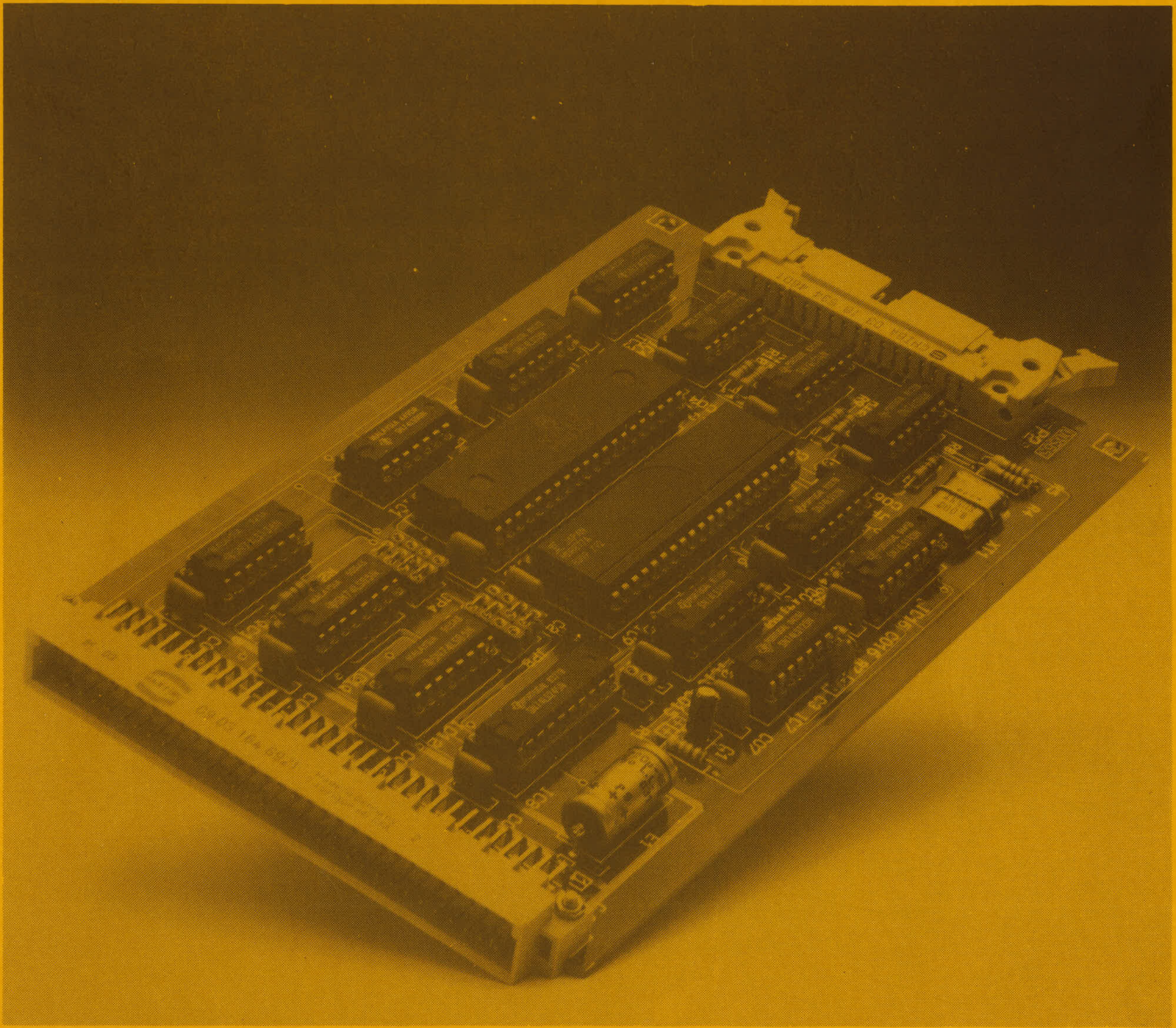
DE 6502 KENNER 51



Elfde Jaargang, Nr. 4
Augustus 1987

DE 6502 KENNER

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

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

Verschijningsdata
DE 6502 KENNER 1985
=================
derde zaterdag van
februari, april, juni,
augustus, oktober, december.

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

Redaktie-adres en informatie over paperware etc.
=========================
Willem L. van Pelt
Jacob Jordaensstraat 15
2923 CK Krimpen/IJssel.
Tel.: 01807 - 19881

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

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

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

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

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

Leden:

Adri Hankel	(05490 - 51151)	Hardware/software/DOS65
Erwin Visschedijk	(05496 - 76764)	Hardware/software/DOS65
Gert van Opbroek	(01729 - 8636)	65802/65816/68000
Nico de Vries	(010 - 4517154)	Hard-/software/68000

Erevoorzitter: Siep de Vries
Ereleden : Mw. H. de Vries - Van der Winden
Anton Mueller
Lidmaatschap : Hfl. 50,=
Subscription : Europe : Hfl. 59,50
(eurocheque = Hfl. 50,=)
Outside Europa:
Hfl. 114,50 (incl.transfers)
Advertenties : Tarieven op aanvraag bij de redaktie.

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

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

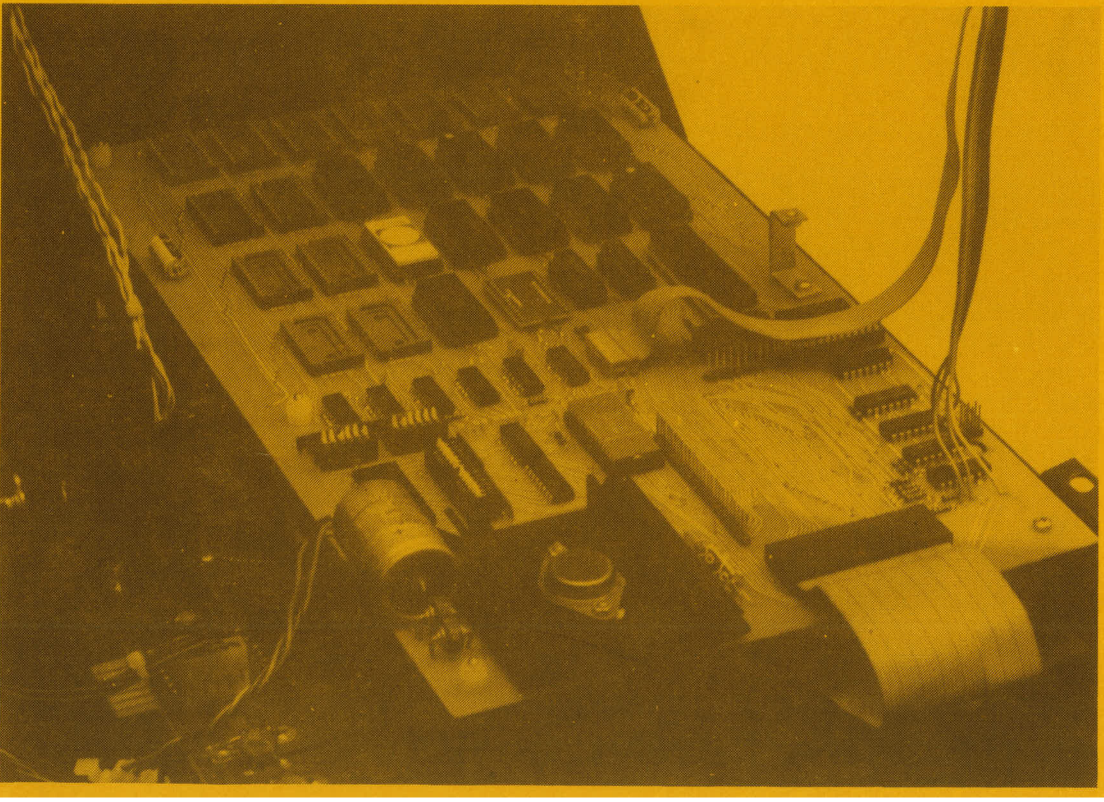
DE 6502 KENNER IS EEN BLAD VAN EN DOOR DE LEDEN

Micro-ADE Assembler/Disassembler/Editor is een produkt van Micro Ware Ltd., geschreven door Peter Jennings en bestemd voor alle 6502-computers. De KIM Gebruikers Club Ned. heeft de copyrights verworven nadat ons lid Sebo Woldringh de 4 K KIM-1 versie uitbreidde tot 8 K KIM-1 versie. Adri Hankel paste deze aan voor de JUNIOR. Willem L. van Pelt stelde een nieuwe 8 K source-listing voor de JUNIOR samen.
De implementatie op andere systemen dan de KIM-1 en JUNIOR kan eenvoudig gebeuren door het aanpassen van de I/O-adressen, welke in de source-listing gemakkelijk te vinden zijn

FATE Format-lister/cond. Assembler/Tape-utilities/Editor is de door ons lid Rob Banen geschreven source-listing van een 12 K universeel systeem voor de JUNIOR-computer aan de hand van het universele disk operating system van de fa. Proton Electronics te Naarden.
FATE wordt beschikbaar gesteld met toestemming van Proton.

DOS65 V2.01 is the new system of our club, build with Elektor's CPU, VDU, RAM-cards and our own professional Floppy-Disk-Controllercard for SS, DS, 40 or 80 tracks and a max. of 720 Kbytes capacity. For use with 6502 or 65C02.
For more information, write to E.J.M. Visschedijk
Dillenlaan 11
NL-7641 CX WIERDEN
The new DOS65 V2.01 is hardware compatible with Elektor's OCTOPUS/EC65 computer, except the controllercard.

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



DE 6502 KENNER

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

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

Illustraties/cartoons:
Antoine Megens
Jitske Voskamp
Frank Vergoossen
en anderen.

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

overnemen

~~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.

INHOUDSOPGAVE DE 6502 KENNER NR. 51 AUGUSTUS 1987.

DE 6502 KENNER

```
*********************************************
**                                         **
** LANDELIJKE BIJEENKOMST DE 6502 KENNERS  **
**                                         **
*********************************************
```

Datum : zaterdag 19 september 1987
Lokatie : Wijkcentrum De Ringvaart
Floris van Adrichemlaan 98
2035 VD Haarlem
Tel.: 023 - 36 38 56

Routebeschrijving:

Wijkcentrum De Ringvaart is te bereiken:

- met het openbaar vervoer, vanaf het station Haarlem: N.Z.H. buslijn 7 - 71 - 72 - 77 - halte Floris van Adrichemlaan.

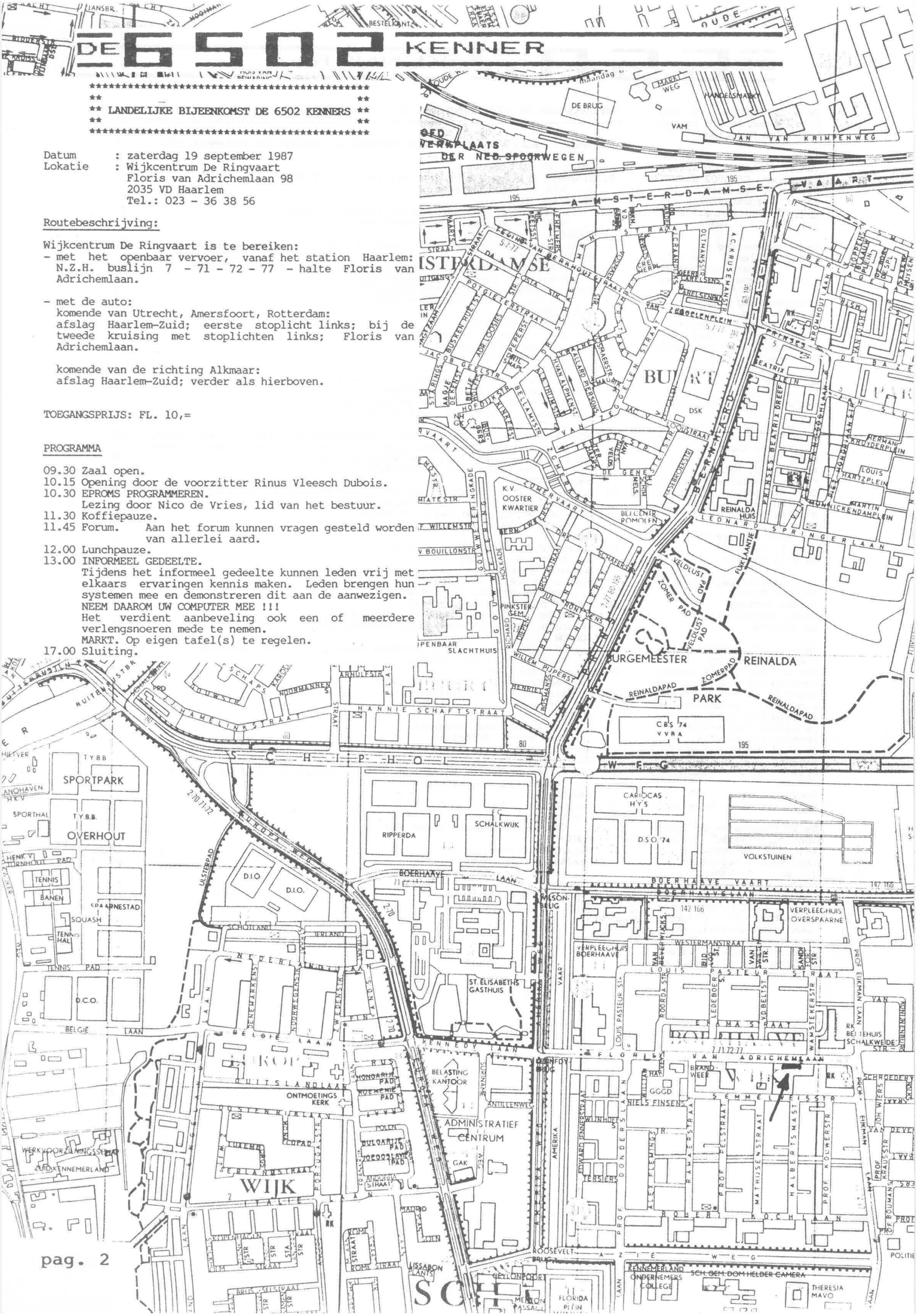
- met de auto:
  komende van Utrecht, Amersfoort, Rotterdam:
  afslag Haarlem-Zuid; eerste stoplicht links; bij de tweede kruising met stoplichten links; Floris van Adrichemlaan.

  komende van de richting Alkmaar:
  afslag Haarlem-Zuid; verder als hierboven.

TOEGANGSPRIJS: FL. 10,=

PROGRAMMA

09.30 Zaal open.
10.15 Opening door de voorzitter Rinus Vleesch Dubois.
10.30 EPROMS PROGRAMMEREN.
Lezing door Nico de Vries, lid van het bestuur.
11.30 Koffiepauze.
11.45 Forum. Aan het forum kunnen vragen gesteld worden van allerlei aard.
12.00 Lunchpauze.
13.00 INFORMEEL GEDEELTE.
Tijdens het informeel gedeelte kunnen leden vrij met elkaars ervaringen kennis maken. Leden brengen hun systemen mee en demonstreren dit aan de aanwezigen.
NEEM DAAROM UW COMPUTER MEE !!!
Het verdient aanbeveling ook een of meerdere verlengsnoeren mede te nemen.
MARKT. Op eigen tafel(s) te regelen.
17.00 Sluiting.



SIDEWAYS PRINTING ROUTINE for ATARI 520 ST

This programme has been written in FAST BASIC (Computer Concepts) and uses 1st WORD PLUS (GST) textfiles, it also requires a printer with quadruple density bit image graphics and the ability to line feed by $^{1}/_{216}$" ie. EPSON or compatible (I used an EPSON FX80)

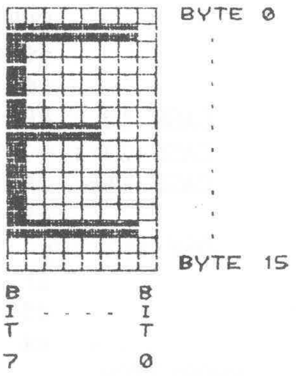
The maximum number of lines per page is 48 (line density is fixed at 8 lines per inch) and the line width is limited to 160 by 1st word plus. Although a page can have 48 lines it looks better if this is limited to 46 or less to allow a margin above and below the text. Blank lines at the end of a page have the effect of pushing text up, in the example printouts at the end of this article I have used 46 lines with one blank line to obtain an even margin above and below the text.

The routine can print in both Elite and Pica size text, however the pica density is not exactly the 10 CPI norm. and the gap between characters is less (This is largely due to resolution limits of the printer, which is also the reason why italics have not been included).

The character font used can either be the one resident in the programme (defined by data statements) or a font from DEGAS (BATTERIES INCLUDED). DEGAS is a drawing package which includes a font editor with several existing fonts. The 1st example at the end is printed with the resident font the second with computer font from disk.

Each character is defined in an 8 by 16 pixel grid (see diagram). Only characters 0 to 127 are defined, however most below 32 cannot be used as they are control characters for the word processor. I have used chr. 26 for the £ sign (1st word uses chr. 156)

The font file begins with character 0 and ends with character 127 with 16 bytes used for each character definition, the bytes are stored in the order indicated on the diagram.



```
DIM line$(48), style$(48), font¦(128,16), b¦(48,15), f3¦(128,16,3)
infile% = FNfileselect( "B:\*.DOC", "*.DOC" )
ruler% = 0
PROCloadfont
OUT 0, 27, 40        :  REM reset printer
OUT 0, 27, 48        :  REM set 1/8" line spacing
OUT 0, 27, 67, 47    :  REM set form length to 47 lines (5 7/8" or half A4)
REPEAT
  PROCloadpage
  IF (ruler% <> 0) AND (fline% <> 0) THEN PROCprint
UNTIL EOF# infile%
CLOSE# infile%
END

DEF FNfileselect( P$, F$ )          :  REM P$ is default path name
  LOCAL ok%, infile%                :  REM F$ default filename and extension
  FSELECT P$, F$, ok%               :  REM file selector
  IF NOT ok% THEN END               :  REM end if 'cancel' was selected
  WHILE RIGHT$( P$, 1 ) <> "\"
    P$ = LEFT$( P$, LEN( P$ ) - 1 )
  WEND
  infile% = OPENIN ( P$+F$ )
  IF infile% < 0 THEN dummy% = ALERT("[1][ No such file ][ OK ]",1) : END
= infile%

DEF PROCsub
  PROCs_script
  OUT 0, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0
ENDPROC
```

```
DEF PROCloadpage
  pica% = FALSE
  d$ = STRING$( 160, " " )
  d$ = ""
  s$ = STRING$( 160, " " )
  s$ = ""
  s¦ = 0
  fline% = 0
  REPEAT
    c¦ = BGET# infile% : REM load single byte from open file
    SWITCH c¦
      CASE 10 : c¦ = ASC( d$ )
                IF c¦ = $1F THEN
                  IF MID$( d$, 2, 2 ) = "9[" THEN
                    ruler% = INSTR( d$, "]" ) - 2
                    IF MID$( d$, ruler% + 3, 1 ) = "0" THEN pica% = TRUE
                    IF ruler% < 4 THEN ruler% = 0
                  ENDIF
                ELSE
                  fline% = fline% + 1
                  IF fline% > 48 THEN PRINT"more than 48 lines !":END
                  line$( fline% ) = d$
                  style$( fline% ) = s$
                ENDIF
                d$ = ""
                s$ = ""
      CASE 13 : REM ignor carriage return (LF is used as a line seperator)
      CASE 27 : s¦ = BGET# infile% : REM loads style change byte
      CASE 156: d$ = d$ + CHR$(26) : REM change   sign char.
                s$ = s$ + CHR$(s¦)
      DEFAULT : c¦ = c¦ AND 127
                IF (c¦ > 27) AND (c¦ < 31) THEN c¦ = 32 : REM convert all
                d$ = d$ + CHR$( c¦ ) : REM 'space' CHR's used by 1st WORD+ to " "
                s$ = s$ + CHR$( s¦ ) : REM add current byte & style from file to string
    ENDSWITCH
  UNTIL (EOF# infile%) OR (c¦ = 12)
ENDPROC

DEF PROCloadfont
  LOCAL chr, row, c¦, fontfile, fontfile%
  fontfile = ALERT("[2][ Load font file from disk ][ Yes ¦ No ]",1) - 2
  IF fontfile THEN fontfile% = FNfileselect( "A:\FONTS\*.FNT", "*.FNT" ) ELSE RESTORE
  FOR chr =0 TO 127
    FOR row = 0 TO 15
      IF fontfile THEN c¦ = BGET# fontfile% ELSE READ c¦
      font¦(chr,row) = c¦      : REM load PICA byte array
      IF c¦ AND 1 THEN f3¦(chr,row,1) = f3¦(chr,row,1) + 1
      IF c¦ AND 2 THEN f3¦(chr,row,3) = f3¦(chr,row,3) + 1
      IF c¦ AND 4 THEN f3¦(chr,row,2) = f3¦(chr,row,2) + 2
      IF c¦ AND 8 THEN f3¦(chr,row,1) = f3¦(chr,row,1) + 4
      IF c¦ AND 16 THEN f3¦(chr,row,3) = f3¦(chr,row,3) + 4
      IF c¦ AND 32 THEN f3¦(chr,row,2) = f3¦(chr,row,2) + 8
      IF c¦ AND 64 THEN f3¦(chr,row,1) = f3¦(chr,row,1) + 16
      IF c¦ AND 128 THEN f3¦(chr,row,3) = f3¦(chr,row,3) + 16
    NEXT row                   : REM convert to ELITE size text
  NEXT chr
  IF fontfile THEN CLOSE# fontfile%
ENDPROC
```

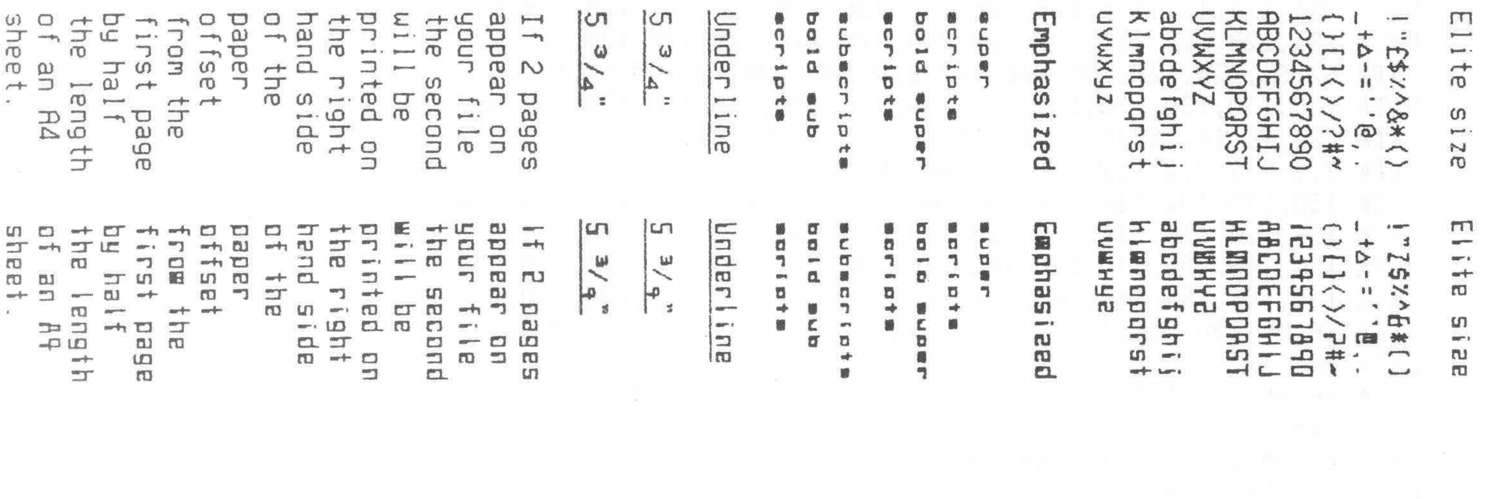
```
DEF PROCprint
  LOCAL lobyte%, hibyte%, col%, pass%, line%, c%, b¦, bl¦, s¦, u¦
  FOR line% = 1 TO fline%
    IF LEN(line$(line%))<ruler% THEN
      line$(line%) = line$(line%)+STRING$( ruler% - LEN(line$(line%))," " )
    ENDIF
    IF LEN(style$(line%))<ruler% THEN
      style$(line%) = style$(line%)+STRING$( ruler% - LEN(style$(line%)),CHR$(0) )
    ENDIF
    FOR i% = 0 TO 15
      b¦(line%,i%) = 0
    NEXT      : REM clear array used for emphasized text
  NEXT line%
  lobyte% = fline% * 40 : REM max. character height in bytes
  hibyte% = lobyte% DIV 256
  lobyte% = lobyte% MOD 256
  FOR col% = 1 TO ruler%
    FOR pass% = 3 TO 1 STEP - 1
      OUT 0, 27, 90, lobyte%, hibyte% : REM quadruple density selection
      FOR line% = fline% TO 1 STEP - 1
        s¦ = ASC(MID$(style$(line%),col%,1))
        IF pica% THEN u¦ = 255 ELSE u¦ = 63 : REM set underline width
        IF (s¦ AND 8) = 0 THEN u¦ = 0       : REM inhibit underline
        IF pass% <> 3 THEN
          IF (s¦ AND 1) = 0 THEN u¦ = 0
          IF (pass% = 1) AND (pica% = FALSE) THEN u¦ = 0
        ENDIF
        OUT 0, 0,0,0,0,0,0,0,u¦ : REM space between lines
        c% = ASC(MID$(line$(line%),col%,1))
        IF (s¦ AND 16) THEN
          PROCsuper             : REM superscript
        ELSE
          IF (s¦ AND 32) THEN
            PROCsub             : REM subscript
          ELSE
            PROCfull            : REM full size text
          ENDIF
        ENDIF
      NEXT line%
      OUT 0, 27,51,1, 13,10     : REM feed 1/216th inch only
    NEXT pass%
    IF pica% THEN
      OUT 0, 27,51,20           : REM set paper feed for PICA
    ELSE
      OUT 0, 27,51,15           : REM set paper feed for ELITE
    ENDIF
    OUT 0, 13,10                : REM CR LF
  NEXT col%
  OUT 0, 12                     : REM form feed to next page
ENDPROC

DEF PROCsuper
  OUT 0, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0
  PROCs_script
ENDPROC
```

```
DEF PROCfull
  FOR i% = 15 TO 0 STEP - 1
    IF pica% THEN
      IF pass% = 3 THEN b¦ = font¦( c%, i% ) ELSE b¦ = 0
    ELSE
      b¦ = f3¦( c%, i%, pass% )
    ENDIF
    b1% = b¦(line%,i%)      : REM emphasize text
    b¦(line%,i%) = b¦       : REM      - " -
    IF (s¦ AND 1) THEN b¦ = (b¦ OR b1%)
    IF pass% = 1 THEN b¦(line%,i%) = 0
    OUT 0, 0, b¦
  NEXT i%
ENDPROC

DEF PROCs_script
  FOR i% = 15 TO 0 STEP - 1
    IF pica% THEN
      IF pass% = 3 THEN b¦ = font¦( c%, i% ) ELSE b¦ = 0
    ELSE
      b¦ = f3¦( c%, i%, pass% )
    ENDIF
    b1% = b¦(line%,i%)      : REM emphasize text
    b¦(line%,i%) = b¦       : REM      - " -
    IF (s¦ AND 1) THEN b¦ = (b¦ OR b1%)
    IF pass% = 1 THEN b¦(line%,i%) = 0
    OUT 0, b¦
  NEXT i%
ENDPROC

REM resident font defintion ( each line defines one character )
DATA 255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
DATA 16,16,56,56,84,84,146,146,16,16,16,16,16,0,0,0
DATA 16,16,16,16,16,146,146,84,84,56,56,16,16,0,0,0
DATA 8,8,4,4,2,2,255,2,2,4,4,8,8,0,0,0
DATA 16,16,32,32,64,64,255,64,64,32,32,16,16,0,0,0
DATA 126,126,60,60,153,153,195,195,195,195,153,153,60,60,126,126
DATA 255,255,255,255,254,254,253,253,251,251,247,247,239,239,223,223
DATA 238,238,238,198,198,146,146,56,56,146,146,198,198,238,238,238
DATA 0,0,1,1,2,2,4,4,136,136,80,80,32,32,0,0
DATA 124,130,162,162,162,162,186,130,130,130,130,130,124,0,0,0
DATA 24,24,60,60,60,60,60,60,126,126,16,16,56,56,16,16
DATA 8,8,12,12,10,10,8,8,56,120,120,120,48,0,0,0
DATA 252,128,128,128,254,144,144,144,158,16,16,16,16,0,0,0
DATA 248,128,128,128,158,146,146,146,254,20,20,18,18,0,0,0
DATA 5,5,5,5,5,5,5,5,9,9,17,17,97,97,0,0
DATA 160,160,160,160,160,160,160,160,144,144,136,136,134,134,0,0
DATA 60,66,66,66,66,66,0,66,66,66,66,66,60,0,0,0
DATA 0,2,2,2,2,2,0,2,2,2,2,2,0,0,0,0
DATA 60,2,2,2,2,2,60,64,64,64,64,64,60,0,0,0
DATA 60,2,2,2,2,2,60,2,2,2,2,2,60,0,0,0
DATA 0,66,66,66,66,66,60,2,2,2,2,2,0,0,0,0
DATA 60,64,64,64,64,64,60,2,2,2,2,2,60,0,0,0
DATA 60,64,64,64,64,64,60,66,66,66,66,66,60,0,0,0
DATA 60,2,2,2,2,2,0,2,2,2,2,2,0,0,0,0
DATA 60,66,66,66,66,66,60,66,66,66,66,66,60,0,0,0
DATA 60,66,66,66,66,66,60,2,2,2,2,2,60,0,0,0
DATA 12,18,33,32,32,32,120,32,32,32,32,32,127,0,0,0
DATA 248,128,128,128,240,128,128,128,254,16,16,16,30,0,0,0
DATA 170,85,170,85,170,85,170,85,170,85,170,85,170,85,170,85
DATA 255,0,255,0,255,0,255,0,255,0,255,0,255,0,255,0
DATA 170,170,170,170,170,170,170,170,170,170,170,170,170,170,170,170
DATA 68,68,136,136,17,17,34,34,68,68,136,136,17,17,34,34
DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DATA 16,16,16,16,16,16,16,16,16,16,0,0,16,0,0,0
DATA 0,36,36,36,0,0,0,0,0,0,0,0,0,0,0,0
DATA 36,36,36,36,126,36,36,36,126,36,36,36,36,0,0,0
DATA 16,16,60,80,144,80,56,20,18,20,120,16,16,0,0,0
DATA 0,0,68,4,8,8,16,16,32,32,68,0,0,0,0,0
DATA 24,36,36,36,24,40,40,69,69,130,130,69,57,0,0,0
DATA 0,16,16,16,0,0,0,0,0,0,0,0,0,0,0,0
DATA 4,4,8,8,16,16,16,16,16,16,8,8,4,4,0,0
DATA 32,32,16,16,8,8,8,8,8,8,16,16,32,32,0,0
DATA 0,0,68,68,40,40,254,40,40,68,68,0,0,0,0,0
DATA 0,0,16,16,16,16,124,16,16,16,16,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,0,16,16,32,32,0
DATA 0,0,0,0,0,0,124,0,0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,0,16,16,0,0,0
DATA 0,0,2,2,4,4,8,8,16,16,32,32,64,64,0,0
DATA 56,68,68,130,130,130,130,130,130,130,68,68,56,0,0,0
DATA 16,48,48,16,16,16,16,16,16,16,16,16,124,0,0,0
DATA 56,68,130,2,2,4,8,16,32,64,128,128,254,0,0,0
DATA 56,68,130,2,2,4,24,4,2,2,130,68,56,0,0,0
DATA 12,12,20,20,36,36,68,68,132,132,254,4,4,0,0,0
DATA 252,128,128,128,128,248,4,2,2,2,130,68,56,0,0,0
DATA 56,68,130,128,128,128,248,132,130,130,130,68,56,0,0,0
DATA 254,130,130,4,4,8,8,16,16,16,16,16,16,0,0,0
DATA 56,68,130,130,130,68,56,68,130,130,130,68,56,0,0,0
DATA 56,68,130,130,130,66,62,2,2,2,2,4,120,0,0,0
DATA 0,0,0,16,16,0,0,0,16,16,0,0,0,0,0,0
DATA 0,0,0,16,16,0,0,0,16,16,32,32,0,0,0,0
DATA 8,8,16,16,32,32,64,64,32,32,16,16,8,8,0,0
DATA 0,0,0,0,124,0,0,0,124,0,0,0,0,0,0,0
DATA 32,32,16,16,8,8,4,4,8,8,16,16,32,32,0,0
DATA 56,68,130,2,2,4,8,16,16,0,0,16,16,0,0,0
DATA 56,68,130,130,158,162,162,162,156,128,128,64,62,0,0,0
DATA 56,68,130,130,130,130,254,130,130,130,130,130,130,0,0,0
DATA 248,132,130,130,130,132,248,132,130,130,130,132,248,0,0,0
DATA 56,68,130,128,128,128,128,128,128,128,130,68,56,0,0,0
DATA 248,132,130,130,130,130,130,130,130,130,130,132,248,0,0,0
DATA 254,128,128,128,128,128,248,128,128,128,128,128,254,0,0,0
DATA 254,128,128,128,128,128,248,128,128,128,128,128,128,0,0,0
DATA 62,64,128,128,128,128,142,130,130,130,130,68,56,0,0,0
DATA 130,130,130,130,130,130,254,130,130,130,130,130,130,0,0,0
DATA 124,16,16,16,16,16,16,16,16,16,16,16,124,0,0,0
DATA 2,2,2,2,2,2,2,2,2,2,130,68,56,0,0,0
DATA 130,130,132,132,136,136,240,136,136,132,132,130,130,0,0,0
DATA 128,128,128,128,128,128,128,128,128,128,128,128,254,0,0,0
DATA 130,198,198,170,170,146,146,130,130,130,130,130,130,0,0,0
DATA 130,194,194,162,162,146,146,138,138,134,134,130,130,0,0,0
DATA 56,68,130,130,130,130,130,130,130,130,130,68,56,0,0,0
DATA 248,132,130,130,130,132,248,128,128,128,128,128,128,0,0,0
DATA 56,68,130,130,130,130,130,130,138,138,132,68,58,0,0,0
DATA 248,132,130,130,130,132,248,136,136,132,132,130,130,0,0,0
DATA 56,68,130,128,128,64,56,4,2,2,130,68,56,0,0,0
DATA 254,16,16,16,16,16,16,16,16,16,16,16,16,0,0,0
DATA 130,130,130,130,130,130,130,130,130,130,130,68,56,0,0,0
DATA 130,130,130,130,130,130,130,68,68,40,40,16,16,0,0,0
DATA 130,130,130,130,130,130,146,146,170,170,198,198,130,0,0,0
DATA 130,130,68,68,40,40,16,40,40,68,68,130,130,0,0,0
DATA 130,130,130,130,68,68,40,40,16,16,16,16,16,0,0,0
DATA 254,4,4,8,8,16,16,32,32,64,64,128,254,0,0,0
DATA 28,16,16,16,16,16,16,16,16,16,16,16,28,0,0,0
DATA 128,128,64,64,32,32,16,16,8,8,4,4,2,0,0,0
DATA 56,8,8,8,8,8,8,8,8,8,8,8,56,0,0,0
DATA 16,16,40,40,68,68,130,130,0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,0,0,254,0,0,0
DATA 0,16,16,16,0,0,0,0,0,0,0,0,0,0,0,0
DATA 0,0,0,0,120,132,4,4,124,132,132,132,122,0,0,0
DATA 128,128,128,128,248,132,132,132,132,132,132,132,248,0,0,0
DATA 0,0,0,0,120,132,128,128,128,128,128,132,120,0,0,0
DATA 4,4,4,4,124,132,132,132,132,132,132,132,124,0,0,0
DATA 0,0,0,0,120,132,132,132,248,128,128,128,120,0,0,0
DATA 14,16,16,16,56,16,16,16,16,16,16,16,16,0,0,0
DATA 0,0,0,0,120,132,132,132,132,132,124,4,4,4,120,0
DATA 128,128,128,128,248,132,132,132,132,132,132,132,132,0,0,0
DATA 16,0,0,0,48,16,16,16,16,16,16,16,56,0,0,0
DATA 8,0,0,0,8,8,8,8,8,8,8,8,8,8,48,0
DATA 128,128,136,136,144,144,224,144,144,136,136,132,132,0,0,0
DATA 48,16,16,16,16,16,16,16,16,16,16,16,56,0,0,0
DATA 0,0,0,0,236,146,146,146,146,130,130,130,130,0,0,0
DATA 0,0,0,0,248,132,132,132,132,132,132,132,132,0,0,0
DATA 0,0,0,0,120,132,132,132,132,132,132,132,120,0,0,0
DATA 0,0,0,0,248,132,132,132,132,132,132,132,248,128,128,128
DATA 0,0,0,0,124,132,132,132,132,132,132,132,124,4,4,4
DATA 0,0,0,0,248,132,132,128,128,128,128,128,128,0,0,0
DATA 0,0,0,0,120,132,128,128,120,4,4,132,120,0,0,0
DATA 16,16,16,16,124,16,16,16,16,16,16,16,12,0,0,0
DATA 0,0,0,0,132,132,132,132,132,132,132,132,124,0,0,0
DATA 0,0,0,0,130,130,130,68,68,40,40,16,16,0,0,0
DATA 0,0,0,0,130,130,130,130,130,146,146,170,68,0,0,0
DATA 0,0,0,0,132,132,72,72,48,72,72,132,132,0,0,0
DATA 0,0,0,0,132,132,132,132,132,132,124,4,4,4,120,0
DATA 0,0,0,0,124,4,8,8,16,16,32,32,124,0,0,0
DATA 24,32,32,32,32,32,64,32,32,32,32,32,24,0,0,0
DATA 16,16,16,16,16,16,16,16,16,16,16,16,16,0,0,0
DATA 48,8,8,8,8,8,4,8,8,8,8,8,48,0,0,0
DATA 0,0,34,84,84,136,136,0,0,0,0,0,0,0,0,0
DATA 0,0,16,16,40,40,68,68,130,130,255,0,0,0,0,0
```



```
Elite size
!"£$%^&*()
_+Δ-=`'@,.
{}[]<>/?#~
1234567890
ABCDEFGHIJ
KLMNOPQRST
UVWXYZ
abcdefghij
klmnopqrst
uvwxyz
Emphasized
super
scripts
bold super
scripts
subscripts
bold sub
scripts
Underline
5 3/4"
5 3/4"
If 2 pages
appear on
your file
the second
will be
printed on
the right
hand side
of the
paper
offset
from the
first page
by half
the length
of an A4
sheet.
```

```
Elite size
!"£$%^&*()
_+Δ-=`'@,.
{}[]<>/?#~
1234567890
ABCDEFGHIJ
KLMNOPQRST
UVWXYZ
abcdefghij
klmnopqrst
uvwxyz
Emphasized
super
scripts
bold super
scripts
subscripts
bold sub
scripts
Underline
5 3/4"
5 3/4"
If 2 pages
appear on
your file
the second
will be
printed on
the right
hand side
of the
paper
offset
from the
first page
by half
the length
of an A4
sheet.
```

CASSETTE MOTOR CONTROL AND BELL ON THE OCTOPUS/EC65

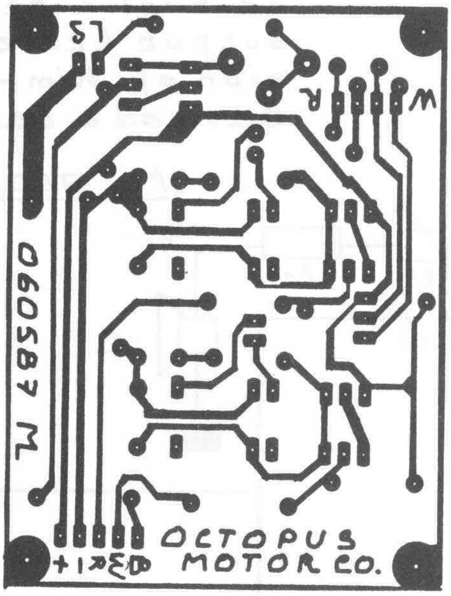
M. Lachaert.

In the 'Computer special' nr 1, Elektor published a well-working cassette interface card, as well for Kim/Jusior as for Basicode format. Unfortunately, they omitted to build in an important feature!The card had no possibilities at all to control the cassette motors on/off. The present circuitry remediates to this gap, and offers beside this a handy possibility to build in a 'bell' feature.

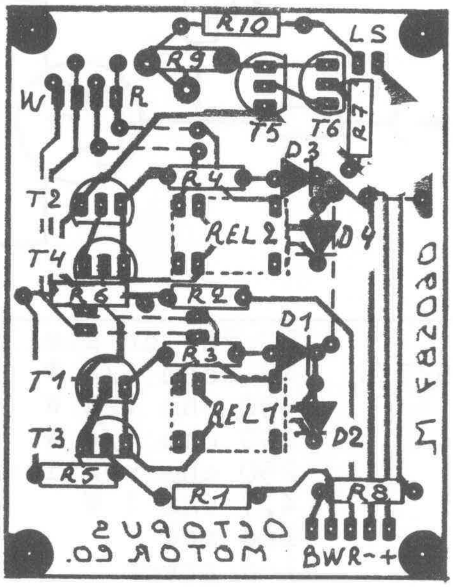
The circuit utilizes three free output lines of the port IC4 on the interface board as described by Elektor. Output S6 (pin 5) controls the 'write' recorder motor, output S7 (pin 4) controls the 'read' recorder motor, while output S8 (pin 3) feeds a small amplifier connected to a miniature speaker, featuring the bell.

The write control hardware and the read control hardware are two very similar PNP-darlington amplifiers, which have both a relay as load. As usual in case of inductive-loaded transistor design, D2 and D4 are incorporated to protect the transistors against inductive voltage peaks. The optional light emitting diodes D1 (red led = write) and D3 (green led = read) can be installed on the front of the computer to indicate the control state. Note that the resistors R3 and R4 are the sole non-identical parts in both write and read logic. This is due to the different working voltages between red and green leds.

I preferred to use single PNP-transistors, rather than real two-in-one-case darlingtons, because of their better availability. The printed circuit board has been designed for low-cost 5 Volt miniature relays, which are very well available too.

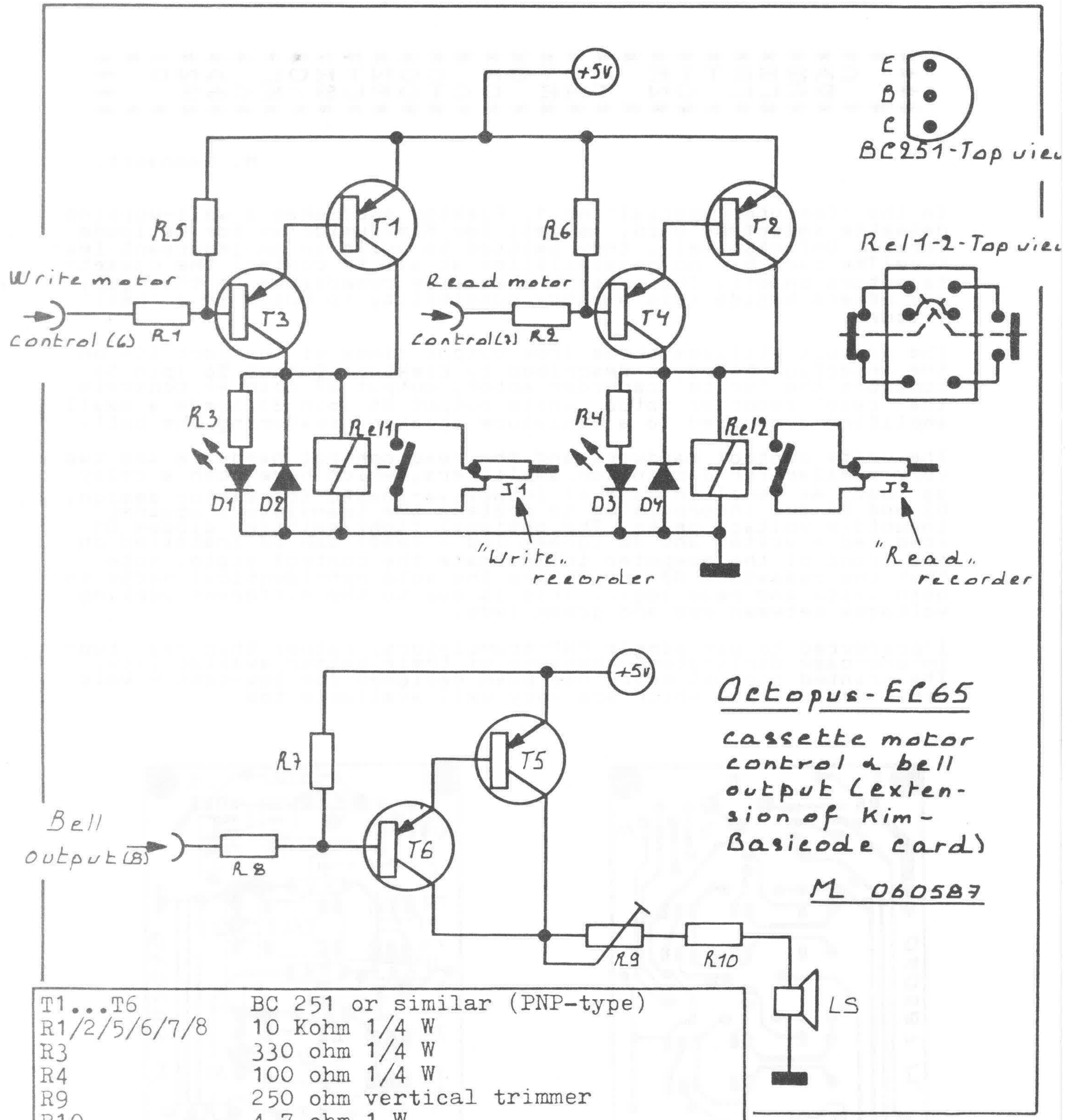


copper side



component side

The control hardware is in fact very simple. As long as the input pin of R1 (R2) remains on HI-level (5 V), the darlington is non-conducting. Even when this pin is open, nothing happens, because of the base of T3 (T4) is pulled HI by R5 (R6). Once the logical level on R1 (R2) is LO (less than 0.7 V), the darlington



Octopus-EC65

cassette motor control & bell output (extension of Kim-Basicode card)

M 060587

T1...T6	BC 251 or similar (PNP-type)
R1/2/5/6/7/8	10 Kohm 1/4 W
R3	330 ohm 1/4 W
R4	100 ohm 1/4 W
R9	250 ohm vertical trimmer
R10	4.7 ohm 1 W
D1	Led red
D3	Led green
D2/4	1N4148 or similar
Rel1/2	Miniature relay 5 V / 80 ohm
J1/2	Subminiature telephone jack
LS	Miniature loudspeaker 8 ohm 150 mW

goes in conduction, Rel1 (Rel2) closes and the selected recorder can start. This means that the control software must be written in order that a HI level (logical '1') must exist at the output lines S6 and S7 of IC4 on the Elektor interface board, as long as the recorders must be stopped. A LO level (logical '0') on output S6 starts the 'write' recorder, while a LO level on output S7 starts the 'read' recorder.

The 'bell' circuitry is even simple. While not operational, a HI level on R8 opens the darlington T5/T6. In operation, the 'bell' software must apply a square-wave in the audible spectrum (at best around 2000 Hertz) on output line 8 of IC4. This signal is buffered by the darlington, and the signal level can be regulated by R9, in order to avoid neighbour complaints...

In a 'normal' Octopus configuration, the KIM I/O port is adressed at $E280. An ennoying detail is that the output status of this port (in fact a simple latch...) can not be read out by the processor. So, for correct operation, the status of the port has to be copied somewhere in the memory, and at each change, this change has to be 'told' also to this 'slave' location.

A sample program that features all the controls described above can have the following form:

1. initialize the port:

```
INIZ      LDA #$FF   ; SET ALL HIGH ---> TURN OFF

SETPRT    STA KIMIO  ; SET/RESET THE DESIRES PORT BITS
          STA GANG   ; SAVE STATUS IN SLAVE LOCATION
          RTS        ; ALL DONE
```

2. turn the 'read' motor on:

```
READON    LDA GANG   ; FETCH PREVIOUS STATUS
          AND #$FE   ; 'READMOTOR' BIT = LOW ---> MOTOR ON
          JMP SETPRT ; GO DO IT
```

3. turn the 'write' motor on:

```
WRON      LDA GANG   ; FETCH PREVIOUS STATUS
          AND #$CF   ; 'WRITEMOTOR' BIT = LOW --> MOTOR ON
          JMP SETPRT ; GO DO IT
```

4. turn the 'read' motor off:

```
READOF    LDA GANG   ; FETCH PREVIOUS STATUS
          ORA #$40   ; 'READMOTOR' BIT = HIGH --> MOTOR OFF
          BNE SETPRT ; GO DO IT
```

5. turn the 'write' motor off:

```
WROFF     LDA GANG   ; FETCH PREVIOUS STATUS
          ORA #$20   ; 'WRITEMOTOR' BIT = HIGH -> MOTOR OFF
          BNE SETPRT ; GO DO IT
```

6. ring the bell:

```
BELL      LDA GANG   ; FETCH OLD PORT STATUS
          ORA #$80   ; SET BELL BIT
          LDY #$80   ; Y = # OF 1/2 PERIODES

TOGGLE    JSR SETPRT ; CHANGE PORT
          LDX #$28   ; X = 1/2 PERIODE LENGHT

PERIOD    DEX        ; MAKE 1 PERIODE
          BNE PERIOD
          EOR #$80   ; TOGGLE BELL-BIT
          DEY        ; NEXT 1/2 PERIODE
          BNE TOGGLE ; NOT YET LAST
          RTS        ; DONE
```

```
****************************************
*  PRINTER INIT FOR ELECTRON AND BBC  *
****************************************
```

By: Ronald van Vugt (PA3EAH), The Netherlands

With this program you are able to set some options for a EPSON-printer. You'll see this options by typing *HELP. The program uses two pages of memory.

Met dit programma kunt u op een makkelijke manier verschillende instellingen op een EPSON-printer verwezelijken. Als u *HELP intikt ziet u alle mogelijkheden. Het programma beslaat precies 2 pagina's.

```
 10 REM Printer init for the ELECTRON and BBC
 20 REM By Ronald van Vugt (PA3EAH), The Netherlands
 30 REM-----------------------------------------
 40 REM startaddress next command
 50 commands_low=&72:commands_high=&73
 60 REM address from oscli (*command) vector
 70 save_vec_low=&70:save_vec_high=&71
 80 REM startaddress where a *command starts (in ASCII)
 90 oscli_low=&74:oscli_high=&75
100 len=&76:REM number of letters in a *command, flag
110         REM (0=command off, not 0=commando on),
120         REM flag (0=printer off, not 0=printer on)
130         REM and a temporary memory place
140 status=&77:REM bit 0 is 0=>pica, bit 0 is 1=>elite
150 REM bit 1=> proportional, bit 2=> compressed
160 REM bit 3=> emphasized, bit 4=> doublestrike
170 REM bit 5=> expanded, bit 6=> italics
180 REM bit 7=> underlining. bit=1 => option on
190                          REM bit=0 => option off
200 oswrch=&FFEE:REM to put the value from the
210                REM accumulator on the screen
220 osnewln=&FFE7:REM print a linefeed + carriage return
230 osbyte=&FFF4:REM invoking OS facilities
240 osasci=&FFE3:REM to put the value from the
250 REM accumulator on the screen. When A is 13
260 REM there'll be print a linefeed and carriage return
270 FOR pass%=0 TO 3STEP3
280 P%=&900:REM P%=&C000 if you've a 'TUBE'
290 [OPT pass% \pass%=0 => no error report
300            \pass%=3 => error report
310 .init
320 LDA &208:STA save_vec_low   \oscli_vector to
330 LDA &209:STA save_vec_high  \save_vec
340 LDA #search MOD256:STA &208 \startaddress form search
350 LDA #search DIV256:STA &209 \to oscli_vector
360 LDA #0:STA status           \all options off, pica
370 RTS
380 .search \when you typed a *command, the program
390         \jumps to search
400 \startaddress from commands to commands_low and high
410 LDA #commands MOD256:STA commands_low
420 LDA #commands DIV256:STA commands_high
430 \start from typed *command to oscli_low and high
440 STX oscli_low:STY oscli_high:LDX #0
450 .lus1
460 \number of letters from commando to len
470 LDY #0:LDA (commands_low),Y:STA len
480 .lus2
490 \next letter from commando.When * => commando found
500 INY:LDA (oscli_low),Y:CMP #ASC"*":BEQ true
510 \make capital from typed letter. When typed letter
520 \and letter from commando are not the same,not found
530 AND #223:CMP (commands_low),Y:BNE false
540 \when all letters from commando compared => found
550 CPY len:BNE lus2
560 .true
570 LDA #0:STA len   \flag (all options off)
580 CPX #9:BEQ reset \when X=9 => reset
590 CPX #10:BEQ help \when X=10 => help
600 .lus6
610 \remove all the spaces after your typed *commando
620 INY:LDA (oscli_low),Y:CMP #32:BEQ lus6
630 \take the second letter after the spaces. When it
640 \is a 'F' or 'f' (from oFf) set len (a flag)
650 INY:LDA (oscli_low),Y:AND #223
660 CMP #ASC"F":BNE no_on_off:STA len
670 .no_on_off
680 CPX #0:BNE no_pica \when X isn't 0 => no pica
690 \change status and send to printer
700 LDA #254:AND status:STA status:JMP printer
710 .no_pica
720 CPX #1:BNE no_elite \when X isn't 1 => no elite
730 \change status and send to printer
740 TXA:ORA status:STA status:JMP printer
750 .no_elite
760 \save len (a flag) to place it into Y
770 LDA #0:LDY len:SEC
780 .lus7
790 \set (X-1)th bit in A (when X=3, A becomes %00000100)
800 ROL A:CLC:DEX:BNE lus7
810 \when option on => jump to on
820 STA len:CPY #0:BEQ on
830 \change status and send to printer
840 LDA #255:EOR len:AND status:STA status:JMP printer
850 .on
860 \change status and send to printer
870 ORA status:STA status:JMP printer
880 .reset
890 \clear status and reset printer
900 LDA #0:STA status:JSR escape
910 LDA #ASC"@":JMP sub_printer
920 .false
930 \next commando. When all commands checked =>
940 \unrecognize command
950 INX:CPX #11:BNE sub_change
960 \jump to old oscli_vector
970 LDX oscli_low:LDY oscli_high:JMP (save_vec_low)
980 .sub_change
990 \change commands_low and high to next commandaddress
1000 JSR change:JMP lus1
1010 .change
1020 LDA len:SEC:ADC commands_low:STA commands_low
1030 LDA #0:ADC commands_high:STA commands_high:RTS
1040 .help
1050 \startaddress copyright to commands_low and high
1060 LDA #copyright MOD256:STA commands_low
1070 LDA #copyright DIV256:STA commands_high
1080 \clear screen
1090 LDA #12:JSR oswrch:LDX #0
1100 .lus3
1110 \number of letters from commando to len
1120 LDY #0:LDA (commands_low),Y:STA len
1130 \print linefeed and carriage return. Fix if
1140 \there must be a '*' for the printed information
1150 JSR osnewln:CPX #0:BEQ lus4:CPX #12:BEQ lus4
1160 \print a '*'
1170 LDA #ASC"*":JSR oswrch
1180 .lus4
1190 \print the information on the screen
1200 INY:LDA (commands_low),Y:JSR osasci
1210 CPY len:BNE lus4
1220 \fix if there must be [ON/OFF] after
1230 \the printed information
1240 CPX #3:BPL on_off
1250 .back
1260 \next information. RTS when finished
1270 INX:CPX #13:BEQ rts
1280 \change command_low and high to next commandaddress
1290 JSR change:JMP lus3
1300 .on_off
1310 \fix if there must be [ON/OFF] after
1320 \the printed information
1330 CPX #10:BPL back
1340 \fix the number of spaces
1350 LDA #17:SEC:SBC len:TAY:LDA #9
1360 .space
1370 \print Y-spaces
1380 JSR oswrch:DEY:BNE space
1390 LDY #0
1400 .lus5
1410 \print [ON/OFF]
1420 LDA sub,Y:JSR oswrch
1430 INY:CMP #32:BNE lus5:BEQ back
1440 .printer
1450 \printer on? send CHR$(27) to printer
1460 JSR status_printer:JSR escape
1470 \send a "!" and status to printer
1480 LDA #ASC"!":JSR sub_printer
1490 LDA status:JSR sub_printer
1500 \when len=0 => printer was off
1510 LDA len:BNE rts
1520 \switch printer off
1530 LDA #3:JMP oswrch
1540 .sub_printer
1550 \send CHR$(1) and the accumulator to printer
1560 PHA:LDA #1:JSR oswrch:PLA:JMP oswrch
1570 .escape
1580 \send CHR$(1) and CHR$(27) to printer
1590 LDA #27:JMP sub_printer
1600 .status_printer
1610 \if the printer is off => len=0, if not len<>0
1620 LDA #&75:STA len:JSR osbyte:TXA:AND #1:BNE rts
1630 \set printer on
1640 STA len:LDA #2:JMP oswrch
1650 .rts
1660 RTS
1670 .copyright
1680 EQUB 16:EQUS "(C) 6502 KENNER"+CHR$(13)
1690 .commands
1700 EQUB 4:EQUS  "PICA"
1710 EQUB 5:EQUS  "ELITE"
1720 EQUB 12:EQUS "PROPORTIONAL"
1730 EQUB 10:EQUS "COMPRESSED"
1740 EQUB 10:EQUS "EMPHASIZED"
1750 EQUB 12:EQUS "DOUBLESTRIKE"
1760 EQUB 8:EQUS  "EXPANDED"
1770 EQUB 7:EQUS  "ITALICS"
1780 EQUB 11:EQUS "UNDERLINING"
1790 EQUB 5:EQUS  "RESET"
1800 EQUB 5:EQUS  "HELP"+CHR$(13)
1810 EQUB 35:EQUS "Wildcarts (*) zijn ook"
1820 EQUS" toegestaan."+CHR$(13)
1830 .sub
1840 EQUS "ON/OFF "
1850 ]
1860 NEXT pass%
```

PAPERWARE & DISKETTE SERVICE

```
*****************************************************************
*        UNIVERSAL TERMINAL V0.23 for DOS65 V0.2x               *
*****************************************************************
```

```
Syntax:  TERMinal [-CDKLMP +v,w,x,y,z]
Options: -C : Print unknown control characters on screen []
         -D : Delay after each character during file trans-
              fer.  For  systems without handshaking  (e.g.
              Elektor's Junior computer)
         -K : Keep  local  copy of  characters  typed  from
              keyboard
         -L : Transmit line feed with CR
         -M : Add line feed to a CR received
         -P : Do not send LF to printer after CR
  +v,w,x,y,z : Communication  parameters (defaults to origi-
              nal settings)

v=transmit baud : 1-External  2-50  3-75  4-110  5-134
                  6-150  7-300  8-600  9-1200  10-1800
                  11-2400   12-3600  13-4800   14-7200
                  15-9600   16-19200
w=word length   : 1-8 bits  2-7 bits  3-6 bits  4-5 bits
x=parity        : 1-None  2-Odd  3-Even  4-Mark  5-Space
y=stop bits     : 1-1 bit  2-1 1.5 or 2 bits
                  (depends on w and x)
z=receive baud  : 0-same as transmitter (default)
                  1-external
```

The program allows a DOS65 version 2 computer to act as a terminal to another machine. It is most conveniently used by calling from a command file e.g.

see JUNIOR

```
; Junior terminal
; 1800 baud
TERMINAL -KDC +10,2,5,1
```

Either all or none of the parameters v,w,x,y,z must be given. If they are not given the values stored at $E734/5 are used with the interrupts turned on automatically. On leaving Yerminal the communication parameters on the host may pass commands to DOS65.
Terminal allows all the characters sent to the screen to be sent to a printer and a disc file. Only certain control codes can be sent to the printer but everything goes to the file. Input may coma from ASCII and binary files instead of the keyboard. Binary files are transmitted in Junior PM format (A9.80.A2.) with or without address information.
The program uses the non-standard routines TONSC and TOFFSC from I/O 65. If they are not at the expected address the program will produce an error message.

UNIVERSAL TERMINAL was made by: Andrew Gregory, England.

The following is available for DOS65:

-DOS65 40/80 trs, SS or DS diskette, only object code:
 Send formatted diskette with label and R/W prot. to the editorial office.
 Europe : Hfl. 72,00   Outside Europe : Hfl. 87,00
 Members: Hfl. 22,00   Members: Hfl. 37,00
 If paying with Eurocheque or on postgiro 841433 of W.L. van Pelt at Krimpen a.d. IJssel, subtract Hfl. 9,50.
-DOS65 40/80 trs, SS or DS diskette, with all sources:
 Send formatted diskette with label and R/W prot. to the editorial office.
 Europe : Hfl. 84,50   Outside Europe : Hfl.101,50
 Members: Hfl. 34,50   Members: Hfl. 51,50
 If paying with Eurocheque or on postgiro 841433 of W.L. van Pelt at Krimpen a.d. IJssel, subtract Hfl. 9,50.

The following is available for other users:

-Source Listing of the UNIVERSAL TERMINAL V0.23 for DOS65.
 Europe : Hfl. 72,00   Outside Europe : Hfl. 87,00
 Members: Hfl. 22,00   Members: Hfl. 37,00
 If paying with Eurocheque or on postgiro 841433 of W.L. van Pelt at Krimpen a.d. IJssel, subtract Hfl. 9,50.

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

****

DATACOMMUNICATION WITH 6502 COMPUTERS B. de Bruine 15-6-87

****

1. Introduction

The only things you need for datacommunication is a modem, a telephone connection and a computer with the right communication software. This article is a very brief introduction in datacommunication.

2. Modems

The development of high speed modems goes very fast. When scientists a few years ago pretend that the maximum available baudrate, usable on ordinary telephone lines is limited to 1200 Baud, nowadays we know, even 9600 Bd is possible! Modulating with several modulation methods at the same time increases the transfer speed. E.g. only AM correspond with 600 Bd, AM & FSK results in 2x600=1200 Bd. Adding PSK increases the speed to 2x1200=2400 Bd. With special encoding and encryption algorithms it is even possible to reach a transferrate of 9600 Bd. Unfortunately there is not yet one uniform standard for 9600 Bd modems. Another technique to spend time is file-compression, like ARC(hive) tools. The speed of the modem is not increased, but the number of data is decreased by this method. Modems can be divided in two categories: Hayes (compatible) or not (transparant modems). The Hayes 'AT' commandset is international standardizised. With those commands it is possible to set the bautrate, to autodial, program the wordformat, etc. etc. Transparant modems are dumb modems. All settings must be done manual.

3. Databanks and bulletin boards

What offer databanks (like the fido's) to the inlogger? The Dutch databanks contains a lot of software for popular computers like IBM-PC, Atari-ST, C-64, etc. Unfortunately there is no software available for DOS-65 computers. The only reason to log in is to download machine independent high level programmes and communication with other users of the databank.

4. Communication protocols

To let a computer 'talk' to another computer, a protocol is needed, to avoid misunderstanding. Very popular and many used protocols are:

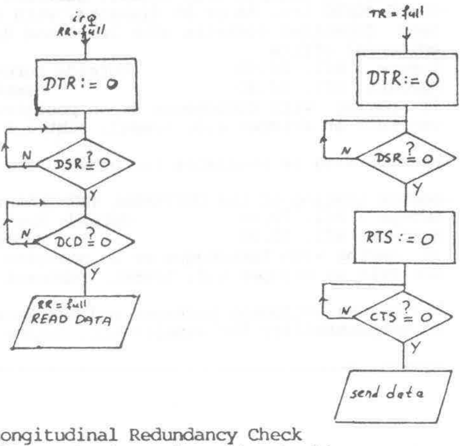
-Ascii : (wait/stop, ^s/^q). Only for textfiles.
-Xmodem: Transfer of all kind of files (read article Xmodem J. Banser soon published)
-Kermit: idem.
-Videotex: Transfer pages with text and graphics (read DE 6502 KENNER nr. 47/48)

5. Errordetection and correction

There are several ways to recognize an error in received data. All of them add redundant information to the data. Errorcorrection is realised by asking the transmitter to transmit the data again. None of the mentioned methods are 100% full-proof (some duplicate errors cancel each other). CRC is better than LRC (statistical 99% error-free!), and LRC is better than VRC.

5.1 Vertical Redundancy Check (VRC or paritycheck)

To every character one bit is added, the so called paritybit. This bit is used to make the number of '1's in the character even (even parity) or odd (odd parity). The paritybit is often called the redundant-bit, because it contain extra information, only used for errorchecking.



5.2 Longitudinal Redundancy Check (LRC or horirontal parity check)

Not every particular character is checked for errors, but a whole datablock is checked. At the end of the block an extra checkbyte is added. This Block Check Character (BCC) consists of horizontal parity bits. Both the sender and receiver calculate the BCC. If the BCC is equal on both sides, the receiver draw the conclusion that the data is received correct.

5.3 Cyclic Redundancy Check (CRC)

According to a polynome a CRC-generator generates a CRC-character. The CRC-character is send as a BCC after the datablock. The receiver calculates an own checkcharacter. With help of mathematics the receiver can detect if the received datablock is error-free (remainder of division CRC(Tx)/CRC(Rx)=constant). For advanced errorchecking CRC(16) is very popular. In the hobbyworld mostly CRC(8) is used. CRC(8)=LRC ==> G(x)=x**8+1.

5.4 Examples

```
VRC:                LRC:                 BCC = Block Check Char.
1010 1010 even  char 1: 1010 1010  FCS = Frame Check
0011 0101 even  char 2: 0101 0101        Sequence
1010 1011 odd   char 3: 1111 0000  BCC = FCS
               even LCR: 0000 1111  FCS is the official
                                         CCITT name.
```

6. Elementary routines of a communication program

Every communication program has a routine to read/write a character from/to the serial port. The following brief description of these routines for a 6502 processor and an 6551 ACIA, are derived from the DOS65 Astrid communication program.

6.1 Interrupt routine (ACIINT)

An interrupt routine has a higher priority than an ordinary routine. This means every time an ACIA-IRQ occurs, the interruptroutine ACIint places a received character in the receiverbuffer, or a character of the transmitter-queue is transmitted. The advantage of this method is, that no characters get lost, because the masterprogram is interrupted every time the ACIA need attention. The flowcharts shows how to read/write a character.

LIST 1 presents the interruptroutine sourcecode.

```
18AB 60        ACIINT   RTS
               ;Aciint
               ;Receive and transmit data via acia
18AC AD 31E1            LDA ACIASR
18AF 29 08              AND #%00001000  ;Transmitter full
18B1 F0 0D              BEQ TRANSMT
18B3 AD 30E1            LDA AREG        ;Dataregister (re-
18B6 AE 2410            LDX RECPNT      ;ceiver)
18B9 9D 0002            STA RECBUF,X
18BC EE 2410            INC RECPNT
18BF 60                 RTS
18C0 AD 2510 TRANSMIT LDA TRANPNT       ;Transmit buffer
                                        ;empty ?
18C3 F0 2C              BEQ DLYINT
18C5 AD 7610            LDA DELAY       ;Conversion delay?
18C8 D0 05              BNE 1.F
18CA AD 7710            LDA CDELAY      ;Wait until conver-
                                        ;sion time elapsed
18CD D0 22              BNE DLYINT
18CF AD 0003 1          LDA TRANBUF
18D2 48                 PHA             ;Save first on stack
18D3 78                 SEI
18D4 A2 01              LDX #1
18D6 EC 2510            CPX TRANPNT
18D9 F0 09              BEQ 99.F
18DB BD 0003 TRANMOV    LDA TRANBUF,X   ;Move
18DE 9D FF02            STA TRANBUF-1,X
18E1 E8                 INX
18E2 D0 F7              BNE TRANMOV
18E4 CE 2510 99         DEC TRANPNT     ;Tranbuf:=tranbuf-1
18E7 68                 PLA
18E8 58                 CLI
18E9 8D 30E1            STA AREG        ;And send byte to
                                        ;modem
18EC 60        DLYINT   RTS (DOS65 system)
or
18EC 40                 RTI (other system)
```

6.2 Transmit a character (EN2)

Since the interruptroutine does the most work, the transmitroutine (EN2) only has to place the character in the transmitbuffer and update the pointer. See List 2.

LIST 2

```
               ;wait until free space in buffer (MAXTRAN)
               ;Entry: a=char to be transmitted
               ;Exit:  a,y unchanged, x destroyed
1838 AE 4C10 EN2        LDX BREAK       ;Breakkey pressed?
183B F0 0D              BEQ 5.F         ;Leave
183D AE 2510            LDX TRANPNT     ;And store in tranbuf
1840 E0 FF              CPX #MAXTRAN    ;Buffer full?
1842 F0 F4              BEQ EN2         ;Wait until free
1844 9D 0003 6          STA TRANBUF,X   ;space
1847 EE 2510            INC TRANPNT
184A 60      5          RTS
```

6.3 Receive a character (GETMOD)

First a check on the receiverpointer is needed. If this pointer equals to zero, no character is received. If there are any characters in the buffer, the first character is loaded in A.

LIST 3

```
184B AE 2410 GETMOD   LDX RECPNT      ;char. received?
184E F0 1A            BEQ 1.F         ;No char in buffer
1850 AD 0002          LDA RECBUF
1853 48               PHA             ;Save first input on
1854 A2 01            LDX #1          ;stack
1856 EC 2410          CPX RECPNT
1859 F0 09            BEQ 2.F
185B BD 0002 QUE      LDA RECBUF,X    ;Move queue
185E 9D FF01          STA RECBUF-1,X
1861 E8               INX
1862 D0 F7            BNE QUE
1864 CE 2410 2        DEC RECPNT      ;Recbuf:=recbuf-1
1867 68               PLA
1868 18               CLC
1869 60               RTS
186A 38       1       SEC             ;Exit with c=1 if no
186B 60               RTS             ;char. received
```

6.4 Receive a character within a specified time interval (READBYT)

To avoid a deadlock, some protocols like Xmodem and Kermit uses a time-out variable. The time-out time is the minimum number of seconds to wait for a databyte. If the databyte is not received within this time, a time-out flag is set, and the software decide to try again or to cancel receiving. Readbyte is entered with the time-out time in A, e.g.:

```
LDA #10
JSR READBYT
```

specifies a time-out of 10 sec.
The variable VIAVRA is on IO65 variable, which is every second decremented by one.

LIST 4

```
               ;Readbyte entry: a=time-out time
               ;          exit : c=1 time-out
               ;          break=0 = breakkey pressed
               ;          c=0 a=received char
2C21 8D 1BE7 READBYT  STA VIAVRA      ;Set time-out time
2C24 AD 4D10 READBUF  LDA CANFLG
2C27 D0 02            BNE READ
2C29 38               SEC
2C2A 60               RTS             ;Cancel exit
2C2B 20 4B18 READ     JSR GETMOD      ;fetch char from
                                      ;recbuf
2C2E B0 01            BCS EMPTY       ;no char in buf
2C30 60               RTS
2C31 20 542E EMPTY    JSR BRKTEST
2C34 90 02            BCC 9.F
2C36 38               SEC             ;Break exit
2C37 60               RTS
2C38 AD 1BE7 9        LDA VIAVRA
2C3B D0 E7            BNE READBUF
2C3D 38               SEC             ;Time-out exit
2C3E 60               RTS
```

6.5 Set speed serial port

To select the wanted baudrate, a table of many used baudrates is made. Entering the table with in X a parameter (1..5) the baudrate is programmed with:

```
LDA BAUDTAB,X
STA ACICR
```

The (s) means split speed, the receiver is disconnected from the internal baudrate generator (read DE 6502 KENNER 49, page 29 about an external baudrate generator).

LIST 5

```
         ;Baudtable predefined baudrates
1A66 00 BAUDTAB FCB   $00 ;Reserved
1A67 1A         FCB   $1A ;2400 baud, x=1
1A68 18         FCB   $18 ;1200 baud, x=2
1A69 16         FCB   $16 ; 300 baud, x=3
1A6A 08         FCB   $08 ;1200 baud, x=4(s)
1A6B 02         FCB   $02 ;  75 baud, x=5(s)
```

6.6 Terminal emulating

With the mentioned routines it is possible to let a computer act as a simple terminal.

```
TERMINAL JSR GETKEY   ;Key from keyboard?
         BCS GMOD     ;C=1 no key
         JSR EN2      ;Transmit key
GMOD     JSR GETMOD   ;Data received?
         BCS TERMINAL ;C=1 nothing received
         JSR OUT      ;Print the character
         JMP TERMINAL ;Stay in loop
```

Of course, it is more fun to make a more intelligent terminal, with breakkey detection, filtering of illegal characters, macro expansion, full/half duplex option, etc., but the base is always a loop like this.

7. DOS65 communication packet

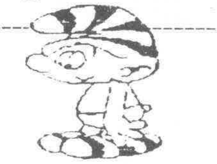
For DOS65 the following communication programs are available:

- Communication 65 (ASTRID)
  ========================
  specifications: -Terminal emulation
  -Up/downloading with Ascii- and Xmodem protocol
  -Macro-expansion
  -Autodial facilities
  -Support Hayes protocol, interspeeder or split-speed
  -Can be configurated for every modem and every systemclock.

- Viditel 65
  ==========
  With this program you can connect your computer to a videotex host. The EC65 viditelprogram is converted to DOS65. Read for more information the articles of Coen Boltjes about this program in DE 6502 KENNER 47/48. No hardware changes needed!
  Extra facilities are:
  -Support Hayes protocol, interspeeder and split-speed operation
  -Disk storage
  -Macro expansion
  -Autodialing
  -Autoreveal mode
  -Editcommands to select a page
  -Configurationprogram available to set parameters to everyones particular wishes.

For both programs a complete manual is written. You can order the software at the usual address. Write for more information to the editor of DE 6502 KENNER.

If you take the trouble to come to a national meeting, you're be able to make a free copy.



```
*************************************************************
ONDERW : HOE HAAL JE  MET DISKDOCTOR EEN DIRECTORY  TERUG?
SYSTEEM: DOS65                 AUTEUR: Bram de Bruine, Holland
*************************************************************
```

HOW TO RECOVER A DIRECTORY WITH DISKDOCTOR?

DOS65 vernietigt soms een subdirectory. Dit is erg lastig, maar het is erg eenvoudig om de subdir terug te halen. Een korte beschrijving.

Op Track 0, sector 1 staat de systeemsector. Adres 32-3F geeft aan waar de subdirectories staan. Is adres 32-3F gevuld met nullen, dan zijn er geen subdirectories. Om een subdirectorie terug te halen, moet men gaan zoeken naar het Track/Sector-nummer van die directory. Men zoekt in feite naar de verzameling filenamen die met een DIR dirspec/ op het scherm geprint worden. Heeft men die gevonden, dan staat bij diskdoctor onder aan het scherm op welke track en welke sector men zich bevind. Deze moeten ingevuld worden in de systeemsector (hexadecimaal).
VOORBEELD: Alle directories zijn verdwenen. 32-3F zijn gevuld met nullen. Met "+" zoekt men tot het volgende verschijnt: (Hee! dat zijn de files die ik mis)
VIDITEL.MAC
ASTRID.MAC
MCONFIG.MAC
DIAL.MAC
Deze informatie bevindt zich bijvoorbeeld op Track 0, Sector 6. In de systeemsector wijzigen we nu:
32: 00 (TRack)
33: 06 (SeCtor)
en de subdirectory is weer hersteld! Was 32/33 niet gevuld met nullen, dan neemt men het eerstvolgende paar binnen 32-3F dat 00 bevat.
OEFEN EERST OP EEN DISK MET GARBAGE!

Copy destroys sometimes subdirectories. To repair the disk, look for the filenames of the subdir on Tr. 0. If you found them, write the Tr/Sc number on the locations 32-3F, e.g. subdir1 = 32/33, subdir2 = 34/35, etc, on Tr. 0, Sc 1. Enter numbers in hex. Use diskdoctor. 00 means: empty subdir. e.g. 34/35=00/00, no subdir2 exists.
FIRST TEST ON A DISK WITH GARBAGE.

```
*************************************
*  H-CODE CALCULATOR FOR JUNIOR  *
*************************************
```

By: M. Nelissen, Belgium

Many autotests of micro-systems, running at the power-on, use H-codes. These checksums (H-codes) are mostly resident at the first or last locations of the firmware (e)proms. Generally they are calculated by a kind of polynome.
In this program i´ve used already existent subroutines in PM, TM and Disassem/Eprutl eproms of the extended JUNIOR-computer. With this program you can make a table with the H-codes of your own (e)proms.

```
0200:                  HCODE  ORG    $0200
0210:
0220:                  * EXISTENT SUBROUTINES *
0230:
0240:          4B 0C   CHKSUM *      $0C4B   resident in TM
0250:          5F 10   LABJUN *      $105F   warm start  PM
0260:          E8 11   CRLF   *      $11E8   carr.ret/l.feed
0270:          68 12   RESIN  *      $1268   0 in databuffer
0280:          8F 12   PRBYTE *      $128F   print byte
0290:          34 13   PRCHA  *      $1334   print character
0300:          87 13   INPAR  *      $1387   input params
0310:          F2 FA   PRINT  *      $FAF2
0320:          C5 FD   CHCK   *      $FDC5
0330:
0340:                  * EXISTENT POINTERS *
0350:
0360:          FA 00   POINTL *      $00FA
0370:          FB 00   POINTH *      $00FB
0380:          63 1A   PARAL  *      $1A63
0390:          64 1A   PARAH  *      $1A64
0400:          65 1A   PARBL  *      $1A65
0410:          66 1A   PARBH  *      $1A66
0420:          6E 1A   CHKL   *      $1A6E
0430:          6F 1A   CHKH   *      $1A6F
0440:          7C 1A   BRKT   *      $1A7C
0450:
0460: 0200 A9 5F       INIT   LDAIM  $5F
0470: 0202 A0 10              LDYIM  $10
0480: 0204 8D 7C 1A           STA    BRKT    set breakpointr
0490: 0207 8C 7D 1A           STY    BRKT    +01
0500: 020A A9 0C       PRTXTA LDAIM  $0C
0510: 020C 20 34 13           JSR    PRCHA   clear screen
0520: 020F 20 FA 02           JSR    ASTER   25 asterisks
0530: 0212 20 E8 11           JSR    CRLF
0540: 0215 20 F2 FA           JSR    PRINT   print title
0550: 0218 2A          TXTA   =      ´*
0560: 0219 2A                 =      ´*
0570: 021A 2A                 =      ´*
0580: 021B 20                 =      ´
0590: 021C 48                 =      ´H
0600: 021D 2D                 =      ´-
0610: 021E 43                 =      ´C
0620: 021F 4F                 =      ´O
0630: 0220 44                 =      ´D
0640: 0221 45                 =      ´E
0650: 0222 20                 =      ´
0660: 0223 43                 =      ´C
0670: 0224 41                 =      ´A
0680: 0225 4C                 =      ´L
0690: 0226 43                 =      ´C
0700: 0227 55                 =      ´U
0710: 0228 4C                 =      ´L
0720: 0229 41                 =      ´A
0730: 022A 54                 =      ´T
0740: 022B 4F                 =      ´O
0750: 022C 52                 =      ´R
0760: 022D 20                 =      ´
0770: 022E 2A                 =      ´*
0780: 022F 2A                 =      ´*
0790: 0230 2A                 =      ´*
0800: 0231 03                 =      $03
0810: 0232 20 E8 11           JSR    CRLF
0820: 0235 20 FA 02           JSR    ASTER   25 asterisks
0830: 0238 20 E8 11           JSR    CRLF
0840: 023B 20 F2 FA           JSR    PRINT   print text B
0850: 023E 0A          TXTB   =      $0A
0860: 023F 42                 =      ´B
0870: 0240 52                 =      ´R
0880: 0241 4B                 =      ´K
0890: 0242 20                 =      ´
0900: 0243 3D                 =      ´=
0910: 0244 20                 =      ´
0920: 0245 52                 =      ´R
0930: 0246 45                 =      ´E
0940: 0247 54                 =      ´T
0950: 0248 55                 =      ´U
0960: 0249 52                 =      ´R
0970: 024A 4E                 =      ´N
0980: 024B 20                 =      ´
0990: 024C 54                 =      ´T
1000: 024D 4F                 =      ´O
1010: 024E 20                 =      ´
1020: 024F 50                 =      ´P
1030: 0250 4D                 =      ´M
1040: 0251 03                 =      $03
```

```
1050: 0252 20 E8 11    NXTCAL JSR    CRLF
1060: 0255 20 F2 FA           JSR    PRINT
1070: 0258 0A                 =      $0A
1080: 0259 47                 =      ´G
1090: 025A 49                 =      ´I
1100: 025B 56                 =      ´V
1110: 025C 45                 =      ´E
1120: 025D 20                 =      ´
1130: 025E 46                 =      ´F
1140: 025F 49                 =      ´I
1150: 0260 52                 =      ´R
1160: 0261 53                 =      ´S
1170: 0262 54                 =      ´T
1180: 0263 2C                 =      ´,
1190: 0264 4C                 =      ´L
1200: 0265 41                 =      ´A
1210: 0266 53                 =      ´S
1220: 0267 54                 =      ´T
1230: 0268 20                 =      ´
1240: 0269 4D                 =      ´M
1250: 026A 45                 =      ´E
1260: 026B 4D                 =      ´M
1270: 026C 4F                 =      ´O
1280: 026D 52                 =      ´R
1290: 026E 59                 =      ´Y
1300: 026F 20                 =      ´
1310: 0270 41                 =      ´A
1320: 0271 44                 =      ´D
1330: 0272 44                 =      ´D
1340: 0273 52                 =      ´R
1350: 0274 45                 =      ´E
1360: 0275 53                 =      ´S
1370: 0276 53                 =      ´S
1380: 0277 20                 =      ´
1390: 0278 3A                 =      ´:
1400: 0279 20                 =      ´
1410: 027A 03                 =      $03
1420: 027B 20 68 12           JSR    RESIN   reset inputbufs
1430: 027E 20 87 13           JSR    INPAR   reset 2 adresse
1440: 0281 30 87              BMI    PRTXTA  repeat if not
1450: 0283 20 C5 FD           JSR    CHCK    done properly
1460: 0286 90 82              BCC    PRTXTA  repeat if last
1470: 0288 A9 00              LDAIM  $00     < 1st address
1480: 028A 8D 6E 1A           STA    CHKL    reset checksum
1490: 028D 8D 6F 1A           STA    CHKH
1500: 0290 AD 63 1A           LDA    PARAL
1510: 0293 AC 64 1A           LDY    PARAH
1520: 0296 85 FA              STAZ   POINTL  prepare work-
1530: 0298 84 FB              STYZ   POINTH  pointers
1540: 029A A0 00       CALCUL LDYIM  $00     start calcula-
1550: 029C B1 FA              LDAIY  POINTL  tion
1560: 029E 20 4B 0C           JSR    CHKSUM
1570: 02A1 E6 FA              INCZ   POINTL
1580: 02A3 D0 02              BNE    CNT
1590: 02A5 E6 FB              INCZ   POINTH
1600: 02A7 A5 FA       CNT    LDAZ   POINTL  restore PARA
1610: 02A9 8D 63 1A           STA    PARAL
1620: 02AC A5 FB              LDAZ   POINTH
1630: 02AE 8D 64 1A           STA    PARAH
1640: 02B1 20 D8 02           JSR    NXT
1650: 02B4 B0 E4              BCS    CALCUL  if not, continue
1660: 02B6 20 F2 FA           JSR    PRINT   else give result
1670: 02B9 0D          TXTC   =      $0D
1680: 02BA 0A                 =      $0A
1690: 02BB 0A                 =      $0A
1700: 02BC 48                 =      ´H
1710: 02BD 2D                 =      ´-
1720: 02BE 43                 =      ´C
1730: 02BF 4F                 =      ´O
1740: 02C0 44                 =      ´D
1750: 02C1 45                 =      ´E
1760: 02C2 20                 =      ´
1770: 02C3 3D                 =      ´=
1780: 02C4 20                 =      ´
1790: 02C5 03                 =      $03
1800: 02C6 AD 6F 1A           LDA    CHKH
1810: 02C9 20 8F 12           JSR    PRBYTE
1820: 02CC AD 6E 1A           LDA    CHKL
1830: 02CF 20 8F 12           JSR    PRBYTE
1840: 02D2 20 E8 11           JSR    CRLF
1850: 02D5 4C 52 02           JMP    NXTCAL
1860: 02D8 18          NXT    CLC
1870: 02D9 AD 63 1A           LDA    PARAL
1880: 02DC 69 01              ADCIM  $01
1890: 02DE 8D 63 1A           STA    PARAL
1900: 02E1 AD 64 1A           LDA    PARAH
1910: 02E4 69 00              ADCIM  $00
1920: 02E6 8D 64 1A           STA    PARAH
1930: 02E9 B0 0C              BCS    NXTB    branch if $FFFF
1940: 02EB 38                 SEC            is crossed
1950: 02EC AD 65 1A           LDA    PARBL   workpointer =
1960: 02EF E5 FA              SBCZ   POINTL
1970: 02F1 AD 66 1A           LDA    PARBH
1980: 02F4 E5 FB              SBCZ   POINTH  carry depends on
                                             PARB minus POINT
1990: 02F6 60          NXTA   RTS            or on crossing
                                             $FFFF, the memo-
2000: 02F7 18          NXTB   CLC            ry boundary
2010: 02F8 90 FC              BCC    NXTA
2020:
```

```
2030:                  * SUB TO PRINT 25 ASTERISKS *
2040:
2050: 02FA A9 2A     ASTER  LDAIM  `*
2060: 02FC A0 19            LDYIM  $19
2070: 02FE 20 34 13  CONT   JSR    PRCHA
2080: 0301 88               DEY
2090: 0302 D0 FA            BNE    CONT
2100: 0304 60               RTS
```

AVAILABLE FOR ELEKTOR'S OCTOPUS/EC65 COMPUTER
ONLY 40 TRACKS FORMAT SS, DD

WORDPROCESSOR VERSION 3.0 (DISK 1)
LOYS 3.1 XTRA's INTEGRATED
INSTALLATION PROGRAM (DISK 2)
SMALL MANUAL (PPWS)

Because OHIO-DOS is part of the system on bootable disks and is not placed in the public domain you must prove you bought it yourself, by sending a copy of the invoice to the editorial's office, before we can deliver the diskettes.

Wordprocessor V3.0 is a powerfull, fast full-screen-editor, or more explicit: a full FILE editor, since it allows 'cruising' around from top to bottom of the file (or even more than one file at a time).
By means of the 'Installation program' on the other diskette, this editor can now be adapted to different dos versions and different machine-configurations. The only requirements are: OHIO-Dos, a 65xxx CPU and a 6845 (6545) CRT-controller. The Installation program allows you to adapt the control-keys to your keyboard and the printer control codes to your printer.
Specifications:
Cursor up/down/left/right/home/1 screen up/1 screen down/ to front of line/to rear of line/toggle writeover/insert /write graphic charater/delete char right/delete char left/ delete line/insert line;
Put file on disk, Load file(s) from disk, Erase filename from directory, Show directory, Select drive, Goto Dos, Status information, Reserve extra tracks, Goto monitor, Help menu, Hard copy, Columns print, Word wrap, Format, Right margin justification, Search and Replace, Goto string, Text copy / insert copy, Kill file, WP/asm, ASM/wp conversion.

```
 1> Directory                      21> Kolorator
 2> Create a new file              22> EDitor-MOnitor
 3> Change a file name             23>
 4> Delete a file                  24> WORDPROCESSOR INSTALL
 5> Create blank diskette          25>
 6> Create diskette with files     26>
 7> Create buffer space for files  27>
 8> Dual disk drive copier         28>
 9> Enter OS-65D system            29>
10> ASS114 (not installed)         30>
11> Word Processor V3.0            31>
12> Basicode Processor             32>
13> Resequencer (RSEQ)             33>
14> Merge basic files              34>
15> Change basics workspace        35>
16> Garbag Collector               36>
17> Arcustangens function          37>
18> Trace basics lines             38>
19> Return to Monitor              39>
20> Track zero r/w                 40>
```

DIRECTORY WORDPROCESSOR V3.0 DISKETTE

V3.3/1	0-0	V3.3/2	1-1	DIR/BO	12-12
BASIC	2-5	B/5V/3	6-6	EDMO	7-9
KOLORA	10-11	V3.3/4	13-13	V3.3/5	14-14
BEXEC*	15-18	GARBAG	19-21	ASS114	22-25
SCRTCH	26-26	WP2.P	27-30	W/RQ/M	31-31
CHANGE	32-33	MERGE	34-34	BSCOD/1	35-35
BSCOD/2	36-36	COPIER	37-37	ATNENB	38-38
COM/TO	39-39				

To order the diskette send 2 diskettes with labels and R/W prots and pay the price as mentioned here:

Europe : Hfl. 87,00	Outside Europe : Hfl. 104,50
Members: Hfl. 37,00	Members: Hfl. 54,50

Members in Holland and Belgium paying on postgiro 841433 of W.L. van Pelt at Krimpen a.d. IJssel only pay Hfl. 27,50.
We also accept Eurocheques. Don't forget to put your number on the back!

Send your order to the editorial office.
All prices including pacckages and postages etc.
We accept no responsibility for damages etc. during transports.

* Patch on Dr. Tietsch's Copier Program *

P. Lindstrøm & L. Rasmussen, Denmark

When you try to copy a disk with no data on track zero, with the org. OHIO-copier, then the system will crash. In his rev. prg. (issue 43-44) Dr. Tietsch tried to remedy that with a check for data on track zero. Unfortunately, this check is not safe. Every once in a while it goes wrong and skip writing track zero to the copy - even when there is a data to write. Then the pointers are set on the track zero data, which goes to track 1, and the data from track 1 goes to track 2, and so on .. This will not be noticed, until you try to use the copy - or you compare the disks.

Solution 1: Remove lines:

```
6390: 4014 08              PHP       ;data on tr 0?
6630: 405E 28    ZERTRC PLP       ;was there any data on
6640: 405F 9003            BCC LAB20;tr 0?
```

Or put NOP in these four addresses. Now you cannot copy disks with empty track zeros, as in the org. Copier.

Solution 2: if you insist on copying empty track zeros, remove lines as in solution 1, and add-in lines:

```
3911: 3E37 F029            BEQ RETA ;data on tr 0?
3601: 3E01 A525            LDA ZRO5
3602: 3E03 F01B            BEQ LAB08;data on tr 0?
```

Now the system will not crash on copying empty track zeros. BUT observe, that if you send such a copy to some one, maybe he will not be able to copy your disk, unless he put something in track zero.
So the solution must be: Use solution 1, and always put data on track zero, - f.ex. Coen Boltjes' message in issue 48 page 49.

* Printer problems with the Octopus *

Maarten den Hertog, The Netherlands.

Mijn Octopus 65 bezit geen 'BUSY'-lijn en de communicatie tussen de printer en de computer liep dan ook hopeloos vast. Hieronder volgt een oplossing die ik overigens heb opgediept uit een oud nummer van Elektuur. Maar niet iedereen zal daar de beschikking over hebben, vandaar.

In principe zou de combinatie computer/printer meteen goed kunnen funktioneren. Het kan echter gebeuren dat de printer gek begint te doen als er data naar de printer wordt gestuurd terwijl deze nog bezig is. In dat geval kan de volgende oplossing worden gebruikt.

1. De 'printer busy'-aansluiting van de Centronics-steker wordt direkt verbonden met de 'clear to send'-aansluiting (CTS) van de seriëele uitgangspoort (ACIA) van de Octopus 65.
2. De CTS-lijn krijgt een pull-down weerstand van 4k7 naar de nul. Deze lijn wordt dan "0" als de printer uitgeschakeld is. Op deze manier kan men ook zonder printer verder werken.
3. In de diverse programma's moeten de I/O-opdrachten dan wel aangepast worden. Bijvoorbeeld in het tekstverwerkingsprogramma WP2.0 moet $149B veranderd worden van 08 in 0D.

Ik heb deze oplossing naar volle tevredenheid toegepast op mijn eigen systeem OCTOPUS - ERICSSON. Ik kan mij voorstellen dat er misschien handiger oplossingen bedacht zijn, die wil ik dan ook graag weten.

Lately I bought a printer for my Octopus 65, but unfortunately it was not working. I only could print one sentence, and then a mysterious printer hang would occur. The problem is that the centronics connector of the Octopus has no 'BUSY-line', so the computer is still sending data even when the printer is not ready to accept them.
The solution is quit simpel. You connect the 'BUSY-line' from the printer directly with the 'clear to send' (CTS) of the ACIA of the computersystem. The result is that the computer will not send any data to the printer whenever the printer is not ready to accept them. After that you connect the CTS-line with a 4k7 resistor to the zero. This will then be "0" when the printer is not connected. In this way you can also work without a printer.
I can imagine someone has better ideas. Please send it to the editorial office.

```
****************************************
*  DATABANK PROGRAM FOR THE JUNIOR  *
****************************************
```

Author: M. Lameij, The Netherlands
Transl: Frank Bens, The Netherlands

>>> THEORY OF OPERATION <<<

This program makes it possible to fill the RAM-memory with text, like for a catalog of cassette tapes or records. The text will be stored in blocks of 1K RAM = 1 full screen. Empty character places at the end of a line will be filled automatically with spaces.

The next commands can be used :

] = Insert of a screen page. The concerning page-vector is looked up in a table. The screen-pages are numbered starting from decimal 1 and to be found in RAM starting from address $2000.
^ = Send a page to screen, which number is previously given by "]".
[= New start. This means, the command menu will appear on the screen and a new choise can be made.
* = Back to JUNIOR monitor.
@ = Search-possibility on screen with the choise jumping half or full lines. Every time this command is given, the cursor will move over the screen.
With the command :
= The cursor will step to the right. This way you can search for a certain character and correct it. The command "^" will store this correction into RAM and display the new page.
$ = Start of new data in a screen-page.

When you are inserting a line of text and you have made a typing error, it is possible to correct this error by using the key <BACKSPACE> to go backwards on this line. When you are ready with a line and hit the key <LINEFEED> automatically a <RETURN> will be given. The EOT-character $03 will automaticalle be generated at the last position of the last line of the screen. The sign can also be placed on another position of the screen when there are less lines needed, by typing <CTRL-C>. There are no secureties build in, therefore be carefull when you are using this program. The possibility exists that everything will go wrong when you hit a wrong key. Also a L.S.-unit can be connected to PB0, a short beeb will then be heard, when the computer is ready with transmitting a page to the screen.

```
0720:                MONITOR SUBROUTINES
0730:
0740:          5F 10 LABJUN *    $105F WARM START JUNIOR
0750:          E8 11 CRLF   *    $11E8 CARR.RET/LINEFEED
0760:          6F 12 HEXNUM *    $126F ASCII TO HEX
0770:          AE 12 RECCHA *    $12AE CHAR FROM KEYB.
0780:          34 13 PRCHA  *    $1334 CHAR TO SCREEN
0790:                VIA ADDRESSES
0800:          00 18 ORB    *    $1800 DATA REGISTER
0810:          02 18 DDRB   *    $1802 DIRECTION REG.
0820:
0830:                PIA ADDRESSES
0840:          F7 1A TIMER  *    $1AF7
0850:          D5 1A RDFLAG *    $1AD5
0860:
0870:                ZERO PAGE ADDRESSES
0880:          00 00 ADDONE *    $0000 CONSTANTS
0890:          01 00 COUNTR *    $0001 CHAR.COUNTER
0900:          02 00 CHARAC *    $0002 CHAR. BUFFER
0910:          03 00 PAGCTR *    $0003 PAGE COUNTER
0920:          70 00 DELAY  *    $0070 DELAY COUNTER
0930:          F8 00 INL    *    $00F8 INPUT BUFFER
0940:
0950:                PROGRAM ADDRESSES
0960:          60 02 WRTPTR *    $0260 WRITE POINTER
0970:          66 02 RDPTR  *    $0266 READ POINTER
0980:          8F 02 SCREND *    $028F SCREEN-END PNTR
0990:          56 03 HALFFU *    $0356 HALF/FULL LINE
1000:
1010:                *** PROGRAM START ***
1020:
1030: 0200           DATA   ORG   $0200
1040: 0200 A9 0C     INPUT  LDAIM $0C
1050: 0202 20 F7 03         JSR   CLRSCR CLEAR SCREEN
1060: 0205 A0 00            LDYIM $00
1070: 0207 20 86 03         JSR   TEXT   PRINT MENU
1080: 020A 20 C9 03         JSR   HALFLI HALF/FULL LINE ?
1090: 020D A5 03     INPUTA LDA   PAGCTR GET PAGE COUNTER
1100: 020F 8D 60 02         STA   WRTPTR
1110: 0212 20 53 02         JSR   ZEROST RESET COUNTERS
1120: 0215 4C D0 02  AGAIN  JMP   KEY    WAIT FOR A CHAR
1130: 0218 85 02     INPUTB STA   CHARAC SAVE CHARACTER
1140: 021A 20 48 02  INPUTC JSR   INCREM CHAR.COUNTER + 1
1150: 021D A5 02            LDA   CHARAC GET CHARACTER
1160: 021F 20 5C 02         JSR   WRITE  STORE IN PAGE
1170: 0222 4C 15 02         JMP   AGAIN  CONTINUE
1180:
1190: 0225 A9 0C     PRINT  LDAIM $0C
1200: 0227 20 F7 03         JSR   CLRSCR CLEAR SCREEN
1210: 022A A5 03            LDA   PAGCTR GET PAGE COUNTER
1220: 022C 8D 66 02         STA   RDPTR
1230: 022F 20 53 02         JSR   ZEROST RESET COUNTERS
1240: 0232 20 0E 03  PRINTA JSR   INCPRI CHAR.COUNTER + 1
1250: 0235 20 62 02         JSR   READ   PRINT PAGES
1260: 0238 C9 03            CMPIM $03    UNTIL EOT-SIGN
1270: 023A F0 06            BEQ   BELL   RING THE BELL
1280: 023C 20 34 13         JSR   PRCHA  PRINT CHARACTER
1290: 023F 4C 32 02         JMP   PRINTA CONTINUE
1300: 0242 20 05 04  BELL   JSR   SOUND  RING THE BELL
1310: 0245 4C 0D 02         JMP   INPUTA WAIT FOR COMMAND
1320:
1330: 0248 A5 01     INCREM LDA   COUNTR GET CHAR.COUNTER
1340: 024A 18               CLC
1350: 024B 65 00            ADC   ADDONE ADD ONE
1360: 024D 85 01            STA   COUNTR STORE IT BACK
1370: 024F 20 68 02         JSR   LINEFU TEST ON FULL LINE
1380: 0252 60               RTS
1390:
1400: 0253 A9 FF     ZEROST LDAIM $FF
1410: 0255 85 01            STA   COUNTR FILL CHAR.COUNTER
1420: 0257 A9 01            LDAIM $01
1430: 0259 85 00            STA   ADDONE FILL CONSTANTS
1440: 025B 60               RTS
1450:
1460: 025C A4 01     WRITE  LDY   COUNTR
1470: 025E 99 00 1F         STAAY $1F00  WRITE CHARACTER
1480: 0261 60               RTS
1490:
1500: 0262 A4 01     READ   LDY   COUNTR
1510: 0264 B9 00 1F         LDAAY $1F00  READ CHARACTER
1520: 0267 60               RTS
1530:
1540: 0268 A5 01     LINEFU LDA   COUNTR
1550: 026A C9 FF            CMPIM $FF    FULL LINE ?
1560: 026C D0 25            BNE   LINEB  IF NOT, NEXT LINE
1570: 026E AD 60 02         LDA   WRTPTR GET WRITE POINTER
1580: 0271 29 0F            ANDIM $0F    MASK BITS
1590: 0273 C9 03            CMPIM $03    IS IT A 3 ? IF SO
1600: 0275 F0 0E            BEQ   LINEA  END OF SCREEN
1610: 0277 C9 07            CMPIM $07    IS IT A 7 ? IF SO
1620: 0279 F0 0A            BEQ   LINEA  END OF SCREEN
1630: 027B C9 0B            CMPIM $0B    IS IT A B ? IF SO
1640: 027D F0 06            BEQ   LINEA  END OF SCREEN
1650: 027F C9 0F            CMPIM $0F    IS IT A F ? IF SO
1660: 0281 F0 02            BEQ   LINEA  END OF SCREEN, IF
1670: 0283 D0 18            BNE   LINEC  NOT STAY ON SCREEN
1680: 0285 AD 60 02  LINEA  LDA   WRTPTR GET WRITE POINTER
1690: 0288 8D 8F 02         STA   SCREND SAVE IN SCREEN-END
1700: 028B A9 03            LDAIM $03    SAVE EOT-SIGN
1710: 028D 8D FF 23         STA   $23FF  IN PAGE AND
1720: 0290 4C 5F 10         JMP   LABJUN RETURN TO MONITOR
1730: 0293 C9 00     LINEB  CMPIM $00    PAGE FULL ?
1740: 0295 D0 06            BNE   LINEC  IF NOT, CONTINUE
1750: 0297 EE 60 02         INC   WRTPTR IF SO, INCREASE
1760: 029A EE 66 02         INC   RDPTR  WRITE & READ POIN-
1770: 029D 60        LINEC  RTS          TERS
1780:
1790: 029E A9 3D     FILLLI LDAIM $3D    CALC. LINE LENGTH
1800: 02A0 38               SEC          IN USE
1810: 02A1 E5 01            SBC   COUNTR
1820: 02A3 10 13            BPL   FILLA
1830: 02A5 A9 7D            LDAIM $7D
1840: 02A7 38               SEC
1850: 02A8 E5 01            SBC   COUNTR
1860: 02AA 10 0C            BPL   FILLA
1870: 02AC A9 BD            LDAIM $BD
1880: 02AE 38               SEC
1890: 02AF E5 01            SBC   COUNTR
1900: 02B1 10 05            BPL   FILLA
1910: 02B3 A9 FD            LDAIM $FD
1920: 02B5 38               SEC
1930: 02B6 E5 01            SBC   COUNTR
1940: 02B8 AA        FILLA  TAX
1950: 02B9 A9 20     FILLC  LDAIM $20    FILL LINE
1960: 02BB 85 02            STA CHARAC   WITH SPACES
1970: 02BD E6 01            INC   COUNTR
1980: 02BF A5 02            LDA   CHARAC
1990: 02C1 20 5C 02         JSR   WRITE
2000: 02C4 A5 01            LDA   COUNTR
2010: 02C6 C9 FE            CMPIM $FE
2020: 02C8 F0 03            BEQ   FILLB
2030: 02CA CA               DEX
2040: 02CB 10 EC            BPL   FILLC
2050: 02CD 4C 1A 02  FILLB  JMP   INPUTC
2060:
2070: 02D0 20 AE 12  KEY    JSR   RECCHA WAIT FOR A CHAR.
2080: 02D3 C9 5B            CMPIM '[     IS IT A [ ?
2090: 02D5 D0 03            BNE   KEYA   IF NOT, NEXT
2100: 02D7 4C 00 02         JMP   INPUT  IF SO, TO START
2110: 02DA C9 2A     KEYA   CMPIM '*     IS IT A * ?
2120: 02DC D0 03            BNE   KEYB   IF NOT, NEXT
2130: 02DE 4C 5F 10         JMP   LABJUN IF SO, TO MONITOR
2140: 02E1 C9 5E     KEYB   CMPIM '^     IS IT A ^ ?
2150: 02E3 D0 03            BNE   KEYC   IF NOT, NEXT
2160: 02E5 4C 25 02         JMP PRINT    IF SO, PRINT PAGES
2170: 02E8 C9 0A     KEYC   CMPIM $0A    IS IT A LINEFEED ?
2180: 02EA D0 08            BNE   KEYD   IF NOT, NEXT
2190: 02EC A9 0D            LDAIM $0D    IF SO, PRINT ALSO
2200: 02EE 20 34 13         JSR   PRCHA  A RETURN
2210: 02F1 4C 9E 02         JMP   FILLLI AND FILL THE LINE
2220: 02F4 C9 08     KEYD   CMPIM $08    IS IT A BACKSPACE?
2230: 02F6 D0 05            BNE   KEYE   IF NOT,NEXT
2240: 02F8 C6 01             DEC    COUNTR IF SO, CHAR.CNTR-1
2250: 02FA 4C D0 02          JMP    KEY
2260: 02FD C9 40     KEYE    CMPIM  $40    IS IT A @ ?
2270: 02FF D0 03             BNE    KEYF   IF NOT, NEXT
2280: 0301 4C 20 03          JMP    CORREC IF SO, CORRECTION
2290: 0304 C9 5D     KEYF    CMPIM  ´]     IS IT A ] ?
2300: 0306 D0 03             BNE    KEYG   IF NOT, NEXT
2310: 0308 4C 95 03          JMP    PAGE   IF SO, PAGENUMBER?
2320: 030B 4C 18 02  KEYG    JMP    INPUTB SAVE CHARACTER
2330:
2340: 030E A5 01     INCPRI  LDA    COUNTR GET CHAR.COUNTER
2350: 0310 18                CLC
2360: 0311 65 00             ADC    ADDONE ADD ONE
2370: 0313 85 01             STA    COUNTR SAVE IT
2380: 0315 C9 00             CMPIM  $00    COUNTER EMPTY ?
2390: 0317 D0 06             BNE    INCEND IF NOT, CONTINUE
2400: 0319 EE 60 02          INC    WRTPTR IF SO, WRITE PNTR+1
2410: 031C EE 66 02          INC    RDPTR  AND READ POINTER+1
2420: 031F 60        INCEND  RTS
2430:
2440: 0320 A9 1C     CORREC  LDAIM  $1C
2450: 0322 20 F7 03          JSR    CLRSCR CURSOR HOME
2460: 0325 A5 03             LDA    PAGCTR GET PAGE COUNTER
2470: 0327 8D 60 02          STA    WRTPTR FILL WRITE POINTER
2480: 032A 8D 66 02          STA    RDPTR  AND READ POINTER
2490: 032D 20 53 02          JSR    ZEROST RESET COUNTERS
2500: 0330 20 AE 12  KEYX    JSR    RECCHA WAIT FOR A CHAR.
2510: 0333 C9 40             CMPIM  $40    IS IT A @ ?
2520: 0335 F0 19             BEQ    CORRA
2530: 0337 C9 23             CMPIM  ´#     IS IT A # ?
2540: 0339 F0 32             BEQ    CORRB
2550: 033B C9 5E             CMPIM  $^     IS IT A ^ ?
2560: 033D F0 44             BEQ    CORRC
2570: 033F C9 08             CMPIM  $08    IS IT A BACKSPACE?
2580: 0341 F0 3B             BEQ    CORRD
2590: 0343 85 02             STA    CHARAC SAVE CHARACTER
2600: 0345 20 0E 03          JSR    INCPRI INCREASE COUNTER
2610: 0348 A5 02             LDA    CHARAC GET CHARACTER
2620: 034A 20 5C 02          JSR    WRITE  AND PRINT IT
2630: 034D 4C 30 03          JMP    KEYX   WAIT FOR A CHAR.
2640: 0350 A9 08     CORRA   LDAIM  $08    DELETE INSERTED
2650: 0352 20 34 13          JSR    PRCHA  CHARACTER
2660: 0355 A2 3F             LDXIM  $3F
2670: 0357 20 0E 03  CORRF   JSR    INCPRI PRINT HALF
2680: 035A 20 62 02          JSR    READ   OR FULL LINE
2690: 035D C9 03             CMPIM  $03    SCREEN END ?
2700: 035F F0 09             BEQ    CORRE  IF SO, STOP
2710: 0361 20 34 13          JSR    PRCHA  IF NOT, PRINT
2720: 0364 CA                DEX           AS LONG AS
2730: 0365 10 F0             BPL    CORRF  LINE NOT FULL AND
2740: 0367 4C 30 03          JMP    KEYX   WAIT FOR A CHAR.
2750: 036A 4C 15 02  CORRE   JMP    AGAIN
2760: 036D 20 0E 03  CORRB   JSR    INCPRI CHAR.COUNTER + 1
2770: 0370 A9 08             LDAIM  $08    DELETE INSERTED
2780: 0372 20 34 13          JSR    PRCHA  CHARACTER AND
2790: 0375 20 62 02          JSR    READ   PRINT 1 CHAR.
2800: 0378 20 34 13          JSR    PRCHA  FROM PAGE
2810: 037B 4C 30 03          JMP    KEYX   WAIT FOR A CHAR.
2820: 037E C6 01     CORRD   DEC    COUNTR CHAR.COUNTER - 1
2830: 0380 4C 30 03          JMP    KEYX   WAIT FOR A CHAR.
2840: 0383 4C 25 02  CORRC   JMP    PRINT  PRINT PAGES
2850:
2860: 0386 B9 23 04  TEXT    LDAAY  MESSAG GET TEXT
2870: 0389 C9 03             CMPIM  $03    EOT-SIGN ?
2880: 038B F0 07             BEQ    TEXEND IF SO, STOP
2890: 038D 20 34 13          JSR    PRCHA  IF NOT, PRINT
2900: 0390 C8                INY
2910: 0391 4C 86 03          JMP    TEXT   NEXT CHARACTER
2920: 0394 60        TEXEND  RTS
2930:
2940: 0395 A9 00     PAGE    LDAIM  $00
2950: 0397 85 FB             STA    INL    CLEAR INPUT BUFFER
2960: 0399 A9 08             LDAIM  $08    DELETE INSERTED
2970: 039B 20 34 13          JSR    PRCHA  CHARACTER
2980: 039E A0 B6     PAGEA   LDYIM  $B6    PRINT
2990: 03A0 20 B6 03          JSR    TEXT   TEXT STRING
3000: 03A3 20 AE 12  PAGEB   JSR    RECCHA WAIT FOR A CHAR.
3010: 03A6 C9 0D             CMPIM  $0D    IS IT A RETURN ?
3020: 03A8 F0 0B             BEQ    PAGEC  IF SO, PRINT PAGE
3030: 03AA C9 24             CMPIM  ´$     IS IT A $ ?
3040: 03AC F0 0D             BEQ    PAGED  IF SO, BACK INPUT
3050: 03AE 20 6F 12          JSR    HEXNUM TRANSFER NUMERIC
3060: 03B1 D0 EB             BNE    PAGEA  ERROR, TRY AGAIN
3070: 03B3 F0 EE             BEQ    PAGEB  NEXT CHARACTER
3080: 03B5 20 C1 03  PAGEC   JSR    VECTOR GET PAGE VECTOR
3090: 03B8 4C 25 02          JMP    PRINT  AND PRINT
3100: 03BB 20 C1 03  PAGED   JSR    VECTOR GET PAGE VECTOR
3110: 03BE 4C 0D 02          JMP    INPUTA WAIT FOR A CHAR.
3120:
3130: 03C1 A4 F8     VECTOR  LDY    INL    GET BUFFER
3140: 03C3 B9 EA 04          LDAAY  VECTAB FETCH PAGE VECTOR
3150: 03C6 85 03             STA    PAGCTR AND SAVE IT
3160: 03C8 60                RTS
3170:
3180: 03C9 20 E8 11  HALFLI  JSR    CRLF   START ON NEW LINE
3190: 03CC A0 BF             LDYIM  $BF    PRINT
3200: 03CE 20 86 03          JSR    TEXT   TEXT STRING
3210: 03D1 20 AE 12          JSR    RECCHA WAIT FOR A CHAR.
3220: 03D4 C9 4A             CMPIM  ´J     IS IT A J ?
3230: 03D6 F0 06             BEQ    HALFA  IF SO, CONTINUE
3240: 03D8 C9 4E             CMPIM  ´N     IS IT A N ?
3250: 03DA F0 0A             BEQ    HALFB  IF SO, CONTINUE
3260: 03DC D0 EB             BNE    HALFLI IF NOT, TRY AGAIN
3270: 03DE A9 1F     HALFA   LDAIM  $1F
3280: 03E0 8D 56 03          STA    HALFFU SET FOR HALF LINE
3290: 03E3 4C EB 03          JMP    HALFC
3300: 03E6 A9 3F     HALFB   LDAIM  $3F
3310: 03E8 8D 56 03          STA    HALFFU SET FOR FULL LINE
3320: 03EB 20 E8 11  HALFC   JSR    CRLF   START ON NEW LINE
3330: 03EE A0 9E             LDYIM  $9E    PRINT
3340: 03F0 20 86 03          JSR    TEXT   TEXT STRING
3350: 03F3 20 E8 11          JSR    CRLF   NEW LINE
3360: 03F6 60                RTS
3370:
3380: 03F7 20 34 13  CLRSCR  JSR    PRCHA  CLEAR
3390: 03FA A9 80             LDAIM  $80
3400: 03FC 8D F7 1A          STA    TIMER  DELAY
3410: 03FF 2C D5 1A  CLRA    BIT    RDFLAG
3420: 0402 10 FB             BPL    CLRA
3430: 0404 60                RTS
3440:
3450: 0405 A9 01     SOUND   LDAIM  $01
3460: 0407 8D 00 18          STA    ORB    SET DATA REGISTER
3470: 040A 8D 02 18          STA    DDRB   SET DIR. REGISTER
3480: 040D A9 7F             LDAIM  $7F    SET DELAY
3490: 040F 85 70             STA    DELAY
3500: 0411 EE 00 18  SOUNA   INC    ORB    SWITCH DATA
3510: 0414 A6 70             LDX    DELAY  REGISTER ON
3520: 0416 E8        SOUNB   INX           AND OFF
3530: 0417 D0 FD             BNE    SOUNB
3540: 0419 C6 70             DEC    DELAY  WAIT
3550: 041B C6 70             DEC    DELAY
3560: 041D 30 03             BMI    SOUNC
3570: 041F 4C 11 04          JMP    SOUNA
3580: 0422 60        SOUNC   RTS
3590:
3600:
3610: 0423           MESSAG  ORG    $0423
3620: 04EA           VECTAB  ORG    $04EA
```

```
                 *** TEXT STRINGS ***

01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

41 41 54 41 42 41 4E 4B 0A 0D 2D 2D 2D 2D 2D DATABANK  ----
2D 2D 2D 0A 0D 56 41 4C 49 44 20 43 4F 4D 4D ---  VALID COMM
41 4E 44 53 3A 0A 0D 5E 20 3D 20 50 52 49 4E ANDS:  ^ = PRIN
54 20 50 41 47 45 0A 0D 5B 20 3D 20 4E 45 57 T PAGE  [ = NEW
20 53 54 41 52 54 0A 0D 2A 20 3D 20 4D 4F 4E  START  * = MON
49 54 4F 52 0A 0D 5D 20 3D 20 50 41 47 45 20 ITOR  ] = PAGE
4E 55 4D 42 45 52 0A 0D 40 20 3D 20 53 45 41 NUMBER  @ = SEA
52 43 48 20 4C 49 4E 45 0A 0D 23 20 3D 20 43 RCH LINE  # = C
55 52 53 4F 52 2D 2D 3E 0A 0D 24 20 3D 20 4E URSOR-->  $ = N
45 57 20 44 41 54 41 0A 0D 03 53 54 41 52 54 EW DATA   START
20 57 49 54 48 20 50 41 47 45 20 4E 55 4D 42  WITH PAGE NUMB
45 52 03 20 50 41 47 45 3F 03 48 41 4C 46 20 ER  PAGE? HALF
4C 49 4E 45 20 53 45 41 52 43 48 20 2B 4A 2F LINE SEARCH (J/
4E 29 20 03 1F 1F 23 27 2B 2F 33 37 3B 3F 3F N)
3F 3F 3F 3F 3F 43 47 4B 4F 53 57 5B 5F 63 67
67 67 67 67 67 67 6B 6F

        IF YOU WISH YOU CAN EXTEND THIS TABLE
```

DISKETTES 80/40 TRS, SS,DD FOR ELEKTOR´S EC65/OCTOPUS
Because OHIO-DOS is part of the system on bootable disks and is not placed in the public domain you must prove you bought it yourself, by sending copy of the invoice, before we can deliver the diskettes. To order, mention the format and pay on postgiro 841433 of W.L. van Pelt, Krimpen aan den IJssel or with Eurocheque. In other cases bankcheque.

Bootable Malach disk with menu-driven **BASICODE**-routines. Send empty diskette with label and R/W-prot.
Europe : Hfl. 72,00   Outside Europe : Hfl. 89,00
Members: Hfl. 22,00   Members: Hfl. 39,00
Members in Holland and Belgium paying on postgiro 841433 only pay Hfl. 12,50. We also accept Eurocheques. Don´t forget to put your number on the back of it.

OCTOPUS-FORTH 1.2 (ONLY 80 TRS) specially build for Elektor´s Octopus/EC65 6502-computer as published in Elektor Germany and Elektuur Holland. It uses the OHIO-DOS which functions as a host for the FORTH-system. OCTOPUS-FORTH 1.2 is based on the model of the FORTH INTEREST GROUP as published in their Fig-Forth 6502 Assembly Source Listing (both of them can be ordered by our club). Several bugs in this model are removed and a lot of high level words are rewritten in code to increase speed.
Send two empty diskettes with label and R/W prots.
Europe : Hfl. 109,50   Outside Europe : Hfl. 126,50
Members: Hfl. 59,50   Members: Hfl. 76,50
Members in Holland and Belgium paying on postgiro 841433 only pay Hfl. 50,00. We also accept Eurocheques. Don´t forget to put your number on the back of it.

****

* OMEGA: THE DESKTOP MAINFRAME *

****

De redaktie was geinteresseerd in enige informatie over een machine welke al enige malen was opgevallen in de literatuur over nieuwe hardware op de markt. Zij vroeg aan en kreeg van Snijders Micro Systems te Vlierden informatie welke wij hier voor u samenvatten.

De Omega is een krachtig 32 bit werkstation gebaseerd op de 68020 CPU terzijde gestaan door een 68881 coprocessor. De toepassingsmogelijkheden beslaan een gebied vanaf software ontwikkelingssysteem en getallenkraker voor wetenschappelijke toepassingen tot procescomputer, data-acquisitiesysteem of besturingscomputer in de single board uitvoering. Een interessante machine voor zowel industrie als universiteiten en HTS'en. Dus ook voor onze club, al vermoeden wij dat de prijs niet uit ieders buidel kan worden getoverd.

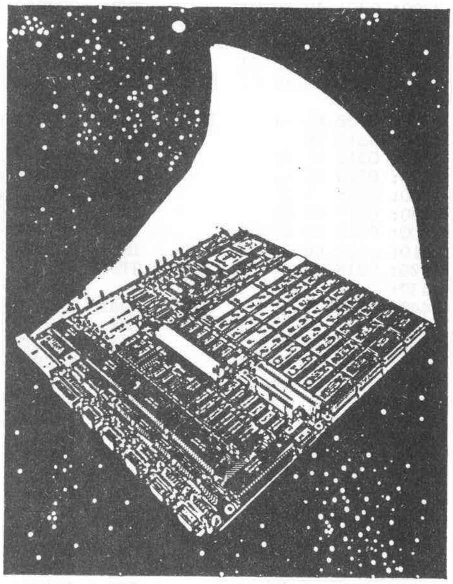
De Omega is een moderne microcomputer opgebouwd rond de 68020 CPU van Motorola met een volledige 32 bits struktuur. Ontwikkeld als single board computer met alle benodigde interfaces op een print is het achtergrondgeheugen het enige externe onderdeel van een professioneel 32 bit werkstation. Tot de standaarduitrusting behoort onder meer een 68881 drijvende komma rekenprocessor. De systeemfrekwentie bedraagt 12,5 Mhz, terwijl hogere frekwenties (16,67 ,20 of 25 Mhz) tegen meerprijs mogelijk zijn. Het geheugen bestaat uit 128/256K byte rom en 1 Megabyte no wait-state statisch, niet vluchtig Cmos Ram. Zelfs in de 25 Mhz uitvoering worden alle lees- en schrijfopdrachten binnen een cyclus uitgevoerd. Een 25,5 Mbyte harde schijf en een 1,2 Mbyte diskette station worden gebruikt als achtergrondgeheugen. De SCSI initiator, die de communicatie met de harde schijf verzorgd, kan maximaal zeven units besturen, bv een tweede harde schijf of een tapestreamer. Verder is het systeem uitgerust met vier RS232 interfaces, een real time clock/calender met battery backup, een netwerkinterface en een I/O expansion bus (16 Mbyte adresseringsruimte).

Als besturingssysteem is gekozen voor OS9/68K van Microware Systems Corporation. OS9/68K heeft een UNIX-achtige struktuur op user nivo (ons eigen DOS65 systeem ging in die gedachte al voor, weet u nog?), is multi-user en multi-tasking en ondersteunt standaard 4 gebruikers (maximaal 12) en een netwerkconfiguratie. OS9/68K biedt een aantal mogelijkheden die in de industrie onontbeerlijk zijn: het is 'real time', kompakt en efficient geschreven en volledig 'rommable', dat wil zeggen het kan in Eprom gezet worden en draaien zonder ondersteuning van hard- of floppy disk, dus zonder mechanische delen.

De opsteekkaarten die op dit moment voorhanden zijn bestaan uit een 8 kanaals RS232 kaart, 4 Megabyte statische Cmos Ram met battery backup, een hoge resolutie grafische kleurenkaart (640x480 punten, 16 kleuren uit een pallet van 4096), een 'prototyping' kaart (ruimte gereserveerd voor ontwikkeling van eigen I/O, etc) en een adapter kaart voor G64...STE...IBM PC... bus, die toegang geeft tot een uitgebreide serie I/O kaarten. In dit geval valt te denken aan AD/DA omzetters, instrumentatie-versterkers ten behoeve van rekstrookjes en PT-100 elementen, servo- of stappenmotor besturingen etc.

Mede door het gebruik van het veelzijdige OS9/68K besturingssysteem lopen de toepassingsmogelijkheden uiteen van een eenvoudige datalogger of besturingscomputer met software in Eprom via een single-user werkstation tot een krachtig mainframe-achtig netwerk met meer dan honderd stations en zonodig nog meer gebruikers. Het hoge prestatienivo, vergelijkbaar met een minicomputer (VAX 11/780) en een uitermate gunstige prijs/prestatieverhouding maakt de Omega tot een alternatief voor zowel een PC (prijs) als een minicomputer (prestaties). De hoge verwerkingssnelheid is met name belangrijk op het wetenschappelijke vlak. De Omega kan ingezet worden als pre-processor bij meetopstellingen of voor digitale signaalverwerking, zoals Fast Fourier Transformatie (FFT) en beeldverwerking. Een niet-vluchtig geheugen van 5 Megabyte en de netwerkfaciliteiten bieden ruime mogelijkheden voor inzet in een industriele omgeving waar het gebruik van mechanische delen (disk drives) uit den boze is en waar toch grote hoeveelheden gegevens verzameld worden. Dit laatste komt veelvuldig voor in bv de procesindustrie.

De Omega is ook leverbaar als SBC voor OEM gebruikers. De statische ram en ingebouwde filters staan garant voor een hoge mate van storingsongevoeligheid in een industriele omgeving. Door het gebruik van spanningsregelaars op de print kan met een ongestabiliseerde voeding volstaan worden bij een zeer lage vermogensopname van 8 Watt.



TIP van Ernst Elderenbosch, Holland

Mijn DOS65 systeem draait op een geschakelde voeding die bij de firma 'Goris Elektronika' (Meek-it) vandaan komt. Deze is maar iets groter dan een eurokaart en levert 5 Volt bij 10 Ampere en 12 Volt bij 2 Ampere en nog een klein beetje -12 Volt. Ruim voldoende en lekker efficient. Geen straalkacheltje zoals de voeding in de eerdere Junior.

****

* A L(LIST) NNNN,<CR> IMPLEMENTATION IN MICRO ADE *

****

By: Fernando Lopes, Portugal

The good old Micro-ADE, which i've made working with bank-switching (red.: ask the editorial office for the price of the paperware), lacks a common and usefull feature of the LIST command. The sort of command necessary to list the lines following a given line-number NNNN, no matter how many they are, because we can BREAK the listing at any moment, or we're using the P(AGE) mode feature (ON). I remember I used to type L NNNN,9999<CR> to command that.

The cure: it's just as easy as changing two byte of code. If in the Junior's Micro-ADE, they are $236B and $236D, that must contain $1E and $1B respectively. Else, if using Marc Lachaert's new version (V2.0), they are $05B2 and $05B4.

The inner workings: the original only checks if the 1st PARAMeter (LOPAR/HIPAR) is null, i.e., L<CR>. Of course, in that case, 2nd PARAMeter is also null. So, we can check only if the latter (LOPAR+01/HIPAR+01) is null; and then 2 cases are acknowledged: L<CR>, as before, and L NNNN,<CR> our new command! In all cases, as was in the original, the 2nd PARAM is raised to maximum 9999, (FFFF in my program) to obtain a listing of all lines following 0000 or NNNN.

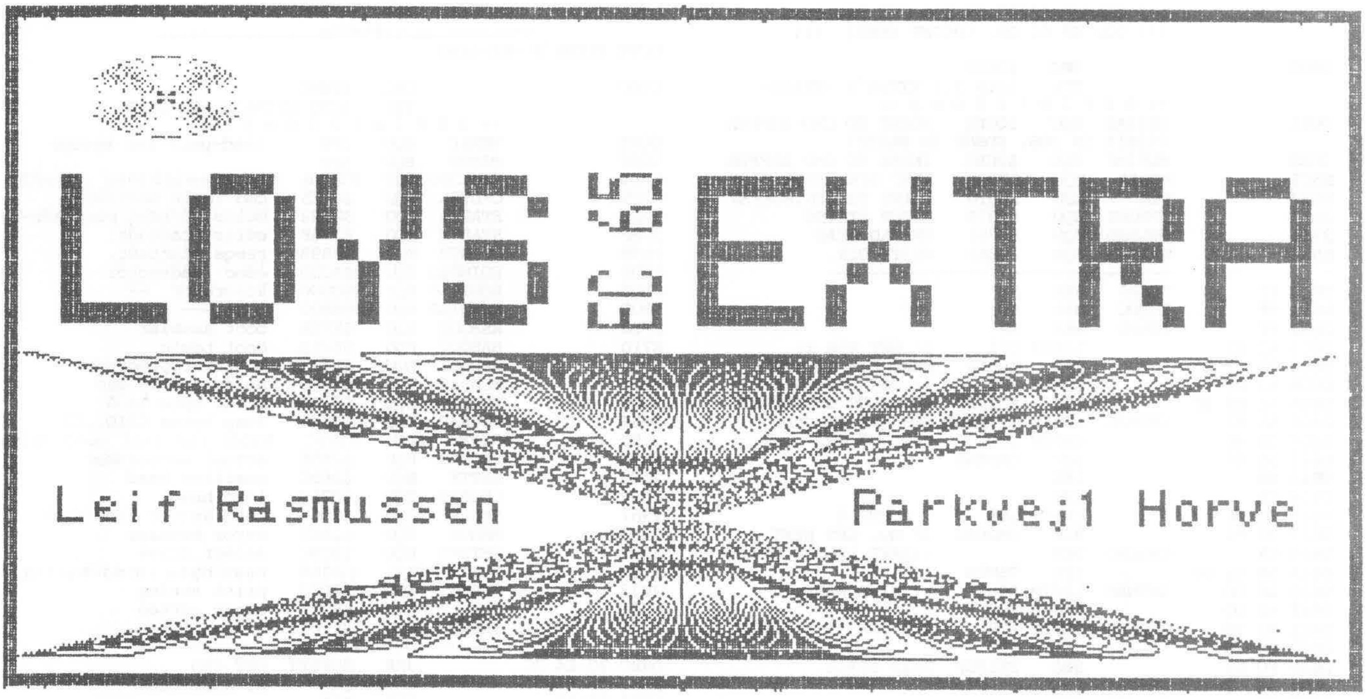
APPLE'S NIEUWE SOFTWARE BEDRIJF 'CLARIS'

De door Apple Computer recentelijk in het leven geroepen software-dochtermaatschappij heeft aangekondigd dat zij software op de markt zal brengen onder de onafhankelijke bedrijfsnaam Claris Corporation.

Apple heeft deze nieuwe onderneming in het leven geroepen voor de marketing van toepassings-software voor de Apple Macintosh en de Apple][personal computers, te beginnen met de pakketten die Apple het meest recent heeft uitgebracht. De bestaande produkten zijn MacWrite, MacDraw, MacProject, MacPaint, AppleWorks en Acces][.

In de nabije toekomst zal de nieuwe onderneming zich volgens president-directeur William C. Campbell concentreren op werkzaamheden die los staan van Apple en op de marketing van bestaande en toekomstige produkten onder eigen naam. Hij zei verder dat Claris zo spoedig mogelijk de overgang wil maken van een Apple dochtermaatschappij naar een onafhankelijk bedrijf dat een complete lijn toepassingen ontwikkelt en ondersteunt. Het ligt in de bedoeling dat Apple een minderheidsaandeel behoudt in het nieuwe bedrijf.

"We hebben voor de naam Claris gekozen omdat deze duidelijkheid ('clarity') en helderheid weerspiegelt. Het herinnert ons eraan dat het vormgeven van de toekomst een heldere visie vereist", aldus Campbell, voorheen Apple's executive vice president voor sales en marketing in de VS.



The object of this paper is to present a proposal on how to bind some of the many utilities of EC 65's systemdisk "LOYS 3.1" together in a comfortable way, and add some extra's.
The following features will be loaded with bootup:

1> ERase a filename
2> RE Assmbl. (always from dr.A)
3> RE Basic --- " ---
4> RE Word p. --- " ---
5> RE Edmo --- " ---
6> RE Kolorator --- " ---
7> RE Resequenser - " ---
8> RE I re-enable edit
9> RE Trace on/off
10> Auto-line numbering on/off
11> Extra short-hand strings
12> Error-text print-out
13> Printer-initialising menu

ad 1: Firstly it's annoying not to be able to erase a filename from directory without loading Bexec* and run Delete a filename! So this feature is now back in Dos 3.3 (in exchange for eXQute which is rarely used. Ps those who wants both, see later).

ad 2-4: Secondly, reloading of another transient processor should always start with: select drive A, as it is now the computer breaks down, trying to load as or wp from drive D!

ad 5: The amazing EDitor/MOnitor by Fred Aubert is right at hand with : DISK!"RE E".

ad 6: The Kolorator software by P. Lavigne can be installed in the system like in this proposal with: DISK!"RE K".

ad 7-8: When you are typing in a Basic program, it happens that you want to change linenumbers. You can do this now just by DISK!"RE R", then resequenser (or !) is enabled, and if you want edIt-command back, you type DISK!"RE I" (or use the much more comfortable full screen editor instead).

ad 9: With DISK!"RE T" you switch on and off the trace of Basic linenumbers.

ad 10: When typing in a Basic program it is sometimes comfortable to have the linenumbers printed automatically by the computer, so this feature is now proposed being toggled on and off with: '@' (commercial at). Then you are asked to give start line nr. and increments.

ad 11: The undispensable Short-hand has some minor defects, f. ex. with RND you have to type in: (1) afterwards. By adding a small detour in the middle of the short hand routine, you can make your own "custom made" short hand strings, f. ex. when experimenting with Kolorator in direct mode, you must type: print#2,chr$(18)" for every command line. This string now comes out just by typing ESC:.

ad 12: It is difficult to remember the 16 different error codes, so now they are printed out in human text as proposed by Gert Klein in DE 6502 KENNER nr. 39.

ad 13: It is comfortable to have a selection of printer modes at hand. This is described in DE 6502 KENNER nr. 44 (NB! there is a bug there at the end: carriage return must come after reset).

INSTALLATION:
All these features together with Full screen editor and print&(x,y) is stored on one track and loaded in address $D800 - $DFFF.
So here is how to do it (on a copy):
-- First you must clear one track on side A (f. ex. DIR or RSEQ).
-- Put the assembled routines in their appropriate addresses and save the 8 pages on this track.
-- Now change the boot routine on track zero (see DE 6502 KENNER nr. 45), so that this new track (sector 1) is loaded to $D800 instead of track 12 sector 5.
-- In dos command table you change:
Address $2E6D ff: E R 02 D8 to point to your new erase routine.
Address $2E6B f: AF D8 to point to your new re-load routine.
(Ps. the track numbers for your EDMO, Kolorator and Resequenser routines could differ from those used here).
(Pps. the edmo you get from disk 18, Kolorator from disk 15 and rseq you get by save the machine code evolving when running the RSEQ from EC 2. (NB: *SA 35,1=BB00/5)).
-- In dos error routine you write in address $2ACE ff:

```
20 40 DC    JSR ERROR-PRINTOUT
20 73 2D    JSR STROUT
20 45 52
52 4F 52
00          "ERROR"
```

to get the new text printed.
(Ps. CAll track 01 to f. ex. $6A00 and make the changes, do not make them 'in situ' - you will loose the RUN" BEXEC*" command).
-- In Full screen editor you write in address $DD0B f: 20 47 DF and in address $DF47 f: 20 70 DA to get your new printer menu and auto line in the 'question round'.
-- Now you save your fully equiped track with *SA XX,1=D800/8.
-- At last (sic!) there is one change in Short hand (track 06,4); you write in address $E678 ff: 20 A0 DB to point to your extra short hands (save it with *SA 06,4=E400/4).

It seemes complicated, but it is worth-while, and the extra routines are not dependend on each other, so you can make one step at a time, and try them out one by one.

For those who wants the XQ command as well as the ER, there is room left to make a small routine that switches between xq and er, f.ex. RE X.

One minor problem arises with "MERGE" by A. Nachtmann, but it is easily relocated to lower address, since it is mostly text (it would be nice to have merge facility right at hand too).

```
LOYS 3.1 EXTRA'S -ERASE-

                 !!! CHANGE $2E6C IN DOS CMD TBL !!!
                 !!! TO: ER 02 D8, (POINT HERE)  !!!

D800                       ORG    $D800
                           TTL    LOYS 3.1 EXTRA'S -ERASE-
                 ++ D E F I N I T I O N S ++
00E1             OSIBAD   EQU    $00E1    POINT TO CMD BUFFER
                 ($2E1E IN DOS, $D800 IN BASIC)
2CE5             BUFINI   EQU    $2CE5    INDEX TO CMD BUFFER
E5C5             RDDIR    EQU    $E5C5    READ DIR. FROM DISK
0010             PTR      EQU    $0010    POINT TO DIR.BUFFER
2D73             STROUT   EQU    $2D73    PRINT STRING
2761             UNLDHD   EQU    $2761    UNLOAD HEAD
E5AE             WRDIR    EQU    $E5AE    WRITE DIR.
                 -------------------------------------
D800 FF          TEMPA    HEX    FF
D801 FF          TEMPC    HEX    FF
D802 FF          TEMPD    HEX    FF
D803 A2 01                LDXIM  $01      :: GET DIR ::
D805 20 C5 E5             JSR    RDDIR    READ DIR TO $2E79
D808 A2 00                LDXIM  $00      :: GET CMD-FN ::
D80A AC E5 2C             LDY    BUFIND   POINT TO CMD-START
D80D B1 E1       CMDBEG   LDAIY  OSIBAD   GET CHR IN CMD
D80F C9 0E                CMPIM  $0E      ?END OF CMD
D811 90 06                BCC    CMDEND
D813 E8                   INX
D814 C8                   INY
D815 E0 06                CPXIM  $06      ? 6 CHR.S
D817 90 F4                BCC    CMDBEG   IF NO, GET NEXT
D819 CA          CMDEND   DEX             ::SAVE LENGTH OF FN::
D81A 8E 02 D8             STX    TEMPD
D81D A0 00       GETNAM   LDYIM  $00      :: FIND NAME ::
D81F A2 00                LDXIM  $00
D821 B1 10       GETCHR   LDAIY  PTR      GET CHR IN DIR.BUF.
D823 C9 20                CMPIM  $20
D825 F0 01                BEQ    SKIPSP   SKIP SPACE
D827 E8                   INX
D828 C8          SKIPSP   INY
D829 C0 06                CPYIM  $06      6 CHR.S?
D82B 90 F4                BCC    GETCHR   IF NO, GET NEXT
D82D CA                   DEX             ::SAVE LENGTH OF NM::
D82E 8E 01 D8             STX    TEMPC
D831 AE E5 2C             LDX    BUFIND   :: COMPARE 2 NAMES ::
D834 A0 00                LDYIM  $00
D836 8C 00 D8    GTNM     STY    TEMPA
D839 8A                   TXA
D83A A8                   TAY
D83B B1 E1                LDAIY  OSIBAD   GET CMD-NAME CHR
D83D AC 00 D8             LDY    TEMPA
D840 C9 0E                CMPIM  $0E      IF END CMP LENGTH
D842 90 32                BCC    CMPLNG
D844 D1 10                CMPIY  PTR      COMPARE DIR-NAME
D846 D0 08                BNE    NXTDN    NO FIT, TRY NEXT
D848 E8                   INX
D849 C8                   INY
D84A C0 06                CPYIM  $06      GET 6 CHR.S
D84C 90 E8                BCC    GTNM
D84E B0 26                BCS    CMPLNG   COMPARE LENGTH'S
D850 A5 10       NXTDN    LDA    PTR      :: NEXT DIR-NAME ::
D852 18                   CLC
D853 69 08                ADCIM  $08      NEXT NAME IN BUFFER
D855 85 10                STA    PTR
D857 90 02                BCC    NOINC
D859 E6 11                INC    PTR      +01
D85B A5 10       NOINC    LDA    PTR
D85D 38                   SEC
D85E E9 79                SBCIM  $79
D860 A5 11                LDA    PTR      +01
D862 E9 2F                SBCIM  $2F
D864 D0 B7                BNE    GETNAM   GET NEXT FILENAME
D866 20 73 2D             JSR    STROUT   :: NO FILE ::
D869 0D 0A                HEX    0D0A
D86B 4E 4F 20             ASC    NO FILE
D86E 46 49 4C
D871 45
D872 0A 0D 00             HEX    0A0D00
D875 60                   RTS
D876 AD 01 D8    CMPLNG   LDA    TEMPC    COMP CMD-FN LENGTH
D879 CD 02 D8             CMP    TEMPD    WITH DIR-NM LENGTH
D87C D0 D2                BNE    NXTDN
D87E A9 00                LDAIM  $00      :: FOUND IT !! ::
D880 A0 07                LDYIM  $07
D882 91 10                STAIY  PTR      WRITE $00 IN TRACKS
D884 88                   DEY
D885 91 10                STAIY  PTR
D887 88                   DEY
D888 A9 23                LDAIM  $23
D88A 91 10       PUTEMP   STAIY  PTR      WRITE 'EMPTY'
D88C 88                   DEY
D88D 10 FB                BPL    PUTEMP
D88F A2 01                LDXIM  $01      :: BUFFER TO DISK ::
D891 20 AE E5             JSR    WRDIR
D894 20 61 27             JSR    UNLDHD
D897 20 73 2D             JSR    STROUT   :: 'FILE ERASED' ::
D89A 0D 0A                HEX    0D0A
D89C 46 49 4C             ASC    FILE ERASED
D89F 45 20 45
D8A2 52 41 53
```

```
D8A5 45 44
D8A7 0A 0D 00                 HEX    0A0D00
D8AA 60                       RTS
              --------------FINISH--------------
LOYS EXTRA'S -RE-LOAD

D8B0                          ORG    $D8B0
                              TTL    LOYS EXTRA'S -RE-LOAD-
                  ++ D E F I N I T I O N S ++
00FE              MEMLO    EQU    $FE      load-pntr for sector
00FF              MEMHI    EQU    $FF
07DB              TRACE0   EQU    $07DB    tracebasiclines on/off
02C5              CMDTBL   EQU    $02C5    cmd table EDIT/RSEQ
0222              STARTA   EQU    $0222    holds ed/rseq startadr-1
376F              STARTE   EQU    $376F    edits startadr.
BB9B              STARTR   EQU    $BB9B    rseqs startadr.
BC00              EDIMOLO  EQU    $BC00    edmo loadvector
C000              KOLOILO  EQU    $C000    kolorator  --
BB00              RSEQILO  EQU    $BB00    rseq ---
E72A              ASBOOT   EQU    $E72A    boot asmbler
E71D              BABOOT   EQU    $E71D    boot basic
E737              WPBOOT   EQU    $E737    boot wordprocessor
2300              MEMSIZ   EQU    $2300    holds top of ram
2343              PRINT    EQU    $2343    print byte in A
2644              SWAPAB   EQU    $2644    swap bytes 0210..13
265C              DRIVES   EQU    $265C    holds the last used drive
265E              SECTNM   EQU    $265E    actual sectornmbr.
26BC              SETTK    EQU    $26BC    position head
2754              LDHEAD   EQU    $2754    load head
2967              READDK   EQU    $2967    read sector from disk
2AC0              ERROR    EQU    $2AC0    error message
2C4C              SETDRV   EQU    $2C4C    select drive
2CE4              BUFBYT   EQU    $2CE4    read byte incmd-buffer
2D73              STROUT   EQU    $2D73    print string
F32F              CLS      EQU    $F32F    clear screen
F707              INIKBD   EQU    $F707
                  -----------------------------
D8B0 20 E4 2C              JSR    BUFBYT   GET CMD
D8B3 C9 41                 CMPIM  $41      A? ASSEMBLER
D8B5 D0 06                 BNE    BAS
D8B7 20 0B D9              JSR    SETA
D8BA 4C 2A E7              JMP    ASBOOT
D8BD C9 42        BAS      CMPIM  $42      B? BASIC
D8BF D0 06                 BNE    WP
D8C1 20 0B D9              JSR    SETA
D8C4 4C 1D E7              JMP    BABOOT
D8C7 C9 57        WP       CMPIM  $57      W? WORD-PROCESSOR
D8C9 D0 06                 BNE    JUNMON
D8CB 20 0B D9              JSR    SETA
D8CE 4C 37 E7              JMP    WPBOOT
D8D1 C9 4D        JUNMON   CMPIM  $4D      M? JUNIOR-MONITOR
D8D3 D0 06                 BNE    EDMON
D8D5 20 44 26              JSR    SWAPAB
D8D8 6C FC FF              JMPI   $FFFC
D8DB C9 45        EDMON    CMPIM  $45      E? EDITOR/MONITOR
D8DD D0 06                 BNE    KOLOR
D8DF 20 0B D9              JSR    SETA
D8E2 4C 2B D9              JMP    EDMLO
D8E5 C9 4B        KOLOR    CMPIM  $4B      K? KOLORATOR
D8E7 D0 06                 BNE    RSEQ?
D8E9 20 0B D9              JSR    SETA
D8EC 4C 77 D9              JMP    KOLORLO
D8EF C9 52        RSEQ?    CMPIM  $52      R? RSEQ
D8F1 D0 06                 BNE    EDIT?
D8F3 20 0B D9              JSR    SETA
D8F6 4C A7 D9              JMP    RSEQLO
D8F9 C9 49        EDIT?    CMPIM  $49      I? EDIT
D8FB D0 03                 BNE    TRACE1
D8FD 4C F8 D9              JMP    EDITLO
D900 C9 54        TRACE1   CMPIM  $54      T? TRACE
D902 D0 03                 BNE    NOMORE
D904 4C 10 DA              JMP    TRACE2
D907 4C C0 2A     NOMORE   JMP    ERROR
                  --------------- SUB ROUTINES -------
D90A 00           SAVDRIV  HEX    00       remember last drive
D90B AD 5C 26     SETA     LDA    DRIVES
D90E 8D 0A D9              STA    SAVDRIV
D911 A9 01                 LDAIM  $01      allways load from dr A
D913 20 4C 2C     SETD     JSR    SETDRV
D916 60                    RTS
D917 AD 0A D9     RETDRIV  LDA    SAVDRIV  return to last dr
D91A 4C 13 D9              JMP    SETD
D91D 85 FE        STAXMEM  STA    MEMLO    store load vector
D91F 86 FF                 STX    MEMHI
D921 60                    RTS
D922 8E 5E 26     RTTSAX   STX    SECTNM   store sectornmbr.
D925 20 BC 26              JSR    SETTK    put head on track
D928 4C 67 29              JMP    READDK   read track and return
                  -------------------------------
D92B 20 54 27     EDMLO    JSR    LDHEAD   load editor/monitor
D92E A9 00                 LDAIM  EDIMOLO
D930 A2 BC                 LDXIM  EDIMOLO /256
D932 20 1D D9              JSR    STAXMEM
D935 A9 07                 LDAIM  $07      TRACK 07,1 -->
D937 A2 01                 LDXIM  $01
D939 20 22 D9              JSR    RTTSAX   $BC00 = EDMO / 1
D93C A9 00                 LDAIM  $00
D93E A2 C4                 LDXIM  $C4
D940 20 1D D9              JSR    STAXMEM
D943 A9 08                 LDAIM  $08      TRACK 08,1 -->
```

```
D945 A2 01                LDXIM $01
D947 20 22 D9             JSR   RTTSAX  $C400 = EDMO / 2
D94A A9 00                LDAIM $00
D94C A2 CC                LDXIM $CC
D94E 20 1D D9             JSR   STAXMEM
D951 A9 09                LDAIM $09     TRACK 09,1 -->
D953 A2 01                LDXIM $01
D955 20 22 D9             JSR   RTTSAX  $CC00 = EDMO / 3
D958 20 17 D9             JSR   RETDRIV
D95B 68                   PLA
D95C 8D 75 D9             STA   SAVE1   save return adr.
D95F 68                   PLA
D960 8D 76 D9             STA   SAVE2
D963 20 03 BC             JSR   EDIMOLO +03 goto editor/monitor
D966 20 2F F3             JSR   CLS     after exit edmo,
D969 20 07 F7             JSR   INIKBD   clear screen
D96C AD 76 D9             LDA   SAVE2   get return adr.
D96F 48                   PHA
D970 AD 75 D9             LDA   SAVE1
D973 48                   PHA
D974 60                   RTS           return to basic or dos
D975 00          SAVE1    HEX   00
D976 00                   HEX   00
                 ------------------------------------------
D977 20 54 27    KOLORLO  JSR   LDHEAD  load kolorator
D97A A9 00                LDAIM KOLOILO
D97C A2 C0                LDXIM KOLOILO /256
D97E 20 1D D9             JSR   STAXMEM
D981 A9 20                LDAIM $20     track 20,1-->
D983 A2 01                LDXIM $01
D985 20 22 D9             JSR   RTTSAX  $C000 KOLOR./1
D988 A9 00                LDAIM $00
D98A A2 C8                LDXIM $C8
D98C 20 1D D9             JSR   STAXMEM
D98F A9 21                LDAIM $21     track 21,1-->
D991 A2 01                LDXIM $01
D993 20 22 D9             JSR   RTTSAX  $C800 KOLOR./2
D996 20 17 D9             JSR   RETDRIV
D999 A9 02                LDAIM KOLOILO +02 set device #2 outp
D99B A2 C0                LDXIM KOLOILO /256 to kolor.
D99D 8D 13 23             STA   MEMSIZ  +13
D9A0 8E 14 23             STX   MEMSIZ  +14
D9A3 20 00 C0             JSR   KOLOILO initiate kolor.
D9A6 60                   RTS           return to basic or dos
                 ------------------------------------------
D9A7 20 54 27    RSEQLO   JSR   LDHEAD
D9AA A9 00                LDAIM RSEQILO
D9AC A2 BB                LDXIM RSEQILO /256
D9AE 20 1D D9             JSR   STAXMEM
D9B1 A9 35                LDAIM $35     track 35,1 -->
D9B3 A2 01                LDXIM $01
D9B5 20 22 D9             JSR   RTTSAX  $BB00 RSEQ
D9B8 20 17 D9             JSR   RETDRIV
D9BB A0 BA .              LDYIM $BA     memory top $BA00
D9BD A9 9B                LDAIM STARTR  start rseq $BB9B -1
D9BF A2 BB                LDXIM STARTR  /256
D9C1 20 CB D9             JSR   STAADR  startadr.to dos cmdtab'
D9C4 A0 FF                LDYIM $FF     counter
D9C6 A2 00                LDXIM TABL1   -TABL1
D9C8 4C D5 D9             JMP   WRCMD   write RSEQ in cmdtab'
D9CB 8D 22 02    STAADR   STA   STARTA
D9CE 8E 23 02             STX   STARTA  +01
D9D1 8C 00 23             STY   MEMSIZ
D9D4 60                   RTS
D9D5 BD 08 DA    WRCMD    LDAX  TABL1
D9D8 20 43 23             JSR   PRINT
D9DB C8                   INY
D9DC C0 03                CPYIM $03
D9DE D0 03                BNE STACMD
D9E0 18                   CLC
D9E1 69 80                ADCIM $80
D9E3 99 C5 02    STACMD   STAY  CMDTBL
D9E6 E8                   INX
D9E7 C0 03                CPYIM $03
D9E9 D0 EA                BNE   WRCMD
D9EB 20 73 2D             JSR   STROUT
D9EE 20 45 4E             ASC    ENABLED
D9F1 41 42 4C
D9F4 45 44
D9F6 00                   HEX   00
D9F7 60                   RTS           return to basic or dos
                 ------------------------------
D9F8 A0 BF       EDITLO   LDYIM $BF     memorytop $BF00
D9FA A9 6F                LDAIM STARTE  ed startadr.to cmdtab'
D9FC A2 37                LDXIM STARTE  /256
D9FE 20 CB D9             JSR   STAADR
DA01 A0 FF                LDYIM $FF     counter
DA03 A2 04                LDXIM TABL2   -TABL1
DA05 4C D5 D9             JMP   WRCMD   write EDIT in cmdtab'
DA08 52 53 45    TABL1    ASC   RSEQ
DA0B 51
DA0C 45 44 49    TABL2    ASC   EDIT
DA0F 54
                 ------------------------------
DA10 A9 00       TRACE2   LDAIM $00     toggle trace on/off
DA12 49 01                EORIM $01
DA14 8D 11 DA             STA   TRACE2  +01
DA17 F0 05                BEQ   TRA-OFF
DA19 A2 00                LDXIM TABL3   -TABL3
DA1B 4C 20 DA             JMP   TRACE3
```

```
DA1E A2 05       TRA-OFF LDXIM TABL4   -TABL3
DA20 A0 00       TRACE3  LDYIM $00
DA22 BD 2F DA    TRACE4  LDAX  TABL3
DA25 99 DB 07            STAY  TRACE0
DA28 E8                  INX
DA29 C8                  INY
DA2A C0 04               CPYIM $04
DA2C D0 F4               BNE   TRACE4
DA2E 60                  RTS          return to basic
DA2F 20 D8 1C    TABL3   HEX   20D81CEAEA
DA32 EA EA
DA34 18 90 02    TABL4   HEX   189002E6C8
DA37 E6 C8
                 ---------------------------------
LOYS 3.1 EXTRA'S -AUTOLINE-

DA70                     ORG   $DA70
                         TTL   LOYS 3.1 EXTRA'S -AUTOLINE-
                     ::: TEMPORARY REGISTERS :::
DA60             TEMPA   EQU   $DA60
DA61             LNL     EQU   TEMPA   +01 LINE NR.
DA62             LNH     EQU   TEMPA   +02
DA63             INCHR   EQU   TEMPA   +03 INCREMENT
DA64             INCRL   EQU   TEMPA   +04
DA65             INPUT   EQU   TEMPA   +05 CHARACTER BUFFER
DA66             LINPRO  EQU   TEMPA   +06 LINE INP IN PROG FLG
DA68             FIGCNT  EQU   TEMPA   +08 CHR COUNTER, LINE NO
DA69             COUNT   EQU   TEMPA   +09
DA6A             OUTLN   EQU   TEMPA   +0A OUT BUFFER LINE NO
DA6B             CLNL    EQU   TEMPA   +0B
DA6C             CLNH    EQU   TEMPA   +0C
DA6D             TEMPX   EQU   TEMPA   +0D
DA6E             TEMPY   EQU   TEMPA   +0E
DA6F             AUTOFL  EQU   TEMPA   +0F AUTOLINE ON/OFF FL
E7C2             PARBL   EQU   $E7C2
E7C3             PARBH   EQU   PARBL   +01
                     ::: EXTERNAL ADDRESSES :::
F32F             RESET   EQU   $F32F   CLEAR SCREEN
FA90             IPB     EQU   $FA90   INPUT MATRIX
FA21             RESPAR  EQU   $FA21   RESET PARAL &PARBL
2D73             STROUT  EQU   $2D73   PRINT STRING
F71D             RECHA   EQU   $F71D   GET CHR FROM KBD
0474             BASIC   EQU   $0474   BASIC WARM
2336             INBAS   EQU   $2336   BASIC INPUTVEC
0588             BASIN   EQU   $0588   BASIC IN
00CC             INL     EQU   $00CC
00CD             INH     EQU   $00CD
                 ---------------------------
DA70 20 1D F7            JSR   RECHA   CHANGE ADR $DF48,49
DA73 C9 40               CMPIM $40      TO 70,DA
DA75 F0 01               BEQ   TSAVER  COMMERCIAL AT TOGGLES
DA77 60                  RTS
                 ---------------------------
DA78 AD 6F DA    TSAVER  LDA   AUTOFL
DA7B 49 FF               EORIM $FF
DA7D 8D 6F DA            STA   AUTOFL
DA80 F0 03               BEQ   AUTO
DA82 4C 7F DB            JMP   BACK    AUTOLINE OFF
                 ---------------------------
DA85 A5 CC       AUTO    LDA   INL     SAVE INL&INH FOR LATER
DA87 8D 6D DA            STA   TEMPX    USE IN BASIC
DA8A A5 CD               LDA   INH
DA8C 8D 6E DA            STA   TEMPY
DA8F 20 73 2D            JSR   STROUT
DA92 0D 0A               HEX   0D0A
DA94 53 54 41            ASC   START LINE
DA97 52 54 20
DA9A 4C 49 4E
DA9D 45 20
DA9F 00                  HEX   00
DAA0 20 BC DB            JSR   TXT     '4 DIGITS'
DAA3 20 21 FA            JSR   RESPAR  GET PARAMETERS
DAA6 20 90 FA            JSR   IPB
DAA9 AD C2 E7            LDA   PARBL
DAAC 8D 62 DA            STA   LNH
DAAF 8D 6C DA            STA   CLNH
DAB2 AD C3 E7            LDA   PARBH
DAB5 8D 61 DA            STA   LNL     START LINE
DAB8 8D 6B DA            STA   CLNL
DABB 20 73 2D            JSR   STROUT
DABE 0D 0A               HEX   0D0A
DAC0 49 4E 43            ASC   INCREMENT
DAC3 52 45 4D
DAC6 45 4E 54
DAC9 20 20
DACB 00                  HEX   00
DACC 20 8C DB            JSR   TXT
DACF 20 21 FA            JSR   RESPAR  GET PARAMETERS
DAD2 20 90 FA            JSR   IPB
DAD5 AD C2 E7            LDA   PARBL
DAD8 8D 64 DA            STA   INCRL   INCREMENTS
DADB AD C3 E7            LDA   PARBH
DADE 8D 63 DA            STA   INCRH
                 ---------------------------
DAE1 AD 6D DA            LDA   TEMPX   RESTORE INL AND INH
DAE4 85 CC               STA   INL
DAE6 AD 6E DA            LDA   TEMPY
DAE9 85 CD               STA   INH
DAEB 18                  CLC
DAEC A9 06               LDAIM BEGIN   CHANGE INPVEC
```

```
DAEE 8D 88 05           STA   BASIN
DAF1 A9 DB              LDAIM BEGIN   /256
DAF3 8D 89 05           STA   BASIN   +01
DAF6 A9 0D              LDAIM $0D
DAF8 8D 65 DA           STA   INPUT
DAFB A9 00              LDAIM $00
DAFD 8D 66 DA           STA   LINPRO  RESET LINEPRO
DB00 20 2F F3           JSR   RESET
DB03 4C 74 04           JMP   BASIC    TO BASIC
                  --------------------------
DB06 98         BEGIN   TYA   NEW     INPUT ROUTINE
DB07 48                 PHA
DB08 AC 65 DA           LDY   INPUT
DB0B C0 0D              CPYIM $0D     IF CR, INC LIN NR
DB0D F0 0C              BEQ   INCREM
DB0F 20 36 23   NEW     JSR   INBAS   GET CHR FROM KBD
DB12 8D 65 DA           STA   INPUT
DB15 68                 PLA
DB16 A8                 TAY
DB17 AD 65 DA           LDA   INPUT
DB1A 60                 RTS           NORMAL PROC. CHR
                  --------------------------
DB1B AD 66 DA   INCREM  LDA   LINPRO  AUTOLINE ROUTINE
DB1E D0 08              BNE   NFIRST
DB20 A9 04              LDAIM $04     4 DIGITS
DB22 8D 66 DA           STA   LINPRO
DB25 8D 68 DA           STA   FIGCNT
DB28 AD 68 DA   NFIRST  LDA   FIGCNT
DB2B F0 2A              BEQ   LAST
DB2D A9 00              LDAIM $00
DB2F 8D 60 DA           STA   TEMPA
DB32 A9 04              LDAIM $04
DB34 8D 69 DA           STA   COUNT
DB37 18         SHIFT   CLC
DB38 2E 62 DA           ROL   LNH     SHIFT NEXT DIGIT TO
DB3B 2E 61 DA           ROL   LNL      LINE NO.
DB3E 2E 60 DA           ROL   TEMPA
DB41 CE 69 DA           DEC   COUNT
DB44 D0 F1              BNE   SHIFT
DB46 CE 68 DA           DEC   FIGCNT
DB49 AD 60 DA           LDA   TEMPA   CONVERT LINENR >
DB4C 69 30              ADCIM $30      ASCII
DB4E 8D 6A DA   NSPLIT  STA   OUTLN
DB51 68                 PLA           RESTORE Y REG.
DB52 A8                 TAY
DB53 AD 6A DA           LDA   OUTLN
DB56 60                 RTS
                 ::: PREPARE FOR NEXT LINE :::
DB57 A9 20      LAST    LDAIM $20
DB59 8D 65 DA           STA   INPUT
DB5C A9 00              LDAIM $00
DB5E 8D 66 DA           STA   LINPRO
DB61 F8                 SED
DB62 18                 CLC
DB63 AD 6C DA           LDA   CLNH
DB66 6D 64 DA           ADC   INCRL
DB69 8D 6C DA           STA   CLNH
DB6C 8D 62 DA           STA   LNH
DB6F AD 6B DA           LDA   CLNL
DB72 6D 63 DA           ADC   INCRH
DB75 8D 6B DA           STA   CLNL
DB78 8D 61 DA           STA   LNL
DB7B D8                 CLD
DB7C B8                 CLV
DB7D 50 90              BVC   NEW     BACK TO INPUT ROUT.
                  --------------------------
DB7F A9 36      BACK    LDAIM $36     RESTORE OLD INVEC
DB81 8D 88 05           STA   BASIN
DB84 A9 23              LDAIM $23
DB86 8D 89 05           STA   BASIN   +01
DB89 4C 74 04           JMP   BASIC
                  --------------------------
DB8C 20 73 2D   TXT     JSR   STROUT
DB8F 20 28 34           ASC    (4 DIGITS)
DB92 20 44 49
DB95 47 49 54
DB98 53 29 20
DB9B 00                 HEX   00
DB9C 60                 RTS
                  --------------------------
LOYS EXTRA'S -SHORTHAND-

DBA0                    ORG   $DBA0
                        TTL   LOYS EXTRA'S -SHORTHAND-
                  --------------------------
F71D            RECCHA  EQU   $F71D   GET CHR. FROM KBD
E660            TEMPY   EQU   $E660
E6A9            OLDRUT  EQU   $E6A9   ORG. SHORTH.
E6A4            GETCMD  EQU   $E6A4
0284            BASCOM  EQU   $0284   BASIC CMD. TABL
                  --------------------
DBA0 20 1D F7           JSR   RECCHA  GET CHR.
DBA3 A0 00              LDYIM TBL1    -TBL1
DBA5 C9 4B              CMPIM 'K      disk!"
DBA7 F0 3B              BEQ   SAVEY
DBA9 A0 06              LDYIM TBL2    -TBL1
DBAB C9 3A              CMPIM ':      print#2,chr$(18)"
DBAD F0 35              BEQ   SAVEY
DBAF A0 13              LDYIM TBL3    -TBL1
DBB1 C9 2F              CMPIM '/      rnd(1)
```

```
DBB3 F0 2F              BEQ   SAVEY
DBB5 A0 19              LDYIM TBL4    -TBL1
DBB7 C9 48              CMPIM 'H      chr$(
DBB9 F0 29              BEQ   SAVEY
DBBB A0 1E              LDYIM TBL5    -TBL1
DBBD C9 26              CMPIM '&      print&(
DBBF F0 23              BEQ   SAVEY
DBC1 A0 25              LDYIM TBL6    -TBL1
DBC3 C9 45              CMPIM 'E      peek(
DBC5 F0 1D              BEQ   SAVEY
DBC7 A0 2A              LDYIM TBL7    -TBL1
DBC9 C9 7B              CMPIM '{      disk!"re
DBCB F0 17              BEQ   SAVEY
DBCD A0 33              LDYIM TBL8    -TBL1
DBCF C9 7D              CMPIM '}      disk!"put
DBD1 F0 11              BEQ   SAVEY
DBD3 A0 3D              LDYIM TBL9    -TBL1
DBD5 C9 7C              CMPIM '|      disk!"lo
DBD7 F0 0B              BEQ   SAVEY
                  --------------------------
DBD9 A0 84              LDYIM BASCOM  if not any, then
DBDB 8C A9 E6           STY   OLDRUT  restore old rout.
DBDE A0 02              LDYIM BASCOM  /256
DBE0 8C AA E6           STY   OLDRUT  +01
DBE3 60                 RTS           and return.
                  --------------------------
DBE4 8C 60 E6   SAVEY   STY   TEMPY   if one of these
DBE7 A9 F5              LDAIM TBL1    -01
DBE9 A0 DB              LDYIM TBL1    /256
DBEB 8D A9 E6           STA   OLDRUT  then set ptr. to
DBEE 8C AA E6           STY   OLDRUT  +01 this routine
DBF1 68                 PLA
DBF2 68                 PLA
DBF3 4C A4 E6           JMP   GETCMD  and write string
                  --------------------------
DBF6 44 49 53   TBL1    ASC   DISK!
DBF9 4B 21
DBFB A2                 HEX   A2
DBFC 50 52 49   TBL2    ASC   PRINT#2,(18)
DBFF 4E 54 23
DC02 32 2C 28
DC05 31 38 29
DC08 A2                 HEX   A2
DC09 52 4E 44   TBL3    ASC   RND(1
DC0C 28 31
DC0E A9                 HEX   A9
DC0F 43 48 52   TBL4    ASC   CHR$
DC12 24
DC13 A8                 HEX   A8
DC14 50 52 49   TBL5    ASC   PRINT&
DC17 4E 54 26
DC1A A8                 HEX   A8
DC1B 50 45 45   TBL6    ASC   PEEK
DC1E 4B
DC1F A8                 HEX   A8
DC20 44 49 53   TBL7    ASC   DISK!"RE
DC23 4B 21 22
DC26 52 45
DC28 A0                 HEX   A0
DC29 44 49 53   TBL8    ASC   DISK!"PUT
DC2C 4B 21 22
DC2F 50 55 54
DC32 A0                 HEX   A0
DC33 44 49 53   TBL9    ASC   DISK!"LO
DC36 4B 21 22
DC39 4C 4F
DC3B A0                 HEX   A0
                  --------------------------
LOYS EXTRA'S -ERROR-

DC40                    ORG   $DC40
                        TTL   LOYS EXTRA'S -ERROR-

2343            PRINT   EQU   $2343

DC40 AA                 TAX
DC41 CA                 DEX
DC42 BC 52 DC           LDYX  TABLE1
DC45 B9 60 DC   PRINTE  LDAY  ERR1
DC48 F0 07              BEQ   EPRINT
DC4A 20 43 23           JSR   PRINT
DC4D C8                 INY
DC4E 4C 45 DC           JMP   PRINTE
DC51 60         EPRINT  RTS
DC52 00         TABLE1  DFB   ERR1      -ERR1
DC53 07                 DFB   ERR2      -ERR1
DC54 0E                 DFB   ERR3      -ERR1
DC55 16                 DFB   ERR4      -ERR1
DC56 24                 DFB   ERR5      -ERR1
DC57 29                 DFB   ERR6      -ERR1
DC58 39                 DFB   ERR7      -ERR1
DC59 40                 DFB   ERR8      -ERR1
DC5A 4E                 DFB   ERR9      -ERR1
DC5B 5B                 DFB   ERRA      -ERR1
DC5C 69                 DFB   ERRB      -ERR1
DC5D 7B                 DFB   ERRC      -ERR1
DC5E 83                 DFB   ERRD      -ERR1
DC5F 95                 DFB   ERRE      -ERR1
DC60 50 41 52   ERR1    ASC   PARITY
DC63 49 54 59
```

```
DC66 00              HEX   00
DC67 52 45 52  ERR2  ASC   REREAD
DC6A 45 41 44
DC6D 00              HEX   00
DC6E 54 52 41  ERR3  ASC   TRACK 0
DC71 43 4B 20
DC74 30
DC75 00              HEX   00
DC76 57 52 49  ERR4  ASC   WRITE PROTECT
DC79 54 45 20
DC7C 50 52 4F
DC7F 54 45 43
DC82 54
DC83 00              HEX   00
DC84 53 45 45  ERR5  ASC   SEEK
DC87 4B
DC88 00              HEX   00
DC89 44 52 49  ERR6  ASC   DRIVE NOT READY
DC8C 56 45 20
DC8F 4E 4F 54
DC92 20 52 45
DC95 41 44 59
DC98 00              HEX   00
DC99 53 59 4E  ERR7  ASC   SYNTAX
DC9C 54 41 58
DC9F 00              HEX   00
DCA0 42 41 44  ERR8  ASC   BAD TRACK NMR
DCA3 20 54 52
DCA6 41 43 4B
DCA9 20 4E 4D
DCAC 52
DCAD 00              HEX   00
DCAE 54 52 41  ERR9  ASC   TRACK HEADER
DCB1 43 4B 20
DCB4 48 45 41
DCB7 44 45 52
DCBA 00              HEX   00
DCBB 53 45 43  ERRA  ASC   SECTOR HEADER
DCBE 54 4F 52
DCC1 20 48 45
DCC4 41 44 45
DCC7 52
DCC8 00              HEX   00
DCC9 42 41 44  ERRB  ASC   BAD SECTOR LENGTH
DCCC 20 53 45
DCCF 43 54 4F
DCD2 52 20 4C
DCD5 45 4E 47
DCD8 54 48
DCDA 00              HEX   00
DCDB 4E 4F 20  ERRC  ASC   NO FILE
DCDE 46 49 4C
DCE1 45
DCE2 00              HEX   00
DCE3 52 2F 57  ERRD  ASC   R/W PAST FILE-END
DCE6 20 50 41
DCE9 53 54 20
DCEC 46 49 4C
DCEF 45 2D 45
DCF2 4E 44
DCF4 00              HEX   00
DCF5 44 49 53  ERRE  ASC   DISK FULL
DCF8 4B 20 46
DCFB 55 4C 4C
DCFE 00              HEX   00
```

```
 10 REM=============================================================
 20 REM= CLOCK FOR ACORN-ATOM               BY JOHN ANIJS    870714 =
 30 REM= THIS PROGRAM HAS BEEN DERIVED FROM THE PROGRAM WRITTEN=
 40 REM= BY R.V.VUGT FOR BBC AND ELECTRON (DE 6502 KENNER 50).  =
 50 REM= THIS PROGRAM WORKS ONLY PROPERLY IN MODE 0, AND SHOWS  =
 60 REM= THE TIME IN THE UPPER RIGHTHAND CORNER OF THE SCREEN.  =
 70 REM= THE PROGRAMCODE MAY BE PLACED IN (E)PROM. (VAR. S&T)   =
 80 REM= RUNTIME VARIABLES ARE ALLOCATED BY VAR. R. (9 BYTES)   =
 90 REM= THE VIA HAS TO BE INSTALLED WITH IRQ-LINE CONNECTED.   =
100 REM= THE CLOCKPROGRAM IS BASED ON 50 mS INTERRUPT.          =
110 REM= THE PROGRAM IS STARTED BY: LINK MM0,OR LINK<ADDR> (=T)=
120 REM= TIMESETTING IS BY MEANS OF COMMAND: *TIME hh mm ss     =
130 REM=============================================================
140 I=#204;REM INTERRUPT VECTOR
150 J=#206;REM OSCLI VECTOR
160 V=#B800;REM VIA ADDRESS
170 S=#3800;REM OBJECT START AFTER ASSEMBLY
180 T=#3800;REM CODE START FOR EXECUTION
190 R=#3800;REM RAM ADDRESS
200 DIMLL31,MM31;F.N=0TO31;LLN=#777:MMN=#777;N.
210 P.'"ASSEMBLY PHASE 1",$21
220 GOS.a
230 F.N=0TO31;MMN=LLN-S+T;N.
240 P.$6,'"ASSEMBLY PHASE 2"',$21
250 GOS.a;P.$6
260 END
270aP=S
280[
290:LL0;SEI;LDA I;STA R;LDA I+1;STA R+1
300LDA @MM2&#FF;STA I;LDA @MM2/256;STA I+1
310LDA J;STA R+2;LDA J+1;STA R+3
320LDA @MM8&#FF;STA J;LDA @MM8/256;STA J+1
330LDA @#C0;STA V+#E;LDA @#40;STA V+#B;LDA @#4E;STA V+6
340LDA @#C3;STA V+5;CLI;RTS
350:LL2;TXA;PHA;TYA;PHA
360DEC R+4;BNE LL1;LDA @20;STA R+4
370SED;LDA @0;SEC;ADC R+5;STA R+5;CMP @#60;BNE LL3
380LDA @0;STA R+5;SEC;ADC R+6;STA R+6;CMP @#60;BNE LL3
390LDA @0;STA R+6;SEC;ADC R+7;STA R+7;CMP @#24;BNE LL3
400LDA @0;STA R+7
410:LL3;LDA R+5;LDX @7;JSR MM4;JSR MM6
420LDA R+6;JSR MM4;JSR MM6;LDA R+7;JSR MM4
430:LL1;LDA @#5A;STA V+#D
440PLA;TAY;PLA;TAX;PLA
450JMP (R)
460:LL4;PHA;JSR MM5;PLA
470LSRA;LSRA;LSRA;LSRA
480:LL5;AND @#F;ORA @#30
490:LL7;STA #8018,X;DEX;RTS
500:LL6;LDA @CH":";JMP MM7
510:LL8;LDY @0;LDX @0;JSR #F876
520:LL9;LDA #100,Y;CMP MM9,X;BEQ LL10;JMP (R+2)
530:LL10;INY;INX;CPX @4;BNE LL9;LDX @2
540:LL11;JSR #F876;LDA #100,Y;ASLA;ASLA;ASLA;ASLA;STA R+8
550INY;LDA #100,Y;AND @#F;ORA R+8;STA R+5,X;INY;DEX;BPL LL11
560RTS
570:LL9;];$P="TIME";P=P+LENP;[
580]
590R.
600****************************************************************
610 PROGRAM-DESCRIPTION
620 L.200:     DEFINITION AND INITIALIZATION OF LABELS.
630 L.290-320: INITIALIZATION OF VECTORS FOR INTERRUPT AND
640            COMMAND LINE INTERPRETER.
650 L.330-340: INITIALIIZATION OF VIA (6522).
660 L.350-450: INTERRUPT SERVICE ROUTINE.
670 L.460-500: CLOCK-DISPLAY ROUTINE.
680 L.510-570: COMMAND INTERPRETER.
690\USED REGISTERS: R/R+1:     INTERRUPT VECTOR
700\                R+2/R+3:   COMAND LINE INTERPRETER VECTOR
710                 R+4:       50 mS COUNTER
720                 R+5:       SEC. COUNTER (BCD)
730                 R+6:       MIN. COUNTER (BCD)
740                 R+7:       HRS. COUNTER (BCD)
750                 R+8:       UTILITY REGISTER
760 EXTRNAL USED ROUTINE: #F876: SKIP SPACES FROM INPUT BUFFER
770****************************************************************
```

OCTOFATE
FATE for the Octopus

By   : Coen Boltjes, The Netherlands
Transl.: Elja v.d. Veer, The Netherlands

The latest offspring of the Octopus software family is OCTOFATE. Even before OCTOFATE is available, it already has become a legend because it is a version of FATE adjusted to Elektor's EC65/Octopus computer by our member Marc Lachaert. And FATE is well-known for Elektor's Junior computer.

OCTOFATE's heart is formed by the Text Editor, an extremely powerful line editor, which can make and change OCTOFATE files in a simple and user-minded way. Closely related to this are the commands of the Disk Operating System especially deviced for OCTOFATE, which can transport files from and to disks.
For reasons of upwards compatability of OCTOFATE tape routines are also implemented, so the FATE files of the Junior can be read without any problems. Naturally, the tape routines can be used for making backup files on tape as well.

Different modules can be called from the Text Editor:

-The Format Lister. This programme can print out texts according to a given format. The programme takes care of page numbers, headlines, transfering to new pages etc. etc. The text to be printed can in principle be infinetely long, because linked files can be used. This enables one to place a command at the end of a file so that a subsequent file can be read.
A consequence of this possibility is that texts can be divided in well-ordered modules (e.g. chapters, paragraphs) while the Format Lister is regarding it as a whole.

-The Assembler is the second module. This concerns a 2-pass conditional assembler using the MOS-Technology notation. In this way it is always clear which addressing mode is refered to, contrary to the Micro-ADE notation.
The time needed for assembling a source is very short: approximately 7.5 seconds for 1K object code on 1 Mhz.
As the Format Lister, the Assembler can also make use of linked files, so that large programmes can be assembled as well. In the OSI- and the Micro-ADE assembler one must limit the sources to about 1500 lines, because otherwise there will be no room in the available memory anymore for the source, symbol table and object code. A nightmare for many programme-makers, which now belongs to the past with OCTOFATE.
In OCTOFATE it is possible to read the sources from disk per track, to assemble them and to store the object code directly on the disk, so that 28K can be reserved for the symbol table.

-In software advertisements it is often claimed that adjustments to individual wishes and the configuration of the user are simple. In that case "only few things" have to be done...
With OCTOFATE efforts were made to make life as easy as possible for the user. Therefore, a configurator is introduced as a third module in order to facilitate adjusting OCTOFATE to any system. In this way all default values (line length, source start etc.), disk drive configurations and so on can be given and stored as a configuration on disk. Additionally, the guiding system of the printer can simply be adjusted to any printer, so that it can underline a text by means of a simple command, or it can switch over to another type of characters in a simple way. All this can be done without having to consult the printer's manual every time. At the system disk there is room for seven configurations. Thus, the use of different kinds of printers will not present any problems.

Future plans are to develop a File Convertor which can change Micro-ADE and OSI-Assembler files into FATE format. At this moment a Full Screen Editor for OCTOFATE is worked on.

For good operating of OCTOFATE a standard EC65/Octopus suffices. However, the use of Tape Utilities and the "Bell" requires the Basicode Interface Card from Elektor's Computing Special 2.

At this moment the following is available for OCTOFATE:

-OCTOFATE System Disk, including the Text Editor, Format Lister, Assembler and Configurator.
-OCTOFATE User Manual containing 90 pages explaining the software (for the time being only a Dutch version).
-OCTOFATE Source Listings in 6502 assembler (English in Micro-ADE and MOS-Technology notation).

1. Editor, Tape I/O	97 pages
2. DOS	67 pages
3. Assembler	80 pages
4. Default Table	10 pages
5. Format Lister	10 pages
6. Configurator	34 pages

At the september meeting a demonstration of OCTOFATE can be given. Those who are interested are welcome.

PAPERWARE & DISKETTE SERVICE OCTOFATE FOR EC65/OCTOPUS

Paying with Eurocheque or on postgiro 841433 of W.L. van Pelt at Krimpen a.d. IJssel: subtract Hfl 9,50. Otherwise:

-OCTOFATE System Disk.
Send an empty diskette to the editorial office, including a label and R/W prot.
Europe : Hfl. 74,50   Outside Europe : Hfl. 91,50
Members: Hfl. 24,50   Members: Hfl. 41,50

-OCTOFATE User Manual.
Europe : Hfl.104,50   Outside Europe : Hfl.121,50
Members: Hfl. 54,50   Members: Hfl. 71,50

-OCTOFATE Source Listings.
1. Editor, Tape I/O.
Europe : Hfl.108,00   Outside Europe : Hfl.125,00
Members: Hfl. 58,00   Members: Hfl. 75,00
2. Disk Operating System.
Europe : Hfl. 93,00   Outside Europe : Hfl.110,00
Members: Hfl. 43,00   Members: Hfl. 60,00
3. Assembler.
Europe : Hfl. 99,50   Outside Europe : Hfl.116,50
Members: Hfl. 49,50   Members: Hfl. 66,50
4. Default Table.
Europe : Hfl. 64,50   Outside Europe : Hfl. 81,50
Members: Hfl. 14,50   Members: Hfl. 31,50
5. Format Lister.
Europe : Hfl. 64,50   Outside Europe : Hfl. 81,50
Members: Hfl. 14,50   Members: Hfl. 31,50
6. Configurator.
Europe : Hfl. 79,50   Outside Europe : Hfl. 96,50
Members: Hfl. 29,50   Members: Hfl. 46,50

Editor, Tape I/O + DOS + Assembler + Default Table + Format Lister + Configurator + MANUAL:
Europe : Hfl.256,50   Outside Europe : Hfl.273,50
Members: Hfl.206,50   Members: Hfl.223,50

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

BRIEF AAN DE REDAKTIE

A.P. Oerlemans, Oss

In editie 50 van DE 6502 KENNER zag ik onder het hoofd CALCUATOR een rekenmachine, welke in de verschillende talstelsels rekent. Misschien is het nuttig om erop te wijzen dat er een SHARP-EL506P calculator bestaat voor de prijs van Hfl. 29,95, bij Kwantum-Hallen, met binaire, octale en hexadecimale berekening. In het binaire stelsel wordt met de 2-complements methode gerekend. De ingevoerde getallen lopen dan van 1000000000 tot en met 111111111 (binair), óf van -512 tot en met 511 (decimaal).

OCTOPUS INPUTVERWERKING

A.P. Oerlemans, Oss.

Bij de OCTOPUS wordt na een INPUT de programmaverwerking gestopt, als uitsluitend de <RETURN>-toets wordt ingedrukt, wat vervelend kan zijn. Ik heb hierop het volgende programma-onderdeel bedacht:

```
10000 REM INPUTVERWERKING
10010 X$="":Z=1
10020 DISK!"GO F71D":Y=PEEK(9059):Y$=CHR$(Y):PRINTY$:IFY=
      13THEN10050
10030 IFY$="-"THENZ=-1:GOTO10020
10040 X$=X$+Y$:GOTO10020
10050 X=VAL(X$)*Z:PRINT:RETURN
```

Z=1 in regel 10010 geeft de mogelijkheid om ook negatieve getallen te kunnen verwerken (zie regel 10030 en 10050).
Subroutine F71D wacht op een toetsaanslag en plaatst de waarde hiervan in het (decimale) adres 9059.
Uiteraard kunnen in 10010 Z=1, de regel 10030 en in 10050 X=VAL(X$) vervallen als men alleen in string-variabelen is geïnteresseerd. Bovenstaande routine keert terug met nul als alleen de <RET>-toets wordt ingedrukt (of als het eerste teken een letterteken is).

* PRINT YOUR GRAPHICS FOR ATARI 600 XL *

By: Henk Speksnijder, The Netherlands.

In addition to the program published in February 1987 here is a program to print what's on your screen.
Most matrix printers use seven needles in a column while the Atari has 8 dots in a row (in graphics 8). When you try to print things, you'll discover that a very large Basic program is needed. This is a task much better to do in machine code (at least some of it).
This program is tested on ATARI 60 XL with ATARI INTERFACE 850 and a SEIKOSHA GP-100A MARK II.
The printer must be able to print graphics.
If your printer needs other commands, change them, this printer has following commands:

CR	carriage return	CHR$(13)
DC4	no linefeed after printing	CHR$(20)
BS	graphics mode	CHR$(8)
SO	double width character	CHR$(14)
Sl	standard character	CHR$(15)
POS	print starting position	CHR$(16)
ESC	escape	CHR$(27)
FS	repeat graphics character	CHR$(28)

```
 30 P=0                                        Machinecode in
 32 FOR I=0 TO 36                              workarea
 34 READ C:P=P+C:POKE1570+I,C
 36 NEXT I
 38 IF P<>4477 THEN STOP                       test checksum
 40 DATA 104,104,133,213,104,133,212
 42 DATA 169,255,32,61,6,160,240
 44 DATA 177,212,32,61,6,152,56
 46 DATA 233,40,168,176,244,96,162
 48 DATA 7,74,62,24,6,202,16
 50 DATA249,96

500 OPEN #2,8,0,"P:"                           printer in
510 PUT #2,8:PUT #2,16:PUT #2,0:PUT #2,15      graphics mode
520 FOR V=0 TO 153 STEP 7                      top to bottom
530 FOR H=0 TO 39                              left to right
540 P=B+H+40*V                                 calculate addr
                                               in video Ram
550 C=USR(1570,P)                              call mach.code
560 FOR I=1560 TO 1567                         bring 8 bytes
570 PUT #2,PEEK(I)                             to the printer
580 NEXT I
590 NEXT H
600 PUT #2,13:PUT #2,16:PUT #2,0:PUT #2,15     send printer
610 NEXT V                                     to next line
620 PUT #2,15                                  printer back
630 CLOSE #2                                   to characters
```

If this program is not used together with the program of the February issue page 26, then add this:

```
100 GRAPHICS 8
110 C=PEEK(560)+256*PEEK(561)
120 B=PEEK(C+4)+256*PEEK(C+5)
```

Between line 120 and 500 you must create something on your screen. Otherwise you'll see nothing printed.
If the computer is not in GRAPHICS 8 when it comes at line 500, then nothing dangerous can happen; all you'll get is that your printer creates anything but graphics.
As mentioned before: the commands at line 510, 600 and 620 may vary, depending on the printer and the interface.

If you want it's possible to change line 520 or 530 but
H must be bigger than 0 and not bigger than 39
V must be bigger than 0 and not bigger than 159
for instance with:

```
520 FOR V=0 TO 79 STEP 7
530 FOR H=20 TO 39
```

Then only the upper right part of the screen will be printed.

This printer needs that in graphics: there are 8 bits, seven correspond with a needle but the MSB must be one. If your printer needs a zero then change line 42:

```
 42 DATA 169,0,32,61,6,160,240
```

* SEABATTLE *

A BASIC PROGRAMME

Transl.: Bart van Pelt, The Netherlands.

In this game you are considered to be the commander of a navy vessel. Your ship is charged with coast-guarding. Your mission is: "destroy every enemy ship in coastal waters". The coastal water are a sea drawn as a square of 100 times 100 points. This square has a horizontal line X and a vertical line Y. You will destroy the enemy vessel by missiles. These missiles will be launched by stating an X and Y coordinate. After the missile is launched, you will be told the distance between the target and the missile impact. A grazing shot will also be stated. After the fifth graze the enemy ship will be moved to another coordinate by the computer. So the searching starts again. The game has a difficulty scale which is defined by 3 variables, that have to be stated by you.

GRAZE AREA	=	VARIABLE A
SPEED AND DIRECTION OF THE SHIP	=	VARIABLE B
CERTAIN CHANGES OF COURSE	=	VARIABLE C

You will also be asked to state a number between 0 and 1. This is to be done by entering a point and thereafter five figures.

ADVICE TO THE PLAYER :	BEGINNER	9	0	0
	AMATEUR	7	3	2
	EXERCISED	6	6	4
	MASTER	4	10	5
	EXPERT	3	12	6

```
 10 REM PUT YOUR CLEAR SCREEN COMMAND IN THIS LINE
 20 PRINT"YOU RECEIVED A MESSAGE THAT AN ENEMY SHIP"
 30 PRINT"HAS INVADED"
 40 PRINT
 50 INPUT"ENTER A,B,C SEPARATED BY COMMA´S  ";A,B,C
 60 INPUT"ENTER ANY NUMBER, E.G.  .12345  ";S
 70 GOSUB 330
 80 A1=D
 90 GOSUB 330
100 A2=D
110 A3=5 : A0=5
120 GOSUB 290
130 A1=A1 + D
140 GOSUB 290
150 A2=A2 + D
160 PRINT"DISTANCE IS ...............";A4
170 INPUT"X COORDINATE";X
180 INPUT"Y COORDINATE";Y
190 W=(Y-ABS(A2))^2 + (X-ABS(A1))^2
200 A4=SQR(W)
210 A4=INT(A4*100+.5)/100
220 IF A4>=A THEN 120
230 A3=A3+A4-5
240 IF A3<=0 THEN 370
250 A0=A0-1
260 PRINT"GRAZE#";ABS(A0-5)
270 IF A0=0 THEN 70
280 GOTO 120
290 RD=(7^9*S*.00001)
300 S=RD-INT(RD)
310 D=-C+B*S
320 RETURN
330 RD=(7^9*S*.00001)
340 S=RD-INT(RD)
350 D=100*S
360 RETURN
370 PRINT:PRINT
380 PRINT TAB(10)"DIRECT HIT !!! SHIP SUNK."
390 PRINT TAB(10)"**************************"
400 PRINT:PRINT
410 PRINT"WANT ANOTHER GAME?"
420 PRINT:PRINT"IF YES, ENTER <Y>; IF NO, ENTER <N>"
430 INPUT Z$
440 IF Z$="Y" OR Z$="y" THEN 20
450 PRINT"END OF THE GAME"
460 END
```

ONTBINDEN IN FAKTOREN

Gerard van Roekel.

Kent u ze nog, die grote getallen welke eindeloos moesten worden gedeeld door 2, 3, 5, 9, 11 enz. Plots hield u dan 323 over en wat dan? Gelukkig hebben we daar een computer voor om dit op te lossen. Met het volgende simpele programma hebben we nooit meer problemen met ´ontbinden in faktoren´.

```
100 REM SCHOONMAKEN BEELDSCHERM
110 PRINT"ONTBINDEN IN FAKTOREN"
120 J=2
130 INPUT"WELK GETAL WILT U ONTBINDEN?";A$
140 A=VAL(A$):IFA<2ORINT(A)<>ATHEN100
150 FORI=JTOA
160 IFINT(A/I)=A/ITHENB=I:B$=B$+STR$(B):I=A
170 NEXT
180 A=A/B:J=B
190 IF A>=2 THEN 150
200 PRINTA$" BESTAAT UIT DE FAKTOREN:"
210 PRINTB$
220 INPUT"NOG EEN KEER? (J/N)";G$
230 IF G$="N" THEN END
240 IF G$<>"J" THEN 220
250 RUN
```

Junior tape save routines on the DOS65 computer.

Some time ago i started building a DOS65 computer and since i switched it on for the first time i have enjoyed working with it. Editing, assembling, loading and saving at tremendous speed with diskettesize up to 720k.
The only thing i couldn't do was loading and saving junior cassettes.
This is no big problem because with diskettedrives in your system there is no great need for an additional cassette recorder. Nevertheless i wanted to be as compatible with the junior system as possible, because my old junior is still standing in the corner for use as an eprom programmer.
This is why i have started adapting the junior cassette routines for use with the DOS65 (or EC65) computer. There is a pcb from Elektuur (EPS65028) with the hardware for a hobbyscope and a junior cassette interface. The following listing contains the modified junior save routines plus the routines used in book 3 to write testtapes plus two routines to check the 2400/3600 Hz output. If these frequencies are incorrect, they can be corrected by changing the values of variables higher and lower. The actual values are for my 1 MHz computer. I use timer 2 in 6522 nr 2 on the CPU piggyback board. The first part (label start) is a little program which shows how to use the routines.

====================

```
                  ; file         juncas.mac
                  ;
                  ; purpose      junior cassette interface for dos65 computer
                  ;
                  ; author       E.R.Elderenbosch
                  ;              De Rijpgracht 49'''
                  ;              1056 XS Amsterdam
                  ;              tel. 020-125386
                  ;
                  ; date         110187  junior cassette write routines
                  ;
                  ;
                          lib    caslib.mac
                  ;
                  ;
           00E0   sal     equ    $00e0
           00E1   sah     equ    sal+$01
           00E2   eal     equ    sal+$02
           00E3   eah     equ    sal+$03
           00E4   pointl  equ    sal+$04
           00E5   pointh  equ    sal+$05
           00E6   chkl    equ    sal+$06
           00E7   chkh    equ    sal+$07
           00E8   id      equ    sal+$08
           00E9   syncnt  equ    sal+$09
           00EA   bits    equ    sal+$0a
           00EB   acc     equ    sal+$0b
           00EC   count   equ    sal+$0c          ; 2 bytes
           00EE   byte    equ    sal+$0e
           00EF   char    equ    sal+$0f
           00F0   sy      equ    sal+$10
           00F1   higher  equ    sal+$11
           00F2   lower   equ    sal+$12
           00F3   first   equ    sal+$13
           00F4   second  equ    sal+$14
                  ;
           E118   vbtbcl  equ    $e118            ; timer 2 latch low, counter low
           E119   vbtbch  equ    $e119            ; timer 2 counter high
           E11B   vbacr   equ    $e11b            ; auxiliary control register
           E11D   vbifr   equ    $e11d            ; interrupt flag register
           E280   juncas  equ    $e280            ; junior cassette port
                                 bit 4 = output
                  ;
                  ;
                  ;
           0200           org    $0200
                  ;
0200 A0 00        start   ldy    #$00             ; dummy program to demonstrate
0202 84 E0                sty    sal              ;  how subroutines are used
0204 84 E2                sty    eal              ; save $2000 - $4000
0206 C8                   iny                     ;  with id = 01
0207 84 E8                sty    id               ; as junior tape format
0209 A9 20                lda    #$20
020B 85 E1                sta    sah
020D A9 40                lda    #$40
020F 85 E3                sta    eah
0211 20 0003              jsr    routine1
0214 60                   rts                     ; end of dummy program
                  ;
                  ;
           0300           org    $0300
                  ;
```

```
                     ;         vector table
                     ;
0300 4C 0F03         routine1 jmp    dump          ; write memory area
0303 4C 4704         routine2 jmp    wrpatrn       ; write alternate ones & zeros
0306 4C 1D04         routine3 jmp    wrsyncs       ; write long sync leader
0309 4C 7C04         routine4 jmp    testhi        ; write 3600 Hz continuously
030C 4C 8C04         routine5 jmp    testlo        ; write 2400 Hz continuously
                     ;
030F A9 6C           dump     lda    #$6c
0311 85 F1                    sta    higher
0313 A9 B0                    lda    #$b0
0315 85 F2                    sta    lower
0317 A9 03                    lda    #$03
0319 85 F3                    sta    first
031B A9 02                    lda    #$02
031D 85 F4                    sta    second
                     ;
031F 78              dumpt    sei
0320 A0 00                    ldy    #$00
0322 8C 1BE1                  sty    vbacr         ; set timer 2 in oneshot mode
0325 84 EB                    sty    acc
0327 84 E6                    sty    chkl          ; reset checksum
0329 84 E7                    sty    chkh
032B C8                       iny
032C 8C 19E1                  sty    vbtbch        ; activate timer 2 initially
032F A5 E0                    lda    sal           ; initialize dumpt pointer
0331 85 E4                    sta    pointl
0333 A5 E1                    lda    sah
0335 85 E5                    sta    pointh
0337 A2 FF                    ldx    #$ff          ; set sync counter
0339 86 E9                    stx    syncnt
                     ;
033B A9 16           syncs    lda    #$16          ; syn character
033D 20 B403                  jsr    outch         ; output 255 syn characters
0340 C6 E9                    dec    syncnt
0342 D0 F7                    bne    syncs
                     ;
0344 A9 2A                    lda    #'*           ; output start character
0346 20 B403                  jsr    outch
0349 A5 E8                    lda    id            ; output id
034B 20 9C03                  jsr    outbt
034E A5 E0                    lda    sal           ; output start address
0350 20 8F03                  jsr    outbtc        ; and start checksum computation
0353 A5 E1                    lda    sah
0355 20 8F03                  jsr    outbtc
                     ;
0358 A5 E5           datatr   lda    pointh
035A C5 E3                    cmp    eah           ; entire file transmitted?
035C D0 21                    bne    hexdat
035E A5 E4                    lda    pointl
0360 C5 E2                    cmp    eal
0362 D0 1B                    bne    hexdat
                     ;
0364 A9 2F                    lda    #'/           ; output end of data character
0366 20 B403                  jsr    outch         ; stop with check sum computation
0369 A5 E6                    lda    chkl          ; output checksum
036B 20 9C03                  jsr    outbt
036E A5 E7                    lda    chkh
0370 20 9C03                  jsr    outbt
0373 A9 04                    lda    #$04          ; eot character
0375 20 B403                  jsr    outch         ; output eot character
0378 A9 04                    lda    #$04          ; eot character
037A 20 B403                  jsr    outch
037D 58                       cli                  ; enable keyboard & clock again
037E 60                       rts
                     ;
037F A0 00           hexdat   ldy    #$00
0381 B1 E4                    lda    [pointl],y    ; fetch current data byte
0383 20 8F03                  jsr    outbtc        ; transmit current data byte
0386 E6 E4                    inc    pointl        ; and compute checksum
0388 D0 CE                    bne    datatr        ; setup for next data byte
038A E6 E5                    inc    pointh
038C 4C 5803                  jmp    datatr
                     ;
038F A8              outbtc   tay                  ; save accu
0390 18                       clc
0391 65 E6                    adc    chkl          ; checksum computation
0393 85 E6                    sta    chkl
0395 A5 E7                    lda    chkh
0397 69 00                    adc    #$00          ; chk := chk + byte
0399 85 E7                    sta    chkh
039B 98                       tya                  ; get accu again
039C A8              outbt    tay                  ; save accu temp
039D 4A                       lsra                 ; get upper nibble
039E 4A                       lsra
```

```
039F 4A                  lsra
03A0 4A                  lsra
03A1 20 AB03             jsr     nibout          ; output upper nibble as ascii char.
03A4 98                  tya                     ; get byte again
03A5 29 0F               and     #$0f            ; get lower nibble
03A7 20 AB03             jsr     nibout          ; output lower nibble as ascii char.
03AA 60                  rts
                ;
03AB C9 0A      nibout   cmp     #$0a            ; convert a nibble to an ascii char.
03AD 18                  clc
03AE 30 02               bmi     nib
03B0 69 07               adc     #$07
03B2 69 30      nib      adc     #$30
03B4 A2 08      outch    ldx     #$08            ; set up for 8 bits
03B6 86 EA               stx     bits
03B8 4A         one      lsra                    ; shift out bit by bit
03B9 48                  pha                     ; save character
03BA 90 0C               bcc     zero
03BC 20 D703             jsr     high            ; start at 3600 Hz
03BF 20 FA03             jsr     low
03C2 20 FA03             jsr     low             ; end at 2400 Hz
03C5 4C D103             jmp     zro
03C8 20 D703    zero     jsr     high            ; start at 3600 Hz
03CB 20 D703             jsr     high
03CE 20 FA03             jsr     low             ; end at 2400 Hz
03D1 68         zro      pla                     ; get character again
03D2 C6 EA               dec     bits            ; all bits shifted out?
03D4 D0 E2               bne     one
03D6 60                  rts
                ;
03D7 A6 F3      high     ldx     first           ; three half periods of 3600 Hz
03D9 A9 20      loop1    lda     #%00100000      ; get mask for timer 2 interrupt flag
03DB 2C 1DE1    2        bit     vbifr           ; timer 2 flag set?
03DE F0 FB               beq     2.              ; no, interval not completed
03E0 AD 18E1             lda     vbtbcl          ; clear interrupt flag
03E3 A5 EB               lda     acc
03E5 49 10               eor     #%00010000      ; 1 = bit to be toggled
03E7 85 EB               sta     acc
03E9 8D 80E2             sta     juncas
03EC A5 F1               lda     higher          ; 3600 Hz half cycle time
03EE 8D 18E1             sta     vbtbcl          ; timer 2 low
03F1 A9 00               lda     #$00
03F3 8D 19E1             sta     vbtbch          ; start timing of timer 2
03F6 CA                  dex
03F7 D0 E0               bne     loop1
03F9 60                  rts
                ;
03FA A6 F4      low      ldx     second          ; two half periods of 2400 Hz
03FC A9 20      loop2    lda     #%00100000      ; get mask for timer 2 int. flag
03FE 2C 1DE1    1        bit     vbifr           ; timer 2 flag set?
0401 F0 FB               beq     1.              ; no, interval not completed
0403 AD 18E1             lda     vbtbcl          ; clear interrupt flag
0406 A5 EB               lda     acc
0408 49 10               eor     #%00010000      ; 1 = bit to be toggled
040A 85 EB               sta     acc
040C 8D 80E2             sta     juncas
040F A5 F2               lda     lower           ; 2400 Hz half cycle time
0411 8D 18E1             sta     vbtbcl          ; timer 2 low
0414 A9 00               lda     #$00
0416 8D 19E1             sta     vbtbch          ; start timing of timer 2
0419 CA                  dex
041A D0 E0               bne     loop2
041C 60                  rts
                ;
                ; the following routines are for test purposes only
                ; and are adapted from junior book 3 routines.
                ;
041D 78         wrsyncs  sei                     ; write four minutes of syncs
041E A9 00               lda     #$00
0420 8D 1BE1             sta     vbacr
0423 A9 01               lda     #$01
0425 8D 19E1             sta     vbtbch
0428 A9 A0               lda     #$a0            ; (00 for eleven minutes)
042A 85 EC               sta     count
042C 85 ED               sta     count+1
042E 18         1        clc
042F A9 01               lda     #$01
0431 65 EC               adc     count
0433 85 EC               sta     count
0435 A9 00               lda     #$00
```

```
0437 65 ED                adc     count+1
0439 85 ED                sta     count+1
043B B0 08                bcs     wrend
043D A9 16                lda     #$16              ; sync character
043F 20 B403              jsr     outch             ; output to tape
0442 4C 2E04              jmp     1.
0445 58          wrend    cli
0446 60                   rts
                 ;
0447 78          wrpatrn  sei                       ; write a few minutes alternate ones & z
0448 A9 00                lda     #$00
044A 8D 1BE1              sta     vbacr
044D A9 01                lda     #$01
044F 8D 19E1              sta     vbtbch
0452 A9 00                lda     #$00
0454 85 EC                sta     count
0456 85 ED                sta     count+1
0458 18          2        clc
0459 A9 01                lda     #$01
045B 65 EC                adc     count
045D 85 EC                sta     count
045F A9 00                lda     #$00
0461 65 ED                adc     count+1
0463 85 ED                sta     count+1
0465 B0 DE                bcs     wrend
0467 20 D703              jsr     high
046A 20 FA03              jsr     low
046D 20 FA03              jsr     low
0470 20 D703              jsr     high
0473 20 D703              jsr     high
0476 20 FA03              jsr     low
0479 4C 9604              jmp     2.
                 ;
                 ; the following routines are usefull for measuring the
                 ; 2400 & 3600 Hz frequencies. (adjust with higher & lower)
                 ;
047C 78          testhi   sei
047D A0 00                ldy     #$00
047F 8C 1BE1              sty     vbacr             ; set timer 2 oneshot
0482 C8                   iny
0483 8C 19E1              sty     vbtbch            ; start timer initially
0486 20 D703     1        jsr     high
0489 4C 8604              jmp     1.
                 ;
048C 78          testlo   sei
048D A0 00                ldy     #$00
048F 8C 1BE1              sty     vbacr             ; set timer 2 oneshot
0492 C8                   iny
0493 8C 19E1              sty     vbtbch            ; start timer initially
0496 20 FA03     2        jsr     low
0499 4C 9604              jmp     2.
                 ;
           0200           end     start
```

Junior tape load routines on the DOS65 computer.

Like with the tape save routines i have tried to make as few changes as possible to the original junior routines so that the old routines can be easily exchanged for the new ones. There are a few changes that had to be made however, because there is no hex display on the DOS65 computer. I have connected three leds to the outputs of three inverters (7406). The inputs of the inverters are connected to the 74173 (IC 4) in the following way :

```
red    led  -  pin2 7406 pin1  -  print hole marked S6.
orange led  -  pin4 7406 pin3  -  print hole marked S7.
green  led  -  pin6 7406 pin5  -  print hole marked S8.
three resistors of 470 ohm from the leds to the +5V.
```

When the program is started, the leds are off. As soon as one bit is received, the red led comes on, indicating that the the program is searching for syncs. When syncs are found, the orange led comes on. If you have a bad tape, the red led comes on again. If the start character is found, the green led is lit. After the data is loaded, the leds are off again. If you cannot read the tape, you have to re-boot the system because control-C doesn't work during running of the load routines. This is necessary for the timing. Just like in the save routines, i use timer 2 of the second 6522 on the cpu board. This change is also necessary because there is no 6532 in the standard system. Instead of one timer in the original program, clocked by the system clock/64, i have used both high and low parts of timer 2, clocked by the system clock. After having detected a change in inputsignal, i only check the high byte of the timer. This should be sufficient. Good luck!

======================

```
                 ; file           junread.mac
                 ;
                 ; purpose        junior cassette interface for dos65 computer
                 ;
                 ; author         E.R.Elderenbosch
                 ;
                 ; date           220187  cassette junior read routines
                 ;
                 ;
                         lib      caslib.mac
                 ;
                 ;
           00E0  sal     equ      $00e0
           00E1  sah     equ      sal+$01
           00E2  eal     equ      sal+$02
           00E3  eah     equ      sal+$03
           00E4  pointl  equ      sal+$04
           00E5  pointh  equ      sal+$05
           00E6  chkl    equ      sal+$06
           00E7  chkh    equ      sal+$07
           00E8  id      equ      sal+$08
           00E9  syncnt  equ      sal+$09
           00EA  bits    equ      sal+$0a
           00EB  acc     equ      sal+$0b
           00EC  count   equ      sal+$0c         ; 2 bytes
           00EE  byte    equ      sal+$0e
           00EF  char    equ      sal+$0f
           00F0  sy      equ      sal+$10
           00F1  higher  equ      sal+$11
           00F2  lower   equ      sal+$12
           00F3  first   equ      sal+$13
           00F4  second  equ      sal+$14
           00B0  timelo  equ      $00b0
           006C  timehi  equ      $006c
                 ;
           E118  vbtbcl  equ      $e118           ; timer 2 latch low, counter low
           E119  vbtbch  equ      $e119           ; timer 2 counter high
           E11B  vbacr   equ      $e11b           ; auxiliary control register
           E11D  vbifr   equ      $e11d           ; interrupt flag register
                 ;
           E280  juncas  equ      $e280           ; junior cassette port
                 ;                bit 4 = output
                 ;                bit 0 = input
                 ;
           0200          org      $0200
                 ;
                 ;
0200 A9 00       start   lda      #$00            ; dummy program to load
0202 85 E0               sta      sal             ; to $2000 upwards
0204 A9 20               lda      #$20            ; in memory
0206 85 E1               sta      sah
0208 A9 FF               lda      #$ff            ; set tape id = ff
020A 85 E8               sta      id
020C 20 1002             jsr      rdtape          ; start reading tape
020F 60                  rts                      ; end of dummy program
                 ;
0210 78          rdtape  sei
0211 20 5003             jsr      noled           ; turn leds off
0214 A9 00               lda      #$00
0216 8D 1BE1             sta      vbacr           ; set timer 2 oneshot
0219 85 E6               sta      chkl
021B 85 E7               sta      chkh
021D A9 FF       sync    lda      #$ff            ; reset for syn character
021F 85 EF               sta      char
0221 20 BC02     synca   jsr      rdbit           ; read a bit from tape
0224 66 EF               ror      char            ; right shift
0226 A5 EF               lda      char            ; get current character
0228 20 5603             jsr      searled         ; display searching (red led)
022B C9 16               cmp      #$16            ; syn character?
022D D0 F2               bne      synca           ; if not, resync
022F A0 0A               ldy      #$0a            ; try it for 10 syncs at least
0231 84 F0               sty      sy              ; sync counter
0233 20 3103     tensyn  jsr      rdch
0236 20 5C03             jsr      syncled         ; display syn character (orange led)
0239 C9 16               cmp      #$15            ; still sync character?
023B D0 E0               bne      sync            ; if not, return
023D C6 F0               dec      sy              ; 10 syncs received?
023F D0 F2               bne      tensyn          ; return if less than 10 syncs
0241 20 3103     star    jsr      rdch            ; wait for '*' character
0244 20 5C03             jsr      syncled         ; display syn character (orange led)
```

```
0247 C9 2A                cmp    #'*
0249 F0 06                beq    stara
024B C9 16                cmp    #$16            ; still sync character?
024D F0 F2                beq    star            ; if yes, then wait
024F D0 BF                bne    rdtape          ; if not, then resync
0251 20 6203    stara     jsr    dataled         ; display data (green led)
0254 20 F002              jsr    rdbyt           ; read id from tape
0257 C5 E8                cmp    id              ; requested id?
0259 D0 40                bne    chkid
025B 20 F002    rdsa      jsr    rdbyt           ; read sal from tape
025E 20 4203              jsr    chksum          ; checksum computation
0261 85 E4                sta    pointl          ; setup store pointer
0263 20 F002              jsr    rdbyt           ; read sah from tape
0266 20 4203              jsr    chksum
0269 85 E5                sta    pointh
026B 20 F002    filmem    jsr    rdbyt           ; read data byte from tape
026E 30 A0                bmi    rdtape          ; not valid hex character
0270 F0 13                beq    check           ; end of data character?
0272 20 4203              jsr    chksum
0275 A0 00                ldy    #$00
0277 91 E4                sta    [pointl],y      ; store byte in memory
0279 E6 E4                inc    pointl          ; set pointer for next byte
027B D0 02                bne    fma
027D E6 E5                inc    pointh
027F 20 6203    fma       jsr    dataled         ; display data (green led)
0282 4C 6B02              jmp    filmem          ; read next data byte from tape
0285 20 F002    check     jsr    rdbyt           ; read checksum from tape
0288 C5 E6                cmp    chkl            ; and compare it
028A D0 0C                bne    synvec
028C 20 F002              jsr    rdbyt
028F C5 E7                cmp    chkh
0291 D0 05                bne    synvec
0293 20 5003              jsr    noled           ; turn leds off
0296 58                   cli
0297 60                   rts                    ; return to caller
0298 4C 1002    synvec    jmp    rdtape
029B A5 E8      chkid     lda    id
029D C9 00                cmp    #$00            ; id = 00?
029F F0 BA                beq    rdsa
02A1 C9 FF                cmp    #$ff            ; id = ff?
02A3 D0 F3                bne    synvec
02A5 20 F002              jsr    rdbyt           ; read sa from tape, but ignore it
02A8 20 4203              jsr    chksum
02AB 20 F002              jsr    rdbyt
02AE 20 4203              jsr    chksum
02B1 A5 E0                lda    sal             ; use sa stored in buffer
02B3 85 E4                sta    pointl
02B5 A5 E1                lda    sah
02B7 85 E5                sta    pointh
02B9 4C 6B02              jmp    filmem
                ;
02BC A9 01      rdbit     lda    #%00000001
02BE 2C 80E2    1         bit    juncas          ; 3600 Hz?
02C1 D0 FB                bne    1.
02C3 AD 19E1              lda    vbtbch          ; load timer 2 high byte
02C6 85 EB                sta    acc
02C8 A0 FF                ldy    #$ff            ; initial timer value $ffff
02CA 8C 18E1              sty    vbtbcl          ; store timer 2 low
02CD 8C 19E1              sty    vbtbch          ; store timer 2 high & start timing
02D0 A0 14                ldy    #$14            ; jitter time
02D2 88         rdba      dey                    ; delay jitter time
02D3 D0 FD                bne    rdba
02D5 A9 01      rdbb      lda    #%00000001
02D7 2C 80E2    2         bit    juncas          ; 2400 Hz?
02DA F0 FB                beq    2.
02DC 38                   sec
02DD A5 EB                lda    acc
02DF ED 19E1              sbc    vbtbch          ; set or reset c-flag
02E2 A0 FF                ldy    #$ff
02E4 8C 18E1              sty    vbtbcl
02E7 8C 19E1              sty    vbtbch
02EA A0 07                ldy    #$07            ; delay for jitter
02EC 88         rdbc      dey
02ED D0 FD                bne    rdbc
02EF 60                   rts
                ;
02F0 20 3103    rdbyt     jsr    rdch            ; read any ascii character from tape
02F3 C9 2F                cmp    #'/             ; end of data character?
02F5 D0 01                bne    rbb
02F7 60         rba       rts                    ; error exit
02F8 20 1403    rbb       jsr    aschex          ; ascii hex conversion
02FB 30 FA                bmi    rba             ; not valid character
02FD 0A                   asla                   ; shift nibble to left
```

```
02FE 0A                       asla
02FF 0A                       asla
0300 0A                       asla
0301 85 EE                    sta     byte          ; save high order nibble
0303 20 3103                  jsr     rdch          ; read next character
0306 C9 2F                    cmp     #'/           ; end of data character
0308 F0 ED                    beq     rba
030A 20 1403                  jsr     aschex        ; ascii hex conversion
030D 30 E8                    bmi     rba           ; not valid character
030F 05 EE                    ora     byte          ; byte = high order and low order nibble
0311 A0 01                    ldy     #$01          ; be shure that character
0313 60                       rts                   ; normal exit
                    ;
0314 C9 30          aschex    cmp     #$30          ; ignore 00...2f
0316 30 0C                    bmi     notval
0318 C9 3A                    cmp     #$3a
031A 30 0B                    bmi     valid
031C C9 41                    cmp     #$41          ; ignore 3a...40
031E 30 04                    bmi     notval
0320 C9 47                    cmp     #$47          ; ignore 47...7f
0322 30 03                    bmi     valid
0324 A0 FF          notval    ldy     #$ff          ; set n-flag
0326 60                       rts                   ; error exit
0327 C9 40          valid     cmp     #$40          ; ascii hex conversion
0329 30 03                    bmi     val
032B 18                       clc
032C 69 09                    adc     #$09
032E 29 0F          val       and     #$0f          ; hex data is low order nibble in accu
0330 60                       rts
                    ;
0331 A2 08          rdch      ldx     #$08          ; set up for 8 bits
0333 20 BC02        read      jsr     rdbit         ; read a bit from tape
0336 66 EF                    ror     char          ; shift bit into character
0338 CA                       dex                   ; all bits read?
0339 D0 F8                    bne     read
033B 26 EF                    rol     char          ; b7 must be zero
033D 46 EF                    lsr     char
033F A5 EF                    lda     char          ; received character to accu
0341 60                       rts
                    ;
0342 48             chksum    pha                   ; save accu
0343 18                       clc
0344 65 E6                    adc     chkl          ; chk := chk + byte
0346 85 E6                    sta     chkl
0348 A5 E7                    lda     chkh
034A 69 00                    adc     #$00
034C 85 E7                    sta     chkh
034E 68                       pla                   ; get accu again
034F 60                       rts
                    ;
0350 A0 00          noled     ldy     #$00
0352 8C 80E2                  sty     juncas
0355 60                       rts
                    ;
0356 A0 20          searled   ldy     #$20
0358 8C 80E2                  sty     juncas
035B 60                       rts
                    ;
035C A0 40          syncled   ldy     #$40
035E 8C 80E2                  sty     juncas
0361 60                       rts
                    ;
0362 A0 80          dataled   ldy     #$80
0364 8C 80E2                  sty     juncas
0367 60                       rts
                    ;
           0200               end     start
                                      label table
```

```
0 LIST
SCR # 0
  0 ****************  FIG-FORTH MODEL  *****************************
  1
  2                 THROUGH THE COURTESY OF
  3
  4                  FORTH INTEREST GROUP
  5                     P. O. BOX 1105
  6                 SAN CARLOS. CA. 94070
  7
  8                     RELEASE V1.1
  9            COMPILER SECURITY & VARIABLE LENGTH NAMES
 10
 11      FURTHER DISTRIBUTION MUST INCLUDE THE ABOVE NOTICE
 12
 13                 REDAKTIE "DE 6502 KENNER"
 14                 JACOB JORDAENSSTRAAT 15
 15                 2923 CK KRIMPEN A.D. IJSSEL.
OK
6 LIST
SCR # 6
  0 ( FIG-FORTH DECOMPILER )
  1 ( CASE CONTROL STATEMENT BY CHARLES E. EAKER )
  2 ( PUBLISHED IN FORTH DIMENSIONS II/3 PAGE 37 )
  3 FORTH DEFINITIONS DECIMAL
  4 : CASE          ?COMP CSP @ !CSP 4 ; IMMEDIATE
  5 : OF            4 ?PAIRS
  6                 COMPILE OVER COMPILE =
  7                 COMPILE OBRANCH HERE 0 .
  8                 COMPILE DROP 5 ; IMMEDIATE
  9 : ENDOF         5 ?PAIRS
 10                 COMPILE BRANCH HERE 0 .
 11                 SWAP 2 [COMPILE] ENDIF 4 ; IMMEDIATE
 12 : ENDCASE       4 ?PAIRS COMPILE DROP
 13                 BEGIN SP@ CSP @ = 0=
 14                 WHILE 2 [COMPILE] ENDIF REPEAT
 15                 CSP ! ; IMMEDIATE                -->
OK
7 LIST
SCR # 7
  0 ( FIG-FORTH DECOMPILER )
  1 0 VARIABLE QUIT.FLAG      0 VARIABLE WORD.PTR
  2 ( FIND RUN-TIME ADDRESSES OF EACH VOCABULARY WORD TYPE )
  3 ' (LOOP)        2 - CONSTANT    LOOP.ADR
  4 ' LIT           2 - CONSTANT    LIT.ADR
  5 ' :             2 - @ CONSTANT  DOCOL.ADR
  6 ' OBRANCH       2 - CONSTANT    OBRANCH.ADR
  7 ' BRANCH        2 - CONSTANT    BRANCH.ADR
  8 ' (+LOOP)       2 - CONSTANT    PLOOP.ADR
  9 ' (.")          2 - CONSTANT    PDOTQ.ADR
 10 ' C/L           2 - @ CONSTANT  CONST.ADR
 11 ' BASE          2 - @ CONSTANT  USERV.ADR
 12 ' USE           2 - @ CONSTANT  VAR.ADR
 13 ' (;CODE)       2 - CONSTANT    PSCODE.ADR
 14 -->
 15
OK
8 LIST
SCR # 8
  0 ( FIG-FORTH DECOMPILER )
  1
  2 : N.            ( PRINT A NUMBER IN DECIMAL AND HEX )
  3                 DUP DECIMAL . SPACE
  4                 HEX 0 ." ( " D. ." H ) "  DECIMAL ;
  5
  6 : PDOTQ.DSP     ( DISPLAY A COMPILED TEXT STRING )
  7                 WORD.PTR @ 2+ DUP >R DUP
  8                 C@ + 1 - WORD.PTR ! ( UPDATE PFA POINTER )
  9                 R> COUNT TYPE ;
 10
 11 : WORD.DSP      ( GIVEN CFA. DISPLAY THE GLOSSARY NAME )
 12                 3 - -1 TRAVERSE DUP 1+ C@ 59 =
 13                 IF 1 QUIT.FLAG ! ENDIF ( IF ";" WE ARE DONE )
 14                 DUP C@ 160 AND 128 =  ( MAKE SURE LEGAL NFA )
 15                 IF ID. ELSE 1 QUIT.FLAG ! ENDIF ;        -->
OK
```

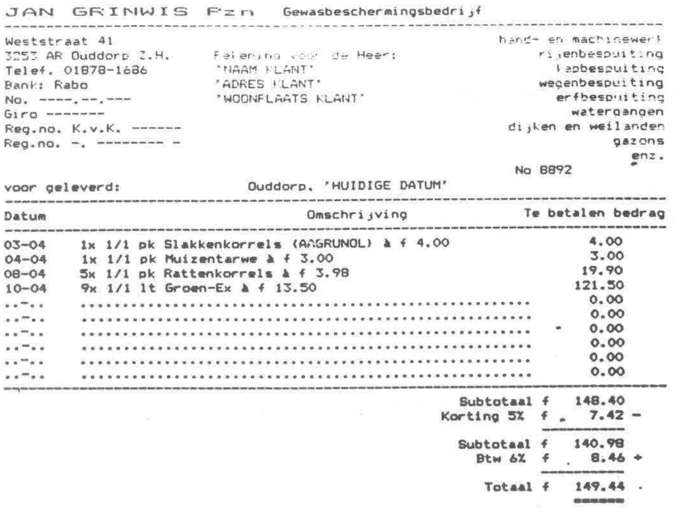
```
9 LIST
SCR # 9
  0 ( FIG-FORTH DECOMPILER )
  1
  2 : BRANCH.DSP      ( GET BRANCH OFFSET. CALCULATE THE )
  3                   ( ACTUAL BRANCH ADDRESS. AND DISPLAY IT )
  4                   ." TO "
  5                   WORD.PTR @ 2+ DUP WORD.PTR ! ( UPDATE PFA PTR )
  6                   DUP @ +    ( OFFSET + PFA = ACTUAL TARGET ADDR )
  7                   0 HEX D. DECIMAL ( PRINT IT )    ;
  8
  9 : USERV.DSP       ( DISPLAY A USER VARIABLE )
 10                   ." USER VARIABLE. CURRENT VALUE = "
 11                   WORD.PTR @ 2+   ( CALCULATE PFA )
 12                   C@ 38 +ORIGIN @ + ( THEN USER AREA ADDRESS )
 13                   @ N.   ( FETCH AND PRINT CONTENTS )
 14                   1 QUIT.FLAG ! ( DONE. SET FLAG )   ;
 15 -->
OK
10 LIST
SCR # 10
  0 ( FIG-FORTH DECOMPILER )
  1
  2 : VAR.DSP         ( DISPLAY A VARIABLE )
  3                   ." VARIABLE. CURRENT VALUE = "
  4                   WORD.PTR @ 2+   ( CALCULATE PFA )
  5                   @ N.       ( FETCH AND PRINT CONTENTS )
  6                   1 QUIT.FLAG ! ( ALL DONE. SET FLAG )   ;
  7
  8 : CONST.DSP       ( DISPLAY A COMPILED CONSTANT )
  9                   ." CONSTANT. VALUE = "
 10                   WORD.PTR @ 2+ ( CALCULATE PFA )
 11                   @ N.       ( FETCH AND PRINT CONTENTS )
 12                   1 QUIT.FLAG ! ( ALL DONE. SET FLAG )   ;
 13 -->
 14
 15
OK
11 LIST
SCR # 11
  0 ( FIG-FORTH DECOMPILER )
  1 : DIS
  2   -FIND 0=            ( IS INPUT WORD IN DICTIONARY? )
  3   IF 3 SPACES ." ? NOT IN GLOSSARY " CR      ( NO. QUIT )
  4   ELSE DROP DUP DUP 2 -    ( YES. CALCULATE CFA )
  5   @ =      ( IF CONTENTS OF CFA = PFA THEN IT IS A PRIMITIVE )
  6   IF ." <PRIMITIVE> " CR      ( SO PRINT MESSAGE AND QUIT )
  7   ELSE    ( OTHERWISE IT'S HIGH LEVEL FORTH SO DECODE IT )
  8   0 QUIT.FLAG !       ( INITIALIZE DONE FLAG )
  9   2 - WORD.PTR !      ( INITIALIZE PSEUDOCODE POINTER )
 10   CR CR               ( PRINT SOME BLANK LINES )
 11   BEGIN               ( NOW LIST THE COMPILED PSEUDOCODE )
 12   WORD.PTR @ DUP      ( FETCH CURRENT PSEUDOCODE POINTER )
 13   0 HEX D. SPACE DECIMAL ( PRINT VALUE OF POINTER )
 14   @                   ( FETCH CURRENT PSEUDOCODE WORD )
 15 -->
OK
12 LIST
SCR # 12
  0 ( FIG-FORTH DECOMPILER )
  1 CASE              ( NOW DECODE ANY SPECIAL WORD TYPES )
  2 LIT.ADR OF        ( COMPILED LITERAL. PRINT ITS VALUE )
  3                   WORD.PTR @ 2+ DUP WORD.PTR ! @ N. ENDOF
  4 DOCOL.ADR OF      ( : POINTS TO THE NESTING ROUTINE )
  5                   ." : "      ENDOF      ( SO JUST PRINT A COLON )
  6 0BRANCH.ADR OF    ( CONDITIONAL BRANCH WITH IN-LINE OFFSET )
  7                   ." BRANCH IF ZERO "    BRANCH.DSP ENDOF
  8 BRANCH.ADR OF     ( UNCONDITIONAL BRANCH WITH IN-LINE OFFSET )
  9                   ." BRANCH "    BRANCH.DSP ENDOF
 10 LOOP.ADR OF       ( END OF A DO...LOOP STRUCTURE )
 11                   ." LOOP "    BRANCH.DSP ENDOF
 12 PLOOP.ADR OF      ( END OF A DO...+LOOP STRUCTURE )
 13                   ." +LOOP "    BRANCH.DSP ENDOF
 14 -->
 15
OK
```

```
13 LIST
SCR # 13
  0 ( FIG-FORTH DECOMPILER )
  1 PDOTQ.ADR OF      ( DISPLAY COMPILE TEXT STRING )
  2                   ." PRINT TEXT: "    PDOTQ.DSP ENDOF
  3 USERV.ADR OF      ( DISPLAY A USER VARIABLE )
  4                   USERV.DSP ENDOF
  5 VAR.ADR OF        ( DISPLAY A GLOBAL VARIABLE )
  6                   VAR.DSP ENDOF
  7 CONST.ADR OF      ( DISPLAY A COMPILED CONSTANT )
  8                   CONST.DSP ENDOF
  9 PSCODE.ADR OF     ( DISPLAY :CODE AND QUIT )
 10                   WORD.PTR @ @ WORD.DSP
 11                   1 QUIT.FLAG ! ENDOF
 12 --)
 13
 14
 15
OK
14 LIST
SCR # 14
  0 ( FIG-FORTH DECOMPILER )
  1
  2                            ( ALL SPECIAL WORD TYPES CHECKED. )
  3 DUP WORD.DSP               ( IF WORD DID NOT MATCH ANY CASES )
  4                            ( JUST PRINT ITS NAME )
  5 ENDCASE CR                 ( DONE DECODING WORD TYPE )
  6 2 WORD.PTR +!              ( UPDATE PSEUDOCODE POINTER )
  7 QUIT.FLAG @                ( CHECK IF FINISHED FLAG SET OR IF )
  8 ?TERMINAL OR               ( INTERRUPTION FROM THE TERMINAL )
  9 UNTIL                      ( OTHERWISE DISPLAY ANOTHER WORD )
 10 ENDIF ENDIF CR :           ( ALL DONE NOW )
 11
 12
 13
 14
 15
OK
15 LIST
SCR # 15
  0 ( FIG-FORTH DECOMPILER )
  1
  2 NFA    = NAME FIELD ADDRESS
  3 CFA    = CODE FIELD ADDRESS
  4 PFA    = PARAMETER FIELD ADDRESS
  5
  6        EXAMPLES :
  7
  8        DIS XXX       ? NOT IN GLOSSARY
  9
 10        DIS C/L       CONSTANT. VALUE = 64   ( 40 H )
 11
 12        DIS DUP       (PRIMITIVE)
 13
 14                                        GOOD LUCK !
 15
OK
```

```
 10REM ########################################
 20REM ### REKENING 09 MSX / 06 april 1987 ###
 30REM ########################################
 40REM ### Peter Grinwis & Simon Voortman  ###
 50REM ### Beatrixweg 28  3253 BB OUDDORP  ###
 60REM ########################################
 70REM ### Made on Acorn Electron 64k with ###
 80REM ###        MSX or STAR printer      ###
 90REM ########################################
100
110MODE0:DIM DATE$(21),OMSCHR$(21),BEDRAG$(21):@%=&90A:data%=0:T%=0:p%=0
120VDU23,133,8,4,62,6,62,102,62,0:REM Define an 'a' with \ on it for MSX or
130VDU23,193,8,4,62,6,62,102,62,0:REM for STAR (on screen only)
140A%=0:B%=0:C%=0:D%=0:Z=0:Bl=6:Bh=20:naam$="":adres$="":place$="":REK%=0
150REPEAT:A%=A%+1:DATE$(A%)="":OMSCHR$(A%)="":BEDRAG$(A%)="":UNTIL A%=21
160REM ################################
170REM # Create strings for heading #
180REM ################################
190F$=CHR$17:ION$=F$+CHR$0+F$+CHR$129:REM Inverse video on
200IOF$=F$+CHR$1+F$+CHR$128:REM Inverse video off
210R$="R E K E N I N G E N"
220VERKOPER$="JAN GRINWIS Pzn":BEDRIJF$="Gewasbeschermingsbedrijf"
230ADRES$="Weststraat 41":ADRES2$="3253 AR Ouddorp Z.H.":BANK$="Bank: Rabo"
240TELF$="Telef. 01878-1686":NR$="No. ----.--.---":GIRO$="Giro -------"
250REGKvK$="Reg.no. K.v.K. ------":REGL$="Reg.no. -. -------- -"
260W1$="hand- en machinewerk":W2$="rijenbespuiting":W3$="kapbespuiting"
270W4$="wegenbespuiting":W5$="erfbespuiting":W6$="watergangen"
280W7$="dijken en weilanden":W8$="gazons":W9$="enz."
290
300PRINTTAB(30,1)R$
310PRINTTAB(23,2);ION$"P R I N T E R  I N S T E L L I N G";IOF$
320PRINT''"PRINTER aangesloten (J/N): ";:c%=GET AND&DF
330IF (c%=74) OR (c%=89) PRINT"Ja":c%=1 ELSE PRINT"Nee":c%=0:GOTO 390
340PRINT'"Is het MSX  (1)"'SPC4"of STAR (2): ";:T%=GET-48:IF T%<>1 T%=2
350IF T%=2 PRINT"STAR" ELSE PRINT"MSX"
360PRINT'"Papier lengte 11 inch (1)"'SPC11"of 12 inch (2): ";
370REPEAT:L%=GET-48:UNTIL L%=1 OR L%=2
380IF L%=1 PRINT"11 inch" ELSE PRINT"12 inch"
390d%=INKEY(200):REM Wait 2 seconds
400
410REM MENU
420REPEAT:CLS:X%=37:PRINT'TAB(X%-7);R$
430PRINTTAB(X%,4);ION$;SPC6;TAB(X%,5);" MENU ";TAB(X%,6);SPC6;IOF$;'''
440RESTORE530:READ D%
450FOR ch%=1 TO D%:READ CH$:PRINT'TAB(X%-5);ION$;ch%;IOF$;". ";CH$:NEXT
460PRINT'''TAB(X%-3)"Uw keuze...";
470REPEAT:ch%=GET-48:UNTIL ch%>0 AND ch%<=D%
480IF ch%=1 PROCinput
490IF ch%=2 PROCprint
500IF ch%=3 PROCold_data
510UNTIL ch%=4:CLS:END
520
530DATA 4,Nieuwe rekening,Printen,Oude data,Stoppen
540
550DEFPROCinput:CLS:A%=0:B%=0:C%=0:D%=0:Z=0:Bl=6:Bh=20
560naam$="":adres$="":place$="":datum$="":REK%=0
570PRINTTAB(3,3)"REKENINGEN":@%=&90A
580INPUT''"Rekening voor "naam$
590INPUT"Adres "adres$
600INPUT"Plaats "place$
610INPUT"Datum "datum$
620INPUT"Rekening "REK%
630INPUT"Btw (bv. 20 ) % "BTW$
640INPUT"Korting J/N ";K$:ko=FALSE
650IF INSTR("Jj",K$) THEN ko=TRUE:INPUT"Hoeveel korting (bv. 5 ) % "kort$
660INPUT"Verkoop (V) of gewerkt (G)";:vg=GET AND&DF:vg$=CHR$(vg):a$=" a f "
670IF vg$="V" THEN VK=TRUE ELSE VK=FALSE
```

```
680IF T%=2 THEN a$=" "+CHR$193+" f " ELSE IF T%=1 THEN a$=" "+CHR$133+" f "
690CLS:PRINT''"<RETURN> bij datum -> Terug naar menu"'':A%=0
700REPEAT:A%=A%+1:DATE$(A%)="":OMSCHR$(A%)="":BEDRAG$(A%)="0"
710PRINT;A%;:INPUT" Datum "DATE$(A%)
720IF DATE$(A%)="" THEN GOTO 810
730IF NOT VK THEN 760
740INPUTSPC(3)"Aantal "num%
750INPUTLINESPC(3)"Omschrijving "omschr$
760IF NOT VK THEN INPUTLINESPC(3)"Omschrijving "OMSCHR$(A%)
770IF NOT VK THEN INPUTLINESPC(3)"Bedrag "BEDRAG$(A%):GOTO 810
780INPUTLINESPC(3)"Bedrag per stuk "bedrag$
790bedrag=VAL(bedrag$):nbedr=bedrag*num%:BEDRAG$(A%)=STR$(nbedr)
800OMSCHR$(A%)=STR$(num%)+"x "+omschr$+a$+bedrag$
810UNTIL DATE$(A%)="":A%=A%-1:N_ART=A%:data%=1
820IF N_ART<10 THEN REPEAT:A%=A%+1:DATE$(A%)="..-..":OMSCHR$(A%)=STRING$(54,"."):BEDRAG$="0":UNTIL A%=10
830ENDPROC
840
850DEFPROCprint:CLS:IF data%<>1 PRINTTAB(31,16)"Geen data aanwezig"''TAB(30)"Druk op een toets..."::d%=GET:ENDPROC
860IF c%=0 VDU3:GOTO 950:REM No printer connected, so no printed output
870PRINT''''"PRINTER on (1) OR off (0)";:p%=GET-48:IF p%<>1 p%=0
880CLS:IF (p%=1 AND c%=1) VDU2
890IF T%=2 PROCstar:GOTO 970
900OSCLI"FX6":REM Generate extra CR's for MSX printer
910IF L%=2:REM MSX code for 12 inch papier (I don't know by now)
920REM ******************************
930REM * Output to screen & printer *
940REM ******************************
950VDU1,27,1,ASC"!",1,14:PRINT'VERKOPER$;:VDU1,15:REM Enlarged/Condensed
960VDU1,27,1,ASC"C",1,ASC"B":PRINTSPC3;BEDRIJF$:VDU1,27,1,ASC"C",1,ASC"b"
970PRINTSTRING$(79,CHR$45)
980PRINTADRES$;TAB(59);W1$'ADRES2$;SPC5;"Rekening voor de Heer:";TAB(64);W2$
990PRINTTELF$;TAB(25);naam$;TAB(66)W3$'BANK$;TAB(25)adres$;TAB(64)W4$
1000PRINTNR$;TAB(25);place$;TAB(66);W5$'GIRO$;TAB(68)W6$'REGKvK$;TAB(60)W7$
1010PRINTREGL$;TAB(73)W8$'TAB(75)W9$'TAB(61);"No ";REK%
1020PRINT"voor geleverd:";SPC15;"Ouddorp, ";datum$'STRING$(79,CHR$45)
1030PRINT"Datum ";SPC30;"Omschrijving";SPC14;"Te betalen bedrag":totaal=0
1040PRINTSTRING$(79,CHR$45):@%=&2020A:B%=0:subtotaal=0
1050
1060REM Print date, description and price for line B%, calculate subtotal
1070FOR B%=1 TO A%:bedrag=VAL(BEDRAG$(B%))
1080PRINT DATE$(B%);TAB(9);OMSCHR$(B%);TAB(74-FNpos(bedrag));bedrag
1090subtotaal=subtotaal+bedrag
1100NEXT B%
1110PRINTSTRING$(79,CHR$45)'TAB(54)"Subtotaal f";
1120
1130subtotaal=INT(subtotaal*100+0.5)/100
1140PRINTTAB(74-FNpos(subtotaal));subtotaal
1150IF ko THEN PROCkorting(subtotaal)
1160PROCbtw(subtotaal)
1170PRINTTAB(64);STRING$(10,CHR$45)
1180PRINTTAB(57)"Totaal f";TAB(74-FNpos(totaal));totaal
1190PRINTTAB(74-FNpos(totaal));STRING$(FNpos(totaal),"="):@%=&90A
1200IF T%=1 VDU1,27,1,34,3 ELSE IF T%=2 VDU1,12,1,27,1,64,3:REM FormFeed & printer reset
1210PRINT''"Druk op een toets":d%=GET:ENDPROC
1220
1230DEFPROCold_data
1240naam$="'NAAM KLANT'":datum$="'HUIDIGE DATUM'":BTW$="6":kort$="5":Z=0
1250adres$="'ADRES KLANT'":place$="'WOONPLAATS KLANT'":REK%=1000+RND(9000):ko=TRUE:BTW=20
1260IF T%=2 THEN a$=CHR$193 ELSE IF T%=1 THEN a$=CHR$133 ELSE a$="a"
1270RESTORE1400:REPEAT:Z=Z+1:READ D$:IF D$="-1" GOTO1310
1280READ NR,Q$,ART$,P$
1290DATE$(Z)=D$:OMSCHR$(Z)=STR$(NR)+"x "+Q$+" "+ART$+" "+a$+" f "+P$
1300BEDRAG$(Z)=STR$(INT(NR*VAL(P$)*100+.5)/100)
1310UNTIL D$="-1":Z=Z-1:N_ART=Z:A%=Z:data%=1
1320IF N_ART<10 THEN REPEAT:A%=A%+1:DATE$(A%)="..-..":OMSCHR$(A%)=STRING$(54,"."):BEDRAG$="0":UNTIL A%=10
1330ENDPROC
1340
```

```
1350DEFFNpos(value):value%=value*100:IF value%<100 THEN =4
1360=LEN(STR$(value%))+1
1370
1380DATA 01-04,2,1/2,lt Groen-Ex,7.00
1390DATA 02-04,3,1/1,sp Vlido,4.00
1400DATA 03-04,1,1/1,pk Slakkenkorrels (AAGRUNOL),4.00
1410DATA 04-04,1,1/1,pk Muizentarwe,3.00
1420DATA 08-04,5,1/1,pk Rattenkorrels,3.98
1430DATA 10-04,9,1/1,lt Groen-Ex,13.50
1440DATA -1
1450
1460DEFPROCkorting(money)
1470KOR=(money/100)*VAL(kort$)
1480KOR=INT(KOR*100+0.5)/100
1490PRINTTAB(52);"Korting ";kort$;"%  f";TAB(74-FNpos(KOR));KOR;" -"
1500subtotaal=money-KOR
1510PRINTTAB(64);STRING$(10,CHR$45)
1520PRINTTAB(54);"Subtotaal f ";TAB(74-FNpos(subtotaal));subtotaal
1530ENDPROC
1540
1550DEFPROCbtw(money):@%=&2020A
1560PRINTTAB(56);"Btw ";BTW$;"%";TAB(64)"f";:btw=(money/100)*VAL(BTW$)
1570btw=INT(btw*100+0.5)/100:PRINTTAB(74-FNpos(btw));btw;" +"
1580totaal=money+btw:ENDPROC
1590
1600DEFPROCstar:VDU1,27,1,64,1,27,1,50:REM init printer, LF = 1/6 inch
1610VDU1,27,1,67,1,0,1,-1*(L%=2)+11:REM 11 or 12 inch paper
1620VDU1,14:PRINTVERKOPER$;:VDU1,27,1,87,1,0,1,27,1,69:PRINTSPC3;BEDRIJF$
1630VDU1,27,1,70:ENDPROC:REM Enlarged/Condensed Normal print style
1640
1650REM This program is written on a Acorn Electron 64k with a Star gemini 10x
1660REM printer, and is also compatible with a MSX2 printer.
1670REM It runs on an 32k Electron without diskdrive too.
1680REM Below follows a list of commands for BBC-BASIC / other BASICs
1690REM
1700REM MODE0       ->  80 x 32 characters mode
1710REM @%=&90A     ->  Print characters with a field with of ten (as normal)
1720REM @%=&2020A   ->  Same as 'PRINT USING "######.##"; .... '
1730REM VDU23,...   ->  Define a character
1740REM Bl,Bh       ->  BTW (VAT) Low (6%) and high (20%)
1750REM PRINT'      ->  PRINT:PRINT
1760REM STRING$(nr,"text")  ->  make string of nr copies of 'text'
1770REM VDU2/VDU3   ->  Printer on / off
1780REM VDU1,...    ->  next code (...) to printer
1790REM PROCname    ->  calls procedure (=subroutine) name
1800REM DEFPROCname...ENDPROC  ->  Definition of procedure
1810
1820REM Idea by    : Peter Grinwis,  Weststraat 41, Ouddorp
1830REM Program by: Simon Voortman, Beatrixweg 28, Ouddorp
```



```
JAN GRINWIS Pzn   Gewasbeschermingsbedrijf
----------------------------------------------------------------------------
Weststraat 41                                              hand- en machinewerk
3253 AR Ouddorp Z.H.     Rekening voor de Heer:                 rijenbespuiting
Telef. 01878-1686        'NAAM KLANT'                            tapbespuiting
Bank: Rabo               'ADRES KLANT'                         wegenbespuiting
No. ----.--.---          'WOONPLAATS KLANT'                       erfbespuiting
Giro -------                                                       watergangen
Reg.no. K.v.K. ------                                     dijken en weilanden
Reg.no. -. -------- -                                                  gazons
                                                                         enz.
                                                        No 8892
voor geleverd:               Ouddorp, 'HUIDIGE DATUM'
----------------------------------------------------------------------------
Datum                               Omschrijving            Te betalen bedrag
----------------------------------------------------------------------------
03-04    1x 1/1 pk Slakkenkorrels (AAGRUNOL) à f 4.00                4.00
04-04    1x 1/1 pk Muizentarwe à f 3.00                               3.00
08-04    5x 1/1 pk Rattenkorrels à f 3.98                            19.90
10-04    9x 1/1 lt Groen-Ex à f 13.50                               121.50
..-..    ...................................................          0.00
..-..    ...................................................          0.00
..-..    ...................................................          0.00
..-..    ...................................................          0.00
..-..    ...................................................          0.00
..-..    ...................................................          0.00
----------------------------------------------------------------------------
                                               Subtotaal f     148.40
                                             Korting 5%  f       7.42 -
                                                         ------------
                                               Subtotaal f     140.98
                                                  Btw 6% f       8.46 +
                                                         ------------
                                                  Totaal f     149.44
                                                               ======
```

```
EPROM BANKSWITCHING            PROTON 650X ASSEMBLER  V4.4   PAGE: 0001

0000              .TIT 'EPROM BANKSWITCHING'
0000              .OPT GEN
0000          ;
0000          ; AUTEUR : F.J.M. SMEEHUIJZEN
0000          ;          LIPPEDAL 19
0000          ;          2904 CL CAPELLE AAN DEN IJSSEL
0000          ;          TEL: 010-4512507
0000          ;
0000          ; ONDANKS HET GEBRUIK VAN FLOPPY DISK'S WAARDOOR HET NAAR
0000          ; BINNEN HALEN VAN PROGRAMMA'S SNEL KAN GEBEUREN, BLIJFT
0000          ; DE SNELSTE METHODE TOCH DOOR DIREKT VANUIT HET GEHEUGEN
0000          ; OP TE STARTEN.
0000          ; WAT EEN LUXE ZOU HET ZIJN OM DIREKT EN EEN EDITOR, EEN
0000          ; ASSEMBLER, EEN BASIC INTERPRETER EN ALLERLEI HANDIGE
0000          ; HULPPROGRAMMA'S DIREKT BESCHIKBAAR TE HEBBEN.
0000          ; DE KONSEQUENTIE IS DAN EVENWEL HET GEBRUIK VAN EEN GROTE
0000          ; HOEVEELHEID INTERN GEHEUGEN IN DE VORM VAN EPROM'S, WAAR-
0000          ; DOOR VOOR HET RAM-GEHEUGEN STEEDS MINDER OVERBLIJFT, OM-
0000          ; DAT WE NU EENMAAL NIET MEER DAN 64KB KUNNEN ADRESSEREN.
0000          ;
0000          ; MET GEBRUIKMAKING VAN HET DOOR ELEKTUUR ONTWORPEN EPROM-
0000          ; SWITCHBOARD IN FEBRUARI 1985 IS HET MOGELIJK OM VAN HET
0000          ; ENE NAAR HET ANDERE EPROM TE SPRINGEN DOOR SIMPELWEG HET
0000          ; BETREFFENDE EPROM NUMMER (0 T/M 3) NAAR EEN DUMMY-ADRES
0000          ; IN EPROM TE SCHRIJVEN.
0000          ; HIERDOOR WORDT DIT EPROM GESELEKTEERD EN KUNNEN DE ZICH
0000          ; DAARIN BEVINDENDE PROGRAMMA'S WORDEN UITGEVOERD.
0000          ; OM EEN EN ANDER IN GOEDE BANEN TE LEIDEN, ZONDER 'HANG-
0000          ; UP'S' TE VEROORZAKEN, DIENEN EEN AANTAL ZAKEN GEREGELD
0000          ; TE WORDEN.
0000          ;
0000          ; ALLEREERST DIENT IN IEDERE EPROM DE RESET-VEKTOR OP DE
0000          ; ADRESSEN FFFC EN FFFD AANWEZIG TE ZIJN WELKE WIJST NAAR
0000          ; BIJVOORBEELD HET 'COLD-START ADRES' VAN DE EDITOR, AS-
0000          ; SEMBLER OF BASIC OF NAAR EEN ROUTINE BINNEN DE EPROM
0000          ; WAARBIJ WEER TERUGGESCHAKELD WORDT NAAR DE MONITOR-EPROM.
0000          ;
0000          ; GEBRUIK MAKEN VAN MONITOR ROUTINES DIENT VIA EEN ZGN.
0000          ; DUMMY JUMP-TABLE TE LOPEN, WELKE TABEL ZICH BUITEN HET
0000          ; GEHEUGENGEBIED VAN DE 4 GEHEUGENBANKS DIENT TE BEVINDEN.
0000          ; IN ONDERSTAANDE VOORBEELDROUTINE IS GEKOZEN VOOR $C000.
0000          ;
0000          ; NU DE PRAKTISCHE KANT VAN DE ZAAK:
0000          ;
0000          ; HIERONDER EEN SCHEMATISCHE WEERGAVE VAN DE VERSCHILLENDE
0000          ; EPROM'S MET DE DAARIN GEPLAATSTE SOFTWARE.
0000          ;
0000          ; EPROM   #1          #2          #3          #4
0000          ;      --------    --------    --------    --------    --------
0000          ;      |$E000 |    |$E000 |    |$E000 |    |$E000 |    |$C000 |
0000          ;      |      |    |      |    |      |    |      |    |      |
0000          ;      |  M   |    |  E   |    |  A   |    |  B   |    |  D   |
0000          ;      |  O   |    |  D   |    |  S   |    |  A   |    |  U   |
0000          ;      |  N   |    |  I   |    |  S   |    |  S   |    |  M   |
0000          ;      |  I   |    |  T   |    |  E   |    |  I   |    |  M   |
0000          ;      |  T   |    |  O   |    |  M   |    |  C   |    |  Y   |
0000          ;      |  O   |    |  R   |    |  B   |    |      |    |  T   |
0000          ;      |  R   |    |------|    |  L   |    |      |    |  A   |
0000          ;      |      |    |  U   |    |  E   |    |      |    |  B   |
0000          ;      |      |    |  T   |    |  R   |    |      |    |  E   |
0000          ;      |      |    |  I   |    |------|    |      |    |  L   |
0000          ;      |      |    |  L   |    |      |    |      |    |      |
0000          ;      --------    --------    --------    --------    --------
0000          ;
0000          ; HET NU VOLGENDE OVERZICHT FUNKTIONEERT UITSTEKEND OP DE
0000          ; CPU/VDU KOMBINATIE VAN ELEKTUUR MET ALS OPERATING SYSTEM
0000          ; HET PROTON DOS IN 2764 EPROM'S.
0000          ; DEZE MONITOR HEEFT VANAF ADRES $E000 EEN INDIREKTE JUMP-
0000          ; TABEL NAAR ROUTINES WELKE DOOR IEDER PROGRAMMA KUNNEN
0000          ; WORDEN AANGEROEPEN.
0000          ; EEN AANTAL VAN DEZE ROUTINES WORDEN IN ONDERSTAAND VOOR-
0000          ; BEELD PROGRAMMA GEBRUIKT OM E.E.A. TE VERDUIDELIJKEN.
0000          ; DE ROUTINES ZELF ZIJN AAN HET EIND VAN HET PROGRAMMA
0000          ; ALLEEN DOOR EEN 'RTS' VOORGESTELD.
```

```
EPROM BANKSWITCHING          PROTON 650X ASSEMBLER  V4.4   PAGE: 0002

0000             ;
0000             ; HET STARTEN VAN DE DIVERSE PROGRAMMA'S GEBEURT MET BE-
0000             ; HULP VAN 'FUNKTIETOETSEN'.
0000             ; DE PROTON MONITOR REGELT NA HET INTOETSEN VAN EEN E,A OF B
0000             ; EEN SPRONG NAAR DE LABELS EKEY, AKEY EN BKEY WAARNA HET
0000             ; OPSTARTEN VAN HET GEWENSTE PROGRAMMA VIA DE VOORAF IN-
0000             ; GEVULDE VECTOR (ADRES $0300 EN HOGER IN HET VOORBEELD),
0000             ; PLAATS VINDT.
0000             ;
0000             ; E-TOETS IS DE EDITOR
0000             ; A-TOETS IS DE ASSEMBLER
0000             ; B-TOETS IS DE BASIC INTERPRETER
0000             ;
0000             ; *** ORIGINAL MONITOR ROUTINE ADDRESSES IN EPROM #1 ***
0000             ;
0000                     *=$E000
E000             ;
E000  00F0               .WOR COMIN        ; WARM RESTART OF MONITOR
E002  01F0               .WOR WHEREI       ; ASK FOR INPUT DEVICE
E004  02F0               .WOR WHEREO       ; ASK FOR OUTPUT DEVICE
E006  03F0               .WOR INALL        ; INPUT FROM ACTIVE INPUT DEVICE
E008  04F0               .WOR OUTALL       ; OUTPUT TO ACTIVE OUTPUT DEVICE
E00A  05F0               .WOR CLOSEI       ; CLOSE ACTIVE INPUT DEVICE
E00C  06F0               .WOR CLOSED       ; CLOSE ACTIVE OUTPUT DEVICE
E00E  07F0               .WOR GETTTY       ; TERMINAL INPUT ROUTINE
E010  08F0               .WOR OUTPUT       ; TERMINAL OUTPUT ROUTINE
E012             ;
E012             ; *** BANK START ADDRESSES ****
E012             ;
E012                     *=$E000
E000             ;
E000             EDITOR  *=*+0             ; EDITOR IN EPROM #2
E000             ASMBL   *=*+0             ; ASSEMBLER IN EPROM #3
E000             BASIC   *=*+0             ; BASIC IN EPROM #4
E000             ;
E000                     *=$FFFF
FFFF             ;
FFFF             DUMMY   *=*+1             ; DUMMY ADDRESS IN EPROM
0000             ;
0000                     *=$0200
0200             ;
0200             BANK    *=*+1             ; SELECTED BANK NUMBER POINTER
0201             ;
0201                     *=$0300
0300             ;
0300             ; *** FUNCTION VECTORS ***
0300             ;
0300  67C0       EDTVEC  .WOR EDT65        ; INDIRECT JUMP TO EDITOR
0302  72C0       ASMVEC  .WOR ASM65        ; INDIRECT JUMP TO ASSEMBLER
0304  7DC0       BASVEC  .WOR BAS65        ; INDIRECT JUMP TO BASIC
0306             ;
0306                     *=$C000
C000             ;
C000             ; *** JUMP TABLE FOR BANK SWITCHED SOFTWARE ***
C000             ;
C000  2056C0 E000        JSR BANK0         ; SWITCH TO MONITOR-EPROM
C003  4C00F0             JMP COMIN         ; RETURN TO MONITOR
C006  2056C0 E002        JSR BANK0         ; SWITCH TO MONITOR-EPROM
C009  2001F0             JSR WHEREI        ; DETERMINE INPUT DEVICE
C00C  205EC0             JSR BANK1         ; RETURN TO SELECTED EPROM
C00F  60                 RTS
C010  2056C0 E004        JSR BANK0         ; SWITCH TO MONITOR-EPROM
C013  2002F0             JSR WHEREO        ; DETERMINE OUTPUT DEVICE
C016  205EC0             JSR BANK1         ; RETURN TO SELECTED EPROM
C019  60                 RTS
C01A  2056C0 E006        JSR BANK0         ; SWITCH TO MONITOR-EPROM
C01D  2003F0             JSR INALL         ; INPUT FROM ACTIVE INPUT DEVICE
C020  205EC0             JSR BANK1         ; RETURN TO SELECTED EPROM
C023  60                 RTS
C024  2056C0 E008        JSR BANK0         ; SWITCH TO MONITOR-EPROM
C027  2004F0             JSR OUTALL        ; OUTPUT TO ACTIVE OUTPUT DEVICE
C02A  205EC0             JSR BANK1         ; RETURN TO SELECTED EPROM
C02D  60                 RTS
C02E  2056C0 E00A        JSR BANK0         ; SWITCH TO MONITOR-EPROM
C031  2005F0             JSR CLOSEI        ; CLOSE INPUT DEVICE
C034  205EC0             JSR BANK1         ; RETURN TO SELECTED EPROM
C037  60                 RTS
```

```
EPROM BANKSWITCHING              PROTON 650X ASSEMBLER  V4.4   PAGE: 0003

0152  C038  2056C0 E00C      JSR BANK0        ; SWITCH TO MONITOR-EPROM
0153  C03B  2006F0           JSR CLOSEO       ; CLOSE OUTPUT DEVICE
0154  C03E  205EC0           JSR BANK1        ; RETURN TO SELECTED EPROM
0155  C041  60               RTS
0156  C042  2056C0 E00E      JSR BANK0        ; SWITCH TO MONITOR-EPROM
0157  C045  2007F0           JSR GETTTY       ; TERMINAL INPUT ROUTINE
0158  C048  205EC0           JSR BANK1        ; RETURN TO SELECTED EPROM
0159  C04B  60               RTS
0160  C04C  2056C0 E010      JSR BANK0        ; SWITCH TO MONITOR-EPROM
0161  C04F  2008F0           JSR OUTPUT       ; TERMINAL OUTPUT ROUTINE
0162  C052  205EC0           JSR BANK1        ; RETURN TO SELECTED EPROM
0163  C055  60               RTS
0164  C056              ;
0165  C056              ; *** SWITCH TO MONITOR-EPROM ***
0166  C056              ;
0167  C056  48          BANK0   PHA          ; SAVE ACCU
0168  C057  A900                LDA #$00     ; NUMBER OF MONITOR-EPROM
0169  C059  8DFFFF              STA DUMMY    ; WRITE TO DUMMY ADDRESS
0170  C05C  68                  PLA          ; RESTORE ACCU
0171  C05D  60                  RTS
0172  C05E              ;
0173  C05E              ; *** SWITCH TO SELECTED EPROM ***
0174  C05E              ;
0175  C05E  48          BANK1   PHA          ; SAVE ACCU
0176  C05F  AD0002              LDA BANK     ; LOAD SELECTED BANK NUMBER
0177  C062  8DFFFF              STA DUMMY    ; AND WRITE TO DUMMY ADDRESS
0178  C065  68                  PLA          ; RESTORE ACCU
0179  C066  60                  RTS
0180  C067              ;
0181  C067              ; *** SWITCH TO EDITOR EPROM ***
0182  C067              ;
0183  C067  A901        EDT65   LDA #$01     ; LOAD EPROM NUMBER
0184  C069  8D0002              STA BANK     ; SAVE SELECTED BANK NUMBER
0185  C06C  8DFFFF              STA DUMMY    ; WRITE TO DUMMY ADDRESS
0186  C06F  4C00E0              JMP EDITOR   ; JUMP TO EDITOR IN EPROM #2
0187  C072              ;
0188  C072              ; *** SWITCH TO ASSEMBLER EPROM ***
0189  C072              ;
0190  C072  A902        ASM65   LDA #$02     ; LOAD EPROM NUMBER
0191  C074  8D0002              STA BANK     ; SAVE SELECTED BANK NUMBER
0192  C077  8DFFFF              STA DUMMY    ; WRITE TO DUMMY ADDRESS
0193  C07A  4C00E0              JMP ASMBL    ; JUMP TO ASSEMBLER IN EPROM #3
0194  C07D              ;
0195  C07D              ; *** SWITCH TO BASIC EPROM ***
0196  C07D              ;
0197  C07D  A903        BAS65   LDA #$03     ; LOAD EPROM NUMBER
0198  C07F  8D0002              STA BANK     ; SAVE SELECTED BANK NUMBER
0199  C082  8DFFFF              STA DUMMY    ; WRITE TO DUMMY ADDRESS
0200  C085  4C00E0              JMP BASIC    ; JUMP TO BASIC IN EPROM #4
0201  C088              ;
0202  C088                      *=$F000
0203  F000              ;
0204  F000              ; *** ROUTINES IN MONITOR EPROM WHERE RTS REPRESENTS ***
0205  F000              ; *** A DUMMY FOR THE ORIGINAL MONITOR ROUTINES      ***
0206  F000              ;
0207  F000  60          COMIN   RTS
0208  F001  60          WHEREI  RTS
0209  F002  60          WHEREO  RTS
0210  F003  60          INALL   RTS
0211  F004  60          OUTALL  RTS
0212  F005  60          CLOSEI  RTS
0213  F006  60          CLOSEO  RTS
0214  F007  60          GETTTY  RTS
0215  F008  60          OUTPUT  RTS
0216  F009              ;
0217  F009              ;
0218  F009  6C0003 EKEY         JMP (EDTVEC) ; FUNKTIETOETS 'E
0219  F00C  6C0203 AKEY         JMP (ASMVEC) ; FUNKTIETOETS 'A'
0220  F00F  6C0403 BKEY         JMP (BASVEC) ; FUNKTIETOETS 'B'
0221  F012              ;
0222  F012                      .END

ERRORS: 0000                     (0000)
```

```
12-Jul-87 16:30  relocate.lis  Page 1

                  ;+-----------------------------------------------------------------+
                  ;                                                                  |
                  ;         RELOCATE start,end,offset,start_adjust,end_adjust        |
                  ;                                                                  |
                  ;         New DOS65 command to relocate the absolute addresses     |
                  ;         in machine code programs                                 |
                  ;                                                                  |
                  ;+-----------------------------------------------------------------+
                  ;
       0400               org     $400
                  ;
                  ; constants
                  ;
       000D       cr      equ     $d
       000C       formf   equ     $c
                  ;
                  ; Zero page addresses
                  ;
       00F0       flag    equ     $f0                 ;flag for newline
       00F2       tema    equ     $f2                 ;temp. store for start-address
       00F4       temb    equ     $f4                 ;temp. store for end-address
       00F6       temc    equ     $f6                 ;temp. store for offset
       00E8       temd    equ     $e8                 ;temp. store for start-adjust
       00EA       teme    equ     $ea                 ;temp. store for end-adjust
                  ;
                  ; DOS65 routines
                  ;
       C02F       crlf    equ     $c02f               ;print CRLF
       C032       space   equ     $c032               ;print space
       C03B       print   equ     $c03b               ;print string till 0
       C06B       spar    equ     $c06b               ;scan parameters
       C023       output  equ     $c023               ;print character
       C038       hexout  equ     $c038               ;print A in hex.
       D0B7       ermes   equ     $d0b7               ;print error message
                  ;------------------------------------------------------------------------
0400 A2 00        relocat ldx     #0
0402 20 6BC0              jsr     spar                ;get parameters
0405 F2F4                 fcc     tema,temb
0407 F6E8EA00             fcc     temc,temd,teme,0
040B B0 07                bcs     1.f                 ;error in parameters
040D E0 F8        reloc   cpx     #$f8
040F F0 49                beq     relgl
0411 4C F104              jmp     erda                ;not enough parameters
0414 4C B7D0      1       jmp     ermes
0417 A2 F2        relb    ldx     #tema
0419 20 7004              jsr     opclen              ;get opcode length
041C C0 03                cpy     #3                  ;absolute?
041E D0 45                bne     reli                ;no
0420 20 B904              jsr     xinc                ;else point to address
0423 A0 01                ldy     #1
0425 A5 EA                lda     teme                ;check if in adjust area
0427 C1 00                cmp     [0,x]
0429 A1 00                lda     [0,x]
042B AA                   tax
042C A5 EB                lda     teme+1
042E F1 F2                sbc     [tema],y
0430 90 31                bcc     relh                ;no, too large
0432 E4 E8                cpx     temd
0434 B1 F2                lda     [tema],y
0436 E5 E9                sbc     temd+1
0438 90 29                bcc     relh                ;no, too small
043A 18                   clc                         ;else add offset
043B 8A                   txa
043C 65 F6                adc     temc                ;low
043E 48                   pha
043F B1 F2                lda     [tema],y
0441 65 F7                adc     temc+1              ;and high part
0443 91 F2                sta     [tema],y            ;adjust in memory
0445 88                   dey
0446 68                   pla
0447 91 F2                sta     [tema],y            ;adjust in memory
0449 A2 F2                ldx     #tema
044B 20 C004              jsr     xdec                ;reset pointer to opcode
044E 20 A204      relg    jsr     outins              ;print adjusted opcode
```

```
12-Jul-87 16:34  relocate.lis  Page 2

0451 B0 1A                    bcs     relia           ;if C then at end
0453 20 E104                  jsr     twospa          ;print two spaces
0456 C6 F0                    dec     flag            ;and decr. flag
0458 D0 BD                    bne     relb            ;if not zero, continue
045A A9 04           relgl    lda     #4              ;else reset flag
045C 85 F0                    sta     flag            ;(four opcode/line)
045E 20 2FC0                  jsr     crlf            ;and do <newline>
0461 90 B4                    bcc     relb            ;then continue
0463 A0 02           relh     ldy     #2              ;point to next opcode
0465 20 CB04         reli     jsr     incbrk
0468 88                       dey
0469 D0 FA                    bne     reli
046B 90 AA                    bcc     relb            ;if C=0 continue
046D 4C 2FC0         relia    jmp     crlf            ;else return to DOS65
                     ;------------------------------------------------------------------
                     ;                                Subroutines
                     ;------------------------------------------------------------------
                     ;
                     ; return opcode length in Y
                     ;
0470 A0 01           opclen   ldy     #1
0472 A1 00                    lda     [0,x]
0474 F0 1B                    beq     opcj
0476 C9 40                    cmp     #$40            ;RTI?
0478 F0 17                    beq     opcj
047A C9 60                    cmp     #$60            ;RTS?
047C F0 13                    beq     opcj
047E A0 03                    ldy     #3
0480 C9 20                    cmp     #$20            ;JSR?
0482 F0 0D                    beq     opcj
0484 29 1F                    and     #$1f
0486 C9 19                    cmp     #$19
0488 F0 07                    beq     opcj
048A 29 0F                    and     #$f
048C A8                       tay
048D B9 9204                  lda     opctb,y         ;get length from table
0490 A8                       tay
0491 60              opcj     rts
                     ;
0492 0202020102      opctb    fcc     2,2,2,1,2,2,2,1,1,2,1,1,3,3,3,1
                     ;
                     ; Print adjusted opcode
                     ;
04A2 20 7004         outins   jsr     opclen          ;get opcode length
04A5 20 E704                  jsr     outxad          ;print current address
04A8 20 3BC0                  jsr     print
04AB 203A2000                 fcc     ' : ',0
04AF 20 DA04         oinsa    jsr     outmeb          ;print byte
04B2 20 CB04                  jsr     incbrk          ;point to next
04B5 88                       dey
04B6 D0 F7                    bne     oinsa           ;until end of opcode
04B8 60                       rts
                     ;
                     ; Incr. 16 bits address where X is pointing to
                     ;
04B9 F6 00           xinc     inc     0,x
04BB D0 02                    bne     xincb
04BD F6 01                    inc     1,x
04BF 60              xincb    rts
                     ;
                     ; Decr. 16 bits address where X is pointing to
                     ;
04C0 48              xdec     pha
04C1 B5 00                    lda     0,x
04C3 D0 02                    bne     xdecb
04C5 D6 01                    dec     1,x
04C7 D6 00           xdecb    dec     0,x
04C9 68                       pla
04CA 60                       rts
                     ;
                     ; Incr. and check if at end
                     ;
04CB A5 F2           incbrk   lda     tema            ;incr. start-address
04CD C5 F4                    cmp     temb            ;and check if at end
```

```
12-Jul-87 16:37   relocate.lis   Page 3

04CF A5 F3                   lda     tema+1
04D1 E5 F5                   sbc     temb+1
04D3 E6 F2                   inc     tema
04D5 D0 02                   bne     incb
04D7 E6 F3                   inc     tema+1
04D9 60              incb    rts                      ;C=1, if at end
                     ;
                     ; Print byte
                     ;
04DA A1 00           outmeb  lda     [0,x]            ;get byte
04DC 20 38C0         outmea  jsr     hexout           ;print in hex
04DF 90 03                   bcc     outspa           ;branch always
                     ;
04E1 20 32C0         twospa  jsr     space            ;print space
04E4 4C 32C0         outspa  jmp     space            ;and another
                     ;
                     ; Print address
                     ;
04E7 B5 01           outxad  lda     1,x              ;get address hi
04E9 20 38C0                 jsr     hexout           ;print in hex.
04EC B5 00                   lda     0,x              ;get address lo
04EE 4C 38C0                 jmp     hexout           ;print it
                     ;
                     ; Help info if error in parameters
                     ;
04F1 20 3BC0         erda    jsr     print
04F4 0C52454C4F         fcc    formf,'RELOCATE HELP',cr,cr
0504 436F6D6D61         fcc    'Command syntax:',cr
0514 52454C4F43         fcc    'RELOCATE aaaa,bbbb,cccc,dddd,eeee',cr,cr
0537 6161616120         fcc    'aaaa  : Startaddress of memory area to be relocated',cr
056B 6262626220         fcc    'bbbb  : Endaddress of memory area',cr
058D 6363636320         fcc    'cccc  : Offset',cr
059C 6464646420         fcc    'dddd  : Start of address area to be adjusted',cr
05C9 6565656520         fcc    'eeee  : End of address area to be adjusted',cr,cr
05F5 4578616D70         fcc    'Example:',cr
05FE 50726F6772         fcc    'Program to be relocated is resident at $2000 to $2200, suppose you want to',cr
0649 7573652074         fcc    'use this program at $3000 to $3200, so only absolute addresses in the area',cr
0694 2432303030         fcc    '$2000 to $2200 have to be adjusted. The offset is $1000 so enter:',cr,cr
06D7 52454C4F43         fcc    'RELOCATE 2000,2200,1000,2000,2200',cr,cr
06FA 5468652061         fcc    'The adjusted addresses are displayed for control purposes (see note)',cr
073F 4E6F74653A         fcc    'Note: Beware for data or tables in the relocated addressarea, the program',cr
0789 2020202020         fcc    '      cannot see the difference between opcodes in a program and characters',c
07D5 2020202020         fcc    '      like $20 (ASCII space) and $4C (ASCII L), so use RELOCATE with care.',cr
0821 60                 rts
          0400          end    relocat
                        global labels

cr        000D  crlf      C02F  erda      04F1  ermes     D0B7  flag      00F0
formf     000C  hexout    C038  incb      04D9  incbrk    04CB  oinsa     04AF
opcj      0491  opclen    0470  opctb     0492  outins    04A2  outmea    04DC
outmeb    04DA  output    C023  outspa    04E4  outxad    04E7  print     C03B
relb      0417  relg      044E  relgl     045A  relh      0463  reli      0465
relia     046D  reloc     040D  relocat   0400  space     C032  spar      C06B
tema      00F2  temb      00F4  temc      00F6  temd      00E8  teme      00EA
twospa    04E1  xdec      04C0  xdecb     04C7  xinc      04B9  xincb     04BF

Errors detected: 0
```

```
SOURCE FILE: CONVERSIE
0000:                1 ;**** HEX/DEC EN DEC/HEC CONVERSIE ****
0000:                2 ;
0000:                3 ;      DOOR HANS BOSCH,
0000:                4 ;      REELAAN 35,
0000:                5 ;      7522 LS ENSCHEDE.
0000:                6 ;
0000:                7 ;APPLESOFT EN MONITOR ROUTINES
DD67:                8 FRMNUM  EQU  $DD67     EXPRESSIE NAAR FAC
E752:                9 GETADR  EQU  $E752     FAC NAAR INTEGER IN LINNUM
ED24:               10 LINPRT  EQU  $ED24     PRINT 2-BYTE NUMMER IN X(LSB) EN A(MSB)
F941:               11 PRNTAX  EQU  $F941     PRINT A EN X REGISTER
F948:               12 PRBLNK  EQU  $F948     PRINT 3 SPATIES
FC22:               13 VTAB    EQU  $FC22     ZET VTAB NAAR CV
FDED:               14 COUT    EQU  $FDED     OUTPUT ROUTINE
FFA7:               15 GETNUM  EQU  $FFA7     CONVERTEER VAN HEX NAAR DEC
0000:               16 ;
0000:               17 ;MEMORY
0024:               18 CH      EQU  $24       CURSOR HORIZONTAAL
0025:               19 CV      EQU  $25       CURSOR VERTICAAL
003E:               20 A2L     EQU  $3E       RESULTAAT HEX/DEC CONVERSIE
0050:               21 LINNUM  EQU  $50       RESULTAAT DEC/HEX CONVERSIE
0200:               22 BUF     EQU  $200      INPUT BUFFER
03F6:               23 AMPERV  EQU  $3F6      AMPERSAND VECTOR
0000:               24 ;
----- NEXT OBJECT FILE NAME IS CONVERSIE.OBJO
6000:               25         ORG  $6000
6000:               26 ;BRUN CONVERSIE.OBJO VOOR GEBRUIK AMPERSAND
6000:A9 0B          27 INIT    LDA  #)START   LSB START ADRES
6002:8D F6 03       28         STA  AMPERV
6005:A9 60          29         LDA  #(START   MSB
6007:8D F7 03       30         STA  AMPERV+1  &VECTOR WIJST NU NAAR START ADRES
600A:60             31         RTS
600B:               32 ;
600B:48             33 START   PHA             BERG A-REGISTER OP
600C:A0 09          34         LDY  #9
600E:C6 25          35         DEC  CV        NAAR VORIGE REGEL
6010:20 22 FC       36         JSR  VTAB
6013:84 24          37         STY  CH        TAB POSITIE OP Y
6015:68             38         PLA             HAAL A-REGISTER TERUG
6016:C9 24          39         CMP  #$24      $(HEX) INVOER?
6018:D0 31          40         BNE  DEC
601A:               41 ;
601A:               42 ;HEX-DEC CONVERSIE
601A:BD 00 02       43 HEX1    LDA  BUF,X     X WIJST  ALTIJD VOORBIJ LAATSTE DIGIT
601D:09 80          44         ORA  #$80
601F:9D 00 02       45         STA  BUF,X     MAAK ALLE 7E BITS IN BUF HOOG
6022:CA             46         DEX
6023:D0 F5          47         BNE  HEX1
6025:A0 02          48         LDY  #2        WIJS NAAR 1E DIGIT IN BUFFER ($0202)
6027:20 A7 FF       49         JSR  GETNUM
602A:A6 3E          50 HEX2    LDX  A2L       LSB
602C:A5 3F          51         LDA  A2L+1     MSB
602E:20 24 ED       52         JSR  LINPRT    UITVOER RESULTAAT
6031:24 3F          53         BIT  A2L+1     RESULTAAT ( 32768?
6033:10 26          54         BPL  KLAAR
6035:8A             55         TXA             A-REGISTER=0, CARRY=SET
6036:E5 3E          56         SBC  A2L       TREK LSB ER VANAF
6038:85 3E          57         STA  A2L
603A:8A             58         TXA             A-REGISTER=0
603B:E5 3F          59         SBC  A2L+1     TREK MSB ER VANAF
603D:30 1C          60         BMI  KLAAR     $8000 INGETIKT?
603F:85 3F          61         STA  A2L+1     NEE,
6041:20 48 F9       62         JSR  PRBLNK    GEEF 3 SPATIES,
6044:A9 AD          63         LDA  #$AD      PRINT ALVAST "-"
6046:20 ED FD       64         JSR  COUT
6049:B0 DF          65         BCS  HEX2      EN PRINT REST VAN NEGATIEVE NOTATIE
604B:               66 ;
604B:               67 ;DEC-HEX CONVERSIE
604B:A9 A4          68 DEC     LDA  #$A4
604D:20 ED FD       69         JSR  COUT      PRINT "$"
6050:20 67 DD       70         JSR  FRMNUM    CONVERTEER INVOER NAAR FAC
6053:20 52 E7       71         JSR  GETADR    MSB IN A EN LINNUM+1
6056:A6 50          72         LDX  LINNUM    LSB
6058:20 41 F9       73         JSR  PRNTAX    UITVOER RESULTAAT
605B:20 D0 03       74 KLAAR   JSR  $3D0      TERUG NAAR APPLESOFT

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
49 54 MLIST
SCR # 49
  0 ( HEX/ASCII-DUMP 1                         GERT KLEIN       )
  1 HEX  0  VARIABLE  ENDAD    0  VARIABLE  POINTER
  2 ( U. PRINTS AN UNSIGNED 16 BIT NUMBER )
  3 :  U.   0   D.   :
  4
  5 ( FETCHBYTE FETCHES A BYTE FROM ADRESS IN POINTER )
  6 :   FETCHBYTE  POINTER   @    C@    :
  7
  8 ( .0 PRINTS n ZERO'S )
  9 : .0    0    DO    30    EMIT  LOOP    :
 10
 11 ( .ROW PRINTS INDEX ROW ON TOP OF DUMP )
 12 :  .ROW  5  SPACES  10  0  DO  I  .  SPACE   LOOP  CR  :
 13
 14 ( CHECK IF BYTE IS A PRINTABLE ASCII CHARACTER )
 15 : ?ASCII   7F  AND   DUP  7F  <  OVER  1F  >  AND  :    -->

SCR # 50
  0 ( HEX/ASCII-DUMP 2                         GERT KLEIN       )
  1 ( PRINT 16 BIT ADRESS WITH LEADING ZERO'S )
  2 : .POINTER  POINTER  @
  3       DUP  10    <  OVER  FFFF  >  AND  IF  3  .0  ENDIF
  4       DUP  100   <  OVER  F     >  AND  IF  2  .0  ENDIF
  5       DUP  1000  <  OVER  FF    >  AND  IF  1  .0  ENDIF
  6       U.   :
  7 ( PRINT 16 HEX BYTES )
  8 : .HEXROW  10  0
  9      DO
 10         FETCHBYTE  DUP  10  <
 11         IF
 12          30 EMIT     ( PRINT LEADING ZERO   )
 13         ENDIF
 14          .  1  POINTER  +!  ( INCREMENT POINTER  )
 15      LOOP  :              -->

SCR # 51
  0 ( HEX/ASCII DUMP 3                           GERT KLEIN       )
  1 ( PRINT 16 ASCII CHARACTERS. IF NOT PRINTABLE OUTPUT DOT )
  2 : .ASCROW
  3    POINTER  @  10  -  POINTER  !  ( POINTER TO BEGIN OF LINE )
  4     3  SPACES  10  0
  5      DO
  6        FETCHBYTE  ?ASCII
  7        IF
  8          EMIT                ( PRINT ASCII CHARACTER  )
  9        ELSE
 10          DROP  2E  EMIT    ( PRINT DOT  )
 11       ENDIF
 12        1  POINTER  +!        ( INCREMENT POINTER  )
 13     LOOP
 14   POINTER  @  ENDAD  @  >   :  ( ENDS WITH FLAG ON STACK )
 15  -->

SCR # 52
  0 ( HEX/ASCII-DUMP 4                         GERT KLEIN       )
  1 ( PRINT HEXDUMP ON CURRENT I/O DEVICE )
  2 : HEXDUMP  ENDAD  !  POINTER  !  CR CR  [COMPILE]  HEX  .ROW
  3         BEGIN
  4            CR  .POINTER  .HEXROW  .ASCROW
  5            ?TERMINAL                     ( BREAKKEY TEST  )
  6            IF
  7              DROP  1     ( SETS TRUE FLAG  )
  8            ENDIF
  9         UNTIL              ( TERMINATE IF FLAG TRUE  )
 10     CR  :
 11
 12 ( PRINT HEXDUMP ON PRINTER )
 13 : PHEXDUMP  PRAAN  HEXDUMP  PRUIT  :
 14
 15  ;S
```

```
SCR # 53
    ( GLOSSARY HEX/ASCII-DUMP                             GERT KLEIN          )
    U.         ( n -- )
               Print an unsioned 16 bit number.
    FETCHBYTE  ( -- bvte )
               Get bvte from adress in POINTER
    .0         ( n -- )
               Print n zero's.
    .ROW       ( -- )
               Print index row on too of dumo.
    ?ASCII     ( ch -- ch f )
               Set true flao if value of ch is between $20 and
               $7E. else false flao.
    .POINTER   ( -- )
               Print adress in variable POINTER with leadino
               zero's.


SCR # 54
    ( GLOSSARY HEX/ASCII-DUMP                             GERT KLEIN          )
    .HEXROW    ( -- )
               Print 16 hexbvtes in the ranoe of adress in variable
               POINTER to POINTER + 16.
    .ASCROW    ( -- f )
               Print 16 ASCII characters in the ranoe of acress in
               variable POINTER to POINTER + 16. Non orintino
               characters are reoresented bv a dot. Flao indicates
               if POINTER > ENDAD.
    HEXDUMP    ( adr1 adr2  -- )
               Print hex/ascii dumo between adr1 and adr2. Terminate
               on terminal break. Adr1 and adr2 mav be entered
               both in hex or in decimal. The dumo is outoutted in
               hex. After termination HEX is the oresent number base
    PHEXDUMP   ( adr1 adr2  -- )
               Print dumo on orinter.

OK
```