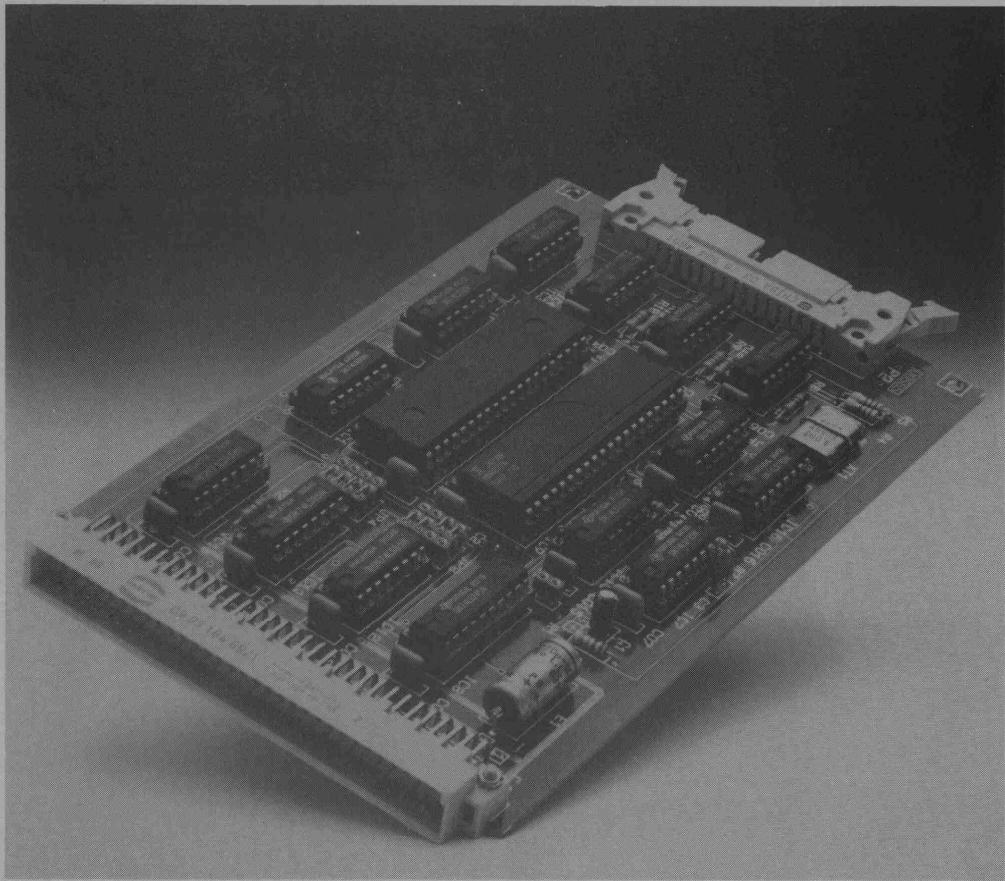


DE6502 KENNER

51



Elfde Jaargang, Nr. 4
Augustus 1987

DE 6502 KENNER

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

De vereniging heeft leden in Nederland, België, 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/IIC/IIGS/III, MACINTOSH, ATARI 600/800XL/512/1024ST, C64-1, PEARCOM, AIM-65, SYM, PET, BBC, VIC-20, BASIS 108, PROTON-computers, IIT-2020, OSI, ACC 8000, ACORN ELECTRON, SYSTEM 65, PC-100, PALLAS, MINIA, FORMOSA, ORIC-1, STARLIGHT, CV-777, ESTATE III, SBC65/68, KIM, NCS, KEMPAC SYSTEM-4, Elektuur-computers (JUNIOR, EC65(K) alias OCTOPUS), LASER, dus ook 6800, 6809 en 68000-computers.

De kennisuitwisseling wordt o.a. gerealiseerd door 6 maal per jaar DE 6502 KENNER te publiceren, door het houden van landelijke clubbijeenkomsten, door het instandhouden van een diskette-service en door het verlenen van paperware-service.

Verschijningsdata

DE 6502 KENNER 1985

=====

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

Redactie-adres en informatie over paperware etc.

Willem L. van Pelt
Jacob Jordaanstraat 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

Penningmeester:

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

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 Europe : Hfl. 114,50 (incl. transfers)
Advertenties : Tarieven op aanvraag bij de redactie.

Bijeenkomsten van de club

=====

derde zaterdag van januari, maart, mei, september, november.

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

DE 6502 KENNER is een uitgave van de KIM Gebruikers Club Nederland. Het blad wordt verstrekt aan leden van de club. DE 6502 KENNER wordt van kopij voorzien door leden van de club, bij de opmaak van een publikatie bijgestaan door de redactie. De inzendingen van programma's dienen voorzien te zijn van commentaar in de listings en zo mogelijk door een inleiding voorafgegaan. Publikaties van een inzending betekent dat de redactie 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 ontwikkelde hardware en/of aanpassingen daarop. Zij schrijven tevens artikelen van algemene aard of reageren op publikaties van andere inzenders.

DE 6502 KENNER IS EEN BLAD VAN EN DOOR DE LEDEN

Micro-ADE Assembler/Disassembler/Editor is een produkt van Micro Ware Ltd., geschreven door Peter Jennings en bestemd voor alle 6502-computers. De KIM Gebruikers Club Ned. heeft de copyrighten verworven nadat ons lid Sebo Wolbring 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.

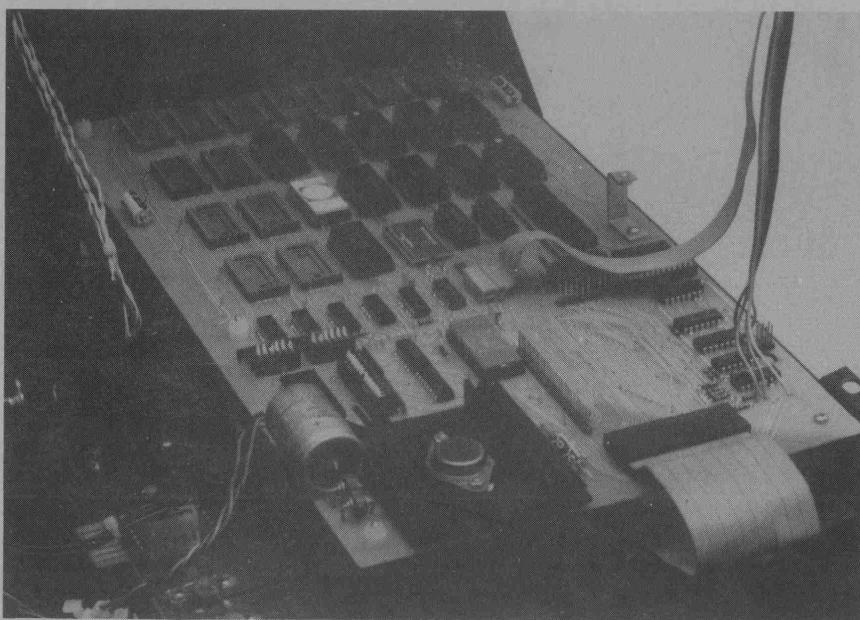
FATE Format-lister/cond. Assembler/Tape-utilities/Editor is de door ons lid Rob Banen geschreven source-listing van een 12 K universeel systeem voor de JUNIOR-computer aan de hand van het universele disk operating system van de fa. Proton Electronics te Naarden. FATE wordt beschikbaar gesteld met toestemming 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 informatie over hetgeen in de handel te koop is. Leden die iets te koop hebben of iets zoeken kunnen dit in de edities van DE 6502 KENNER bekend maken. Ook worden brieven aan de redactie gepubliceerd, evenals specifieke vragen van leden. De edities worden samengesteld op basis van een groot aantal prioriteiten, welke door een redactievergadering worden gehanteerd. Deze vergadering bestaat uit de vaste medewerkers zoals in de colofon vermeld. Het aantal inzendingen is groter dan in een enkele editie van minimaal 48 pagina's is te verwerken. Hierdoor kan het voorkomen dat een inzending eerst na enige tijd kan worden gepubliceerd.



DE 6502 KENNER

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

Adres voor het inzenden van en reacties op artikelen voor DE 6502 KENNER:
Willem L. van Pelt
Jacob Jordaanstraat 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 Oppen
Leif Rasmussen (Denmark)
Ruud Uphoff
Frans Verberkt
Herman Zondag (drawings)

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

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

Gehele of gedeeltelijke overname van de inhoud van DE 6502 KENNER zonder toestemming van het bestuur is verboden. Toepassing van gepubliceerde programma's, hardware etc. is alleen toegestaan voor persoonlijke gebruik.
DE 6502 KENNER verschijnt 6 x per jaar en heeft een oplage van 500 exemplaren.

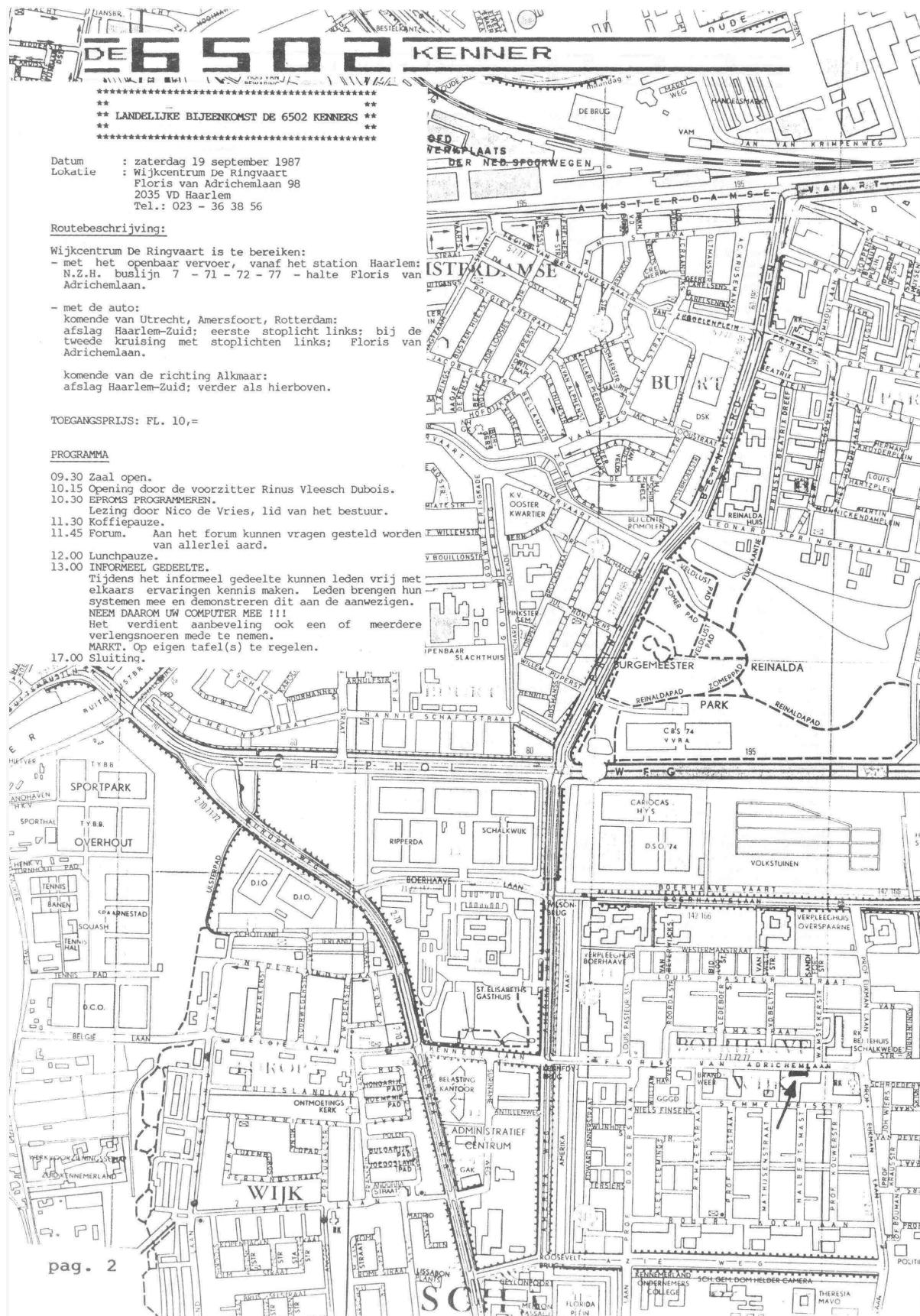
Copyright (C) 1987 KIM Gebruikers Club Nederland.

De voorpagina is de DOS65-controllerkaart ontwikkeld door Ad Brouwer.
CAD/CAM: E. Visschedijk.
i.s.m.: A. Hankel
Fotogr.: Fr. Visschedijk.
I.v.m. auteurswetgeving aanvaardt de redactie geen aansprakelijkheid voor inzendingen. Tenzij anders aangegeven, dient de inzending afkomstig te zijn van de inzender.

Oversamen

INHOUDSOPGAVE DE 6502 KENNER NR. 51 AUGUSTUS 1987.

1. Landelijke Bijeenkomst DE 6502 KENNERS te Haarlem, 19 september 1987	2.
ATARI 520 ST (68000)	
2. Sideways Printing; a program written in FAST BASIC	3.
... John C. Rix, England	
ATARI 600 XL	
3. Print Your Graphics; screenprint routine in Basic	27.
... Henk Speksnijder, The Netherlands	
68020	
4. OMEGA, the desktop mainframe; general view on a 32 bit workstation	20.
6502 general	
5. Eprom Bankswitching; handige hulpprogramma's direct beschikbaar	41.
... Frans J.M. Smeehuijzen, Nederland	
6. Datacommunication with 6502 computers	14.
... Bram de Bruine, The Netherlands	
EC65/OCTOPUS	
7. Cassette Motor Control and Bell; hardware design	9.
... Marc Lachaert, Belgium	
8. Wordprocessor V3.0 available	17.
... Leif Rasmussen, Denmark	
9. Patch on Dr. Tietsch's Copier Program	17.
... P. Lindström & L. Rasmussen, Denmark	
10. Printer problems with the Octopus	17.
... Maarten den Hertog, The Netherlands	
11. Loy's Extra; how to bind the systemdisk utilities	21.
... Leif Rasmussen, Denmark	
12. OCTOFATE; FATE for the Octopus. A new assembler/editor and much more.	26.
... Coen Boltjes, The Netherlands, about Marc Lachaert's exercices.	
13. Octopus Inputverwerking; a solution to a problem	26.
... A.P. Oerlemans, The Netherlands.	
BBC/ELECTRON	
14. Printer Init for Electron and BBC; Basic program to set some options	12.
... Ronald van Vugt, The Netherlands	
15. Rekening 09 MSX; faktureren in Basic	38.
... Peter Grinwis en Simon Voortman, Nederland	
DOS65	
16. Universal Terminal V0.23 available	13.
... Andrew P. Gregory	
17. Hoe haal je met Diskdoctor een directory terug?	15.
How to recover a directory with Diskdoctor?	
... Bram de Bruine, The Netherlands	
18. JUNIOR Tape Routines on DOS65	28.
... Ernst R. Elderenbosch, The Netherlands	
19. Relocate; new DOS65 command to relocate absolute addresses	44.
JUNIOR	
20. H-code Calculator; make a checksum table of your own (e)proms	16.
... Maurice Nelissen, Belgium	
21. Databank Program	18.
... Maarten Lamey, The Netherlands	
22. A L(list) NNN, <CR> Implementation in Micro-ADE	20.
... Fernando Lopes, Portugal	
APPLE	
23. Hex/Dec en Dec/Hex Conversie	47.
... Hans Bosch, Nederland	
24. APPLE nieuws	20.
ACORN ATOM	
25. Clock; conversion of the BBC program of Ronald van Vugt in issue 50	25.
... John Anijs, The Netherlands	
BASIC	
26. Seabattle	27.
Ontbinden in Factoren	
... Gerard van Roekel, Nederland	
FORTH	
27. Piq-FORTH Decompiler	35.
Hex/Ascii-Dump	
... Gert Klein, The Netherlands	
Diversen	48.
28. Tip van Ernst Elderenbosch	26.
Tips van A.P. Oerlemans	



DE 6502 KENNER

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 $\frac{1}{2}^{\text{nd}}$ ie. EPSON or compatible (I used an EPSON FX80)

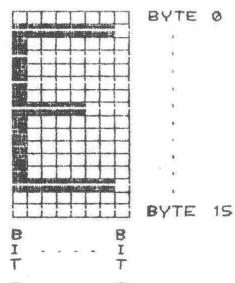
The maximum number of lines per page is 48 (line density is fixed at 8 lines per inch) and the line width is limited to 160 by 1st word plus. Although a page can have 48 lines it looks better if this is limited to 46 or less to allow a margin above and below the text. Blank lines at the end of a page 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 f 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

```

DE 6502 KENNER

```
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
```

DE 6502 KENNER

```
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$(line%),col%,1))
        IF pica% THEN u! = 255 ELSE u! = 63 : REM set underline width
        IF (s! AND 8) = 0 THEN u! = 0 : REM inhibit underline
        IF pass% <> 3 THEN
          IF (s! AND 1) = 0 THEN u! = 0
          IF (pass% = 1) AND (pica% = FALSE) THEN u! = 0
        ENDIF
        OUT 0, 0,0,0,0,0,u! : REM space between lines
        c% = ASC(MID$(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
  PROCs_script
ENDPROC
```

DE 6502 KENNER

```
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 255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
DATA 16,16,56,56,84,84,146,146,16,16,16,16,16,16,0,0,0
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 124,130,162,162,162,162,186,130,130,130,130,124,0,0,0
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 5,5,5,5,5,5,5,9,9,17,17,97,97,0,0
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 60,2,2,2,2,2,60,2,2,2,2,60,0,0,0
DATA 0,66,66,66,66,66,60,2,2,2,2,0,0,0,0,0
DATA 60,64,64,64,64,64,60,2,2,2,2,60,0,0,0
DATA 60,64,64,64,64,64,64,60,66,66,66,66,60,0,0,0
DATA 60,2,2,2,2,0,2,2,2,2,0,0,0,0
DATA 60,66,66,66,66,66,60,66,66,66,66,66,60,0,0,0
DATA 60,66,66,66,66,66,60,2,2,2,2,60,0,0,0
DATA 12,18,33,32,32,32,120,32,32,32,32,127,0,0,0
```

DE 65 02 KENNER

DATA 248,128,128,128,240,128,128,254,16,16,16,30,0,0,0
DATA 170,85,170,85,170,85,170,85,170,85,170,85,170,85
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
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 16,16,16,16,16,16,16,16,0,0,16,0,0,0
DATA 0,36,36,36,0,0,0,0,0,0,0,0,0,0,0,0
DATA 36,36,36,36,126,36,36,126,36,36,36,36,0,0,0
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
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
DATA 4,4,8,8,16,16,16,16,8,8,4,4,0,0
DATA 32,32,16,16,8,8,8,8,16,16,32,32,0,0
DATA 0,0,68,68,40,40,254,40,40,68,68,0,0,0,0
DATA 0,0,16,16,16,16,124,16,16,16,16,0,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,16,16,32,32,0
DATA 0,0,0,0,0,124,0,0,0,0,0,0,0,0
DATA 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 56,68,68,130,130,130,130,130,68,68,56,0,0,0
DATA 16,48,48,16,16,16,16,16,16,16,16,124,0,0,0
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 252,128,128,128,248,4,2,2,2,130,68,56,0,0,0
DATA 56,68,130,128,128,248,132,130,130,68,56,0,0,0
DATA 254,130,130,4,4,8,8,16,16,16,16,16,0,0,0
DATA 56,68,130,130,130,68,56,68,130,130,130,68,56,0,0,0
DATA 56,68,130,130,130,66,62,2,2,2,4,120,0,0,0
DATA 0,0,0,16,16,0,0,0,16,16,0,0,0,0,0
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 56,68,130,130,130,130,254,130,130,130,130,130,0,0,0
DATA 248,132,130,130,130,132,248,132,130,130,130,132,248,0,0,0
DATA 56,68,130,128,128,128,128,128,128,130,68,56,0,0,0
DATA 248,132,130,130,130,130,130,130,130,130,132,248,0,0,0
DATA 254,128,128,128,128,248,128,128,128,128,254,0,0,0
DATA 254,128,128,128,128,248,128,128,128,128,128,0,0,0
DATA 62,64,128,128,128,128,142,130,130,130,130,68,56,0,0,0
DATA 130,130,130,130,130,130,254,130,130,130,130,130,0,0,0
DATA 124,16,16,16,16,16,16,16,16,16,16,124,0,0,0
DATA 2,2,2,2,2,2,2,2,130,68,56,0,0,0
DATA 130,130,132,132,136,136,240,136,136,132,132,130,0,0,0
DATA 128,128,128,128,128,128,128,128,128,128,254,0,0,0
DATA 130,198,198,170,170,146,146,130,130,130,130,130,0,0,0
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 248,132,130,130,130,132,248,128,128,128,128,128,0,0,0
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 254,16,16,16,16,16,16,16,16,16,16,0,0,0

DE 6502 KENNER

DATA 130,130,130,130,130,130,130,130,130,130,130,68,56,0,0,0
 DATA 130,130,130,130,130,130,130,130,68,68,40,40,16,16,0,0,0
 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 28,16,16,16,16,16,16,16,16,16,28,0,0,0
 DATA 128,128,64,64,32,32,16,16,8,8,4,4,2,0,0,0
 DATA 56,8,8,8,8,8,8,8,8,8,56,0,0,0
 DATA 16,16,40,40,68,68,130,130,0,0,0,0,0,0,0,0,0
 DATA 0,0,0,0,0,0,0,0,0,0,0,0,254,0,0,0
 DATA 0,16,16,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 DATA 0,0,0,120,132,4,4,124,132,132,122,0,0,0
 DATA 128,128,128,128,248,132,132,132,132,132,248,0,0,0
 DATA 0,0,0,0,120,132,128,128,128,128,132,120,0,0,0
 DATA 4,4,4,4,124,132,132,132,132,132,124,0,0,0
 DATA 0,0,0,0,120,132,132,132,248,128,128,120,0,0,0
 DATA 14,16,16,16,56,16,16,16,16,16,16,16,16,0,0,0
 DATA 0,0,0,0,120,132,132,132,132,124,4,4,4,120,0
 DATA 128,128,128,128,248,132,132,132,132,132,132,0,0,0
 DATA 16,0,0,0,48,16,16,16,16,16,16,16,56,0,0,0
 DATA 8,0,0,0,8,8,8,8,8,8,8,8,48,0
 DATA 128,128,136,136,144,144,224,144,144,136,132,132,0,0,0
 DATA 48,16,16,16,16,16,16,16,16,16,56,0,0,0
 DATA 0,0,0,0,236,146,146,146,146,130,130,130,130,0,0,0
 DATA 0,0,0,0,248,132,132,132,132,132,132,132,0,0,0
 DATA 0,0,0,0,120,132,132,132,132,132,132,120,0,0,0
 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 0,0,0,0,130,130,130,130,146,146,170,68,0,0,0
 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 48,8,8,8,8,4,8,8,8,8,8,8,48,0,0,0
 DATA 0,0,34,84,84,136,136,0,0,0,0,0,0,0,0,0,0,0
 DATA 0,0,16,16,40,40,68,68,130,130,255,0,0,0,0,0,0

Elite size	Elite size
! "£\$%^&*{ }	! "£\$%^&*{ }
-+Δ-=.,@,	-+Δ-=.,@,
{ }[]⟨⟩?#~	{ }[]⟨⟩?#~
1234567890	1234567890
ABCDEFGHIJ	ABCDEFGHIJ
KLMNOPQRST	KLMNOPQRST
UVWXYZ	UVWXYZ
abcdefghijklmnopqrstuvwxyz	abcdefghijklmnopqrstuvwxyz

Emphasized

Emphasized

super
scripts
bold super
scripts
subscripts
bold sub
scripts

super
scripts
bold super
scripts
subscripts
bold sub
scripts

Underline

Underline

5 ¾"

5 ¾"

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/E C65 ***

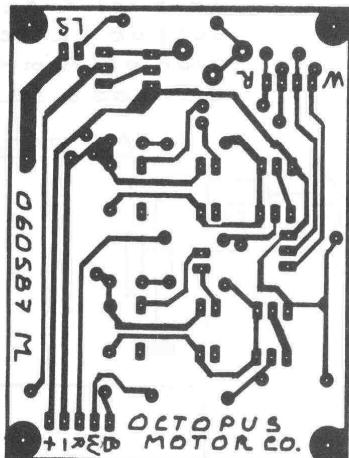
M. Lachaert.

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 remedies to this gap, and offers beside this a handy possibility to build in a 'bell' feature.

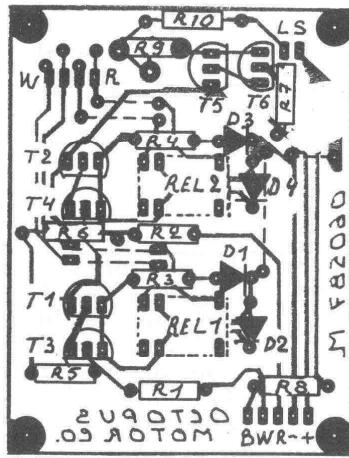
The circuit utilizes three free output lines of the port IC4 on the interface board, as described by Elektor. Output S6 (pin 5) controls the 'write' recorder motor, output S7 (pin 4) controls the 'read' recorder motor, while output S8 (pin 3) feeds a small amplifier connected to a miniature speaker,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



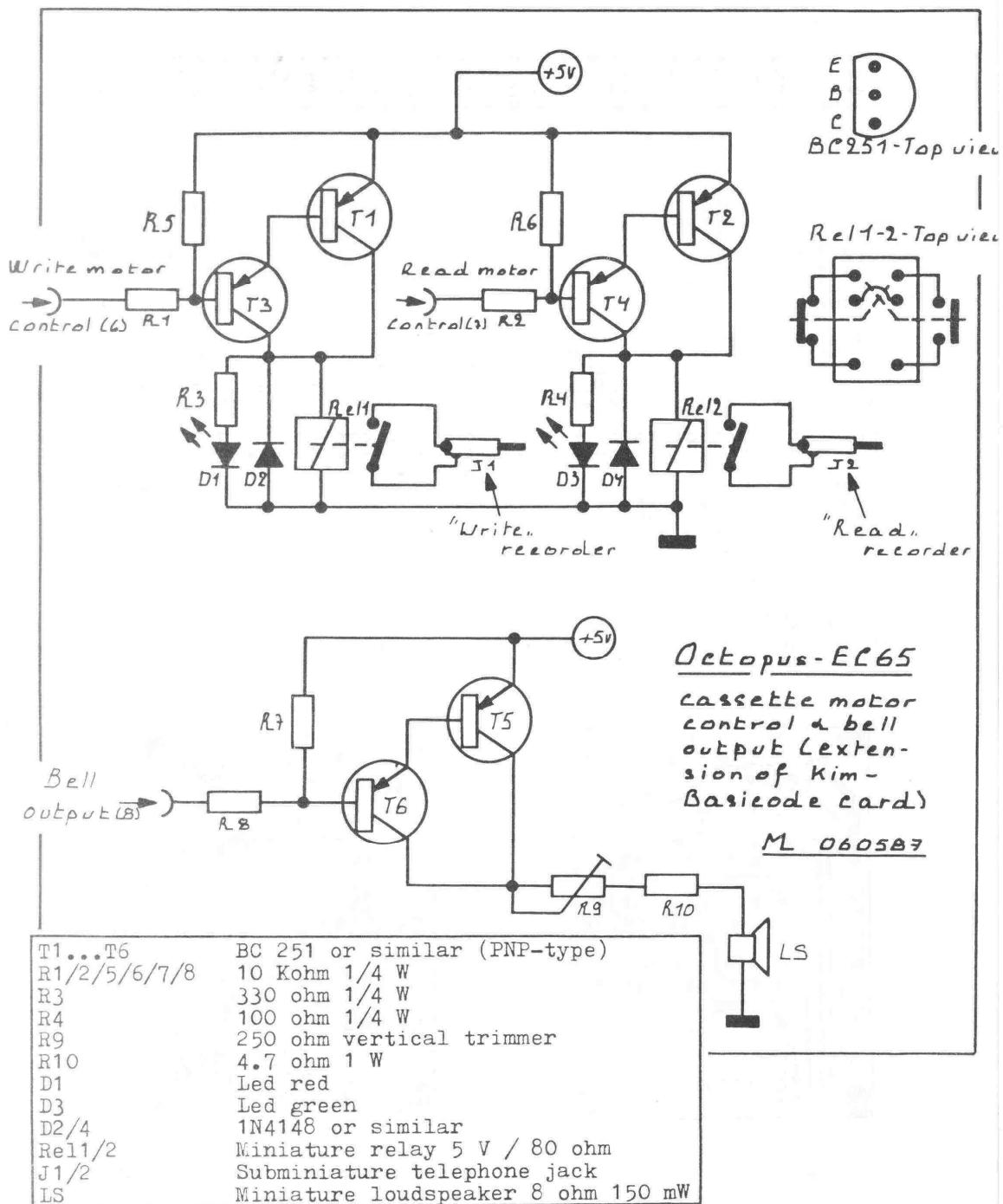
COPPER SIDE



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

DE 6502 KENNER



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 addressed at \$E280. An annoying 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:

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

DE 65 02 KENNER

 * 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 verwezelen. 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 oscll (*command) vector
70 save_vec_low=&70:save_vec_high=&71
80 REM startaddress where a *command starts (in ASCII)
90 oscll_low=&74:oscll_high=&75
100 len=&76:REM number of letters in a *command, flag
110     REM (0=command off, not 0=command 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=&FFFA:REM invoking OS facilities
240 osascii=&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 \oscll_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 oscll_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 oscll_low and high
440 STX oscll_low:STY oscll_high:LDX #0
450 .lus1
460 \number of letters from command to len
470 LDY #0:LDA (commands_low),Y:STA len
480 .lus2
490 \next letter from command. When * => command found
500 INY:LDA (oscll_low),Y:CMR #ASC"*:BEQ true
510 \make capital from typed letter. When typed letter
520 \and letter from command are not the same, not found
530 AND #223:CMR (commands_low),Y:BNE false
540 \when all letters from command 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 *command
620 INY:LDA (oscll_low),Y:CMR #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 (oscll_low),Y:AND #223
660 CMR #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:EDR 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"0":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 oscll_vector
970 LDX oscll_low:LDY oscll_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:CPY #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 osascii
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

```

DE 6502 KENNER

```

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:EQU "C 6502 KENNER"+CHR$(13)
1690 .commands
1700 EQUB 4:EQU "PICA"
1710 EQUB 5:EQU "ELITE"
1720 EQUB 12:EQU "PROPORTIONAL"
1730 EQUB 10:EQU "COMPRESSED"
1740 EQUB 10:EQU "EMPHASIZED"
1750 EQUB 12:EQU "DOUBLESTRIKE"
1760 EQUB 8:EQU "EXPANDED"
1770 EQUB 7:EQU "ITALICS"
1780 EQUB 11:EQU "UNDERLINING"
1790 EQUB 5:EQU "RESET"
1800 EQUB 5:EQU "HELP"+CHR$(13)
1810 EQUB 35:EQU "Wildcards (*) zijn ook"
1820 EQU "toegestaan."+CHR$(13)
1830 .sub
1840 EQU "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 transfer. 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 original 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 2-1.5 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 interrupt 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 come 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 TOPFSC 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.

DE 6502 KENNER

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 Baud is possible! Modulating with several modulation methods at the same time increases the transfer speed. E.g. only AM correspond with 600 Baud, AM & FSK results in 2x600=1200 Baud. Adding PSK increases the speed to 2x1200=2400 Baud. With special encoding and encryption algorithms it is even possible to reach a transferrate of 9600 Baud. Unfortunately there is not yet one uniform standard for 9600 Baud 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 (transparent modems). The Hayes 'AT' commandset is international standardized. With those commands it is possible to set the bautrate, to autodial, program the wordformat, etc. etc. Transparent 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

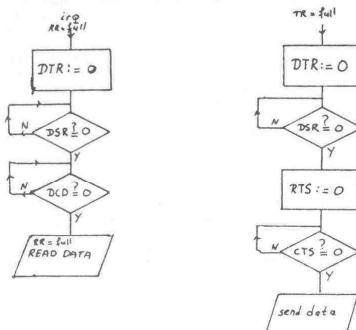
To let a computer 'talk' to another computer, a protocol is needed, to avoid misunderstanding. Very popular and many used protocols are:
 -Ascii : (wait/stop, ^s/^q). Only for textfiles.
 -Xmodem: Transfer of all kind of files (read article Xmodem J. Bansen soon published)
 -Kermit: idem.
 -Videotex: Transfer pages with text and graphics (read DE 6502 KENNER nr. 47/48)

5. Error detection 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 parity-bit. 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 horizontal 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 $\text{CRC}(Tx)/\text{CRC}(Rx)=\text{constant}$). For advanced errorchecking $\text{CRC}(16)$ is very popular. In the hobbyworld mostly $\text{CRC}(8)$ is used. $\text{CRC}(8)=\text{LRC} \Rightarrow G(x)=x^8+1$.

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 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
              ;Acint
              ;Receive and transmit data via acia
18AC AD 31E1  LDA ACIASR
18AF 29 08    AND #00001000 ;Transmitter full
18B1 F0 0D    BEQ TRANSMIT
18B3 AD 30E1  LDA AREG   ;Dataregister (re-
18B6 AE 2410  LDX RECPTN ;ceiver)
18B9 9D 0002  STA RECBUF,X
18BC EE 2410  INC RECPTN
18BF 60       RTS
18C0 AD 2510  TRANSMIT LDA TRANPNT ;Transmit buffer
              ;empty ?
18C3 F0 2C    BEQ DLYINT
18C5 AD 7610  LDA DLLAY ;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 ;Save first on stack
18D2 48       PHA
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 FFO2  STA TRANBUF-1,X
18E1 F8       INX
18E2 D0 F7    BNE TRANMOV
18E4 CE 2510  DEC TRANPNT ;Tranbuf:=tranbuf-1
18E7 68       PLA
18E8 58       CLI
18E9 8D 30E1  STA AREG ;And send byte to
              ;REGIST
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    LDY 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 (GEIMOD)

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.

DE 6502 KENNER

LIST 3

```

184B AE 2410 GETMOD LDX RECPNT ;char. received?
184E FO 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 . CPA RECPNT
1859 FO 09 BEQ 2.F
185B BD 0002 QUE LDA RECBUF,X ;Move queue
185E 9D FF01 STA RECBUF-1,X
1861 E2 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
2C2B 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 3B 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?
      LBS GMOD ;C=1 no key
      JSR EN2 ;Transmit key
      BCS TERMINAL ;Data 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.

- Videl 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 everyone's 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'll be able to make a free copy.



ONDERW : HOE HAAL JE MET DISKDOCTOR EEN DIRECTORY TERUG?
SISTEEM: 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 subdirectory terug te halen, moet men gaan zoeken naar het Track/Sector-nummer van die directory. Men zoekt in feite naar de verzameling filennamen die met een DIR dirspec/ op het scherm geprint worden. Hooft men die gevonden, dan staat bij diskdoctor onder aan het scherm op welke track en welke sector men zich bevindt. 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: (Heel! dat zijn de files die ik mis)

VIDTEL.MAC

ASTRID.MAC

MCNFIG.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/Sector 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.

DE 6502 KENNER

* B-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 eeproms 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		1050: 0252 20 E8 11	NXTCAL	JSR	CRLF
0210:					1060: 0255 20 F2 FA		JSR	PRINT
0220:	* EXISTENT SUBROUTINES *				1070: 0258 0A		=	\$OA
0230:					1080: 0259 47		=	'I
0240:	4B OC	CHKSUM	*	\$0C4B	resident in TM	1090: 025A 49		'V
0250:	5F 10	LABJUN	*	\$105F	warm start PM	1100: 025B 56		'E
0260:	E8 11	CRLF	*	\$11E8	carr.ret/l.feed	1110: 025C 45		'G
0270:	68 12	RESIN	*	\$1268	0 in databuffer	1120: 025D 20		'I
0280:	8F 12	PRBYTE	*	\$128F	print byte	1130: 025E 46		'F
0290:	34 13	PRCHA	*	\$1334	print character	1140: 025F 49		'I
0300:	87 13	INPAR	*	\$1387	input params	1150: 0260 52		'R
0310:	F2 FA	PRINT	*	\$FAF2		1160: 0261 53		'S
0320:	C5 FD	CHCK	*	\$FD5		1170: 0262 54		'T
0330:						1180: 0263 2C		,
0340:						1190: 0264 4C		'L
0350:						1200: 0265 41		'A
0360:	FA 00	POINTL	*	\$00FA		1210: 0266 53		'S
0370:	FB 00	POINTH	*	\$00FB		1220: 0267 54		'T
0380:	63 1A	PARAL	*	\$1A63		1230: 0268 20		
0390:	64 1A	PARAH	*	\$1A64		1240: 0269 4D		'M
0400:	65 1A	PARBL	*	\$1A65		1250: 026A 45		'E
0410:	66 1A	PARBH	*	\$1A66		1260: 026B 4D		'M
0420:	6E 1A	CHKL	*	\$1A6E		1270: 026C 4F		'O
0430:	6F 1A	CHKH	*	\$1A6F		1280: 026D 52		'R
0440:	7C 1A	BRKT	*	\$1ATC		1290: 026E 59		'Y
0450:						1300: 026F 20		,
0460:	0200 A9 5F	INIT	LDAIM	\$5F		1310: 0270 41		'A
0470:	0202 A0 10		LDYIM	\$10		1320: 0271 44		'D
0480:	0204 8D 7C 1A	STA	BRKT		set breakpointr	1330: 0272 44		'D
0490:	0207 8C 7D 1A	STY	BRKT	+01		1340: 0273 52		'R
0500:	020A A9 OC	PRTXTA	LDAIM	SOC		1350: 0274 45		'E
0510:	020C 20 34 13	JSR	PRCHA		clear screen	1360: 0275 53		'S
0520:	020E 20 FA 02	JSR	ASTER		25 asterisks	1370: 0276 53		'S
0530:	0212 20 EB 11	JSR	CRLF			1380: 0277 20		:
0540:	0215 20 F2 FA	JSR	PRINT		print title	1390: 0278 3A		:
0550:	0218 2A	TXTA	=	*		1400: 0279 20		
0560:	0219 2A		=	*		1410: 027A 03		\$03
0570:	021A 2A		=	*		1420: 027B 20 68 12	JSR	RESIN
0580:	021B 20		=	,		1430: 027B 20 87 13	JSR	INPAR
0590:	021C 48		=	'H		1440: 0281 30 87	BMI	PRTXTA
0600:	021D 2D		=	-		1450: 0283 20 C5 FD	JSR	CHCK
0610:	021E 43		=	'C		1460: 0286 90 82	BCC	PRTXTA
0620:	021F 4F		=	'O		1470: 0288 A9 00	LDAIM	reset inputbufs
0630:	0220 44		=	'D		1480: 028A 8D 6E 1A	STA	reset 2 adresse
0640:	0221 45		=	'E		1490: 028D 8D 6F 1A	STA	reset checksum
0650:	0222 20		=			1500: 0290 AD 63 1A	LDA	PARAL
0660:	0223 43		=	'C		1510: 0293 AC 64 1A	LDY	PARAH
0670:	0224 41		=	'A		1520: 0296 85 FA	STAZ	POINTL
0680:	0225 4C		=	'L		1530: 0298 84 FB	STYZ	POINTH
0690:	0226 43		=	'C		1540: 029A 00 00	CALCUL	prepare work-
0700:	0227 55		=	'U		1550: 029C B1 FA	LDYIM	pointers
0710:	0228 4C		=	'L		1560: 029E 20 4B OC	POINTL	start calcula-
0720:	0229 41		=	'A		1570: 02A1 E6 FA	JSR	tion
0730:	022A 54		=	'T		1580: 02A3 D0 02	INCZ	CHKSUM
0740:	022B 4F		=	'O		1590: 02A5 E6 FB	POINTL	
0750:	022C 52		=	'R		1600: 02A7 A5 FA	CNT	INCZ
0760:	022D 20		=	'-		1610: 02A9 8D 63 1A	STA	POINTL
0770:	022E 2A		=	*		1620: 02AC A5 FB	LDAZ	restore PARA
0780:	022F 2A		=	*		1630: 02AE 8D 64 1A	STA	POINTH
0790:	0230 2A		=	*		1640: 02B1 20 D8 02	JSR	NXT
0800:	0231 03		=	\$03		1650: 02B4 B0 E4	BCS	CALCUL
0810:	0232 20 EB 11	JSR	CRLF			1660: 02B6 20 F2 FA	JSR	if not, continue
0820:	0235 20 FA 02	JSR	ASTER		25 asterisks	1670: 02B9 0D	PRINT	else give result
0830:	0238 20 EB 11	JSR	CRLF			1680: 02BA 0A	=	\$OD
0840:	023B 20 F2 FA	JSR	PRINT		print text B	1690: 02BB 0A	=	\$OA
0850:	023E 0A	TXTB	=	\$OA		1700: 02BC 48		'H
0860:	023F 42		=	'B		1710: 02BD 2D		'-
0870:	0240 52		=	'R		1720: 02BE 43		'C
0880:	0241 4B		=	'K		1730: 02BF 4F		'O
0890:	0242 20		=			1740: 02C0 44		'D
0900:	0243 3D		=	'=		1750: 02C1 45		'E
0910:	0244 20		=			1760: 02C2 20		'-
0920:	0245 52		=	'R		1770: 02C3 3D		'=
0930:	0246 45		=	'E		1780: 02C4 20		
0940:	0247 54		=	'T		1790: 02C5 03		\$03
0950:	0248 55		=	'U		1800: 02C6 AD 6F 1A	LDA	CHKH
0960:	0249 52		=	'R		1810: 02C9 20 8F 12	JSR	PRBYTE
0970:	024A 4E		=	'N		1820: 02CC AD 6E 1A	LDA	CHKL
0980:	024B 20		=			1830: 02CF 20 8F 12	JSR	PRBYTE
0990:	024C 54		=	'T		1840: 02D2 20 E8 11	JSR	CRLF
1000:	024D 4F		=	'O		1850: 02D5 4C 52 02	JMP	NXTCAL
1010:	024E 20		=			1860: 02D8 18	NXT	CLC
1020:	024F 50		=	'P		1870: 02D9 AD 63 1A	LDA	PARAL
1030:	0250 4D		=	'M		1880: 02DC 69 01	ADCIM	\$01
1040:	0251 03		=	\$03		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 OC	BCS	NXTB
						1940: 02EB 38	SEC	branch if \$FFFF
						1950: 02EC AD 65 1A	LDA	is crossed
						1960: 02EF E5 FA	SBCZ	workpointer =
						1970: 02F1 AD 66 1A	LDA	
						1980: 02F4 E5 FB	SBCZ	
						1990: 02F6 60	NXTA	POINTH
						2000: 02F7 18	NXTB	carry depends on
						2010: 02F8 90 FC	CLC	PARB minus POINT
						2020:	BCC	or on crossing
								\$FFFF, the memory boundary

DE 65 02 KENNER

```

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 powerful, 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/l screen up/l screen down/to front of line/to rear of line/toggle writeover/insert/write graphic character/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/asim, 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
SCRATCH	26-26	WP2.P	27-30	W/ROM	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 packages and postages etc.
We accept no responsibility for damages etc. during transports.

```
*****
* Patch on Dr. Tietsch's Copier Program *
*****
```

P. Lindström & 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: 405B 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 ZR05
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 direct verbonden met de 'clear to send'-aansluiting (CTS) van de seriële 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 OD.

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 quite simple. 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.

DE 6502 KENNER

```
*****
* DATABASE 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 automatically 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 securities build in, therefore be careful 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 PBO, a short beep will then be heard, when the computer is ready with transmitting a page to the screen.

0720: MONITOR SUBROUTINES

0730:	5F 10	LARJUN *	\$105F	WARM START JUNIOR	
0750:	E8 11	CRLF *	\$11E8	CARRK.RET/LINEFEED	
0760:	6F 12	HEXNUM *	\$126F	ASCII TO HEX	
0770:	AE 12	RECHHA *	\$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	DDR8 *	\$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 PTR	
0990:	56 03	HALFFU *	\$0356	HALF/FULL LINE	
1000:					
1010:	*** PROGRAM START ***				
1020:					
1030: 0200	DATA	ORG	\$0200		
1040: 0200 A9 0C	INPUT	LDAM	\$0C		
1050: 0202 20 F7 03		JSR	CLRSR	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	INCRIM	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	LDAM	\$0C		
1200: 0227 20 F7 03		JSR	CLRSR	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		INCREMENT	LDA	COUNTR	GET CHAR.COUNTER
1340: 024B 18			CLC		
1350: 024B 65 00			ADC	ADDONE	ADD ONE
1360: 024B 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	\$1FOO	WRITE CHARACTER
1480: 0261 60			RTS		
1490:					
1500: 0262 A4 01		READ	LDY	COUNTR	
1510: 0264 B9 00 1F			LDAY	\$1FOO	READ CHARACTER
1520: 0267 60			RTS		
1530:					
1540: 0268 A5 01		LINEFU	LDA	COUNTR	
1550: 026A C9 FF			CMPIM	\$FF	FULL LINE ?
1560: 026C DO 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 DO 18			BNE	LINEC	NOT STAY ON SCREEN
1680: 0285 AD 60 02			LDA	WRTPTR	GET WRITE POINTER
1690: 0288 8D 6F 02			STA	SCREND	SAVE IN SCREEN-END
1700: 028A EE 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: 02A0 D6 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 POINTERS
1770: 029D 60			RTS		
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 99 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		FILLLA	TAX		
1950: 02B9 A9 20		FILLLC	LDAIM	\$20	FILL LINE
1960: 02BB 85 02			STA	CHARAC	WITH SPACES
1970: 02B6 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	RECHHA	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	\$OA	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

DE6502 KENNER

```

2240: 02F8 C6 01 DEC COUNTR IF SO, CHAR.CNTR-1 3060: 03B1 DO EB BNE PAGEA ERROR, TRY AGAIN
2250: 02FA 4C DO 02 JMP KEY IS IT A @ ? 3070: 03B3 FO EB BEQ PAGEB NEXT CHARACTER
2260: 02FD C9 40 KEYE CMPIM $40 IS IT A @ ? 3080: 03B5 20 C1 03 PAGEC JSR VECTOR GET PAGE VECTOR
2270: 02FF DO 03 BNE KEYF IF NOT, NEXT 3090: 03B8 4C 25 02 JMP PRINT AND PRINT
2280: 0301 4C 20 03 JMP CORREC IF SO, CORRECTION 3100: 03BB 20 C1 03 PAGED JSR VECTOR GET PAGE VECTOR
2290: 0304 C9 5D KEYF CMPIM 'J' IS IT A J ? 3110: 03B2 4C OD 02 JMP INPUTA WAIT FOR A CHAR.
2300: 0306 DO 03 BNE KEYG IS NOT, NEXT 3120:
2310: 0308 4C 95 03 JMP PAGE IF SO, PAGENUMBER? 3130: 03C1 A4 F8 VECTOR LDY INL GET BUFFER
2320: 030B 4C 18 02 KEYG JMP INPUTB SAVE CHARACTER 3140: 03C3 B9 EA 04 LDAAY VECTAB FETCH PAGE VECTOR
2330: 3100: 03C6 85 03 STA PAGCTR AND SAVE IT
2340: 030E A5 01 INCPRI LDA COUNTR GET CHAR.COUNTER 3160: 03C8 60 RTS
2350: 0310 18 CLC 3170:
2360: 0311 65 00 ADC ADDONE ADD ONE 3180: 03C9 20 E8 11 HALFLI JSR CRLF START ON NEW LINE
2370: 0313 85 01 STA COUNTR SAVE IT 3190: 03CC AO BF LDYIM $BF PRINT
2380: 0315 C9 00 CMPIM $00 COUNTER EMPTY ? 3200: 03CE 20 86 03 JSR TEXT TEXT STRING
2390: 0317 DO 06 BNE INCEND IF NOT, CONTINUE 3210: 03D1 20 AE 12 JSR RECHA WAIT FOR A CHAR.
2400: 0319 EE 60 02 INC WRTPTR IF SO, WRITE PTRN+1 3220: 03D4 C9 4A CMPIM 'J' IS IT A J ?
2410: 031C EE 66 02 INC RDPTR AND READ POINTER+1 3230: 03D6 FO 05 BEQ HALFA IF SO, CONTINUE
2420: 031F 60 INCEND RTS 3240: 03D8 C9 4E CMPIM 'N' IS IT A N ?
2430: 3100: 03D9 FO 0A BEQ HALFB IF SO, CONTINUE
2440: 0320 A9 1C CORREC LDAIM $1C 3250: 03DA FO 0A BEQ HALFC IF NOT, TRY AGAIN
2450: 0322 20 F7 03 JSR CLRSCR CURSOR HOME 3260: 03DC DO EB BNE HALFLI IF NOT, TRY AGAIN
2460: 0325 A5 03 LDA PAGCTR GET PAGE COUNTER 3270: 03DE A9 1F HALFA LDAIM $1F
2470: 0327 8D 60 02 STA WRTPTR WRITE FULL LINE 3280: 03E0 8D 56 03 STA HALFFU SET FOR HALF LINE
2480: 032A 8D 66 02 STA RDPTR AND READ POINTER 3290: 03E3 4C EB 03 JMP HALFC
2490: 032D 20 53 02 JSR ZEROST RESET COUNTERS 3300: 03E6 A9 3F HALFB LDAIM $3F
2500: 0330 20 AE 12 KEYX JSR RECHA WAIT FOR A CHAR. 3310: 03E8 8D 56 03 STA HALFFU SET FOR FULL LINE
2510: 0333 C9 40 CMPIM $40 IS IT A @ ? 3320: 03E9 20 E8 11 HALFC JSR CRLF START ON NEW LINE
2520: 0335 FO 19 BEQ CORRA 3330: 03EE A0 9E LDYIM $9E PRINT
2530: 0337 C9 23 CMPIM '#' IS IT A # ? 3340: 03F0 20 86 03 JSR TEXT TEXT STRING
2540: 0339 F9 32 BEQ CORRB 3350: 03F2 20 E8 11 JSR CRLF NEW LINE
2550: 033B C9 5E CMPIM '$' IS IT A ~ ? 3360: 03F6 60 RTS
2560: 033D FO 44 BEQ CORRC 3370:
2570: 033F C9 08 CMPIM $08 IS IT A BACKSPACE? 3380: 03F7 20 34 13 CLRSCR JSR PRCHA CLEAR
2580: 0341 FO 3B BEQ CORRD 3390: 03FA A9 80 LDAIM $80
2590: 0343 85 02 STA CHARAC SAVE CHARACTER 3400: 03FC BD F7 1A STA TIMER DELAY
2600: 0345 20 0E 03 JSR INCPRI INCREASE COUNTER 3410: 03FF 2C D5 1A CLR A BIT RDFLAG
2610: 0348 A5 02 LDA CHARAC GET CHARACTER 3420: 0402 10 FB BPL CLRA
2620: 034A 20 5C 02 JSR WRITE AND PRINT IT 3430: 0404 60 RTS
2630: 034D 40 30 03 JMP KEYX WAIT FOR A CHAR. 3440:
2640: 0350 A9 08 CORRA LDAIM $08 DELETE INSERTED 3450: 0405 A9 01 SOUND LDAIM $01
2650: 0352 20 34 13 JSR PRCHA CHARACTER 3460: 0407 BD 00 18 STA ORB SET DATA REGISTER
2660: 0355 A2 3F LDIXM $3F 3470: 040A BD 02 18 STA DDRB SET DIR. REGISTER
2670: 0357 20 0E 03 CORRF JSR INCPRI PRINT HALF 3480: 040D A9 7F LDAIM $7F SET DELAY
2680: 035A 20 62 02 JSR READ OR FULL LINE 3490: 040F B5 70 STA DELAY
2690: 035D C9 03 CMPIM $03 SCREEN END ? 3500: 0411 EE 00 18 SOUNA INC ORB SWITCH DATA
2700: 035P FO 09 BEQ CORRE IF SO, STOP 3510: 0414 A6 70 LDX DELAY REGISTER ON
2710: 0361 20 34 13 JSR PRCHA IF NOT, PRINT 3520: 0416 EB SOUNB INX AND OFF
2720: 0364 CA DEX AS LONG AS 3530: 0417 DO FD BNE SOUNB
2730: 0365 10 FO CORRF LINE NOT FULL AND 3540: 0419 C6 70 DEC DELAY WAIT
2740: 0367 40 30 03 JMP KEYX WAIT FOR A CHAR. 3550: 041B C6 70 DEC DELAY
2750: 036A 45 15 02 CORRE JMP AGAIN 3560: 041D 30 03 BMI SOUNC
2760: 036B 20 0E 03 CORRB JSR INCPRI CHAR.COUNTER + 1 3570: 041F 4C 11 04 JMP SOUNA
2770: 0370 A9 08 LDAIM $08 DELETE INSERTED 3580: 0422 60 SOUNC RTS
2780: 0372 20 34 13 JSR PRCHA CHARACTER AND 3590:
2790: 0375 20 62 02 JSR READ PRINT 1 CHAR. 3600:
2800: 0378 20 34 13 JSR PRCHA FROM PAGE 3610: 0423 MESSAG ORG $0423
2810: 037B 4C 30 03 JMP KEYX WAIT FOR A CHAR. 3620: 04EA VECTAB ORG $04EA
2820: 037E C6 01 CORRD DEC COUNTER 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 ? 41 41 54 41 42 41 4E 4B OA OD 2D 2D 2D 2D DATABANK ----
2880: 038B FO 07 BEQ TEXEND IF SO, STOP 2D 2D 2D OA OD 56 41 4C 49 44 20 43 4F 4D 4D --- VALID COMM
2890: 038D 20 34 13 JSR PRCHA IF NOT, PRINT 41 4E 44 53 3A OA OD 58 20 3D 20 50 52 49 4E ANDS: ^ = PRIN
2900: 0390 C8 INY 54 20 50 41 47 45 04 OD 5B 20 30 20 48 45 57 T PAGE: [ = NEW
2910: 0391 4C 86 03 JMP TEXT NEXT CHARACTER 20 53 54 41 52 54 0A OD 2A 20 3D 20 4D 4F 4E START: * = MON
2920: 0394 60 TEXEND RTS 49 54 4F 52 0A OD 5D 20 3D 20 50 41 47 45 20 ITOR ] = PAGE
2930:
2940: 0395 A9 00 PAGE LDAIM $00 4E 55 4D 42 52 0A OD 40 20 3D 20 53 45 41 NUMBER @ = SEA
2950: 0397 85 FB STA INL CLEAR INPUT BUFFER 52 43 48 20 49 4E 45 0A OD 23 20 3D 20 43 RCH LINE # = C
2960: 0399 A9 08 LDAIM $08 DELETE INSERTED 55 52 53 4F 52 2D 2B 3B OA OD 24 20 3D 20 4E USROR--> $ = N
2970: 039B 20 34 13 JSR PRCHA CHARACTER 45 57 20 44 41 54 41 0A OD 03 53 54 41 52 54 EW DATA START
2980: 039E A6 B6 PAGEA LDYIM $B6 PRINT 20 57 49 48 20 50 41 47 45 20 48 55 4D 42 WITH PAGE NUMB
2990: 03A0 2B 03 03 JSR TEXT TEXT STRING 45 52 03 20 50 41 47 45 3F 03 48 41 4C 46 20 ER PAGE? HALF
3000: 03A3 20 AE 12 PAGEB JSR RECHA WAIT FOR A CHAR. 4C 49 48 20 53 45 41 52 43 48 20 2B 4A 2F LINE SEARCH (J/
3010: 03A6 C9 09 JMP CMPIM $OD IS IT A RETURN ? 4E 29 20 03 1F 23 27 2B 2F 33 37 3B 3F N)
3020: 03A8 FO 0B BEQ PAGEC IF SO, PRINT PAGE 3F 3F 3F 3F 43 47 48 4F 53 57 5B 5F 63 67
3030: 03AA C9 24 CMPIM '$' IS IT A $ ? 67 67 67 67 67 6B 6F
3040: 03AC FO 0D BEQ PAGED IF SO, BACK INPUT
3050: 03AE 20 6F 12 JSR HEXNUM TRANSFER NUMERIC

```

DISKETTES 80/40 TRS, SS,DD FOR ELECTOR'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.

DE 6802 KENNER

***** * OMEGA: THE DESKTOP MAINFRAME * *****

De redactie was geïnteresseerd 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-acquisitionssysteem 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 structuur. 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 systeemfrequentie bedraagt 12,5 Mhz, terwijl hogere frequenties (16,67 ,20 of 25 Mhz) tegen meerprijs mogelijk zijn. Het geheugen bestaat uit 128/256k byte Ram en 1 Megabyte no wait-state statisch, niet vluchting Cmos Ram. Zelfs in de 25 Mhz uitvoering worden alle lees- en schrijffopdrachten 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 structuur op user niveau (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 efficiënt geschreven en volledig 'rommable', dat wil zeggen het kan in Eeprom gezet worden en draaien zonder ondersteuning van hard- of floppy disk, dus zonder mechanische delen.

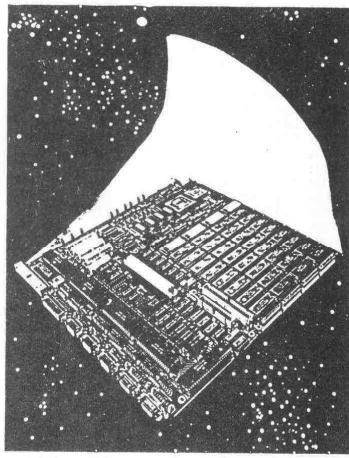
De opstekkaarten die op dit moment vorhanden zijn bestaan uit een 8 kanals RS232 kaart, 4 Megabyte statische Cmos Ram met battery backup, een hoge resolutie grafische kleurenaart (640x480 punten, 16 kleuren uit een pallet van 4096), een 'prototyping' kaart (ruimte gereserveerd voor ontwikkeling van eigen I/O, etc) en een adapter kaart voor G64...STE..IBM PC... bus, die toegang geeft tot een uitgebreide serie 1/I 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 Eeprom via een single-user werkstation tot een krachtig mainframe-achtig netwerk met meer dan honderd stations en zondig 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 verwerkingsnelheid 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-vluchting geheugen van 5 Megabyte en de netwerkfaciliteiten bieden ruime mogelijkheden voor inzet in een industriële 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 storingsgevoeligheid in een industriële 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 'Coris 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 Lachae'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+OL/HIPAR+OL) 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.

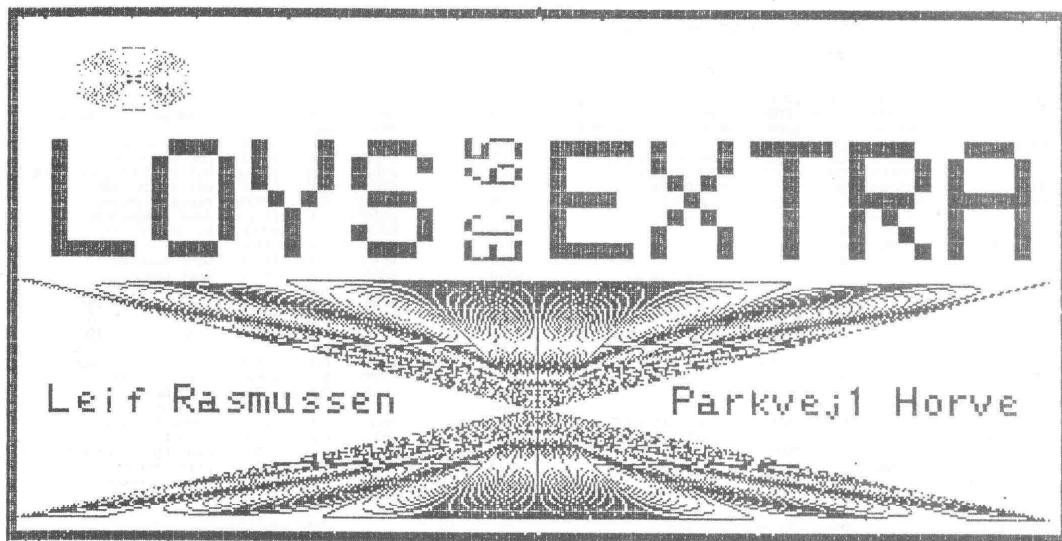
APPLE'S NIEUWE SOFTWARE BEDRIJP 'CLARIS'

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 producten zijn MacWrite, MacDraw, MacProject, MacPaint, AppleWorks en Accees].

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 producten 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, voorhen 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 "LOVS 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 Asmbl. (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 : DISK1"RE E".

ad 6: The Kolorator software by P. Lavigne can be installed in the system like in this proposal with: DISK1"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 DISK1"RE R", then resequencer (or !) is enabled, and if you want edit-command back, you type DISK1"RE I" (or use the much more comfortable full screen editor instead).

ad 9: With DISK1"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 indispensable 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 a f t e r reset).

I N S T A L L A T I O N :

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 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 read you get by save the machine code evolving when running the RSEQ from EC 2. (NB: *SA 35,l=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 equipped track with *SA XX,l=DB00/8.

-- 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 seems 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).

DE 6502 KENNER

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 ++
OOE1      OSIBAD EQU $00E1   POINT TO CMD BUFFER
($2E1E IN DOS, $D800 IN BASIC)
2CE5      BUFINC 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      UNLHDH EQU $2761   UNLOAD HEAD
E5AE      WRDIR EQU $E5AE   WRITE DIR.

D800 FF    TEMPB HEX FF
D801 FF    TEMPBC 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 BUFINC POINT TO CMD-START
D80D B1 E1    CMDBEG LDAIY OSIBAD GET CHR IN CMD
D80F C9 0E    CMPIM $OE    ?END OF CMD
D811 90 06  BCC CMDEND
D813 E8    INX
D814 C8    INY
D815 E0 06  CPYIM $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 CO 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 TEMPB
D831 AE E5 2C LDX BUFIND :: COMPARE 2 NAMES ::
D834 A0 00  LDYIM $00
D836 8C 00 D8 GTNM STY TEMPB
D839 8A    TXA
D83A A8    TAY
D83B B1 E1  LDAIY OSIBAD GET CMD-NAME CHR
D83D AC 00 D8 LDY TEMPB
D840 C9 0E    CMPIM $OE    IF END CMP LENGTH
D842 90 32  BCC CMPLNG
D844 D1 E0    CMPYI PTR COMPARE DIR-NAME
D846 D0 08  BNE NXTDN NO FIT, TRY NEXT
D848 E8    INX
D849 C8    INY
D84A CO 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
D85A 55 10  NOINC LDA PTR
D85D 38    SEC
D85E 89 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 :: FILE :::
D869 OD 0A  HEX ODOA
D86B 4E 4F 20 ASC NO FILE
D871 45
D872 OA OD 00  HEX OAODOO
D875 60
D876 AD 01 D8 CMPLNG LDA TEMPB COMP CMD-FN LENGTH
D879 CD 02 D8 CMP TEMPD WITH DIR-NM LENGTH
D87D DO D2  BNE NXTDN
D87B 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  LDYIM $01    :: BUFFER TO DISK :::
D891 20 AE E5 JSR WRDIR
D894 20 61 27 JSR UNLHDH
D897 20 73 2D JSR STROUT :: 'FILE ERASED' :::
D89A OD 0A  HEX ODOA
D89C 46 49 4C ASC FILE ERASED
D89F 45 20 45
D8A2 52 41 53
```

D8A5 45 44
D8A7 OA OD 00
D8AA 60

HEX OAODOO
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 ++
OOPE      MEMLO EQU $00E1   SPE load-pntr for sector
MEMHI EQU $FF
00FF      TRACEO ECU $07DB tracebasiclines on/off
07DB      CMTBL EQU $02C5 cmd table EDIT/RSEQ
02C5      STARTA EQU $0222 holds ed/rseq startadr-1
0222      376F      STARTE EQU $376F edits startadr.
376F      STARTR EQU $BB9B rseqs startadr.
BB9B      EDIMOLO EQU $BC00 edmo loadvector
BC00      KOLOILO EQU $C000 kolorator --
BB00      RSEQILLO EQU $B800 rseq --
E72A      ASBOOT EQU $E72A boot assembler
E71D      BABOOT EQU $E71D boot basic
E737      WPBOOT EQU $E737 boot wordprocessor
2300      MENSIZ 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 sectornrmb.
26BC      LDHEAD EQU $2754 load head
2754      SETTNM EQU $26BC position head
2967      READDK EQU $2967 read sector from disk
2AC0      ERROR EQU $2AC0 error message
2C4C      SETDRV EQU $2C4C select drive
2C4E      BUFBYT EQU $2CE4 read byte incmd-buffer
2D73      STROUT EQU $2D73 print string
F32F      CLS EQU $F32F clear screen
F707      INIKBD EQU $F707
-----SUB ROUTINES-----
D8B0 20 E4 2C JSR BUFBYT GET CMD
D8B3 C9 41 CMPIM $41 A? ASSEMBLER
D8B5 DO 06 BNE BAS
D8B7 20 08 D9 JSR SETA
D8B8 4C 2A E7 JMP ASBOOT
D8B9 C9 42 BAS CMPIM $42 B? BASIC
D8B7 20 08 D9 BNE WP
D8B8 4C 2A E7 JSR SETA
D8C1 20 08 D9 JMP BABOOT
D8C4 4C 1D E7 D8C4 4C 1D E7
D8C7 C9 57 WP CMPIM $57 W? WORD-PROCESSOR
D8C9 DO 06 BNE JUNMON
D8CB 20 08 D9 JSR SETA
D8CE 4C 37 E7 JMP WPBOOT
D8D1 C9 4D JUNMON CMPIM $4D M? JUNIOR-MONITOR
D8D3 DO 06 BNE EDMON
D8D5 20 44 26 JSR SWAPAB
D8D8 6C FC FF JMP $FFFC
D8DB C9 45 EDMON CMPIM $45 E? EDITOR/MONITOR
D8DD DO 06 BNE KOLOR
D8DF 20 08 D9 JSR SETA
D8E2 4C 2B D9 KOLOR CMPIM $4B K? KOLORATOR
D8E5 C9 4B D8E5 C9 4B
D8E7 DO 06 BNE RSEQ?
D8E9 20 08 D9 JSR SETA
D8F6 4C A7 D9 JMP RSEQLO
D8EF 4C 77 D9 JMP KOLORLO
D8F3 C9 52 RSEQ? CMPIM $52 R? RSEQ
D8F1 DO 06 BNE EDIT?
D8F3 20 08 D9 JSR SETA
D8F6 4C A7 D9 JMP RSEQLO
D8F9 C9 49 EDIT? CMPIM $49 I? EDIT
D8FB DO 03 BNE TRACEL
D8FD 4C F8 D9 JMP EDITLO
D900 C9 54 TRACE1 CMPIM $54 T? TRACE
D902 DO 03 BNE NOMORE
D904 4C 10 DA JMP TRACE2
D907 4C CO 2A NOMORE JMP ERROR
-----SUB ROUTINES-----
D90A 00 SAVDRV HEX OO remember last drive
D90B AD 5C 26 SETA LDA DRIVES
D90E 8D 0A D9 STA SAVDRV
D911 A9 01 LDAIM $01 allways load from dr A
D913 20 4C 2C SETD JSR SETDRV
D916 60 RTS
D917 AD 0D 09 RETDRV LDA SAVDRV return to last dr
D91A C4 13 D9 JMP SETD
D91D 85 FE STAXMEM STA MEMLO store load vector
D91F 86 FF STA MEMHI
D921 60 RTS
D922 8E 5E 26 RTTSAX STX SECINNM store sectornrmb.
D925 20 BC 26 JSR SETIK put head on track
D928 4C 67 29 JMP READRK 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 -->
```

DE 6502 KENNER

D945 A2 01	LDXIM \$01	DA1E A2 05	TRA-OFF	LDXIM TABL4	-TABL3
D947 20 22 D9	JSR RTTSAX SC400 = EDMO / 2	DA20 A0 00	TRACE3	LDVIM \$00	
D94A A9 00	LDAIM \$00	DA22 BD 2F DA	TRACE4	LDAX TABL3	
D94C A2 CC	LDXIM \$CC	DA25 99 DB 07	STAY	TRACE0	
D94E 20 1D D9	JSR STAXMEM	DA28 E8	INX		
D951 A9 09	LDAIM \$09 TRACK 09,1 -->	DA29 C8	INY		
D953 A2 01	LDXIM \$01	DA2A C0 04	CPYIM \$04		
D955 20 22 D9	JSR RTTSAX SCC00 = EDMO / 3	DA2C D0 F4	BNE	TRACE4	
D958 20 17 D9	JSR RETDRIV	DA2E 60	RTS		return to basic
D959 68	PLA	DA2F 20 D8 1C TABL3	HEX	20D81CEAEA	
D95C 8D 75 D9	STA SAVE1 save return adr.	DA32 EA EA			
D95F 68	PLA	DA34 18 90 02 TABL4	HEX	189002E6C8	
D960 8D 76 D9	STA SAVE2	DA37 E6 C8			
D963 20 03 BC	JSR EDIMOLO +03 goto editor/monitor	LOYS 3.1 EXTRA'S -AUTOLINE-			
D966 20 2F F3	JSR CLS after exit edmo,	DA70	ORG	\$DA70	
D969 20 07 F7	JSR INIKBD clear screen	TTL	LOYS 3.1 EXTRA'S -AUTOLINE-		
D96C AD 76 D9	LDA SAVE2 get return adr.	TEMPA	EQU	\$DA60	
D96F 48	RHA	DA61	LNL	TEMPA	+01 LINE NR.
D970 AD 75 D9	LDA SAVE1	DA62	LNH	TEMPA	+02
D973 48	RHA	DA63	INCHR	TEMPA	+03 INCREMENT
D974 60	RTS return to basic or dos	DA64	INCR	TEMPA	+04
D975 00	SAVE1 HEX 00	DA65	INPUT	TEMPA	+05 CHARACTER BUFFER
D976 00	HEX 00	DA66	LINPRO	TEMPA	+06 LINE INP IN PROG FLG
D977 20 54 27	KOLORLO JSR LDHEAD load kolorator	DA68	FIGNT	TEMPA	+08 CHR COUNTER, LINE NO
D97A A9 00	LDAIM KOLOILO	DA69	COUNT	TEMPA	+09
D97C A2 C0	LDXIM KOLOILO /256	DA6A	OUTLN	TEMPA	+0A OUT BUFFER LINE NO
D97E 20 1D D9	JSR STAXMEM	DA6B	CLNL	TEMPA	+0B
D981 A9 20	LDAIM \$20 track 20,1-->	DA6C	CLNH	TEMPA	+0C
D983 A2 01	LDXIM \$01	DA6D	TEMPX	TEMPA	+0D
D985 20 22 D9	JSR RTTSAX \$C000 KOLOR./1	DA6E	TEMPY	TEMPA	+0E
D988 A9 00	LDAIM \$00	DA6F	AUTOFL	TEMPA	+0F AUTOLINE ON/OFF FL
D98A A2 C8	LDXIM \$C8	E7C2	PARBL	E\$7C2	
D98C 20 1D D9	JSR STAXMEM	E7C3	PARBH	PARBL	+01
D98F A9 21	LDAIM \$21 track 21,1-->	::: EXTERNAL ADDRESSES :::			
D991 A2 01	LDXIM \$01	F32F	RESET	E\$32F	CLEAR SCREEN
D993 20 22 D9	JSR RTTSAX \$C800 KOLOR./2	FA90	IPB	E\$FA90	INPUT MATRIX
D996 20 17 D9	JSR RETDRIV	FA21	RESPAR	E\$FA21	RESET PARAL &PARBL
D999 A9 02	LDAIM KOLOILO +02 set device #2 outp	2D73	STROUT	E\$2D73	PRINT STRING
D99B A2 C0	LDXIM KOLOILO /256 to kolor.	F71D	RECHA	E\$71D	GET CHR FROM KBD
D99D 8D 13 23	STA MEMSIZ +13	0474	BASIC	E\$0474	BASIC WARM
D9A0 8B 14 23	STX MEMSIZ +14	2336	INBAS	E\$2336	BASIC INPUTVEC
D9A3 20 00 CO	JSR KOLOILO initiate kolor.	0588	BASIN	E\$0588	BASIC IN
D9A6 60	RTS return to basic or dos	00CC	INL	E\$00CC	
D9A7 20 54 27	RSEQLO JSR LDHEAD	00CD	INH	E\$00CD	
D9AA A9 00	LDAIM RSBQILO	TSAVER LDA AUTOFL			
D9AC A2 BB	LDXIM RSBQILO /256	DA70 20 1D F7	JSR RECHA	CHANGE ADR \$DF48,49	
D9AB 20 1D D9	JSR STAXMEM	DA73 C9 40	CMPIM \$40	TO 70,DA	
D9B1 A9 35	LDAIM \$35 track 35,1 -->	DA75 F0 01	BBQ TSAVER	COMMERCIAL AT TOGGLES	
D9B3 A2 01	LDXIM \$01	DA77 60	RTS		
D9B5 20 22 D9	JSR RTTSAX \$BBOO RSEQ	TSAVER LDA AUTOFL			
D9B8 20 17 D9	JSR RETDRIV	DA78 AD 6F DA	STA TEMPX		
D9B9 A9 BA	LDYIM \$BA memory top \$BA00	DA79 40 FF	EORIM \$FF		
D9BD A9 9B	LDAIM STARTR start rseq \$BB9B -1	DA7D 8D 6F DA	STA AUTOFL		
D9BF A2 BB	LDXIM STARTR /256	DA80 F0 03	BEQ AUTO		
D9C1 20 CB D9	JSR STAADR startadr.to dos cmdtab'	DA82 4C 7F DB	JMP BACK		AUTOLINE OFF
D9C4 A0 FF	LDYIM \$FF counter	DA85 A5 CC	AUTO	LDA INL	SAVE INL&INH FOR LATER
D9C6 A2 00	LDXIM TABL1 -TABL1	DA87 8D 6D DA	STA TEMPX		USE IN BASIC
D9C8 4C D5 D9	JMP WRCMD write RSEQ in cmdtab'	DA88 A5 CD	STA INH		
D9CB 8D 22 02	STAADR STA STARTA	DA8A 8D 6E DA	STA TEMPY		
D9CE 8E 23 02	STX STARTA +01	DA8C 8D 6E DA	STA DABF		
D9D1 8C 00 23	STY MEMSIZ	DA8F 20 73 2D	JSR STROUT		
D9D4 60	RTS	DA92 00 OA	HEX ODOA		
D9D5 BD 08 DA	WRCMD LDAX TABL1	DA94 53 54 41	ASC START LINE		
D9D8 20 43 23	JSR PRINT	DA97 52 54 20			
D9D9 C8	INY	DA9A 4C 49 4B			
D9DC CO 03	CPYIM \$03	DA9D 45 20			
D9DE DO 03	BNE STACMD	DA9F 00	HEX 00		
D9E0 18	CLC	DAAO 20 BC DB	JSR TXT	'4 DIGITS'	
D9E1 69 80	ADCIM \$80	DAA3 20 21 FA	JSR RESPAR	GET PARAMETERS	
D9E3 99 C5 02	STACMD STAY CMDTBL	DAA6 20 90 FA	JSR IPB		
D9E6 E8	INX	DAA9 AD C2 E7	LDA PARBL		
D9E7 CO 03	CPYIM \$03	DAAC 8D 62 DA	STA LNH		
D9E9 D0 EA	BNE WRCMD	DAAF 8D 6C DA	STA CLNH		
D9E8 20 73 2D	JSR STROUT	DAAB 8D C3 E7	LDA PARBH		
D9EE 20 45 4E	ASC ENABLED	DA5B 8D 61 DA	STA LNL	START LINE	
D9F1 41 42 4C		DA8B 8D 6B DA	STA CLNL		
D9F4 45 44		DAAB 20 73 2D	JSR STROUT		
D9F6 00	HEX 00	DAE8 0D 0A	HEX ODOA		
D9F7 60	RTS return to basic or dos	DAE9 49 4E 43	ASC INCREMENT		
D98 A0 BF	EDITLO LDYIM \$BF memorytop \$BF00	DAE3 52 45 4D			
D9FA A9 6F	LDAIM STARTE ed startadr.to cmdtab'	DAE6 45 4E 54			
D9FC A2 37	LDXIM STARTE /256	DAE9 20 20			
D9FE 20 CB D9	JSR STAADR	DABC 00	HEX 00		
DA01 A0 FF	LDYIM \$FF counter	DACC 20 8C DB	JSR TXT		
DA03 A2 04	LDXIM TABL2 -TABL1	DACF 20 21 FA	JSR RESPAR	GET PARAMETERS	
DA05 4C D5 D9	JMP WRCMD write EDIT in cmdtab'	DAD2 20 90 FA	JSR IPB		
DA08 52 53 45	TABL1 ASC RSEQ	DAD5 AD C2 E7	LDA PARBL		
DA0B 51		DADB 8D 64 DA	STA INCR	INCREMENTS	
DA0C 45 44 49	TABL2 ASC EDIT	DADB AD C3 E7	LDA PARBH		
DA0F 54		DADE 8D 63 DA	STA INCRH		
DA10 A9 00	TRACE2 LDAIM \$00 toggle trace on/off	DAE1 AD 6D DA	LDA TEMPX	RESTORE INL AND INH	
DA12 49 01	EORIM \$01	DAE4 85 CC	STA INL		
DA14 8D 11 DA	STA TRACE2 +01	DAE6 AD 6E DA	LDA TEMPY		
DA17 F0 05	BEQ TRA-OFF	DAE9 85 CD	STA INH		
DA19 A2 00	LDXIM TABL3 -TABL3	DAEB 18	CLC		
DA1B 4C 20 DA	JMP TRACE3	DAEC A9 06	LDAIM BEGIN	CHANGE INPVEC	

DE 6502 KENNER

```

DAEE 8D 88 05 STA BASIN      /256
DAF1 A9 DB LDAIM BEGIN +01
DAF3 BD 89 05 STA BASIN      +01
DAF6 A9 0D LDAIM $0D
DAF8 BD 65 DA STA INPUT
DAFB A9 00 LDAIM $00
DAFD BD 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 FO 0C BEQ INCREM
DB0F 20 36 23 NEW JSR INBAS GET CHR FROM KBD
DB12 BD 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 DO 08 BNE NFIRST
DB20 A9 04 LDAIM $04 4 DIGITS
DB22 BD 66 DA STA LINPRO
DB25 BD 68 DA STA FIGCNT
DB28 AD 68 DA NFIRST LDA FIGCNT
DB2B FO 2A BEQ LAST
DB2D A9 00 LDAIM $00
DB2F 8D 60 DA STA TEMPA
DB32 A9 04 LDAIM $04
DB34 BD 69 DA STA COUNT
DB37 18 SHIFT CLC
DB38 2E 62 DA ROL LNH SHIFT NEXT DIGIT TO
DB38 2E 61 DA ROL LNL LINE NO.
DB38 2E 60 DA ROL TEMPA
DB41 CE 69 DA DEC COUNT
DB44 DO F1 BNE SHIFT
DB46 CE 68 DA DEC FIGCNT
DB49 AD 60 DA LDA TEMPA CONVERT LINENR >
DB4C 69 30 ADCIM $30 ASCII
DB48 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 INCR
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
DB9C 60       HEX 00
               RTS

LOYS EXTRA'S -SHORTHAND-
DBAO ORG $DBAO
TTL LOYS EXTRA'S -SHORTHAND-
F71D RECHHA EQU $F71D GET CHR. FROM KBD
E660 TEMPY EQU $E660
E6A9 OLDRUT EQU $E6A9 ORG. SHORTH.
E6A4 GETCMD EQU $E6A4 BASIC CMD. TABL
O284 BASCOM EQU $0284

DBAO 20 1D F7 JSR RECHHA GET CHR.
DBA3 A0 00 LDYIM TBL1 -TBL1
DBA5 C9 4B CMPIM 'K disk!"'
DBA7 FO 3B BEQ SAVEY
DBA9 A0 06 LDYIM TBL2 -TBL1
DBAB C9 3A CMPIM ';' print#2,chr$(18)"
DBAD FO 35 BEQ SAVEY
DBAF A0 13 LDYIM TBL3 -TBL1
DBB1 C9 2F CMPIM '/' rnd(l)

DBB3 FO 2F BEQ SAVEY
DBB5 A0 19 LDYIM TBL4 -TBL1
DBB7 C9 48 CMPIM 'H chr$(1
DBB9 FO 28 BEQ SAVEY
DBBB A0 1E LDYIM TBL5 -TBL1
DBBD C9 28 CMPIM '& print&(1
DBBF FO 23 BEQ SAVEY
DBC1 A0 25 LDYIM TBL6 -TBL1
DBC3 C9 45 CMPIM 'E peek(1
DBC5 FO 1D BEQ SAVEY
DBC7 A0 2A LDYIM TBL7 -TBL1
DBC9 C9 7B CMPIM 'I disk!"' re
DBC8 FO 17 BEQ SAVEY
DBC9 A0 33 LDYIM TBL8 -TBL1
DBC9 C9 7D CMPIM ') disk!"' put
DBC1 FO 11 BEQ SAVEY
DBC3 A0 3D LDYIM TBL9 -TBL1
DBC5 C9 7C CMPIM 'J disk!"' lo
DBC7 FO 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 3B 29
DC08 A2 HEX A2
DC09 52 4E 44 TBL3 ASC RND(1
DC0C 2B 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 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 SDC40
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 FO 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 DEB ERR1 -ERR1
DC53 07 DEB ERR2 -ERR1
DC54 0E DEB ERR3 -ERR1
DC55 16 DEB ERR4 -ERR1
DC56 24 DEB ERR5 -ERR1
DC57 29 DEB ERR6 -ERR1
DC58 39 DEB ERR7 -ERR1
DC59 40 DEB ERR8 -ERR1
DC5A 4E DEB ERR9 -ERR1
DC5B 5B DEB ERRA -ERR1
DC5C 69 DEB ERRB -ERR1
DC5D 7B DEB ERRC -ERR1
DC5E 83 DEB ERRD -ERR1
DC5F 95 DEB ERRE -ERR1
DC60 50 41 52 ERR1 ASC PARITY
DC63 49 54 59

```

DE 6502 KENNER

DC66 00		HEX	00		200 DIMLL31;MM31;F,N=0T031;LLN=#777;MMN=#777;N.
DC67 52 45 52	ERR2	ASC	REREAD		210 P."ASSEMBLY PHASE 1", \$21
DC6A 45 41 44					220 GDS,a
DC6D 00		HEX	00		230 F,N=0T031;MMN=LLN-S+T;N.
DC6E 54 52 41	ERR3	ASC	TRACK 0		240 P,\$6."ASSEMBLY PHASE 2", \$21
DC71 43 4B 20					250 GDS,a;P,\$6
DC74 30					260 END
DC75 00		HEX	00		270aP=S
DC76 57 52 49	ERR4	ASC	WRITE PROTECT		280I
DC79 54 45 20					290:LL0:SET:LDA I;STA R;LDA I+1;STA R+1
DC7C 50 52 4F					300LDA @MM2&#FF;STA I;LDA @MM2/256;STA I+1
DC7F 54 45 43					310LDA J;STA R+2;LDA J+1;STA R+3
DC82 54					320LDA @MM8&#FF;STA J;LDA @MM8/256;STA J+1
DC83 00		HEX	00		330LDA @#C0;STA V+#+E:LDA @#40;STA V+#+B:LDA @#4E;STA V+6
DC84 53 45 45	ERR5	ASC	SEEK		340LDA @#C3;STA V+5;CLI;RTS
DC87 4B					350:LL2:TXA;PHA;TYA;PHA
DC88 00		HEX	00		360DEC R+4;BNE LL1;LDA @20;STA R+4
DC89 44 52 49	ERR6	ASC	DRIVE NOT READY		370SED;LDA @0;SEC;ADC R+5;STA R+5;CMP @#60;BNE LL3
DC8C 56 45 20					380LDA @0;STA R+5;SEC;ADC R+6;STA R+6;CMP @#60;BNE LL3
DC8F 4E 4F 54					390LDA @0;STA R+6;SEC;ADC R+7;STA R+7;CMP @#24;BNE LL3
DC92 20 52 45					400LDA @0;STA R+7
DC95 41 44 59					410:LL3:LDA R+5;LDX @7;JSR MM4;JSR MM6
DC98 00		HEX	00		420LDA R+6;JSR MM4;JSR MM6;LDA R+7;JSR MM4
DC99 53 59 48	ERR7	ASC	SYNTAX		430:LL1:LDA @#5A;STA V+#+D
DC9C 54 41 58		HEX	00		440PLA;TAY;PLA;TAX;PLA
DCA0 42 41 44	ERR8	ASC	BAD TRACK NMR		450JMP (R)
DCA3 20 54 52					460:LL4:PHA;JSR MM5;PLA
DCA6 41 43 48					470LSRA;LSRA;LSRA;LSRA
DCA9 20 4E 4D					480:LL5:AND @#F:DRA @#30
DCAC 52					490:LL7:STA #B018,X;DEX;RTS
DCAD 00		HEX	00		500:LL6:LDA @CH":;JMP MM7
DCAE 54 52 41	ERR9	ASC	TRACK HEADER		510:LL8:LDY @0;LDX @0;JSR #F876
DCB1 43 4B 20					520:LL9:LDA #100,Y;CMP MM9,X;BED LL10;JMP (R+2)
DCB4 48 45 41					530:LL10:INY;INX;CPX @4:BNE LL9;LDX @2
DCB7 44 45 52					540:LL11:JSR #FB76:LDA #100,Y;ASLA;ASLA;ASLA;ASLA;STA R+8
DCBA 00		HEX	00		550:INY;LDA #100,Y;AND @#F:DRA R+8;STA R+5,X;INY;DEX;BPL LL11
DCBB 53 45 43	ERRA	ASC	SECTOR HEADER		560:RTS
DCB8 54 48 52					570:LL9;];\$P="TIME";P=P+LENP;I
DCC1 20 48 45					580I
DCC4 41 44 45					590R.
DCC7 52					600*****
DCC8 00		HEX	00		610 PROGRAM-DESCRIPTION
DCC9 42 41 44	ERRB	ASC	BAD SECTOR LENGTH		620 L.200: DEFINITION AND INITIALIZATION OF LABELS.
DCCC 20 53 45					630 L.290-320: INITIALIZATION OF VECTORS FOR INTERRUPT AND
DCCF 43 54 4F					640 COMMAND LINE INTERPRETER.
DCD2 52 20 4C					650 L.330-340: INITIALIZATION OF VIA (6522).
DCD5 45 48 47					660 L.350-450: INTERRUPT SERVICE ROUTINE.
DCD8 54 48					670 L.460-500: CLOCK-DISPLAY ROUTINE.
DCDA 00		HEX	00		680 L.510-570: COMMAND INTERPRETER.
DCDB 4E 4F 20	ERRC	ASC	NO FILE		690 USED REGISTERS: R/R+1: INTERRUPT VECTOR
DCDE 46 49 4C					700 R+2/R+3: COMMAND LINE INTERPRETER VECTOR
DCE1 45					710 R+4: 50 ms COUNTER
DCE2 00		HEX	00		720 R+5: SEC. COUNTER (BCD)
DCE3 52 2F 57	ERRD	ASC	R/W PAST FILE-END		730 R+6: MIN. COUNTER (BCD)
DCE6 20 50 41					740 R+7: HRS. COUNTER (BCD)
DCE9 53 54 20					750 R+8: UTILITY REGISTER
DCEC 46 49 4C					760 EXTRNAL USED ROUTINE: #F876: SKIP SPACES FROM INPUT BUFFER
DCEF 45 2D 45					770*****
DCF2 4E 44					
DCF4 00					
DCF5 44 49 53	ERRE	ASC	DISK FULL		
DCFB 4B 20 46					
DCF8 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 RIGHHAND 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 M0,OR LINK<ADDR> (=T)=
120 REM= TIMESETTING IS BY MEANS OF COMMAND: *TIME nn 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

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: INITIALIZATION 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: COMMAND 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*****

```

DE 6502 KENNER

* OCTOPATE *
* 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 OCTOPATE. Even before OCTOPATE 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.

OCTOPATE's heart is formed by the Text Editor, an extremely powerful line editor, which can make and change OCTOPATE files in a simple and user-minded way. Closely related to this are the commands of the Disk Operating System especially devised for OCTOPATE, which can transport files from and to disks. For reasons of upwards compatibility of OCTOPATE 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, transferring to new pages etc. etc. The text to be printed can in principle be infinitely 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 referred 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 OCTOPATE.

In OCTOPATE 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 OCTOPATE 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 OCTOPATE 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 OCTOPATE is worked on.

For good operating of OCTOPATE 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 OCTOPATE:

- OCTOPATE System Disk, including the Text Editor, Format Lister, Assembler and Configurator.
- OCTOPATE User Manual containing 90 pages explaining the software (for the time being only a Dutch version).
- OCTOPATE 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 OCTOPATE can be given. Those who are interested are welcome.

PAPERWARE & DISKETTE SERVICE OCTOPATE 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:

-OCTOPATE 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

-OCTOPATE User Manual.
Europe : Hfl.104,50 Outside Europe : Hfl.121,50

Members: Hfl. 54,50 Members: Hfl. 71,50

-OCTOPATE 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 CALCULATOR 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 100000000 tot en met 11111111 (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 inge-

drukt, wat vervelend kan zijn. Ik heb hierop het volgende programma-onderdeel bedacht:

```
10000 REM INPUTVERWERKING
10010 X$="" : Z=1
10020 DISK!"GO F71b":Y=PEEK(9059):Y$=CHR$(Y):PRINT Y$:IF Y=
13THEN10050
10030 IF Y$="" THEN Z=-1:GOTO10020
10040 X$=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).

DE 6502 KENNER

* PRINT YOUR GRAPHICS FOR ATARI 600 XL *

By: Henk Spekenijder, 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)
SI standard character	CHR\$(15)
POS print starting position	CHR\$(16)
ESC escape	CHR\$(27)
FS repeat graphics character	CHR\$(28)

```

30 P=0
32 FOR I=0 TO 36
34 READ C:P+C:POKE1570+I,C
36 NEXT I
38 IF P<>4477 THEN STOP
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 DATA 249,96

```

```

500 OPEN #2,8,0,"P:"
510 PUT #2,8:PUT #2,16:PUT #2,0:PUT #2,15
520 FOR V=0 TO 153 STEP 7
530 FOR H=0 TO 39
540 P=B+H*40*V
550 C=USR(1570,P)
560 FOR I=1560 TO 1567
570 PUT #2,PEEK(I)
580 NEXT I
590 NEXT H
600 PUT #2,13:PUT #2,16:PUT #2,0:PUT #2,15
610 NEXT V
620 PUT #2,15
630 CLOSE #2

```

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

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$

```

* 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

```

```

140 A=VAL(A$):IFA<20INT(A)<>ATHEN100
150 FOR I=JTOA
160 IF INT(A/I)=A/ITHENB=I:B$=B$+STR$(B):I=A
170 NEXT
180 A=A/B:I=B
190 IF A>=2 THEN 150
200 PRINT B$" BESTAAT UIT DE FAKTOREN:"
210 PRINT B$#
220 INPUT "NOG EEN KEER? (J/N)":G$
230 IF G$="N" THEN END
240 IF G$<>"J" THEN 220
250 RUN

```

DE6502 KENNER

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 disketterdrives 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 hobby scope 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 4911
;               1056 XS Amsterdam
;               tel. 020-125386
; date         110187 junior cassette write routines
;
; lib          caslib.mac
;
;
; 00C0 sal    equ    $00e0
; 00C1 sah    equ    sal+$01
; 00C2 eat    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
;
; E114 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 vbitc   equ    $e11d      ; interrupt flag register
; E280 juncas  equ    $e280      ; junior cassette port
;                                bit 3 = 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    sal    ; save $2000 - $4000
; 0206 08 iny    id    ; 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
;
```

DE6502 KENNER

```

;      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    #$00
0322 8C 1BE1   sty    vbacr        ; set timer 2 in oneshot mode
0325 84 EB   sty    acc
0327 84 E6   sty    chkl         ; reset checksum
0329 84 E7   sty    chkh
032B C8   iny
032C 8C 19E1   sty    vbtbch      ; activate timer 2 initially
032F A5 E0   lda    sal           ; initialize dumpt pointer
0331 85 E4   sta    pointl
0333 A5 E1   lda    sah
0335 85 E5   sta    pointh
0337 A2 FF   idx    #$ff
0339 86 E9   stx    syncnt      ; set sync counter
;
033B A9 16   syncs   lda    #$16
033D 20 B403   jsr    outch        ; syn character
0340 C6 E9   dec
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
035C D0 21   bne    hexdat      ; entire file transmitted?
035E A5 E4   lda    pointl
0360 C5 E2   cmp
0362 D0 1B   bne    eah
;
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
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
0388 D0 CE   bne    pointl      ; and compute checksum
038A E6 E5   inc
038C 4C 5803   jmp    datatr     ; setup for next data byte
;
038F A8   outbtc  tay
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
0399 85 E7   sta    chkh
039B 98   tya
039C A8   outbt  tay
039D 4A   lsra
039E 4A   lsra
;
; save accu
; checksum computation
; chk := chk + byte
; get accu again
; save accu temp
; get upper nibble

```

DE6502 KENNER

```

039F 4A           Isra
03A0 4A           Isra
03A1 20 AB03      jsr    nibout   ; output upper nibble as ascii char.
03A4 98           tya
03A5 29 0F         and   #$0f    ; get byte again
03A7 20 AB03      jsr    nibout   ; get lower nibble
03AA 60           rts    ; output lower nibble as ascii char.

; 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   Isra
03B9 48           pha
03BA 90 0C         bcc    zero
03BC 20 D703      jsr    high   ; start at 3600 Hz
03BF 20 FA03      jsr    low
03C2 20 FA03      jsr    low
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
03D1 68           zro
03D2 C6 EA         pla
03D4 D0 E2         dec    bits
03D6 60           bne    one   ; all bits shifted out?

; 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.
03E0 AD 18E1      lda
03E3 A5 EB         lda
03E5 49 10         eor    #00010000 ; no, interval not completed
03E7 85 EB         sta   acc
03E9 8D 80E2      sta   juncas ; clear interrupt flag
03EC A5 F1         lda   higher ; 1 = bit to be toggled
03EE 8D 18E1      sta   vbtbcl ; 3600 Hz half cycle time
03F1 A9 00         lda   #$00 ; timer 2 low
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.
0403 AD 18E1      lda
0406 A5 EB         lda
0408 49 10         eor    #00010000 ; no, interval not completed
040A 85 EB         sta   acc
040C 8D 80E2      sta   juncas ; clear interrupt flag
040F A5 F2         lda   lower ; 2400 Hz half cycle time
0411 8D 18E1      sta   vbtbcl ; timer 2 low
0414 A9 00         lda   #$00 ; start timing of timer 2
0416 8D 19E1      sta   vbtbch
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   vbaqr
0423 A9 01         lda   #$01
0425 8D 19E1      sta   vbtbch ; (00 for eleven minutes)
0428 A9 A0         lda   #$a0
042A 85 EC         sta   count
042C 85 ED         sta   count+1
042E 18           clc
042F A9 01         lda   #$01
0431 65 EC         adc   count
0433 85 EC         sta   count
0435 A9 00         lda   #$00

```

DE6502 KENNER

```

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      idy    #$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      idy    #$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!

DE6502 KENNER

```

; file          junread.mac
; purpose       junior cassette interface for dos65 computer
; author        E.R.Elderenbosch
; date         220137 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 vbtocl  equ    $e118 ; timer 2 latch low, counter low
E119 vbtch   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    #$20 ; in memory
0206 85 E1 sta    sah    ; set tape id = ff
0208 A9 FF sta    id     ; start reading tape
020A 85 E8 rts    rdtape ; end of dummy program
020F 60
;
0210 78 rdtape sal    noted ; turn leds off
0211 20 5003 jsr    lda    #ff00
0214 A9 00 lda    #ff00 ; turn leds off
0216 80 1BE1 sta    vbacr ; set timer 2 oneshot
0219 85 E6 sta    chkl
021B 85 E7 sta    chkh
021D A9 FF sync  lda    #$ff ; reset for syn character
021F 85 EF sta    char
0221 20 BC02 synca jsr    rdbit ; read a bit from tape
0224 66 EF nor   char ; right shift
0226 A5 EF ldy   char ; got current character
0228 20 5603 jsr    searled ; display searching (red led)
022B C9 15 cmp   #15 ; syn character?
022D D0 F2 bne   synca ; if not, resync
022F A0 0A ldy   #0a ; try it for 10 syncs at least
0231 94 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)

```

DE6502 KENNER

```

0247 C9 2A      cmp    #'*'
0249 F0 06      beq    stara
024B C9 16      cmp    #$16          ; still sync character?
024D F0 F2      beq    star
024F D0 BF      bne    rdtape
0251 20 6203    stara   jsr    dataled
0254 20 F002    jsr    rdbyt
0257 C5 E8      cmp    id
0259 D0 40      bne    chkid
025B 20 F002    rdsd   jsr    rdbyt
025E 20 4203    jsr    chksum
0261 85 E4      sta    pointl
0263 20 F002    jsr    rdbyt
0266 20 4203    jsr    chksum
0269 85 E5      sta    pointh
026B 20 F002    filmem jsr    rdbyt
026E 30 A0      bmi    rdtape
0270 F0 13      beq    check
0272 20 4203    jsr    chksum
0275 A0 00      ldy    #$00
0277 91 E4      sta    [pointl],y
0279 E6 E4      inc    pointl
027B D0 02      bne    fma
027D E6 E5      inc    pointh
027F 20 6203    fma    jsr    dataled
0282 4C 6B02    jmp    filmem
0285 20 F002    check   jsr    rdbyt
0288 C5 E6      cmp    chkl
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
0296 58          cli    rts
0297 60          rts
0298 4C 1002    synvec jmp    rdtape
029B A5 E8      chkid  lda    id
029D C9 00      cmp    #$00          ; id = 00?
029F F0 BA      beq    rdsd
02A1 C9 FF      cmp    #$ff          ; id = ff?
02A3 D0 F3      bne    synvec
02A5 20 F002    jsr    rdbyt
02A8 20 4203    jsr    chksum
02AB 20 F002    jsr    rdbyt
02AE 20 4203    jsr    chksum
02B1 A5 E0      lda    sal
02B3 85 E4      sta    pointl
02B5 A5 E1      lda    sah
02B7 85 E5      sta    pointh
02B9 4C 6B02    jmp    filmem
; rdbit 1
02BC A9 01      rdbit  lda    #%00000001
02BE 2C 80E2    1      bit    juncas
02C1 D0 FB      bne    1.
02C3 AD 19E1    lda    vbtbch
02C6 85 EB      sta    acc
02C8 A0 FF      ldy    #$ff
02CA 8C 18E1    sty    vbtbcl
02CD 8C 19E1    sty    vbtbch
02D0 A0 14      ldy    #$14
02D2 88          rdba   dey    rdba
02D3 D0 FD      bne    #%00000001
02D5 A9 01      rdbb   lda    juncas
02D7 2C 80E2    2      bit    2.
02DA F0 FB      beq    sec
02DC 38          sec
02DD A5 EB      lda    acc
02DF ED 19E1    sbc    vbtbch
02E2 A0 FF      ldy    #$ff
02E4 8C 18E1    sty    vbtbcl
02E7 8C 19E1    sty    vbtbch
02EA A0 07      ldy    #$07
02EC 88          rdbc   dey    ; delay for jitter
02ED D0 FD      bne    rdbc
02EF 60          rts
; rdbyt
02F0 20 3103    rdbyt jsr    rdch
02F3 C9 2F      cmp    #'/'
02F5 D0 01      bne    rbb
02F7 60          rba    rts
02F8 20 1403    rbb    jsr    aschex
02FB 30 FA      bmi    rba
02FD 0A          asla
; read any ascii character from tape
; end of data character?
; error exit
; ascii hex conversion
; not valid character
; shift nibble to left

```

DE6502 KENNER

```

02FE 0A      asla
02FF 0A      asia
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
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
0338 CA      dex
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  ; juncas
0352 8C 80E2  sty
0355 60      rts

; 0356 A0 20  searled ldy   #$20  ; juncas
0358 8C 80E2  sty
035B 60      rts

035C A0 40  syncled ldy   #$40  ; juncas
035E 8C 80E2  sty
0361 60      rts

; 0362 A0 80  dateded ldy   #$80  ; juncas
0364 8C 80E2  sty
0367 60      rts

; 0200      end   start label-table

```

DE 6502 KENNER

```
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)
9 , (.)
10 , C/L        2 - @ CONSTANT CONST.ADR
11 , BASE       2 - @ CONSTANT USERV.ADR
12 , USE        2 - @ CONSTANT VAR.ADR
13 , (:CODE)    2 - CONSTANT PS CODE.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 X>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 :      -->
```

```
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 O=          ( 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 OBRANCH.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
 3 DUP WORD.DSP      ( ALL SPECIAL WORD TYPES CHECKED. )
 4                      ( IF WORD DID NOT MATCH ANY CASES )
 5 ENCASE CR          ( JUST PRINT ITS NAME )
 6 2 WORD.PTR +!      ( DONE DECODING WORD TYPE )
 7 QUIT.FLAG @        ( UPDATE PSEUDOCODE POINTER )
 8 ?TERMINAL OR       ( CHECK IF FINISHED FLAG SET OR IF )
 9 UNTIL              ( INTERRUPTION FROM THE TERMINAL )
10 ENDIF ENDIF CR :   ( OTHERWISE DISPLAY ANOTHER WORD )
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
15
GOOD LUCK !
```


DOS 5.02 KENNER

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

```

1      ;-----+
2      ; RELOCATE start,end,offset,start_adjust,end_adjust
3      ;-----+
4      ; New DOS5 command to relocate the absolute addresses
5      ;-----+
6      ; in machine code programs
7
8
9      ;-----+
10     org    $400
11     ; constants
12
13     cr     equ    $d
14     formf  equ    $c
15
16     ; Zero page addresses
17
18     000D   cr     equ    $d
19     000C   formf  equ    $c
20
21     00F0   flag   equ    $f0      ;flag for newline
22     00F2   tema   equ    $f2      ;temp. store for start-address
23     00F4   temb   equ    $f4      ;temp. store for end-address
24     00F6   temc   equ    $f6      ;temp. store for offset
25     00E8   temd   equ    $e8      ;temp. store for start-adjust
26     00EA   teme   equ    $ea      ;temp. store for end-adjust
27
28     ; DOS65 routines
29     C02F   crlf   equ    $c02f   ;print CRLF
30     C032   space  equ    $c032   ;print space
31     C03B   print   equ    $c03b   ;print string till 0
32     C06B   spar   equ    $c06b   ;scan parameters
33     C023   output  equ    $c023   ;print character
34     C038   hexout equ    $c038   ;print A in hex.
35     DOB7   ermes  equ    $d0b7   ;print error message
36
37     relocat ldx    #0
38     jsr    spar   ;get parameters
39     fcc    tema,temb
40     fcc    temc,temd,teme,0
41     bcs    l.f    ;error in parameters
42     reloc   cpx    #$f8
43     beq    relgl
44     jmp    erda
45     l     jmp    ermes
46     relb   ldx    #tema
47     jsr    opclen
48     cpy    #3
49     bne    reli
50     jsr    xinc
51     ldy    #1
52     lda    teme
53     cmp    [0,x]
54     lda    [0,x]
55     tax
56     lda    teme+
57     sbc    [tema],y
58     bcc    relh
59     cpx    temd
60     lda    [tema],y
61     sbc    temd+
62     bcc    relh
63     clc
64     txa
65     adc    temc
66     pha
67     lda    [tema],y
68     adc    temc+
69     sta    [tema],y
70     dey
71     pla
72     sta    [tema],y
73     ldx    #tema
74     jsr    xdec
75     jsr    outins

```

DE6502 KENNER

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

```

0451 B0 1A      76     bcs    relia      ;if C then at end
0453 20 E104    77     jsr    twospa    ;print two spaces
0456 C6 F0      78     dec    flag       ;and decr. flag
0458 D0 BD      79     bne    relb      ;if not zero, continue
045A A9 04      80     relgl   lda      #4        ;else reset flag
045C 85 F0      81     sta    flag      ;(four opcode/line)
045E 20 2FC0    82     jsr    crlf      ;and do <newline>
0461 90 B4      83     bcc    relb      ;then continue
0463 A0 02      84     relh   ldy      #2        ;point to next opcode
0465 20 CB04    85     reli   jsr      incbrk
0468 88          86     dey
0469 D0 FA      87     bne    reli
046B 90 AA      88     bcc    relb      ;if C=0 continue
046D 4C 2FC0    89     relia  jmp      crlf      ;else return to DOS65
90 ;
91 ;
92 ;
93 ;
94 ; return opcode length in Y
95 ;
0470 A0 01      96     opclen ldy      #1
0472 A1 00      97     lda      [0,x]
0474 F0 1B      98     beq    opcj
0476 C9 40      99     cmp    #$40      ;RTI?
0478 F0 17      100    beq    opcj
047A C9 60      101    cmp    #$60      ;RTS?
047C F0 13      102    beq    opcj
047E A0 03      103    ldy    #3
0480 C9 20      104    cmp    #$20      ;JSR?
0482 F0 0D      105    beq    opcj
0484 29 1F      106    and    #$1f
0486 C9 19      107    cmp    #$19
0488 F0 07      108    beq    opcj
048A 29 OF      109    and    #$f
048C A8          110    tay
048D B9 9204    111    lda      opctb,y    ;get length from table
0490 A8          112    tay
0491 60          113    opcj   rts
114 ;
115 opctb   fcc   2,2,2,1,2,2,2,1,1,2,1,1,3,3,3,1
116 ;
117 ; Print adjusted opcode
118 ;
119 outins  jsr    opclen      ;get opcode length
120         jsr    outxad      ;print current address
121         jsr    print
122         fcc    : ,0
123 oinsa   jsr    outmcb      ;print byte
124         jsr    incbrk      ;point to next
125         dey
126         bne    oinsa      ;until end of opcode
127         rts
128 ;
129 ; Incr. 16 bits address where X is pointing to
130 ;
131 xinc    inc    0,x
132         bne    xincb
133         inc    1,x
134 xincb  rts
135 ;
136 ; Decr. 16 bits address where X is pointing to
137 ;
138 xdec    pha
139         lda    0,x
140         bne    xdecb
141         dec    1,x
142 xdecb  dec    0,x
143         pla
144         rts
145 ;
146 ; Incr. and check if at end
147 ;
148 incbrk  lda    tema      ;incr. start-address
149         cmp    temb      ;and check if at end

```

DE6502 KENNER

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

```

04CF A5 F3      150    lda     tema+1
04D1 E5 F5      151    sbc     temb+1
04D3 E6 F2      152    inc     tema
04D5 D0 02      153    bne     incb
04D7 E6 F3      154    inc     tema+1
04D9 60          155    incb    rts           ;C=1, if at end
156    ;
157    ; Print byte
158    ;
04DA A1 00      159    outmeb   lda     [0,x]       ;get byte
04DC 20 38C0      160    outmea   jsr     hexout      ;print in hex
04DF 90 03      161    bcc     outspa      ;branch always
162    ;
04E1 20 32C0      163    twospa   jsr     space        ;print space
04E4 4C 32C0      164    outspa   jmp     space        ;and another
165    ;
166    ; Print address
167    ;
04E7 B5 01      168    outxad   lda     1,x       ;get address hi
04E9 20 38C0      169    jsr     hexout      ;print in hex
04EC B5 00      170    lda     0,x       ;get address lo
04EE 4C 38C0      171    jmp     hexout      ;print it
172    ;
173    ; Help info if error in parameters
174    ;
175    erda   jsr     print
176    fcc    formf, "RELOCATE HELP", cr, cr
177    fcc    Command syntax: ,cr
178    fcc    'RELOCATE aaaa,bbbb,cccc,dddd,eeee', cr, cr
179    fcc    aaaa : Startaddress of memory area to be relocated', cr
180    fcc    bbbb : Endaddress of memory area', cr
181    fcc    cccc : Offset', cr
182    fcc    dddd : Start of address area to be adjusted', cr
183    fcc    eeee : End of address area to be adjusted', cr, cr
184    fcc    'Example:', cr
185    fcc    'Program to be relocated is resident at $2000 to $2200, suppose you want to', cr
186    fcc    'use this program at $3000 to $3200, so only absolute addresses in the area', cr
187    fcc    '$2000 to $2200 have to be adjusted. The offset is $1000 so enter:', cr, cr
188    fcc    'RELOCATE 2000,2200,1000,2000,2200', cr, cr
189    fcc    'The adjusted addresses are displayed for control purposes (see note)', cr
190    fcc    'Note: Beware for data or tables in the relocated address area, the program', cr
191    fcc    'cannot see the difference between opcodes in a program and characters', cr
192    fcc    'like $20 (ASCII space) and $4C (ASCII L), so use RELOCATE with care.', cr
193    rts
194    end   relocat
195    global labels

cr      0000 crlf  002F erda   04F1 exres  D0B7 flag   00F0
formf  000C hexout 0038 indo   04D9 inbck  04CB oinsa  04AF
opcj   0491 opclen 0470 opctb  0492 outins 04A2 outnea 04DC
outmeb 04DA output 0023 outspa 04F4 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   0400 xdecdb 04C7 xinc   04B9 xindb  04BF

```

Errors detected: 0

DE 6502 KENNER

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 AMPERAND 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 NAAR VORIGE REGEL
600E:C6 25 35 DEC CV
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:DO 31 40 BNE DEC
601A: 41 ;
601A:BD 00 02 42 ;HEX-DEC CONVERSIE X WIJST ALTIJD VOORBIJ LAATSTE DIGIT
601A:43 HEX1 LDA BUF,X
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:DO F5 47 BNE HEX1 WIJS NAAR 1E DIGIT IN BUFFER ($0202)
6025:A0 02 48 LDY #2
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 A-REGISTER=0
603A:8A 58 TXA TREK MSB ER VANAF
603B:E5 3F 59 SBC A2L+1 $8000 INGETIKT?
603D:30 1C 60 BMI KLAAR NEE
603F:85 3F 61 STA A2L+1 GEEF 3 SPATIES,
6041:20 48 F9 62 JSR PRBLNK PRINT ALVAST "-"
6044:A9 AD 63 LDA #$AD
6046:20 ED FD 64 JSR COUT
6049:80 DF 65 BCS HEX2 EN PRINT REST VAN NEGATIEVE NOTATIE
604B: 66 ;
604B:67 ;DEC-HEX CONVERSIE
604B:A9 A4 68 DEC LDA #$A4 PRINT "$"
604D:20 ED FD 69 JSR COUT CONVERTEER INVOER NAAR FAC
6050:20 67 DD 70 JSR FRMNUM MSB IN A EN LINNUM+
6053:20 52 E7 71 JSR GETADR LSB
6056:A6 50 72 LDX LINNUM UITVOER RESULTAAT
6058:20 41 F9 73 JSR PRNTAX TERUG NAAR APPLESOFT
605B:20 DD 03 74 KLAAR JSR $3D0

```

*** SUCCESSFUL ASSEMBLY: NO ERRORS

DE6502 KENNER

```
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 ADDRESS 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 ADDRESS WITH LEADING ZERO'S )
2 : .POINTER POINTER @
3     DUP 10 ( OVER FFFF ) AND IF 3 .0 ENDF
4     DUP 100 ( OVER F ) AND IF 2 .0 ENDF
5     DUP 1000 ( OVER FF ) AND IF 1 .0 ENDF
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    ENDF
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    ENDF
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     ENDF
9     UNTIL ( TERMINATE IF FLAG TRUE )
10    CR :
11
12 ( PRINT HEXDUMP ON PRINTER )
13 : PHEXDUMP PRAAN HEXDUMP PRUIT :
14
15 :S
```

DE 6502 KENNER

```
SCR # 53
0 ( GLOSSARY HEX/ASCII-DUMP           GERT KLEIN      )
1 U.          ( n -- )
2 Print an unsisioned 16 bit number.
3 FETCHBYTE   ( -- bvte )
4 Get bvte from adress in POINTER
5 .O          ( n -- )
6 Print n zero's.
7 .ROW         ( -- )
8 Print index row on top of dumo.
9 ?ASCII       ( ch -- ch f )
10 Set true flag if value of ch is between $20 and
11 $7E. else false flag.
12 .POINTER    ( -- )
13 Print adress in variable POINTER with leading
14 zero's.
15
```

```
SCR # 54
0 ( GLOSSARY HEX/ASCII-DUMP           GERT KLEIN      )
1 .HEXROW      ( -- )
2 Print 16 hexbytes in the range of adress in variable
3 POINTER to POINTER + 16.
4 .ASCROW      ( -- f )
5 Print 16 ASCII characters in the range of address in
6 variable POINTER to POINTER + 16. Non printing
7 characters are represented by a dot. Flag indicates
8 if POINTER > ENDAD.
9 .HEXDUMP     ( adr1 adr2 -- )
10 Print hex/ascii dumo between adr1 and adr2. Terminate
11 on terminal break. Adr1 and adr2 may be entered
12 both in hex or in decimal. The dumo is outputted in
13 hex. After termination HEX is the present number base
14 .PHEXDUMP    ( adr1 adr2 -- )
15 Print dumo on printer.
```

OK

