D0 02     )569             BNE  BRC
2B78: 86 91     )570             STX  OZET
2B7A: A6 0F     )571  BRC        LDX  PZET
2B7C: E4 91     )572             CPX  OZET
2B7E: 60        )573             RTS
                )574                                  ;
                )575  * NMI *                         ;
                )576                                  ;
2B7F: A9 00     )577  NMI        LDA  #0              ; KOM HIERNA STOPTOETS
2B81: 8D 82 1A  )578             STA  PBD
2B84: A9 81     )579             LDA  #$81
2B86: 85 BF     )580             STA  POSWRDHOB       ; POSVALTELLER GROTE NEGATIEVE WAARDE
2B88: EA        )581             NOP
2B89: EA        )582             NOP
2B8A: EA        )583             NOP
2B8B: EA        )584             NOP
2B8C: EA        )585             NOP
2B8D: EA        )586             NOP
2B8E: EA        )587             NOP
2B8F: EA        )588             NOP
2B90: EA        )589             NOP
2B91: EA        )590             NOP
2B92: EA        )591             NOP
2B93: EA        )592             NOP
2B94: EA        )593             NOP
2B95: 4C 08 2A  )594             JMP  HFD             ; JUMP INPUT
2B98: 20 8E 1D  )595  TDDS1      JSR  SCANDS          ; STUUR DISPLAYS
2B9B: D0 FB     )596             BNE  TDDS1
2B9D: 20 8E 1D  )597  TDDS2      JSR  SCANDS          ; ONDERZOEK TOETSENBORD
2BA0: F0 FB     )598             BEQ  TDDS2           ; ZET TOETSWAARDE IN
2BA2: 20 8E 1D  )599             JSR  SCANDS          ; KEYBUFFER
2BA5: F0 F6     )600             BEQ  TDDS2
2BA7: 20 F9 1D  )601             JSR  GETKEY
2BAA: 85 E1     )602             STA  KEYBUF
2BAC: 60        )603             RTS
2BAD: A5 F9     )604  MMI        LDA  DISP1           ; STUUR 4 DISPLAY ALS $F9=0: ANDERS 6 DISPLAYS
2BAF: F0 06     )605             BEQ  MM2
2BB1: A9 03     )606             LDA  #3
2BB3: 85 F6     )607             STA  $F6
2BB5: D0 04     )608             BNE  STARTMON
2BB7: A9 02     )609  MM2        LDA  #2
2BB9: 85 F6     )610             STA  $F6
2BBB: 20 98 2B  )611  STARTMON   JSR  TDDS1
```

```
2BBE: 29 F0     )613             AND  #%11110000 ; TEST SOORT TOETS
2BC0: D0 12     )614             BNE  MM4
2BC2: A2 04     )615             LDX  #4         ; NUM.TOETS - SCHUIF DISPLAYS EEN POS. NAAR LINKS
2BC4: 06 FA     )616 MM3         ASL  DISP2
2BC6: 26 FB     )617             ROL  DISP3
2BC8: CA        )618             DEX
2BC9: D0 F9     )619             BNE  MM3
2BCB: A5 E1     )620             LDA  KEYBUF     ; ZET NIEUWE WAARDE IN RECHTER DISPLAY
2BCD: 05 FA     )621             ORA  DISP2
2BCF: 85 FA     )622             STA  DISP2
2BD1: 4C B7 2B  )623             JMP  MM2
2BD4: A5 E1     )624 MM4         LDA  KEYBUF     ; FUNKTIETOETS
2BD6: C9 14     )625             CMP  #$14       ; PC-TOETS ?
2BD8: D0 03     )626             BNE  MM5
2BDA: 4C 95 2C  )627             JMP  PCMON      ; JUMP PCMON
2BDD: C9 12     )628 MM5         CMP  #$12       ; + TOETS ?
2BDF: D0 06     )629             BNE  MM6
2BE1: 20 31 2C  )630             JSR  PM1        ; VOER ZET UIT
2BE4: 4C B7 2B  )631             JMP  MM2        ; JUMP STARTMON
2BE7: C9 11     )632 MM6         CMP  #$11       ; DA-TOETS ?
2BE9: D0 06     )633             BNE  MM7
2BEB: 20 57 2C  )634             JSR  DA1        ; VOER STUK IN
2BEE: 4C B7 2B  )635             JMP  MM2        ; JUMP STARTMON
2BF1: C9 10     )636 MM7         CMP  #$10       ; AD-TOETS ?
2BF3: D0 03     )637             BNE  MM8
2BF5: 20 8B 2C  )638             JSR  AD1        ; MAAK BORD LEEG
2BF8: 4C B7 2B  )639 MM8         JMP  MM2        ; JUMP STARTMON
                )640 * CODEMON *
2BFB: A9 C0     )641 CODEMON     LDA  #$C0       ; ZET CODE IN DISPLAYBUFFERS
2BFD: 85 FB     )642             STA  DISP3
2BFF: A9 DE     )643             LDA  #$DE
2C01: 85 FA     )644             STA  DISP2
2C03: 60        )645             RTS
                )646 * VELDKODEMON *
2C04: A5 92     )647 VK1         LDA  TEMP2      ; REKEN HET BORDVELD OM
2C06: 18        )648             CLC             ; NAAR INTERN FORMAAT
2C07: 69 60     )649             ADC  #$60       ; KONTROLEER DE GELDIGHEID
2C09: 30 23     )650             BMI  VK2        ; VELD ONGELDIG ALS LINKER NIBBLE + 60
2C0B: A5 92     )651             LDA  TEMP2      ; OPLEVERT MSB=1
2C0D: 29 0F     )652             AND  #%00001111 ; IDEM ALS RECHTER NIBBLE -1 ) 7
2C0F: 38        )653             SEC
2C10: E9 01     )654             SBC  #1
2C12: 29 08     )655             AND  #%00001000
2C14: D0 18     )656             BNE  VK2
2C16: A5 92     )657             LDA  TEMP2      ; KODEER
2C18: 18        )658             CLC             ; DEZE ROUTINE REKENT HET VELD OM NAAR
2C19: 69 5F     )659             ADC  #$5F       ; INTERN FORMAAT
2C1B: 85 92     )660             STA  TEMP2      ; A1 = 00   H8 = 3F
2C1D: 4A        )661             LSRA
2C1E: 4A        )662             LSRA
2C1F: 4A        )663             LSRA
2C20: 4A        )664             LSRA
2C21: 85 93     )665             STA  TEMP2+1
2C23: A5 92     )666             LDA  TEMP2
2C25: 29 07     )667             AND  #%00000111
2C27: 0A        )668             ASLA
2C28: 0A        )669             ASLA
2C29: 0A        )670             ASLA
2C2A: 05 93     )671             ORA  TEMP2+1
2C2C: 10 02     )672             BPL  VK2A
2C2E: A9 FF     )673 VK2         LDA  #$FF       ; KEER TERUG MET A=FF
2C30: 60        )674 VK2A        RTS             ; INDIEN VELD = ONGELDIG
                )675 * PLUSMON *
2C31: A2 01     )676 PM1         LDX  #1         ; PLUSMON VERPLAATST EEN STUK
2C33: B5 FA     )677 PM2         LDA  DISP2,X    ; KONTROLEER GELDIGHEID VELD
2C35: 85 92     )678             STA  TEMP2      ; EN KODEER VELD
2C37: 20 04 2C  )679             JSR  VK1
2C3A: C9 FF     )680             CMP  #$FF
2C3C: F0 15     )681             BEQ  PM3
2C3E: 95 94     )682             STA  TEMP2+2,X
2C40: CA        )683             DEX
2C41: 10 F0     )684             BPL  PM2        ; DISPLAY CODE INDIEN ZET = ONGELDIG
```

```
2C43: A0 00     )686            LDY  #0
2C45: A6 95     )687            LDX  TEMP2+3
2C47: B5 16     )688            LDA  BORD,X      ; VOER ZET UIT
2C49: F0 08     )689            BEQ  PM3         ; ZET VAN-VELD OP NAAR-VELD
2C4B: 94 16     )690            STY  BORD,X
2C4D: A6 94     )691            LDX  TEMP2+2     ; ZET VAN-VELD OP 00
2C4F: 95 16     )692            STA  BORD,X
2C51: 10 03     )693            BPL  PM4
2C53: 20 FB 2B  )694  PM3       JSR  CODEMON
2C56: 60        )695  PM4       RTS
                )696  * DAMON *
2C57: A6 FA     )697  DA1       LDX  DISP2       ; DAMON PLAATST EEN STUK
2C59: F0 19     )698            BEQ  DA2         ; KONTROLEER GELDIGHEID STUK
2C5B: 8A        )699            TXA
2C5C: 29 38     )700            AND  #%00111000  ; VOOR ALLE STUKKEN GELDT:
2C5E: D0 27     )701            BNE  DA3         ; B3, B4, B5 = 01
2C60: 8A        )702            TXA
2C61: 29 80     )703            AND  #%10000000  ; EN OOK MSB = 1
2C63: F0 22     )704            BEQ  DA3
2C65: 8A        )705            TXA
2C66: C9 C0     )706            CMP  #$C0        ; C0 IS GEEN STUK
2C68: F0 1D     )707            BEQ  DA3
2C6A: C9 81     )708            CMP  #$81        ; 81 IS GEEN STUK
2C6C: F0 19     )709            BEQ  DA3
2C6E: 29 07     )710            AND  #%00000111
2C70: C9 07     )711            CMP  #7
2C72: F0 13     )712            BEQ  DA3
2C74: A5 FB     )713  DA2       LDA  DISP3       ; KONTROLEER EN KODEER VELD
2C76: 85 92     )714            STA  TEMP2
2C78: 20 04 2C  )715            JSR  VK1
2C7B: C9 FF     )716            CMP  #$FF
2C7D: F0 08     )717            BEQ  DA3
2C7F: AA        )718            TAX              ; VOER STUK IN
2C80: A5 FA     )719            LDA  DISP2       ; ZET STUK VAN FA
2C82: 95 16     )720            STA  BORD,X      ; OP VELD VAN FB
2C84: E8        )721            INX
2C85: 10 03     )722            BPL  DA4
2C87: 20 FB 2B  )723  DA3       JSR  CODEMON     ; DISPLAY CODE INDIEN STUK
2C8A: 60        )724  DA4       RTS              ; OF VELD ONGELDIG
                )725  * ADMON *
2C8B: A2 3F     )726  AD1       LDX  #$3F        ; ADMON VEEGT BORD LEEG
2C8D: A9 00     )727            LDA  #0          ;ZET ALLE VELDEN OP 0
2C8F: 95 16     )728  AD2       STA  BORD,X
2C91: CA        )729            DEX
2C92: 10 FB     )730            BPL  AD2
2C94: 60        )731            RTS
2C95: A9 00     )732  PCMON     LDA  #0          ; KILL DISPLAY                 LABEL=PC1
2C97: 8D 82 1A  )733            STA  PBD
2C9A: A2 02     )734            LDX  #2          ; STUUR VIER DISPLAYS
2C9C: 86 F6     )735            STX  $F6
2C9E: A2 77     )736            LDX  #$77        ; SET NMI-VECTOR
2CA0: 8E 7A 1A  )737            STX  NMIVECTLOB
2CA3: A2 2C     )738            LDX  #$2C
2CA5: 8E 7B 1A  )739            STX  NMIVECTHOB
2CA8: A2 24     )740            LDX  #$24        ; SET IRQ-VECTOR
2CAA: 8E 7E 1A  )741            STX  IRQVECTLOB
2CAD: A2 2B     )742            LDX  #$2B
2CAF: 8E 7F 1A  )743            STX  IRQVECTHOB
2CB2: A2 55     )744            LDX  #$55        ; RESET ZERO-PAGE              LABEL=PC2
2CB4: 95 00     )745  RESZP     STA  ZETI,X
2CB6: CA        )746            DEX
2CB7: 10 FB     )747            BPL  RESZP
2CB9: 86 FA     )748            STX  DISP2       ; DISPLAY = FFFF
2CBB: 86 FB     )749            STX  DISP3
2CBD: A2 00     )750            LDX  #0          ; INITIEER ZEROPAGE
2CBF: BD F0 2C  )751  INITZP    LDA  PCLIJST,X                                  LABEL=PC3
2CC2: 95 15     )752            STA  CKAZ,X
2CC4: E8        )753            INX
2CC5: E0 CB     )754            CPX  #$CB
2CC7: D0 F6     )755            BNE  INITZP
2CC9: 20 98 2B  )756  KEY?      JSR  TDDS1       ; WACHT OP EEN TOETS           LABEL=PC4
2CCC: A5 E1     )757            LDA  KEYBUF
2CCE: F0 0E     )758            BEQ  ZWRT        ; TOETS 0 = ZWART
2CD0: C9 03     )759            CMP  #3          ; TOETS 1 = WIT
2CD2: D0 F5     )760            BNE  KEY?        ; LOOP INDIEN NIET 0 OF 3
2CD4: A0 00     )761            LDY  #0          ; JUNIOR SPEELT WIT
2CD6: 84 A0     )762            STY  KAZ         ; ZET POSVAL OP WIT
```

```
2CD8: EA        )763            NOP             ; HIER KAN DE KLEURINFORMATIE
2CD9: EA        )764            NOP             ; VOOR HET OPENINGENPROGRAMMA
2CDA: EA        )765            NOP             ; GEZET WORDEN
2CDB: 4C 2C 2A  )766            JMP  BREAK      ; JUMP INPUT VOOR EERSTE ZET
2CDE: A9 40     )767  ZWRT      LDA  #$40       ; JUNIOR SPEELT ZWART          LABEL=PC5
2CE0: 85 A0     )768            STA  KAZ        ; ZET POSVAL OP ZWART
2CE2: EA        )769            NOP             ; HIER KAN DE KLEURINFORMATIE
2CE3: EA        )770            NOP             ; VOOR HET OPENINGENPROGRAMMA
2CE4: EA        )771            NOP             ; GEZET WORDEN
2CE5: EA        )772            NOP
2CE6: EA        )773            NOP
2CE7: A9 00     )774            LDA  #0         ; DISPLAY = 0000
2CE9: 85 FA     )775            STA  DISP2
2CEB: 85 FB     )776            STA  DISP3
2CED: 4C B7 2B  )777            JMP  MM2        ; JUMP MAINSMON EN WACHT OP EERSTE ZET VAN WIT
2CF0: FF 84 83
2CF3: 85 86 82
2CF6: 85 83 84
2CF9: 80 80 80
2CFC: 80 80 80
2CFF: 80        )778  PCLIJST   HEX  FF848385868285838480808080808080
2D00: 80 00 00
2D03: 00 00 00
2D06: 00 00 00
2D09: 00 00 00
2D0C: 00 00 00
2D0F: 00        )779            HEX  80000000000000000000000000000000
2D10: 00 00 00
2D13: 00 00 00
2D16: 00 00 00
2D19: 00 00 00
2D1C: 00 00 00
2D1F: 00        )780            HEX  00000000000000000000000000000000
2D20: 00 C1 C1
2D23: C1 C1 C1
2D26: C1 C1 C1
2D29: C4 C3 C5
2D2C: C6 C2 C5
2D2F: C3        )781            HEX  00C1C1C1C1C1C1C1C1C4C3C5C6C2C5C3
2D30: C4 00 05
2D33: 0A 13 1C
2D36: 0E 0A 01
2D39: 01 00 03
2D3C: 05 03 08
2D3F: 05        )782            HEX  C400050A131C0E0A0101000305030805
2D40: 0F A4 24
2D43: 00 15 1F
2D46: B4 34 00
2D49: 06 16 22
2D4C: A2 26 36
2D4F: A6        )783            HEX  0FA42400151FB43400061622A22636A6
2D50: B6 00 2E
2D53: 3E AE BE
2D56: 66 76 E6
2D59: F6 00 06
2D5C: 16 22 A2
2D5F: 00        )784            HEX  B6002E3EAEBE6676E6F600061622A200
2D60: 04 07 38
2D63: 3C 3F 40
2D66: C0 80 10
2D69: 30 20 FF
2D6C: 00 00 00
2D6F: 00        )785            HEX  0407383C3F40C080103020FF00000000
2D70: 00 00 00
2D73: 00 00 00
2D76: 00 00 00
2D79: 00 00 00
2D7C: 00 00 00
2D7F: 00        )786            HEX  00000000000000000000000000000000
2D80: 00 00 00
2D83: 00 00 00
2D86: 00 00 00
2D89: 00 00 00
```

```
2D8C: 00 00 00
2D8F: 00          )787          HEX  00000000000000000000000000000000
2D90: 00 00 00
2D93: 00 00 00
2D96: 00 00 00
2D99: 00 80 00
2D9C: 00 00 27
2D9F: 46          )788          HEX  00000000000000000000800000002746
2DA0: 66 17 00
2DA3: 00 05 05
2DA6: 00 10 19
2DA9: 12 2D 07
2DAC: FF F7 F8
2DAF: F9          )789          HEX  66170000050500101912 2D07FFF7F8F9
2DB0: 01 09 08
2DB3: 91 90 89
2DB6: 99 09 10
2DB9: 11 01 00
2DBC: 00 00 00
2DBF: 00          )790          HEX  01090891908999091011010000000000

---End assembly---

3674 bytes

Errors: 0
```

PATCH IN THE OCTOPUS/EC65 STANDARD MONITOR

Author: Marc Lachaert, Belgium

If you want to connect a centronics interfaced printer to the Octopus, you will discover that when your printer is 'off line' or 'out of paper', the computer will still continue to send characters, so you will loose a whole part of your text. To remediate at this uncomfortable situation you will have to re-programm the 2732-Eprom at the 'centronics' routine.

Changes are made in the branch-instructions at addresses $F7B5, $F7BC, $F7C9 and $F7D0, according the below listing

```
F7B0 AD 10 E1  CENTRO LDA   VBPBD
F7B3 29 08            ANDIM $08    IS THE PRINTER SELECTED?
F7B5 F0 F9            BEQ   CENTRO WAIT FOR 'ONLINE'
F7B7 AD 10 E1         LDA   VBPBD
F7BA 29 10            ANDIM $10    DO WE HAVE STILL PAPER?
F7BC D0 F2            BNE   CENTRO WAIT FOR PAPER
F7BE AD 63 23  CENTRA LDA   AHOLD
F7C1 8D 00 E1         STA   VAPBD  ALL O.K., LET'S PRINT..
F7C4 AD 10 E1  WAIT   LDA   VBPBD
F7C7 29 08            ANDIM $08    STILL SELECTED?
F7C9 F0 F9            BEQ   WAIT   WAIT FOR 'ONLINE'
F7CB AD 10 E1         LDA   VBPBD
F7CE 29 10            ANDIM $10    STILL PAPER?
F7D0 D0 F2            BNE   WAIT   WAIT FOR PAPER
F7D2 AD 0D E1         LDA   VAIFR  WAIT FOR ACKNOWLEDGE
F7D5 29 10            ANDIM $10    SAMPLE PRINTER ANSWER
F7D7 F0 EB            BEQ   WAIT   NO ANSWER, WAIT UNTIL
F7D9 AD 0D E1         LDA   VAIFR  RESET FLAGS
F7DC 09 18            ORAIM $18
F7DE 8D 0D E1         STA VAIFR
F7E1 60               RTS
```

PATCHES ON DISK 5a (ORIGINAL OSI V3.3)

Author: Marc Lachaert, Belgium

On the disk I got from the club, I discovered a small amount of errors:

A. In Basic, the 'PRINT #(device)' does not work properly, because on track zero, at $25BF, an $61 was written in stead of $A9. Please change !

B. The Init, Call and Save functions do not work properly, because of the precense of $80 or $79 at the following locations on track zero:

$26CA ---) CHANGE TO $40 !
$2769 ---) CHANGE TO $39 !
$2779 ---) CHANGE TO $39 !

C. The 'Read/Write' utility on track 6, sector 4 did not have been transformed to the Octopus location for the disk controller !
Change in this programm all references to $C0XX (the controller... in $E0XX !
This must be done at locations $327, $32C, $333, $336, $339 and $365.

D. To make to work properly the breakpoint setting in the OSI extended monitor, change on track 9 the following locations:

$173A ---) 5 * NOP
$173F ---) LDA $18D5
$1742 ---) STA $E7CB
$1745 ---) LDA $18D6
$1748 ---) STA $E7CC

(This patch sets the IRQ vector on $1B25, where all registers are saved, and EM does his 'breakpoint handling').

At the same time, change $18DA from $10 to $16, to bring the number of disassembled instructions pro screen to 24.

Problemen met de Octopus 65 met RS 232 Printer

Oorzaak:

Het signaal DTR is verbonden met DSR en DCD en jumper 15-16 van PL-3 is aangebracht.
Bij een hardware reset worden de interne registers van de ACIA 6551 gereset.
In het Command Register (Hex E132) staat dan de waarde Hex ØØ.
De Monitor van de Octopus 65 initialiseert de ACIA. In het Command Register komt een waarde die wordt bepaald door de instelling van PL-3.
Als bit Ø van het Command Register van "Ø" naar "1" gaat , dan gaat DTR van "Ø" naar "1". Als bit Ø van het Command Register een "1" is , dan geeft de ACIA een $\overline{IRQ}$ wanneer DSR of DCD van status verandert.
DTR is verbonden met DSR en DCD , dus de ACIA geeft een $\overline{IRQ}$.

Gevolgen:

De Assembler start op adres Hex Ø2ØØ (= cold start assembler).
Hij vraagt "NEW?". Men kan deze vraag beantwoorden met "Y" of "N". Bij "Y" wordt op het scherm "CLEAR" geprint. Daarna ziet men zowel bij "Y" als bij "N" een oneindige reeks foutmeldingen op het scherm. Dit komt doordat op adres Hex Ø238 een Clear Interrupt Disable Bit instructie staat. Na deze instructie reageert de micro-processor op een $\overline{IRQ}$. Deze $\overline{IRQ}$ wordt gegeven door de ACIA.
De micro-processor springt naar het adres aangegeven door de $\overline{IRQ}$-vector.
De Assembler gebruikt de $\overline{IRQ}$-vector om bij foutieve invoer d.m.v. een BRK instructie te springen naar een routine op adres Hex Ø6Ø4.
Deze routine zorgt voor een foutmelding op het scherm en springt hierna naar adres Hex Ø231 (= warm start assembler).
De ACIA maakt de $\overline{IRQ}$ weer hoog wanneer het Status Register (Hex E131) wordt uitgelezen. Dit gebeurt echter niet door de Assembler , zodat op het scherm een oneindige reeks foutmeldingen verschijnt.

Oplossingen:

1: Verbreek de verbinding van DTR met DSR en DCD.

2: Verwijder jumper 15-16 van PL-3.

3: Lees voor het Booten (.B) eerst het Status Register uit m.b.v. de Monitor.

4: Toets voor het Booten eerst .S en .RF in. De routine die .S uitvoert leest het Status Register uit.

5: In ieder geval moet men er voor zorgen dat de ACIA geen $\overline{IRQ}$ geeft.
Bit Ø t/m 3 van het Command Register (PL-3 9-1Ø 11-12 13-14 15-16) controleren de interrupts.

Voorbeeld van de instelling PL-3 , PL-4 en PL-8 voor een RS 232 Printer

Gegevens : 30 char/sec = 300 Baud , 1 start bit , 7 data bits , 1 parity bit , 1 stop bit

0 = open 1 = jumper X = maak niet uit

PL-3

```
 1- 2  0
 3- 4  1 ─ even parity
 5- 6  1
 7- 8  0 ─ normal mode
 9-10  1 ─ Interrupt t.g.v. leeg Transmit Data Register disabled
11-12  0 ─ RTS = "0"   Transmitter on
13-14  1 ─ IRQ t.g.v. vol Receive Data Register disabled
15-16  X
```

PL-4

```
 1- 2  0 ─ 1 stop bit
 3- 4  0 ─ 7 data bits
 5- 6  1
 7- 8  1 ─ Baud rate generator intern
 9-10  0
11-12  1 ─ 300 Baud
13-14  1
15-16  0
```

PL-8

```
 1- 2  1 ─ Low speed modem
 3- 4  0
```

25-P D-Connector voor de Octopus 65

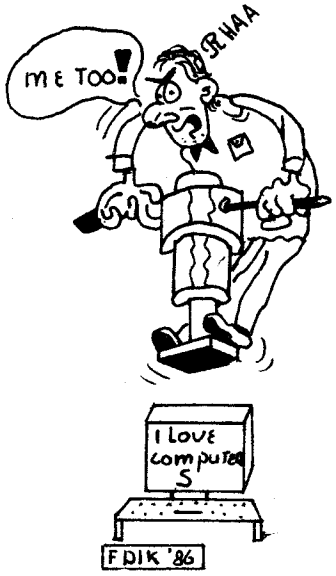
Transmitted Data (TD)	output	2	● ──→	Transmitted Data
Received Data (RD)	input	3	●	
Request To Send (RTS)	output	4	● ┐	
Clear To Send (CTS)	input	5	● ┘	
Data Set Ready (DSR)	input	6	●	
Signal GrouND (SGND)		7	● ──→	Signal GrouND
Data Carrier Detect (DCD)	input	8	●	
Data Terminal Ready (DTR)	output	20	●	

```
57 70 MLIST
SCR # 57
  0 ( FORTH 6502 assembler bv W.F. Raosdale  iulv 1980          )
  1 ( Uodated for 65C02 bv Gert Klein        okt  1984          )
  2
  3 ( This version conforms to the CMOS 6502 CPU manufactured   )
  4 ( bv ROCKWELL. You can however use this assembler for the   )
  5 ( SYNERTEK and GTE versions of the 65C02 as well as for the )
  6 ( NMOS 6502 if vou avoid usino non existino instructions and)
  7 ( adressino modes.                                          )
  8
  9 ( This assembler is an uodated version of the orioinal 6502 )
 10 ( assembler orovided throuoh the courtesv of the            )
 11
 12 (           * Forth interest oroup     *                   )
 13 (           * PO box 1105              *                   )
 14 (           * San Carlos. CA 94070     *                   )
 15 --)


SCR # 58
  0 ( 65C02 ASSEMBLER   PART 1  )
  1 HEX
  2 VOCABULARY ASSEMBLER IMMEDIATE   ASSEMBLER DEFINITIONS
  3
  4 ( REGISTER ASSIGNMENT CONFORMS TO fig-FORTH SOURCE LISTING )
  5
  6 '  (LOOP)    1+   C@         CONSTANT  XSAVE
  7 '  LIT       1+   C@         CONSTANT  IP
  8 '  LIT   22  +    C@         CONSTANT  W
  9 '  COLD  16  +    C@         CONSTANT  UP
 10 '  EXECUTE  NFA  6  -  C@  1+  CONSTANT  N
 11
 12 ( NUCLEUS LOCATIONS ALSO CONFORM TO fig-FORTH SOURCE LISTING )
 13
 14 '  (DO)    0E  +  CONSTANT  POP
 15 --)


SCR # 59
  0 ( 65C02 ASSEMBLER  PART 2  )
  1
  2 '  (DO)    0C  +  CONSTANT  POPTWO
  3 '  LIT     13  +  CONSTANT  PUT
  4 '  LIT     11  +  CONSTANT  PUSH
  5 '  LIT     18  +  CONSTANT  NEXT
  6 '  EXECUTE  NFA  11  -  CONSTANT  SETUP
  7
  8 ( INDEX TABLE FOR GENERATING OPCODES  )
  9
 10 0  VARIABLE  INDEX  -2  ALLOT
 11 0909  ,  1505  ,  0115  ,  8011  ,
 12 8009  ,  1D0D  ,  1219  ,  8080  ,
 13 0080  ,  1404  ,  8014  ,  8080  ,
 14 8080  ,  1C0C  ,  801C  ,  2C3C  ,
 15 --)


SCR # 60
  0 ( 65C02 ASSEMBLER  PART 3  )
  1
  2 2 VARIABLE  MODE     ( DEFAULT  )
  3
  4 : .A    0 MODE ! ;   ( ACCU ADRESSING )
  5 : #     1 MODE ! ;   ( IMMEDIATE )
  6 : MEM   2 MODE ! ;   ( Z-PAGE AND ABSOLUTE )
  7 : ,X    3 MODE ! ;   ( Z-PAGE AND ABSOLUTE X INDEXED )
  8 : ,Y    4 MODE ! ;   ( Z-PAGE AND ABSOLUTE Y INDEXED )
  9 : X)    5 MODE ! ;   ( Z-PAGE X INDEXED INDIRECT )
 10 : )Y    6 MODE ! ;   ( Z-PAGE INDIRECT Y INDEXED )
 11 : Z)    D MODE ! ;   ( Z-PAGE INDIRECT )
 12 : )X    E MODE ! ;   ( ABSOLUTE INDEXED INDIRECT )
 13 : )     F MODE ! ;   ( ABSOLUTE INDIRECT )
 14
 15 --)
```



```
SCR # 61
  0 ( 65C02 ASSEMBLER  PART 4  )
  1
  2 ( PSEUDO MACRO'S FOR STACK ACCES )
  3
  4 : BOT   .X    0  :  ( ADRESS THE BOTTOM OF THE STACK )
  5 : SEC   .X    2  :  ( ADRESS SECOND ITEM ON THE STACK )
  6 : RP)   .X  101  :  ( ADRESS BOTTOM OF RETURN STACK )
  7
  8 ( ERROR MESSAGE FOR ILLEGAL OPERAND OR ADRESSING MODE )
  9
 10 : ASMERROR  MEM  CR  LATEST  ID.  3  ERROR  :
 11
 12 --)
 13
 14
 15


SCR # 62
  0 ( 65C02 ASSEMBLER  PART 5  )
  1
  2 ( UPMODE SETS MODE. CHECKS FOR ILLEGAL ADRESSING MODES )
  3
  4 : UPMODE  IF  MODE  @  8  AND  0=  IF  8  MODE  +! ENDIF  ENDIF
  5           1  MODE  @  OF  AND  -DUP
  6              IF  0  DO  DUP  +  LOOP  ENDIF
  7           OVER  1+  @  AND  0=  :
  8
  9 ( DEFINE SINGLE BYTE INSTRUCTIONS  )
 10
 11 :  CPU  (BUILDS  C.  DOES)  C@  C.  MEM  :
 12
 13   00 CPU BRK.    18 CPU CLC.    D8 CPU CLD.    58 CPU CLI.
 14   B8 CPU CLV.    CA CPU DEX.    88 CPU DEY.    E8 CPU INX.
 15 --)


SCR # 63
  0 ( 65C02 ASSEMBLER  PART 6  )
  1
  2   C8 CPU INY.    EA CPU NOP.    48 CPU PHA.    08 CPU PHP.
  3   DA CPU PHX.    5A CPU PHY.    68 CPU PLA.    28 CPU PLP.
  4   FA CPU PLX.    7A CPU PLY.    40 CPU RTI.    60 CPU RTS.
  5   38 CPU SEC.    F8 CPU SED.    78 CPU SEI.    AA CPU TAX.
  6   A8 CPU TAY.    BA CPU TSX.    8A CPU TXA.    9A CPU TXS.
  7   98 CPU TYA.    3A CPU DEA.    1A CPU INA.
  8
  9 ( DEFINE MULTI MODE INSTRUCTIONS  )
 10
 11 : M/CPU  (BUILDS  C.  .  DOES)  DUP  1+  @  80  AND
 12          IF  10  MODE  +!  ENDIF
 13          OVER  FF00  AND  UPMODE UPMODE  IF ASMERROR  ENDIF
 14          C@  MODE  C@  INDEX  +  C@  +  C.  MODE  C@  7  AND
 15  --)


SCR # 64
  0 ( 65C02 ASSEMBLER  PART 7  )
  1
  2       IF  MODE  C@  OF  AND  DUP  7  (  SWAP  D  =  OR
  3         IF  C.  ELSE  .  ENDIF
  4       ENDIF   MEM  :
  5
  6  3C6E 60 M/CPU ADC.    3C6E 20 M/CPU AND.    3C6E C0 M/CPU CMP.
  7  3C6E 40 M/CPU EOR.    3C6E A0 M/CPU LDA.    3C6E 00 M/CPU ORA.
  8  3C6E E0 M/CPU SBC.    3C6C 80 M/CPU STA.    0D0D 01 M/CPU ASL.
  9  0C0C C1 M/CPU DEC.    0C0C E1 M/CPU INC.    0D0D 41 M/CPU LSR.
 10  0D0D 21 M/CPU ROL.    0D0D 61 M/CPU ROR.    0414 81 M/CPU STX.
 11  0486 E0 M/CPU CPX.    0486 C0 M/CPU CPY.    1496 A2 M/CPU LDX.
 12  0C8E A0 M/CPU LDY.    048C 80 M/CPU STY.    0480 14 M/CPU JSR.
 13  C480 40 M/CPU JMP.    0484 10 M/CPU TRB.    0484 00 M/CPU TSB.
 14
 15 --)
```

```
SCR # 65
  0 ( 65C02 ASSEMBLER  PART 8  )
  1
  2 ( DEFINE STZ. AND BIT.  )
  3
  4 : M2/CPU  (BUILDS  .  .  .  DOES)  MODE  @  DUP  0= SWAP 3 ) OR
  5           IF  ASMERROR  ENDIF  OVER  FF00  AND
  6           IF  MODE  @  1  =  0=
  7           IF  2  MODE  +!  ENDIF  ENDIF
  8           MODE  @  +  C@  DUP  0=  IF  ASMERROR  ENDIF
  9           C.  MODE  @  4  (  IF  C.  ELSE  .  ENDIF  MEM  :
 10
 11  3C2C 3424 8900 M2/CPU BIT.
 12  9E9C 7464 0000 M2/CPU STZ.
 13
 14 --)
 15


SCR # 66
  0 ( 65C02 ASSEMBLER  PART 9  )
  1
  2 ( ROCKWELL 65C02 ONLY ! )
  3
  4 00 CONSTANT M0    10 CONSTANT M1    20 CONSTANT M2
  5 30 CONSTANT M3    40 CONSTANT M4    50 CONSTANT M5
  6 60 CONSTANT M6    70 CONSTANT M7
  7
  8 ( DEFINE RMB. AND SMB.  )
  9
 10 : BITGEN  C@  +  OVER  FF00  AND  MODE  @  2  =  0=  OR
 11           IF  ASMERROR  ENDIF  :
 12
 13 : M3/CPU  (BUILDS  C.  DOES)  BITGEN  C.  C.  :
 14
 15 07 M3/CPU RMB.    87 M3/CPU SMB.     --)


SCR # 67
  0 ( 65C02 ASSEMBLER  PART 10  )
  1
  2 ( DEFINE CONDITIONALS FOR BBS AND BBR INSTRUCTIONS )
  3
  4 : M4/CPU  (BUILDS  C.  DOES)  BITGEN  C.  :
  5
  6  0F M4/CPU SET    8F M4/CPU CLR
  7
  8 ( ASSEMBLER CONDITIONALS. ALL VERSIONS  )
  9
 10 : BEGIN.  HERE  1  :  IMMEDIATE
 11 : UNTIL.  ?EXEC  )R  1  ?PAIRS  R)  C.  HERE  1+  -  C.  :
 12 IMMEDIATE
 13
 14 : IF.  C.  HERE  0  C.  2  :  IMMEDIATE
 15 --)


SCR # 68
  0 ( 65C02 ASSEMBLER  PART 11  )
  1
  2 : ENDIF.  ?EXEC  2  ?PAIRS  HERE  OVER  C@
  3           IF  SWAP  !  ELSE  OVER  1+  -  SWAP  C!  ENDIF :
  4 IMMEDIATE
  5 : ELSE.  2  ?PAIRS  HERE  1+  1  JMP.  SWAP HERE  OVER
  6          1+  -  SWAP  C!  2  :  IMMEDIATE
  7
  8
  9 :  NOT  20  +  :  ( DON'T USE AFTER SET OR CLR )
 10
 11 ( FLAG TESTING PSEUDO MACRO'S  )
 12
 13 90 CONSTANT CS
 14 D0 CONSTANT 0=
 15 --)
```

```
SCR # 69
  0 ( 65C02 ASSEMBLER  PART 12  )
  1
  2 10 CONSTANT 0(   ( TEST FOR LESS THEN ZERO )
  3 90 CONSTANT )=   ( TEST FOR GREATER OR EQUAL TO ZERO )
  4                  ( VALID ONLY AFTER SBC. OR CMP.          )
  5
  6 : END-CODE  CURRENT  @  CONTEXT  !  ?EXEC  ?CSP  SMUDGE  ;
  7 IMMEDIATE
  8
  9 FORTH DEFINITIONS DECIMAL
 10
 11 : CODE  ?EXEC  CREATE  [COMPILE]  ASSEMBLER  ASSEMBLER
 12         MEM  !CSP  :   IMMEDIATE
 13
 14 ' ASSEMBLER  CFA  '  :CODE  8  +   ( ALTER  :CODE  )
 15 --)


SCR # 70
  0 ( 65C02 ASSEMBLER  PART 13  )
  1
  2 ( LOCK ASSEMBLER INTO SYSTEM  )
  3
  4 LATEST   12  +ORIGIN  !   ( TOP NFA )
  5 HERE     28  +ORIGIN  !   ( FENCE   )
  6 HERE     30  +ORIGIN  !   ( DP      )
  7 '  ASSEMBLER  6  +  32  +ORIGIN  !   ( VOC-LINK  )
  8 HERE     FENCE  !
  9
 10 :S
 11
 12
 13
 14
 15

OK
```

乾杯

Tips and Tricks: C-16 / PLUS4
==========

A Cursor for GET

```
10 A=PEEK(200)+PEEK(201)*256+PEEK(202) :REM LOCATION OF CURSOR
20 POKE65292,A/256                     :REM CURSOR ON (HIGH) IN TED
30 POKE65293,A-256*INT(A/256)          :REM CURSOR ON (LOW) IN TED
40 GETA$                               :REM CHARACTER
50 IFA$=""THEN40                       :REM WAIT FOR INPUT
60 PRINTA$;                            :REM ECHO ON SCREEN
70 POKE65292,255                       :REM CURSOR OFF (HIGH) IN TED
80 POKE65293,255                       :REM CURSOR OFF (LOW) IN TED
90 RETURN                              :REM BACK TO MAIN PROGRAM
```

Clearly, line 50 can be changed such that the program continues only if some particular character is typed in, or else GETKEY could be used.

Fred Behringer, München

```
23 35 MLIST
SCR # 23
  0 ( ********** SCREEN EDITOR **********                                  )
  1 ( SCREEN EDITOR FOR FIG-79 FORTH V1.1                                  )
  2 ( RUNNING ON EVERY JUNIOR WITH ELEKTERMINAL AND STANDARD               )
  3 ( 79 FORTH AS DISTRIBUTED BY THE KIM-CLUB.                             )
  4 (                                                                      )
  5 ( THE PROGRAM IS DEVELOPED ON A SENIOR BY:                             )
  6 (   G. VAN OPBROEK                                                     )
  7 (   HOOGLANDEN 20                                                      )
  8 (   9801 LB ZUIDHORN                                                   )
  9 (   TEL. 05940-5627                                                    )
 10 (                                                                      )
 11 CR CR ." GEVOP SCREEN EDITOR."
 12 CR    ." VERSION 1.0          29/08/84" CR CR
 13 VOCABULARY SCR EDITOR DEFINITIONS
 14 HEX
 15 --)


SCR # 24
  0 ( ********** SCREEN EDITOR **********                                  )
  1 (                                                                      )
  2 ( USED CONSTANTS AND VARIABLES:                                        )
  3 ( B/SCR       BLOCKS PAR SCREEN =   8   08 HEX                         )
  4 ( B/BUF       BYTES  PAR BUFFER = 128   80 HEX                         )
  5 ( C/L         CHAR.  PAR LINE   =  64   40 HEX                         )
  6 ( SCR         SCREEN NUMBER                                            )
  7 (                                                                      )
  8 VARIABLE ACT_LINE         ( ACTUAL LINE                                )
  9 VARIABLE CHAR_POS         ( POSISTION OF CURSOR                        )
 10 --)
 11
 12
 13
 14
 15


SCR # 25
  0 ( ********** SCREEN EDITOR **********                                  )
  1 : STOPBITS  1+ 1A59 C! ;          ( SEE LISTING OF PM: BOOK 4          )
  2 : CLS'                  ( CLEAR TERMINAL SCREEN                        )
  3   A0 STOPBITS 0C EMIT 02 STOPBITS ( 160 STOPBITS = 133 MS.             )
  4   0 ACT_LINE ! 0 CHAR_POS ! ;
  5 (                                                                      )
  6 : HOME                  ( CURSOR HOME ON TERMINAL                      )
  7   A0 STOPBITS 1C EMIT 02 STOPBITS ( 160 STOPBITS = 133 MS.             )
  8   0 ACT_LINE ! 0 CHAR_POS ! ;
  9 (                                                                      )
 10 : C_POS                ( MOVE CURSOR TO DESIRED PLACE                 )
 11   ( (HORIZONTAL POS.) (LINE NR.)                                       )
 12   HOME CR ACT_LINE ! DUP CHAR_POS ! 0>
 13   IF CHAR_POS @ 0 DO 09 EMIT LOOP ENDIF
 14   10 ACT_LINE @ - -1 DO 0B EMIT LOOP ;
 15 --)


SCR # 26
  0 ( ********** SCREEN EDITOR **********                                  )
  1 : SCR SAVE              ( SAVE CURRENT SCREEN                          )
  2   SCR @ B/SCR * B/SCR 0
  3     DO DUP I + BLOCK UPDATE DROP LOOP DROP SAVE-BUFFERS ;
  4 : SCR CLEAR             ( CLEAR CURRENT SCREEN TO SPACES               )
  5   SCR @ B/SCR * B/SCR 0
  6     DO DUP I + BLOCK B/BUF BLANKS LOOP DROP ;
  7 (                                                                      )
  8 : ACT_ADDRESS           ( CALCULATE ACTUAL ADDRESS OF THE CURSOR       )
  9   ACT_LINE @ 2 /MOD SCR @ B/SCR * + BLOCK SWAP
 10   C/L * + CHAR_POS @ + ;
 11 (                                                                      )
 12 --)
 13
 14
 15
```

```
SCR # 27
  0 ( ********** SCREEN EDITOR **********                                  )
  1 : SCR_LIST                ( LIST ACTUAL SCREEN                        )
  2   CLS' SCR @ 0 0E C_POS   ( START AT LINE 14 TO OVERCOME              )
  3   SCR @ B/SCR * B/SCR 1-  ( PROBLEMS AT POSITION 63 OF                )
  4   + BLOCK C/L 2 * TYPE    ( OF LINE 15                                )
  5   HOME
  6   SCR @ B/SCR *
  7   B/SCR 1- 0 DO DUP I + BLOCK B/BUF TYPE LOOP DROP
  8   HOME DROP ;
  9 : LINE_END                ( CALCULATE THE ADDRESS OF THE END OF       )
 10                           ( THE ACTUAL LINE                           )
 11   ACT_ADDRESS C/L + CHAR_POS @ - 1- ;
 12 : PRINT_FIRST             ( PRINT FIRST LINE AGAIN                    )
 13   CHAR_POS @ ACT_LINE @   ( SAVE CURSOR POSITION ON THE STACK         )
 14   HOME ACT_ADDRESS C/L TYPE
 15   C_POS ; --)             ( PUT CURSOR ON THE OLD POSITION            )


SCR # 28
  0 ( ********** SCREEN EDITOR **********                                  )
  1 : SI                      ( DELETE ACTUAL LINE AND SHIFT THE          )
  2   ACT_LINE @ DUP 0F ( ( OTHER LINES IN.                               )
  3   IF DUP 1+ 10 SWAP
  4     DO I ACT_LINE ! 0 CHAR_POS ! ACT_ADDRESS
  5        I 1- ACT_LINE ! ACT_ADDRESS C/L CMOVE
  6   LOOP ENDIF 0 CHAR_POS ! 0F ACT_LINE !
  7   ACT_ADDRESS C/L BLANKS SCR_LIST 0 SWAP C_POS ;
  8 : SO                      ( SPREAD AT THE CURRENT LINE                )
  9   ACT_LINE @              ( AND INSERT A BLANK LINE                   )
 10   0F ACT_LINE ! 0 CHAR_POS ! 0 0=       ( LAST LINE EMPTY?            )
 11   LINE_END 1+ ACT_ADDRESS DO I C@ 20 = AND LOOP
 12   IF DUP 0F DO I DUP 1- ACT_LINE ! ACT_ADDRESS SWAP
 13     ACT_LINE ! ACT_ADDRESS C/L CMOVE -1 +LOOP DUP ACT_LINE !
 14     ACT_ADDRESS C/L BLANKS SCR_LIST 0 SWAP C_POS
 15   ELSE 0 SWAP C_POS 07 EMIT ENDIF ; --) ( IF NOT THEN BELL            )


SCR # 29
  0 ( ********** SCREEN EDITOR **********                                  )
  1 : DEL                     ( DELETE ACTUAL CHARACTER                   )
  2   CHAR_POS @ C/L 1- (
  3   IF LINE_END ACT_ADDRESS DO I DUP 1+ C@ DUP EMIT SWAP C! LOOP
  4   ENDIF LINE_END 1 BLANKS SPACE
  5   ACT_LINE @ 0F = IF PRINT_FIRST ENDIF
  6   CHAR_POS @ ACT_LINE @ C_POS ;
  7 : SUB                     ( INSERT CHARACTER AT CURSOR POSITION       )
  8   LINE_END C@ 20 = CHAR_POS @ C/L 1- ( AND
  9   IF ACT_ADDRESS LINE_END DO I DUP 1- C@ SWAP C! -1 +LOOP
 10     ACT_ADDRESS 1 BLANKS
 11     LINE_END 1+ ACT_ADDRESS DO I C@ EMIT LOOP
 12   ELSE 07 EMIT ENDIF
 13   ACT_LINE @ 0F = IF PRINT_FIRST ENDIF
 14   CHAR_POS @ ACT_LINE @ C_POS ;
 15 --)


SCR # 30
  0 ( ********** SCREEN EDITOR **********                                  )
  1 : KEY_INT                 ( KEY INTERPRETER                           )
  2   SCR ! SCR_LIST
  3   BEGIN KEY DUP 1F ) OVER 7F ( AND
  4   IF                      ( PRINTABLE CHARACTER                       )
  5     DUP EMIT DUP ACT_ADDRESS C! CHAR_POS @ 1+ DUP 3F )
  6     IF DROP 07 EMIT 08 EMIT ACT_LINE @ 0F = IF PRINT_FIRST ENDIF
  7     ELSE CHAR_POS ! 3F = ACT_LINE @ 0F = AND
  8     ENDIF
  9   ELSE DUP                ( NON PRINTABLE CHARACTER                   )
 10   DUP 08 = IF DROP CHAR_POS @ 1- DUP 0(     ( CURSOR LEFT             )
 11     IF DROP 07 EMIT ELSE CHAR_POS ! DUP EMIT ENDIF ELSE
 12   DUP 09 = IF DROP CHAR_POS @ 1+ DUP 3F )   ( CURSOR RIGHT            )
 13     IF DROP 07 EMIT ELSE CHAR_POS ! DUP EMIT ENDIF ELSE
 14   DUP 0A = IF DROP ACT_LINE @ 1+ DUP 0F )   ( CURSOR DOWN             )
 15     IF DROP 07 EMIT ELSE ACT_LINE ! DUP EMIT ENDIF ELSE --)
```

```
SCR # 31
  0 ( ********** SCREEN EDITOR **********                                  )
  1     DUP 08 = IF DROP ACT LINE @ 1- DUP 0(      ( CURSOR UP           )
  2        IF DROP 07 EMIT ELSE ACT LINE ! DUP EMIT ENDIF ELSE
  3     DUP 0D = IF DROP ACT ADDRESS                ( CARRIAGE RETURN)
  4        C/L CHAR POS @ - BLANKS CHAR POS @ 0= IF SPACE ENDIF
  5        DUP EMIT 0 CHAR POS !
  6        0A EMIT ACT LINE @ 1+ DUP 0F )
  7        IF DROP PRINT FIRST HOME ELSE ACT LINE ! ENDIF ELSE
  8     DUP 0C = IF DROP CLS' SCR CLEAR ELSE        ( CLEAR SCREEN    )
  9     DUP 1C = IF DROP HOME ELSE                  ( CURSOR HOME     )
 10     DUP 1D = IF DROP DUP EMIT 0 CHAR POS !      ( CARRIAGE RETURN)
 11        0A EMIT ACT LINE @ 1+ DUP 0F )           ( NO ERASURE      )
 12        IF DROP PRINT FIRST HOME ELSE ACT LINE ! ENDIF ELSE
 13 --)
 14
 15


SCR # 32
  0 ( ********** SCREEN EDITOR **********                                  )
  1     DUP 7F = IF DROP DEL ELSE          ( DEL = DELETE CHARACTER )
  2     DUP 1A = IF DROP SUB ELSE          ( SUB = INSERT CHARACTER )
  3     DUP 0F = IF DROP SI  ELSE          ( SI  = DELETE LINE      )
  4     DUP 0E = IF DROP SO  ELSE          ( SO  = INSERT LINE      )
  5 (                                                                 )
  6 (                                                                 )
  7 (                                                                 )
  8     DROP 07 EMIT
  9   ENDIF ENDIF ENDIF ENDIF
 10   ENDIF ENDIF ENDIF ENDIF ENDIF ENDIF ENDIF ENDIF ENDIF
 11   03 = UNTIL CLS'
 12   BEGIN CR ." SAVE SCR: " SCR @ . ." ? (N/Y) " KEY DUP EMIT
 13     DUP 4E = ( "N" ) OVER 59 = ( "Y" ) OR
 14   UNTIL 59 = IF SCR SAVE ELSE EMPTY-BUFFERS ENDIF :
 15 FORTH DEFINITIONS  -->


SCR # 33
  0 ( ********** SCREEN EDITOR **********                                  )
  1 : SCR EDIT ( SCR# -- )      ( EDIT ENTRY-POINT                        )
  2   EMPTY-BUFFERS SCR EDITOR
  3   BEGIN KEY INT
  4     BEGIN CR ." NEXT SCREEN? (N/Y) "
  5       KEY DUP EMIT DUP 4E = OVER 59 = OR  ( "N" OR "Y"               )
  6     UNTIL
  7     DUP 59 = IF SCR @ 1+ SWAP ENDIF       ( INCREASE SCR#           )
  8   4E = UNTIL FORTH :
  9 DECIMAL
 10 (                                                                   )
 11 ." SCREEN EDITOR LOADED" CR CR
 12 ." USER INTERFACE: SCR# SCR EDIT" CR CR
 13 :S
 14
 15


SCR # 34
  0 ( ********** SCREEN EDITOR : INSTRUCTIONS **********               )
  1 (  1: START THE EDITOR BY SCR-NUMBER SCR EDIT.                     )
  2 (  2: CURSOR CONTROL:                                              )
  3 (     BS [CNTRL-H]  CURSOR LEFT                                    )
  4 (     HT [CNTRL-I]  CURSOR RIGHT                                   )
  5 (     LF [CNTRL-J]  CURSOR DOWN                                    )
  6 (     VT [CNTRL-K]  CURSOR UP                                      )
  7 (     FS [CNTRL-\]  CURSOR HOME                                    )
  8 (  3: CHARACTER EDITING:                                           )
  9 (     DEL           DELETE CHARACTER                               )
 10 (     SUB [CNTRL-Z] INSERT BLANK CHARACTER                         )
 11 (  4: LINE EDITING:                                               )
 12 (     SO [CNTRL-N]  INSERT BLANK LINE                              )
 13 (     SI [CNTRL-O]  DELETE LINE                                    )
 14 (     FF [CNTRL-L]  CLEAR SCREEN TO BLANKS                         )
 15 (                                                                  )
```

```
SCR # 35                                                                    )
  0 ( ********** SCREEN EDITOR: INSTRUCTIONS **********                    )
  1 (  5: CARRIAGE RETURN:                                                  )
  2 (     CR:              CARRIAGE RETURN AND ERASE                        )
  3 (                      UNTIL END-OF-LINE                                )
  4 (     GS:  [CNTRL-]] CARRIAGE RETURN WITHOUT ERASURE                    )
  5 (  6: EXIT EDIT MODE:                                                   )
  6 (     ETX: [CNTRL-C]                                                    )
  7 (                                                                       )
  8 ( THERE EXISTS ALSO A VERSION FOR THE SENIOR COMPUTER WITH              )
  9 ( PDS-65 FORTH.                                                         )
 10 ( CONTACT THE AUTHOR OF THIS PROGRAM.                                   )
 11 (                                                                       )
 12 ( THIS PROGRAM CAN ONLY BE LOADED IF VOCABULARY AND +LOOP ARE           )
 13 ( WORKING PROPERLY. SEE THE PATCHES THAT ARE PUBLISHED BY THE           )
 14 ( AUTHOR OF THIS PROGRAM.
 15 :S

 OK
```

AVAILABLE FOR YOUR OCTOPUS/EC65 DISKETTE :

DISKETTE 11

To show you what's on the diskette, we give here the output on screen after booting diskette 11 and the output on screen of the directory, followed by a list of new files, compared with the System Loys diskette 1.

```
-- UTIL 5 -- DRIVE A                          -- July 9, 1985 --

1) Directory                                  :  10) Load assembler
2) Create a new file                          :  11) Load wordprocessor
3) Change a new file                          :  12) Basicode processor (BSCOD1)
4) Delete file from diskette                  :
5) Create blank data diskette                 :
6) Create data diskette with files            :
7) Create buffer space for data files         :
8) Single or dual disk drive copier           :
9) Enter OS-65D system                        :

Type the number of your selection and depress RETURN ?
```

```
-- Directory of drive A --
-------------------------------------------------------------------------------
V3.3/1      0-0     V3.3/2      1-1     DIRECT      12-12
BAS/1       2-2     BAS/2       3-3     BAS/3        4-4
BAS/4       5-5     B/5V/3      6-6     ASM/1        7-7
ASM/2       8-8     ASM/3       9-9     TOV3.1      10-10
T1V3.1     11-11    V3.3/4     13-13    BEXEC*      14-17
COPIER     18-19    CHANGE     20-21    GARBAG      22-24
DIR        25-25    SCRTCH     26-26    MERGE       27-27
WP2.0      28-31    BSCOBJ     32-32    NEWCOP      33-34
ATNENB     35-35    BSCOD1     37-37    BSCOD4      38-38
COP/TO     36-36    COM/TO     39-39
-------------------------------------------------------------------------------

35 entries free out of 64

Depress RETURN to continue ?

---
NEW FILES :          SCRTCH    26-26    BSCOD1    37-37
                     NEWCOP    33-34    BSCOD4    38-38
                     BSCOBJ    32-32    WP2.0     28-31
```

If you ordered the OS-65D patch-diskettes earlier:
Send empty diskette with label and R/W protect sticker to the editor's office.
The diskette format is 40 tracks.
Send cheque of Hfl. 22,00 to W.L.v.Pelt (eurocheque 12,50)
Price only for European (C.E.P.T.) countries.

BASICODE-2

1 WAT IS BASICODE-2

BASICODE is een gestandariseerde audiocode waarmee BASIC-programma' s op cassette worden bewaard. Deze code is ontwikkeld door HOBBYSCOOP van de NOS. Het uitwisselen van BASIC-programma' s tussen verschillende computers wordt hiermee mogelijk, aangezien iedereen de zelfde geluidscode gebruikt.

2 SPECIFICATIES

Het BASIC-programma wordt in ASCII-formaat op de band gezet. Het most significant bit (bit 8) = 1. De baudrate bedraagt 1200. Een byte wordt voorafgegaan door 1 startbit (0) en gevolgd door 2 stopbits (1).
Een logische 1 wordt omgezet in twee perioden van 2400 Hz. Een logische 0 wordt omgezet in een periode van 1200 Hz.
Het totale programma wordt als volgt weggeschreven:

- 5 seconden 2400 Hz,
- STX ($ 82),
- BASIC-info,
- ETX ($ 83),
- checksum,
- 5 seconden 2400 Hz.

De checksum is het resultaat van de exclusive-or van alle voorgaande bytes. Het 8-ste bit kan wel 0 zijn.

2.1 PROTOCOL

Omdat de gebruikte BASIC' s niet gelijk zijn schrijft het BASICODE-2 protocol voor hoe het programma opgebouwd moet zijn en welke statements gebruikt mogen worden.

2.1.1 ALGEMENE AFSPRAKEN

De afspraken waaraan BASICODE-programma's moeten voldoen zijn:

- Alleen die BASCIC-statements gebruiken die alle computers kennen (zie 2.1.2).
- De regelsnummers tot 1000 zijn gereserveerd voor speciale functies die niet voor elke computer gelijk hoeft te zijn (zie 2.1.4).
- Er wordt uitgegaan van een beeldscherm van 24 regels met elk 40 tekens.
- Een programmaregel mag inclusief het regelnummer en de spaties maximaal 60 tekens lang zijn.

2.1.2 BASIC-STATEMENTS

De volgende commando' s en operatoren mogen gebruikt worden:

ABS	AND	ASC	ATN	CHR$	COS	DATA	DIM
END	EXP	FOR	GOSUB	GOTO	IF	INPUT	INT
LEFT$	LEN	LET	LOG	MID$	NEXT	NOT	ON
OR	PRINT	READ	REM	RESTORE	RETURN	RIGHT$	RUN
SGN	SIN	SQR	STEP	STOP	TAB	TAN	THEN
TO	VAL						

+	-	*	/	^	=	<	>
<>	<=	>=					

2.1.3 KORTE BESCHRIJVING VAN DE STATEMENTS

ABS(X) Geeft de absolute waarde van een variabele X.

AND Dit is de logische AND, die alleen gebruikt mag worden voor logische variabelen. Gebruik haakjes om de bewerkingsvolgorde aan te geven.
```
IF (A=3) AND (B=1) THEN ....
```

ASC(X$) Geeft de ASCII-waarde van het eerste karakter van X$.
```
A$="HALLO":B=ASC(A$) ==> B=72
```

ATN(X) Geeft de arctangens in radialen van de variabele X.

CHR$(X) Geeft het karakter waarvan de ASCII-waarde gelijk is aan de variabele X (32 <= X <= 127). De waarden kleiner dan 32 zijn niet voor alle computers gelijk. Alleen de RETURN-toets heeft altijd de ASCII-waarde 13.

COS(X) Geeft de cosinus van de hoek (in radialen) X.

DATA Hierna volgen de variabelen die met het statement READ gelezen kunnen worden. De variabelen worden gescheiden door een komma. Stringvariabelen moeten tussen aanhalingstekens staan.
```
DATA 100,200,300,"HALLO","DAAG","GROETJES",1,2,3
```

DIM Hiermee worden array' s gedimensioneerd. Een array moet voor gebruik gedimensioneerd worden. In BASICODE moeten allen array' s gedimensioneerd worden. Het aantal dimensie' s bedraagt maximaal twee.
```
DIM A$(2),B$(15),TL$(20,50)
```

END Hiermee wordt het programma gestopt.

FOR TO STEP NEXT Herhalingsconstructie. De lus wordt minimaal 1 maal doorlopen. STEP mag weggelaten worden als de stapgrootte 1 is. Na NEXT moet de variabele opgegeven worden.
```
FOR A=10 TO 100 STEP 5
..... \
..... ! Programma dat hehaald doorlopen moet worden.
..... /
NEXT A
```

GOSUB Hiermee wordt de subroutine aangeroepen waarvan het regelnummer achter GOSUB staat.

GOTO Hiermee wordt gesprongen naar het regelnummer volgend op dit statement.

IF THEN Voorwaardelijke sprong. Tussen IF en THEN staat een logische variabele of een logische vergelijking. Is deze 'waar' dan wordt het gedeelte achter THEN uitgevoerd; is deze 'niet waar' dan wordt het volgende statement uitgevoerd.
```
IF A=3 THEN B=5:C$="WAAR":GOSUB 2000:GOTO 2500
```

INPUT Hiermee wordt aan de gebruiker gevraagd om invoer, welke wordt toegekend aan de variabele of string-variabele volgend op INPUT. Een ingevoerde string mag geen komma' s of dubbele punten bevatten. Er mag maar een variabele staan achter INPUT.
```
PRINT"WAT IS UW NAAM";:INPUT N$
PRINT"GEEF DE X- EN Y-WAARDE";:INPUT X:INPUT Y
INPUT"HOE VAAK";A is VERBODEN !!
```

INT(X) Geeft het grootste gehele getal kleiner dan of gelijk aan X.

LEFT$ LEFT$(X$,X) haalt X karakters uit X$ te beginnen met het meest linker karakter.
A$="RODE FIETSEN":B$=LEFT$(A$,4) ==> B$="RODE"

LEN(X$) Geeft de lengte van de string X.

LET Hiermee kan een waarde aan een variabele worden toegekend.
LET A=3. Ook mag A=3.

LOG(X) Berekent de natuurlijke logaritme van X.

MID$ MID$(X$,X,Y) haalt Y karakters uit X$ te beginnen met het X-ste karakter.
A$="HOBBYSCOOP BASICODE":B$=MID$(A$,12,5) ==> B$="BAS

NEXT Zie FOR

NOT Logische ontkenning. Dit is allen toepasbaar op logische variabelen.
A=5:B=NOT(A=6) ==> B="waar"

ON GOSUB GOTO Hiermee kan een sprong naar een subroutine of een programma-regel gemaakt worden. Na ON volgt een uitdrukking of variabele.
ON A-2 GOTO 5000,6000,7000
A moet 3, 4 of 5 zijn !!. Als A=4 dan wordt gesprongen naar 6000

OR Logische OR. Zie ook AND

PRINT Hiermee kan een variabele of string op het scherm worden afgedrukt. Meerdere variabelen in een PRINT-statement moeten gescheiden worden met een puntkomma. Als aan het eind van de opdracht een puntkomma staat wordt geen automatische regelovergang gegeven.

READ Leest de gegevens van een DATA-statement. Meerdere variabelen worden gescheiden met een komma. De READ-variabelen moeten van het zelfde type zijn als de gegevens in het DATA-statement.
DATA 1,"HALLO",2
READ A,B$,C

REM Hiermee kan het programma voorzien worden van kommentaar. Er mag geen dubbele punt in de kommentaar staan.

RESTORE Hierna wordt met een READ-statement weer vanaf het eerste DATA-statement gelezen.

RETURN Geeft het einde van een subroutine aan.

RIGHT$ RIGHT$(X$,X) geeft X karakters uit X$ eindigend bij het meest rechter karakter.
A$="RODE FIETSEN":B$=RIGHT$(A$,7) ==> B$="FIETSEN"

RUN Start het programma. Alle variabelen worden gewist. Er mag geen regelnummer achter RUN staan.

SIN(X) Berekent de sinus van de hoek (in radialen) X.

SGN(X) Geeft -1 als X negatief is en 0 als X positief is.

SQR(X) Berekent de wortel van X.

STEP Zie FOR.

TAB(X) Hiermee kan in een PRINT-statement de cursor naar rechts verschoven worden. X geeft de nieuwe positie aan.
PRINT"A";TAB(5);"B";TAB(15);"C" ==> A....B.....C (.=spatie)

TAN(X) Berekent de tangens van de hoek (in radialen) X.

THEN Zie IF.

TO Zie FOR.

VAL(X$) Bepaalt de numerieke waarde van X$. Als de string niet numeriek is, kan de uitkomst per computer verschillend zijn.
A$="1.4E6":A=VAL(A$) ==> A=1.4E6
B$="12D":B=VAL(B$) ==> B=0 of B=12

2.1.4 STANDAARD ROUTINES

GOSUB 100 Deze subroutine wist het scherm en plaatst de cursor linksboven op het scherm (positie 0,0).

GOSUB 110 Hiermee wordt de cursor op een bepaalde plaats op het scherm gezet. Hiervoor worden de variabelen HO en VE gebruikt. In HO moet de positie op een regel (0=uiterst links) staan en in VE moet het regelnummer (0=bovenste) staan.

GOSUB 120 Hiermee wordt de positie van de cursor op het scherm bepaalt en in de variabelen HO en VE gezet.

GOSUB 200 Hiermee wordt gekeken of er een toets ingedrukt is. De waarde komt in IN$ te staan. Is er geen toets ingedrukt, dan is IN$ een lege string.

GOSUB 210 Deze subroutine wacht tot er een toets wordt ingedrukt en zet dan de waarde ervan in IN$.

GOSUB 250 Deze subroutine zorgt er voor dat de computer een piep afgeeft.

GOSUB 260 Na aanroep van deze subroutine bevat RV een willekeurig getal tussen 0 en 1.

GOSUB 270 Hiermee wordt de variabelenruimte opgeruimt en wordt bepaalt hoeveel geheugenruimte er nog vrij is. De variabelen worden niet gewist. Het aantal vrije bytes komt te staan in FR.

GOSUB 300 Hiermee wordt de waarde van SR omgezet in een stringwaarde SR$.

GOSUB 310 Deze routine levert SR$ die bepaalt wordt door CT,CN en SR. SR$ is in waarde gelijk aan SR en altijd in de fixed-point notatie. De totale lengte van SR$ bedraagt CT karakters, waarvan CN karakters na de decimale punt. Als het getal niet in het opgegeven formaat past, bestaat SR$ uit CT sterren.
CT=7:CN=3:SR=2/3 :GOSUB 310 ==> SR$="0.667"
CT=3:CN=0:SR=23.6:GOSUB 310 ==> SR$=" 24"
CT=3:CN=1:SR=100 :GOSUB 310 ==> SR$="***"

GOSUB 350 Hiermee wordt SR$ op de printer afgedrukt, de regel wordt niet afgesloten.

GOSUB 360 Hiermee wordt de regel op de printer afgesloten en wordt er op een nieuwe regel begonnen.

2.1.5 VARIABELEN

De variabelen die in een programma gebruikt worden zijn aan enkele beperkingen gebonden. Deze zijn:

- Numerieke variabelen zijn real en single precision.
  De nauwkeurigheid zal niet groter zijn dan zes decimalen.

- De namen van variabelen mogen niet langer zijn dan twee karakters.
  In deze namen mogen alleen hoofdletters gebruikt worden.
  Voor stringvariabelen wordt de naam gevolg door $, alle andere toevoegingen zijn verboden !!.

- Er mag geen gebruik worden gemaakt van de numerieke waarde van logische variabelen. Het resultaat kan dus alleen in een IF THEN constructie gebruikt worden.

- Voordat een variabele gebruikt wordt moet hij een waarde krijgen. Na RUN hebben de variabelen dus NIET automatisch de waarde nul.

- Stringvariabelen mogen niet langer zijn dan 255 karakters.

- Namen van variabelen mogen niet beginnen met een O. Deze zijn gereserveerd voor de standaardroutines.

- Tevens zijn uitgesloten de namen: AS, AT, FN, GR, IF, PI, ST, TI, TI$ en TO.

- Voor communicatie met de standaardroutines worden gebruikt: HO, VE, FR, SN, CN, CT, RV, IN$ en SR$.

2.1.6 DOCUMENTATIE

Basicode-2 Hans G. Janssen
Hilversum: Nederlandse Omroep Stichting
ISBN 90-6833-001-2
SISO 365.3 UDC 681.3.06

2.2 HOBBYSCOOP

HOBBYSCOOP zendt op de radio programma' s uit in BASICODE.
De uitzendtijden zijn:

- Woensdag RADIO 1+2 19.02 - 19.30
- Donderdag RADIO 5 17.30 - 17.36

Naast BASICODE-programma's zendt HOBBYSCOOP ook de BASICODE BEELDKRANT uit met veel informatie over computers en de landelijke computer agenda.

8-May-86 10:17 SEARCH.TXT Page 1

```
0001:
0002:
0003: 9000                    ORG   $9000
0004:
0005:                     **********
0006:                     SEARCH.OBJ  Written by Jean De Noyette
0007:                     **********             Ch.de Kerchovelaan,87
0008:                                            9000 - Gent
0009:                                            Belgium
0010:
0011:               THIS SHORT PROGRAM, ESPECIALLY WRITTEN FOR DOS 65, IS
0012:               INTENDED TO BE CALLED EVERYTIME YOU BOOT-UP. IT WORKS
0013:               IN COMBINATION WITH ANOTHER FILE (ANNI), WHICH IS
0014:               USED AS AN AGENDA WHEREIN BIRTHDAYS AND APPOINTMENTS
0015:               ARE STORED WITH EDITOR'S HELP.
0016:               SO IT'S VERY USEFUL FOR THOSE WHO ARE USED TO FORGET
0017:               THEIR WIFE'S BIRTHDAY !.....
0018:
0019:
0020:
0021:                     <<< DOS MONITOR ROUTINES >>>
0022:
0023:
0024:          4E E0  CLS    *     $E04E    CLEAR SCREEN
0025:          B0 F3  PNTXIN *     $F3B0    INCREASE POINTER
0026:          00 E0  PRINT  *     $E000    PRINT A CHARACTER IN ACCU
0027:          9C F3  CRLF   *     $F39C    CR/LF
0028:          43 F4  BDPNTX *     $F443    MAKE RAMPROG.
0029:          06 F4  COPYPA *     $F406    COPY PARAMETERS IN RAMPROG.
0030:          F5 F3  CHECKE *     $F3F5    CHECK FOR END
0031:          0F E0  PRSTR  *     $E00F    PRINT STRING
0032:          A9 F0  HEXDE  *     $F0A9    CONVERT TO DECIMAL IN X AND Y
0033:
0034:                     <<< VARIABLES >>>
0035:
0036:          6B CE  WHBUF  *     $CE6B    "LOOK FOR "BUFFER
0037:          87 CE  DNTCR  *     $CE87    WILD CHARACTER
0038:          6A CE  NMB    *     $CE6A    NUMBER OF CHARACTERS TO LOOK FOR
0039:          35 CE  PNTX   *     $CE35    PROGRAM IN RAM, SELF MODIFYING
0040:          1B CE  PARAL  *     $CE1B    PARAMETER A
0041:          1C CE  PARAH  *     $CE1C
0042:          1D CE  PARBL  *     $CE1D    PARAMETER B
0043:          1E CE  PARBH  *     $CE1E
0044:          C2 CE  DAY    *     $CEC2    DAY BUFFER
0045:          C3 CE  MONTH  *     $CEC3    MONTH BUFFER
0046:
0047: 9000 20 4E E0          JSR   CLS
0048: 9003 20 0F E0          JSR   PRSTR    PLACE CURSOR IN CENTER OF SCREEN
0049: 9006 14                =     $14
0050: 9007 01                =     $01
0051: 9008 08                =     $08
0052: 9009 00                =     $00
0053: 900A AD C2 CE          LDA   DAY      GET VALUE OF DAY BUFFER(HEX)
0054: 900D 20 A9 F0          JSR   HEXDE    CONVERT IT TO DECIMAL IN X AND Y
```

```
0055: 9010 18              CLC
0056: 9011 8A              TXA             CONVERT X IN ASCII
0057: 9012 69 30           ADCIM $30
0058: 9014 8D 6B CE        STA   WHBUF     AND STORE IT IN BUFFER
0059: 9017 98              TYA
0060: 9018 69 30           ADCIM $30       CONVERT Y IN ASCII
0061: 901A 8D 6C CE        STA   WHBUF  +01   AND STORE IT IN BUFFER
0062: 901D A9 2D           LDAIM $2D
0063: 901F 8D 6D CE        STA   WHBUF  +02   STORE "-" IN BUFFER
0064: 9022 AD C3 CE        LDA   MONTH     GET VALUE OF MONTH BUFFER (HEX)
0065: 9025 20 A9 F0        JSR   HEXDE
0066: 9028 18              CLC
0067: 9029 8A              TXA
0068: 902A 69 30           ADCIM $30       STORE ASCII VALUE OF MONTH
0069: 902C 8D 6E CE        STA   WHBUF  +03   IN "LOOK FOR" BUFFER
0070: 902F 98              TYA
0071: 9030 69 30           ADCIM $30
0072: 9032 8D 6F CE        STA   WHBUF  +04
0073: 9035 A9 00           LDAIM $00       SET START ADDRESS OF ASCII FILE
0074: 9037 8D 1B CE        STA   PARAL     IN PARAMETER A ($4000)
0075: 903A A9 40           LDAIM $40
0076: 903C 8D 1C CE        STA   PARAH
0077: 903F A9 00           LDAIM $00       SET END ADDRESS OF ASCII FILE IN
0078: 9041 8D 1D CE        STA   PARBL     PARAMETER B ($8000)
0079: 9044 A9 80           LDAIM $80
0080: 9046 8D 1E CE        STA   PARBH
0081: 9049 A9 05           LDAIM $05       SET NUMBER OF CHARACTERS TO FIND
0082: 904B 8D 6A CE        STA   NMB       ( DD-MM ) = 5
0083: 904E 20 9C F3        JSR   CRLF
0084: 9051 20 43 F4        JSR   BDPNTX    MAKE RAMPROG.
0085: 9054 20 06 F4        JSR   COPYPA    COPY PARAMETERS IN RAMPROG.
0086: 9057 A2 00           LDXIM $00
0087: 9059 20 F5 F3  CMWF  JSR   CHECKE    READY?
0088: 905C B0 0A           BCS   CMWFAA
0089: 905E 20 9C F3        JSR   CRLF
0090: 9061 20 0F E0        JSR   $E00F
0091: 9064 1B              =     $1B
0092: 9065 6E              =     $6E       SET NORMAL VIDEO
0093: 9066 00              =     $00
0094: 9067 60              RTS             AND RETURN TO CALLER
0095: 9068 20 35 CE  CMWFAA JSR   PNTX      GET CHARACTERS FROM SELECTED AREA
0096: 906B DD 6B CE        CMPAX WHBUF     COMPARE THEM WITH BUFFER
0097: 906E F0 0F           BEQ   CMWG
0098: 9070 AD 87 CE        LDA   DNTCR     IS IT A DON'T CARE CHARACTER ?
0099: 9073 DD 6B CE        CMPAX WHBUF     COMPARE WITH BUFFER
0100: 9076 F0 07           BEQ   CMWG
0101: 9078 20 B0 F3  CMWFA JSR   PNTXIN    INCREASE POINTER
0102: 907B A2 00           LDXIM $00
0103: 907D F0 DA           BEQ   CMWF      BRANCH ALWAYS BACK IN LOOP
0104: 907F E8        CMWG  INX             CHARACTER THE SAME
0105: 9080 EC 6A CE        CPX   NMB       DO ALL CHARACTERS MATCH ?
0106: 9083 D0 D4           BNE   CMWF      BRANCH IF NOT
0107: 9085 20 9C F3        JSR   CRLF      STRING FOUND !
0108: 9088 20 0F E0        JSR   $E00F     SET INVERS VIDEO
```

```
0109: 908B 1B              =     $1B
0110: 908C 69              =     $69
0111: 908D 07              =     $07       RING THE BELL
0112: 908E 00              =     $00
0113: 908F A2 00           LDXIM $00
0114: 9091 20 35 CE  START JSR   PNTX       GET CHARACTERS FROM  SELECTED AREA
0115: 9094 C9 40           CMPIM $40        IS IT "@" ?
0116: 9096 F0 E0           BEQ   CMWFA      IF YES, SEARCH ANOTHER STRING
0117: 9098 C9 0D           CMPIM $0D        IS IT A "CR" ?
0118: 909A F0 07           BEQ   PRI        IF YES, PRINT ANOTHER LINE
0119: 909C 20 00 E0        JSR   PRINT      PRINT IT OUT
0120: 909F E8              INX
0121: 90A0 4C 91 90        JMP   START
0122: 90A3 20 9C F3  PRI   JSR   CRLF       PRINT ON ANOTHER LINE
0123: 90A6 E8              INX
0124: 90A7 4C 91 90        JMP   START
0125:
0126:              - AFTER ASSEMBLING  PUT THIS PROGRAM IN A FILE
0127:                (SEARCH.OBJ) AND SET IT ON DRIVE 0 IN THE COMMAND
0128:                MODE.
0129:              - CREATE AN ASCII FILE (ANNI) ON DRIVE 1 SUCH AS:
0130:              18-08-47 BIRTHDAY NADINE.....@
0131:              23-06-43 BIRTHDAY JEAN.......
0132:                       DE KERCHOVELAAN, 87...
0133:                       9000 GENT...........@
0134:              - DON'T FORGET THE "@" AFTER EVERY TEXT.
0135:              - PUT "LO ANNI 4000" FOLLOWED BY "SEARCH.OBJ"
0136:                IN LOGIN.COM.
0137:
0138:            NOW WHEN YOU'LL START UP YOUR SYSTEM, THE PROGRAM
0139:            SEARCH.OBJ WILL COMPARE EVERY DATE (DAY-MONTH) OF THE
0140:            FILE "ANNI" WITH THE ACTUAL DATE, (OF COURSE YOU NEED
0141:            A REAL TIME CLOCK !) AND IF THE COMPARISON IS TRUE,
0142:            WILL PRINT OUT THE INFORMATION ABOUT THAT DAY (AND
0143:            RING THE BELL)...
0144:
```