De 6502 Kenner 28 - October 1983

Elektuur 16K Dyn RAM kaart adapatation for reliability

De 6502 Kenner 39 - Augustus 1985

Another DOS for the Junior, DOS65. Coen Kleipool

De 6502 Kenner 40 – Oktober 1985

DOS65 new for your system. overview. Coen Kleipool

De 6502 Kenner 41 – December 1985

DOS65 DOS Editor, Coen Kleipool

De 6502 Kenner 42

FORTH screen editor F. Jonkman

De 6502 Kenner 43 – April 1985

RS232 ACIA configure program. A.S. Hankel

De 6502 Kenner 44 _juni 1986

DOS65 Version 2, Coen Kleipool

De 6502 Kenner 45 – Augustus 1986

6502 tracer for the DOS65 computer, Rene Hettfleisch

De 6502 Kenner 46

A Macro loader and saver for ED, Bram de Bruine

DOS65 AIM-Basic, Bram de Bruine

VDU card in color, Phons Bloemen

Basicode 2

Search Birthdays, Jean de Noyette

FORTH 6502 assembler, updated for 65C02 Gert Klein

Screen Editor FORTH G van Opbroek

De 6502 Kenner 47

Basicode for DOS65

BELL for DOS, Lindstrom and Rasmussen

Apple compatible cassette interface. P. de Visser

De 6502 Kenner 48

Basicode for DOS65, P. Laser

De 6502 Kenner 49

ACIA problem, split speed, Bram de Bruine

Centronics input for DOS65. E. R. Elderenbosch

De 6502 Kenner 50

Problems DOS65, bug in COPY, Andrew Gregory

Small C videocontrol routines for DOS65, A. Megens

SETPRINT printer configuration, H.A.J. Quast

Disc controller card error above 1 MHz, N. De vries

COMAL V2.1 for DOS65, A. Megens

Locator DOS-Junior, C. Boltjes

RAM Test program, M. Lachaert

Amazing Maze V.1 Comal, A. Megens

timedate.mac alternative for real time clock, P. Roesssingh

GETPAR, get decimal parameter from inputbuffer and convert to hex.

Ingangsprotectie Logic Analyzer, H.A.J. Quast

ELEKTUUR 16-K DYN.RAM KAART.

Door: Dirk Pickee

Deze kaart is beschreven in Elektuur nr.222, blz.50 e.v. Een wijziging hierop werd beschreven in Elektuur nr.230, blz.79 en 80.
Het nadeel van deze kaart, voor mij tenminste, was dat deze inclusief de genoemde wijzigingen niet betrouwbaar werkte.
Van vrienden hoorde ik van de volgende wijzigingen, welke er bij mij voor zorgden, dat er nu drie dyn.Ram kaarten naar tevredenheid werken:

1) Maak de printwijziging zoals beschreven in Elektuur nr.230, ongedaan (herstel het doorgesneden printspoor). De massa-doorverbindingen mogen blijven zitten, dat kan zeker geen kwaad.
2) Verwijder draadbrug 1 - 1'
3) Verbindt 1' met pen 29c van de connector, dit is R/$\overline{W}$.
4) C 5 was 120, wordt 470 pF.
5) C 4 was 470, wordt 820 pF.
6) Verwijder draadbrug A - B.
7) Verbindt punt B met pen 27a van de connector, dit is Φ_2 .

Na bovenstaande aanpassingen werkten de drie kaarten bij mij goed. Ik ben benieuwd of deze aanpassingen ook bij anderen tot het gewenste resultaat voeren.
Met dank aan de heren Guus Assman en William Derks voor hun informatie.

(Red.: De dyn.Ram kaart voorzien van snelle dyn.Rams is wellicht een andere oplossing.)

R_E_V_I_E_W

ANOTHER DOS FOR THE JUNIOR

by Coen Kleipool, Val de Périer, F-83310 Cogolin, France. t.(94) 54.43.82

For those of you who think that things come cheaper by the dozen, here is some good news. Another Junior Dos has come out, which makes this Dos65 the fourth of its kind after Elektor's Ohio Dos, Proton Dos and Coen van Nieuwenhoven Dos. Why another Dos ? Not so easy to answer but apparently our members were still not satisfied and are willing to improve.

I can wholeheartedly endorse this new Dos65; speed, command structure and flexibility are impressive and are of another class compared to the previous three Dos.

Brouwers Baby. It has been written by our member Ad Brouwer who is a graduate Delft computing student and as such a man of experience on big machines. It is not surprising that this Dos has a lot of similarities with Unix. Unix is basically a multi-user Dos for big systems and Brouwer has boiled this down to a single-user Dos with a simple and easy-to-learn command system.
Unix has the reputation to be a powerful tool for the professional and a headache to the occasional user, but this simplified Unix is easy to work with.

Unix, being a multi-user system, has been conceived to work with one or more intelligent terminals and the basic Dos65 also needs a terminal. Later versions were developed to be used with a VDU-card and a screen memory in the computor itself. This solution simplifies matters and reduces costs to some extent, but has the disadvantage that the screen memory occupies valuable computor memory and furthermore microprocessor time is needed to process the screen after each operation. If your computor has only 64 Kbytes of memery, this is a serious consideration and I would myself prefer the terminal solution at any time.

Hardware. To implement Dos65 you will need to construct an Eurocard which carries the 1793 Floppy Controller and a 6821 I/O port. Our member Cees Bootsma has a PC-board available but it is also feasable to produce this in wirewrap or roadrunner. I had a lot of trouble building the PC-board, mainly due to a lack in documentation. The +5 volt and the ground grid need to be improved and some parts do not fit very well in the proper places, but after having made these modifications it works fine.

The 1793 FDC is a very powerful chip. It provides both single and double density (IBM system 34 MFM) and has on-chip track and sector registers which enables automatic sector search. The 372 FDC, used in

Proton Dos, does not have these facilities and is incapable of writing in true double-density.

Double density packs 320 Kbytes on a double-sided 40-tracks diskette and as data are more condensed, you will also get a higher transfer rate ! Dos65 still surprises me with its speed; loading Moser or Micro-ade (8 Kbytes) takes less than 3 seconds and most of this time is spend searching for the proper track. My son uses Proton Dos and together we concluded that Dos65 is at least 4 times as fast.

Software. Dos65 software consists of 4 different parts:

- the monitor
- the kernel
- the utilities
- the editor.

The Monitor. This occupies 4 Kbytes in page F and comprises the basic I/O routines which are very conveniently accessed through jump tables. Monitor commands are special in the sense that they are not needed during normal Dos operation; the enable the user to insert or delete data in memory, call hex- and asccidumps, blockmove and relocate, read and write cassettes, show microprocessor internal registers, search for certain data in memory, etc. All this is very useful during programming and debugging.

The kernel. The kernel is the very cornerstone of the Dos, it is 16 Kbytes long and is read from disk into ram right after startup. Here you will find the vital routines and commands which must be available all the time. There are the typical Unix redirect commands which will allow you to manage in- and output of a command in other directions that the usual keyboard in- and screen out- routines. For instance, you can call input from a disk and redirect the output to the printer or to another disk file, or a modem for that matter.

You can also input commands from a diskfile instead of typing in long and frequent commands on the keyboard such as startup and initialising sequences.

During all these operations there is no need to create, open of close files; Dos65 takes care of all this automatically. Nor is there a "compress" command to fill out empty spaces on your disks, the file handler uses an ISAM (Index Sequential Access Method) to put the file in the first available open space and normally no rearranging is necessary.

At this time Dos65 is incapable of writing 80-tracks disks; in the area where the Track Sector Lists of the 4 open files are located ($1000 - $17FF) is insufficient memory space to accommodate 80 tracks, but Ad Brouwer told me he will soon solve this problem.

Utilities. The utilities comprises a number of commands that remain on disk and are only read into memory when needed. Most of them reside in the same place, that is $ 800 for the smaller and $ 4000 for the larger programs. Their infrequent use does not merit a permanent stay in active memory. In a Dos computor diskfiles are considered a direct extension of memory and everything is done to occupy the least possible space of active ram which should be available to the user for work in progress.

In any case reading-in a utility only takes a fraction of a second and the software remains in memory until written over by the next utility. The only inconvenience is that it makes the use of two drives almost mandatory. Otherwise a user-drive is not immediately available and you will have to change disks frequently.

But the system can be used with a single diskdrive as a starter. You can read in a utility from the systemdisk, then change disks and execute the utility which now resides in memory with a LC (last command) instruction.

Typical utilities are: delete, list, print or format a file, rename a file, change a filemode, etc. The filemode, which is written on the diskette with the file allows you to protect a file against deleting or writing and to designate it as an ascci-, binary- or command file. You will find this feature very useful once you start using it.

The Brouwer Editor. I will save this item for a next article. Ad Brouwers Editor is a full-screen editor and my 12-year old Hazeltine is not quite ready for its impact. It is a tired old terminal and I will need to inject some fresh blood before he can cope with the full capabilities of this editor.

My comments: A truly excellent Dos which merits to be implemented by all KIM-members. It transforms your Junior into a real professional machine. We will need better hardware and a good manual (especially for Dos beginners) but these are minor problems.

A more serious problem will be to come to a unified Kimclub memory mapping in order to render our club-software compatible. More and more people are scrapping the standard and interface cards, thereby leaving the old junior compatibility. We should now agree on a standard memory-map, starting from page FF downwards.

This should provide us with an uninterupted ram coverage from $00 till the start-address of Dos. I have some ideas on this matter and I will come back to this soon.

°°°°°°°°°°°°°°°°°°°°

****
* *
* NIEUW VOOR UW 6502 SYSTEEM: *
* *
* D O S 6 5 *
* --------- *
****

Wat is DOS65:

DOS65 is een operating-system, in principe geschikt voor elk 6502-systeem. Het is ontwikkeld door A.B.M. Brouwers, ten behoeve van implementatie op de de zogenaamde uitgebreide Junior. Door E.J.M. Visschedijk en A.S. Hankel werd het aangepast voor implementatie op een konfiguratie van Elektuur's CPU/VDU kaarten.
DOS65 bestaat uit een 'Kernel', 8Kbyte groot, en een groot aantal hulproutine's (utility's), waaronder een full screen editor/word processor. DOS65 ondersteunt de programmeertalen BASIC, FORTH, alsmede Micro-Ade en de Moser assembler, en in de toekomst wellicht nog andere, wo. PASCAL en 'C'.
DOS65 is geschikt voor alle type's disk-drives met 40 tracks per kant, dus zowel single- als double-sided, single- als double density. De maximale schijfkapaciteit (geformatteerd) is 304 Kbyte.
DOS65 wordt, afhankelijk van de implementatie, ondersteund door de monitor van A.B.M. Brouwer, of de MON65 monitor van HaViSOFT.

Wat er aan hardware nodig is:

Wordt DOS65 ge-implementeerd voor de CPU/VDU konfiguratie, dan dient de volgende hardware aanwezig te zijn:
-Elektuur's CPU-kaart inkl. 2 Kbyte RAM en 8 Kbyte ROM (Deze ROM bevat MON65).
-Elektuur's VDU-kaart met speciale karakter-ROM.
-DOS65 floppy-controller-kaart.
-48 Kbyte RAM (van $0000 tot $C000).

De print voor de floppy-controller, de monitor-ROM, de speciale karakter-ROM, en DOS65 zijn leverbaar via de software-service van onze club.
Alle beschikbare handleidingen zijn geschreven in het nederlands.

Wat het systeem kan:

Verkort kommando overzicht DOS65.

APPEND	plaats een file achter een andere
CAT	geef catalog van een schijf
COPY	kopieer file of file's
CREATE	kreeer een ascii-file
DELETE	verwijder file of file's
DIR	geef de directory van een schijf
DUMP	laat file als hexdump zien
EDITOR	start de full screen editor
FORMAT	formatteer een schijf
LIST	laat een ascii-file zien
LOAD	laad een file in het geheugen
MEMFILL	vul een stuk geheugen
MEMMOVE	verplaats een stuk geheugen
PLIST	laat ascii-file zien op pagina formaat
PRINT	print een file op de printer
RENAME	geef een file een andere naam
RUN	executeer een file
SAVE	schrijf geheugen naar schijf
SDIR	laat een gesorteerde directory zien
SETDRIVES	zet motor-on en headload tijd
SETMODE	zet de mode van een file
TIME	definieer tijd en datum

Verkort mogelijkheden-overzicht full screen editor.
-Tekst invoeren, wijzigen en verwijderen.
-Door de tekst stappen:
 per karakter, woord, tab positie, regel of sektie, zowel voor- als achteruit.
-Zoeken naar een string, zowel voor- als achteruit. De gevonden string kan eventueel worden vervangen door een andere (zoek & vervang).
-Plaats een gedeelte van de tekst in een tijdelijke buffer.
-Voeg de buffer toe aan, of tussen, de tekst.
-Verander hoofdletters in kleine letters vice versa.
-Vul de tekst uit met spaties, zodat een rechte rechterkantlijn ontstaat.
-Verwijder overtollige spaties.
-Voer een DOS65 kommando uit.
-Lees een file in.
-Schrijf de tekst naar een file.
-Definieer/executeer een macro.
-Geef een kort mogelijkheden-overzicht (help-funktie).

Kommando overzicht MON65 (HaViSOFT)

@	open een adres, laat de inhoud zien
A	laat datum zien / wijzig datum
B	zet breakpoint
C	controleer een geheugengebied
D	disassembleer een geheugengebied
E	executeer vanaf een adres
F	vul een geheugengebied
H	hexdump
L	laat de CPU registers zien
M	verplaats een geheugengebied
P	maak het scherm schoon
Q	verlaat MON65 middels een RTS instruktie
S	bepaal de checksum van een geheugengebied
T	laat tijd zien / wijzig tijd
V	vergelijk twee geheugengebieden
W	zoek naar een data-patroon
X	door systeemgebruiker te definieren
Y	door systeemgebruiker te definieren
+	tel twee hex getallen bij elkaar op
-	trek twee hex getallen van elkaar af
$	reken om van decimaal naar hex
#	reken om van hex naar decimaal
<	herhaal het kommando / de kommando's

Verder beschikt MON65 over een aantal I/O-routines, waaronder:
-Keyboard (parallel)
-RS 232
-Centronics
-Viacom
Viacom is een routine, waarmee twee computers, (die beiden over Viacom beschikken), razendsnel data uit kunnen wisselen.

Wat DOS65 kost:

DOS65 en alles wat erbij hoort, is door clubleden ontwikkeld, zonder winstoogmerk. DOS65 is dan ook tegen kostprijs verkrijgbaar. Ter indikatie:

-floppy-controller print: ca. f 50,-
(dubbelzijdig, doorgemetaliseerd, met tekstopdruk)
-ROM's, schijven: bestaande verkoopprijzen
-handleidingen: ca 15 ct. per pagina

Beschikbare handleidingen:
-DOS65 handleiding
-Editor handleiding
-Technische handleiding (hardware)
-Bouwbeschrijving floppy-controller
-MON65 handleiding
-Sources MON65
-Binnenkort: Sources DOS65

Prijzen van het MON65/DOS65 systeem:
-Handleiding MON65
Handleiding DOS65
Handleiding editor
Hardware beschrijving
Samen ongeveer 110 pagina's Fl. 50,=
-2764 Monitor Eprom Fl. 35,=
-2732 Karaktergenerator Eprom Fl. 25,=
-Diskette met o.a. Micro-ADE, FORTH, Editor, en diverse utilities Fl. 15,=
-Floppy Disk Controller kaart Fl. 50,=
-Source MON65 Fl. 25,=
-Source DOS65 Fl. 25,=
-Source utilities Fl. 25,=
Deze prijzen zijn exclusief verzendkosten en gelden alleen voor leden van de KIM Gebruikers Club Ned. Ieder lid komt slechts eenmaal voor deze prijzen in aanmerking.
Bestellingen moet niet zoals gebruikelijk bij de redaktie worden gedaan, maar als volgt:

E.J.M. Visschedijk (Havisoft)
Drakesteyn 299
7608 TR ALMELO
Postrekening 225746
of per eurocheque
(Voor buitenland geldt: indien niet per eurocheque wordt betaald, dan moet het bedrag met Fl. 7.50 extra transfers worden verhoogd)

Vermeldt duidelijk wat gewenst wordt.

==========

MERGEN VAN APPLESOFT PROGRAMMA'S

Heeft U niet de beschikking over een routine om Applesoft Basic programma's te mergen (= aan elkaar plakken), dan kan dat blijkbaar ook met de volgende truc:
1. Laadt het eerste programma
2. CALL-151
3. Maak pointer op $67-$68 gelijk aan pointer op $69-$6A minus 3
4. Terug naar Basic en laadt tweede programma
5. CALL-151 en geef pointer op $67-$68 de oude waarde terug
6. Terug naar Basic door CTRL-C of door E003G

Aldus Pieter de Visser uit Veldhoven.

==========

1 BIT/DRUPPEL geeft op regenachtige dagen minstens een subroutine.

Fr. Verberkt

==========

HCC-Computerdagen

De in de vorige editie afgedrukte reduktiebon blijkt een vergissing te zijn welke niet meer te herstellen bleek. Door het tijdstip waarop de edities moeten worden samengesteld is herstel niet meer mogelijk. Excuses.

DATE: 26.08.85

DRAFT KIM-KENNER ARTICLE.

*** DOS-65 CORNER ***

BY: COEN KLEIPOOL, VAL DE PERIER, F-83310 COGOLIN. T.(33)94-544382

THE DOS EDITOR

IN THE FIRST ARTICLE OF THIS SERIES ON DOS-65 I BRIEFLY MENTIONED THE EDITOR, WHICH WE WILL NOW EXAMINE MORE IN DETAIL. IN FACT, THIS EDITOR IS SO EXTENSIVE THAT I COULD EASILY FILL SEVERAL PAGES.

TO ANSWER THE OBVIOUS QUESTION RIGHTAWAY, AN EDITOR SHOULD COME WITH DOS-65 TO ENSURE A PERFECT COORDINATION BETWEEN WRITING AND FILING AND TO MAKE SURE THAT RE-EDITING A FILE ALREADY WRITTEN ON DISK HAPPENS SMOOTHLY. THE FILE IS READ FROM DISK INTO MEMORY, CHANGED AND WRITTEN BACK IN SUCH A WAY THAT YOU HARDLY REALISE YOU ARE DEALING WITH A DISK-FILE. THEREFORE DOS AND EDITOR SOFTWARE SHOULD BE INTEGRATED.

MOST OF US WILL PROBABLY HAVE A LINE EDITOR INCORPORATED IN ASSEMBLER SOFTWARE. PERHAPS THIS IS A GOOD MOMENT TO EXPLAIN A FEW NOTIONS ABOUT EDITORS TO NEWCOMERS.

A LINE EDITOR STORES THE FILE IN MEMORY IN LINES, EACH LINE BEING OPENED BY A LINE NUMBER AND CLOSED BY AN END-OF-LINE CHARACTER. IF YOU WISH TO CHANGE A LINE, YOU FIRST HAVE TO GO INTO COMMAND MODE AND TYPE IN THE LINE NUMBER. THE COMPUTOR WILL THEN SEARCH THROUGH MEMORY FOR THAT LINENUMBER AND PLACE THE LINE IN A BUFFER WHERE IT CAN BE EDITED, AFTER WHICH THE LINE IS PUT BACK IN ITS PROPER PLACE IN MEMORY BY USING ITS LINE NUMBER. IF THE EDITED LINE IN DIFFERENT IN LENGTH, THE COMPUTOR MUST THEN PERFORM SOME BLOCKMOVING.

ALL THIS CAN BE USED, BUT IS NOT VERY PRACTICAL. REAL TROUBLE STARTS HOWEVER IF YOU TRY TO MOVE A BLOCK OF SOURCE TO A DIFFERENT LOCATION IN THE FILE. THE FIRST TIME I DID THIS IN MICRO-ADE, I THOUGHT MY COMPUTOR HAD BROKEN DOWN ! IT TOOK HIM MORE THAT 20 MINUTES TO MOVE 30 LINES, AMPLE TIME TO GO DOWN TO THE LOCAL BISTRO AND HAVE A DRINK.

YOU WONT HAVE THESE PROBLEMS WITH A FULL-SCREEN EDITOR. THE FILE IS WRITTEN IN MEMORY AS A CONTINUOUS STRING AND THE SCREEN IS A WINDOW WHICH YOU CAN PLACE ANYWHERE OVER THE FILE TO READ A PAGE. YOU CAN SCROLL UP AND DOWN AS IF YOU ARE READING PAGES OF A BOOK. TO CHANGE THE TEXT, JUST PLACE THE CURSOR OVER A CHARACTER AND DELETE OR INSERT.

ANY EDIT COMMAND IS EXECUTED BY MANIPULATING THE CURSOR, WHICH IS LINKED BY SOFTWARE TO THE IMAGINARY CURSOR IN THE COMPUTOR'S MEMORY.

I HAVE USED THE DOS-65 EDITOR FOR SOME TIME AND I AM STILL IMPRESSED BY ITS MANY FACILITIES. IT CAN BE USED AS A WORD PROCESSOR, JUST HAVE A LOOK AT THE FOLLOWING COMMANDS:

PASTE: ALLOWS TO PUT A PARAGRAPH OF THE FILE IN PASTE-BUFFER AND INSERT THIS PASTE IN ANOTHER PART OF THE FILE;

SEARCH & CHANGE: AFTER YOU HAVE DEFINED A SEARCH AND A CHANGE-STRING, THE COMPUTOR WILL SEARCH TROUGH MEMORY AND REPLACE THE SEARCH STRING EVERYWHERE BY THE CHANGE STRING.

MACRO: THE PROGRAM CAN LEARN A STRING OF EDIT-COMMANDS WHICH CAN BE USED MANY TIMES OVER.

FILL: REARRANGES THE LINES IN SUCH A WAY THAT EACH LINE BECOMES A STANDARD LENGHT.

SFILL: INSERTS SPACES TO PROVIDE A STRAIGHT RIGHTHAND EDGE TO THE TEXT.

NEEDLESS TO SAY THAT ALL THESE FACILITIES MAKE THE DOS-65 EDITOR ALMOST A WORDPROCESSOR. BUT THERE IS MORE, THE REAL PURPOSE IS OBTAIN A POWERFUL TOOL TO WRITE SOURCE WHICH CAN BE USED WITH SEVERAL ASSEMBLERS. THIS IS NOT TOO OBVIOUS AS EACH ASSEMBLER USES A DIFFERENT SYSTEM.

AD BROUWER HAS WRITTEN A CONVERSION PROGRAM FOR BOTH MOSER AND MICRO-ADE WHICH MAKES THIS POSSIBLE. YOU HAVE TO WRITE THE SOURCE IN ACCORDANCE WITH THE MOSER OR MICRO-ADE FORMAT (AS FAR AS SPACES AND TABS GO) BUT THERE IS NO NEED TO WORRY ABOUT LINENUMBERS OR END-OF-LINE CHARACTERS. IF YOU USE THE NEW "GET" COMMAND IN MOSER, LINENUMBERS AND END-OF-LINE CHARACTERS WILL BE AUTOMATICALLY INSERTED IN THE MOSER SOURCE BUFFER BY THE CONVERSION PROGRAM. THE MOSER OR MICRO-ADE LINE-EDITOR IS NO LONGER USED.

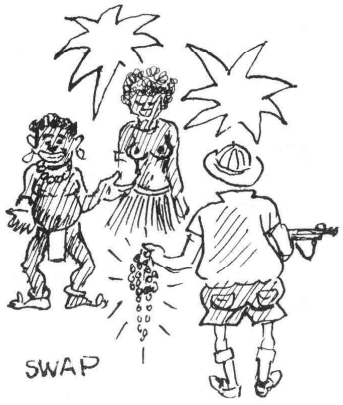
TO DEBUG, THE SOURCE IS RE-EDITED IN DOS-65, AFTER WHICH YOU CAN MAKE ANOTHER ATTEMPT TO ASSEMBLE. IF MOSER SHOWS AN ERROR MESSAGE IN A CERTAIN LINE, DOS-65 CAN JUMP TO THAT LINE NUMBER ALTHOUGH IT WONT SHOW THE ACTUAL NUMBER.

THE "NEW" MOSER ALSO HAS SPECIAL COMMANDS FOR DOS REDIRECTS. THE INPUT REDIRECT MAKES IT POSSIBLE TO PUT ALL THE ASSEMBLY COMMANDS IN ONE INPUT FILE; RUNNING THAT FILE MAKES ASSEMBLY AUTOMATIC. AT THE SAME TIME THE ASSEMBLED LISTING IS WRITTEN ON THE OUTPUT FILE FOR FURTHER PROCESSING. YOU CAN IMAGE THAT IS IT QUITE A THRILL TO WORK WITH THIS PROFESSIONAL SYSTEM !

THERE IS A LOT MORE TO TELL, BUT THAT GOES BEYOND THE SCOPE OF AN INTRODUCTORY ARTICLE. I SUGGEST THAT YOU HAVE A GOOD LOOK AT DOS-65 AT THE NEXT KIM-CLUB MEETING !

NEXT ISSUE: DOS-65 MEMORY MAPPING.
A DOS-65 DISASSEMBLER.

4651 CHARACTERS, 96 LINES.



```
78 88 MLIST
SCR # 78
  0 ( F O R T H   S C R E E N   E D I T O R                          )
  1 (        Fridus Jonkman , Stijn Streuvelslaan 9                  )
  2 (        5242 GD Rosmalen , tel. 04192-16146                     )
  3 ( ======================================================== )
  4 ( De line-editor van W. Ragsdale voor fig-FORTH is niet erg      )
  5 ( plezierig in het gebruik: reden om een full-screen-editor      )
  6 ( te gaan bouwen. Ik heb gebruik gemaakt van mijn publikatie     )
  7 ( met woorden voor cursorbesturing en simpele grafische routi-   )
  8 ( nes voor de Junior met VDU-kaart. In deze vorm is de editor    )
  9 ( geschikt voor fig-systemen met diskbuffers kleiner dan 1K.     )
 10 ( Omdat de diskbuffers elk een tussenruimte hebben van 4 bytes,  )
 11 ( heb ik een extra buffer gemaakt van 1K, waarin tekst wordt     )
 12 ( gemanipuleerd. Via de constante TEMP krijgt deze een plaats.   )
 13 ( In de laatste screen wordt aangegeven hoe je de editor een-    )
 14 ( voudig geschikt kunt maken voor systemen met diskbuffers van   )
 15 ( 1K.                                                           )


SCR # 79
  0 FORTH DEFINITIONS HEX
  1 ( \  ((   ---   )) ; commentaar na \ i.p.v. tussen haakjes      )
  2 : \  IN @ C/L / 1+ C/L * IN ! ;  IMMEDIATE
  3 \ WHERE en COPY komen uit de Ragsdale-editor
  4 \ WHERE (( block# offset ---   ))
  5 \ gebruiken na error bij compileren van screens
  6 : WHERE DUP B/SCR / DUP SCR ! ." SCR # " DECIMAL . SWAP C/L /MOD
  7   C/L * ROT BLOCK + CR C/L TYPE CR HERE C@ - SPACES 5E
  8   EMIT [COMPILE] QUIT ;
  9 \ COPY (( van-scr# naar-scr# ---   ))
 10 : COPY B/SCR * OFFSET @ + SWAP B/SCR * B/SCR OVER + SWAP DO DUP
 11   FORTH I BLOCK 2 - ! 1+ UPDATE LOOP DROP FLUSH ;
 12 \ (CMOVE (( van naar #bytes ---   ))
 13 \ kan i.t.t. CMOVE ook blokken overlappend vooruit schuiven
 14 : (CMOVE -DUP IF >R R + 1 - SWAP 1 - R> OVER + DO I C@ OVER C!
 15          1 - -1 +LOOP DROP ELSE DROP DROP ENDIF ;  -->


SCR # 80
  0 \ Gebruik is gemaakt van de CASE-konstruktie van Charles Eaker
  1 : CASE    ?COMP CSP @ !CSP 4 ;  IMMEDIATE
  2 : OF      4 ?PAIRS COMPILE OVER COMPILE = COMPILE 0BRANCH
  3           HERE 0 , COMPILE DROP 5 ;  IMMEDIATE
  4 : ENDOF   5 ?PAIRS COMPILE BRANCH HERE 0 , SWAP 2
  5           [COMPILE] ENDIF 4 ;  IMMEDIATE
  6 : ENDCASE 4 ?PAIRS COMPILE DROP BEGIN SP@ CSP @ = 0= WHILE 2
  7           [COMPILE] ENDIF REPEAT CSP ! ;  IMMEDIATE
  8 \ Specifieke editorwoorden komen in EDIT
  9 VOCABULARY EDIT IMMEDIATE    EDIT DEFINITIONS DECIMAL
 10 512  CONSTANT TEMP      \ de extra buffer begint hier op $0200
 11 0    VARIABLE CUR       \ cursorpositie t.o.v. TEMP
 12 0    VARIABLE INSERT    \ insert-mode = 1
 13 \ DISPL-I-ON  ((   ---   )) ; display insert-on
 14 : DISPL-I-ON  63 1 !CURSOR ." INSERT-ON"                    ;
 15               10 1 62 0 RECTANGLE ;                  -->


SCR # 81
  0 \ DISPL-I-OFF  ((   ---   )) ; display spaties
  1 : DISPL-I-OFF  3 0 DO 62 I !CURSOR 14 SPACES LOOP ;
  2 \ CUR)CHAR-LINE  ((   --- char# line# ))
  3 \ zet de cursorpositie om naar x-y-waarden voor de screen
  4 : CUR)CHAR-LINE  CUR @ C/L /MOD ;
  5 \ .CUR  ((   ---   )) ; display cursor
  6 : .CUR  CUR)CHAR-LINE 5 + >R 9 + R> !CURSOR ;
  7 \ DISPL-LINE  (( line# ---   )) ; display line
  8 : DISPL-LINE  9 OVER 5 + !CURSOR
  9               C/L * TEMP + C/L TYPE ;
 10 \ DISPL-SCR  (( line# ---   )) ; display screen vanaf line#
 11 : DISPL-SCR  16 SWAP DO I DISPL-LINE LOOP ;
 12 \ GET-SCR  ((   ---   )) ; haal screen naar diskbuffer
 13 \                          en naar extra buffer
 14 : GET-SCR  SCR @ 16 0 DO I OVER (LINE) TEMP I C/L * + SWAP
 15                         CMOVE LOOP DROP ;          -->
```

```
SCR # 82
  0 \ SETUP  ((   ---   )) ; display scherm-lay-out editor
  1 : SETUP  CLS ( clear screen ) 10 1 !CURSOR
  2          ." F o r t h  S c r e e n  E d i t o r"
  3          37 1 8 0 RECTANGLE
  4          9 3 !CURSOR ." SCR# " SCR ?   \ display scr-nr
  5          16 0 DO 5 I OVER + !CURSOR I 2 .R LOOP  \ display
  6                                                  \ line-nr's
  7          64 16 8 4 RECTANGLE   0 INSERT !
  8          0 22 !CURSOR ;  \ vanaf deze positie kan je informatie
  9                          \ kwijt over te gebruiken kommando's
 10 \ nu volgen alle commando's die kunnen worden gegeven
 11 \ MODE  ((   ---   )) ; toggle mode en display
 12 : MODE INSERT @ 0= DUP INSERT !                \ toggle
 13        IF DISPL-I-ON ELSE DISPL-I-OFF ENDIF  \ display
 14        .CUR  ;
 15                                                 -->

SCR # 83
  0 \ HOME  ((   ---   )) ; cursor-home binnen screen
  1 : HOME  9 5 !CURSOR 0 CUR ! ;
  2 \ CLEAR  ((   ---   )) ; clear/display screen en cursor-home
  3 : CLEAR  TEMP 1024 BLANKS 0 DISPL-SCR HOME ;
  4 \ LEFT  ((   ---   )) ; cursor-left binnen screen
  5 : LEFT CUR)CHAR-LINE DROP 0 ) IF -1 CUR +! 8 EMIT ENDIF ;
  6 \ RIGHT  ((   ---   )) ; cursor-right binnen screen
  7 : RIGHT CUR)CHAR-LINE DROP 63 ( IF 1 CUR +! 9 EMIT ENDIF ;
  8 \ UP  ((   ---   )) ; cursor-up binnen screen
  9 : UP  CUR)CHAR-LINE SWAP DROP 0 ) IF -64 CUR +! 11 EMIT ENDIF ;
 10 \ DOWN  ((   ---   )) ; cursor-down binnen screen
 11 : DOWN CUR)CHAR-LINE SWAP DROP 15 ( IF 64 CUR +! 10 EMIT ENDIF ;
 12 \ I-LINE ((   ---   )) ; voeg lege regel tussen op cursorpositie
 13 : I-LINE  CUR)CHAR-LINE SWAP DROP DUP C/L * TEMP + DUP DUP C/L
 14           + OVER 1024 TEMP + SWAP - (CMOVE C/L BLANKS
 15           DISPL-SCR .CUR ;        -->

SCR # 84
  0 \ D-LINE  ((   ---   )) ; verwijder regel op cursorpositie
  1 : D-LINE  CUR)CHAR-LINE SWAP DROP DUP C/L * TEMP + DUP C/L +
  2           SWAP OVER 1024 TEMP + SWAP - CMOVE 960 TEMP + C/L
  3           BLANKS DISPL-SCR .CUR ;
  4 \ H-LINE  ((   ---   )) ; bewaar regel op cursorpositie in PAD
  5 : H-LINE  CUR)CHAR-LINE SWAP DROP C/L * TEMP + PAD 1+ C/L DUP
  6           PAD C! CMOVE ;
  7 \ R-LINE  ((   ---   )) ; PAD naar regel op cursorpositie
  8 : R-LINE  CUR)CHAR-LINE SWAP DROP DUP PAD 1+ SWAP C/L * TEMP
  9           + C/L CMOVE DISPL-LINE .CUR ;
 10 \ (ERASE  ((   ---   )) ; spaties in regel voor cursorpositie
 11 : (ERASE  CUR)CHAR-LINE SWAP DROP DUP C/L * TEMP + DUP CUR @
 12           TEMP + SWAP - DUP IF BLANKS DISPL-LINE .CUR
 13           ELSE DROP DROP DROP ENDIF ;
 14
 15                                                 -->

SCR # 85
  0 \ ERASE)  ((   ---   )) ; spaties in regel vanaf cursorpositie
  1 : ERASE)  CUR)CHAR-LINE SWAP DROP DUP CUR @ TEMP + DUP ROT 1+
  2           C/L * TEMP + SWAP - BLANKS DISPL-LINE .CUR ;
  3 \ D-CHAR  ((   ---   )) ; verwijder karakter op cursorpositie
  4 : D-CHAR  CUR @ TEMP + DUP 1+ SWAP CUR)CHAR-LINE SWAP DROP DUP
  5           )R 1+ C/L * TEMP + 1 - DUP )R OVER - CMOVE BL R) C! R)
  6           DISPL-LINE .CUR ;
  7 \ NEXT-LINE  ((   ---   )) ; cursor naar begin volgende regel
  8 : NEXT-LINE  CUR)CHAR-LINE SWAP DROP DUP 15 ( IF 1+ C/L * CUR !
  9              .CUR ELSE DROP ENDIF ;
 10 \ CHAR  (( ascii-waarde ---   )) ; voeg karakter toe op cursor-
 11 \ positie en schuif rest van regel op indien insert = on
 12 : CHAR  CUR @ TEMP + INSERT @ IF DUP DUP 1+ CUR)CHAR-LINE SWAP
 13         DROP 1+ C/L * TEMP + OVER - (CMOVE C! ELSE C! ENDIF
 14         CUR)CHAR-LINE )R 63 ( IF 1 CUR +! ENDIF R)
 15         DISPL-LINE .CUR ;
```

```
SCR # 86
  0 EDIT DEFINITIONS DECIMAL \ i.v.m. laden tweede deel
  1 \ SAVE  (( ---  )) : schrijf screen naar (oseudo-)disk(-ram)
  2 : SAVE  SCR @ B/SCR * DUP B/SCR + SWAP DO TEMP I B/SCR MOD B/BUF
  3         * + I BUFFER B/BUF CMOVE UPDATE LOOP FLUSH ;
  4 \ COMMAND  (( ---  )) : dit is de hoofdlooo van de editor
  5 : COMMAND  BEGIN KEY DUP 32 ( OVER 127 ) OR 0= IF DUP CHAR ENDIF
  6            DUP 21 - WHILE CASE
  7    2  OF H-LINE  ENDOF 3  OF R-LINE  ENDOF 4  OF I-LINE     ENDOF
  8    5  OF D-LINE  ENDOF 6  OF D-CHAR  ENDOF 7  OF (ERASE     ENDOF
  9    8  OF LEFT    ENDOF 9  OF RIGHT   ENDOF 10 OF DOWN       ENDOF
 10   11 OF UP      ENDOF 12 OF CLEAR   ENDOF 13 OF NEXT-LINE ENDOF
 11   16 OF SAVE    ENDOF 17 OF ERASE)  ENDOF 22 OF MODE       ENDOF
 12   28 OF HOME    ENDOF
 13            ENDCASE REPEAT DROP ;
 14                                                          --)
 15


SCR # 87
  0 FORTH DEFINITIONS
  1 \ Met een van beide voloende commando's kan je vanuit FORTH de
  2 \ editor starten.
  3 \ E  (( scr# ---  )) of (( ---  )) : indien de stack leeo is.
  4 \    wordt de te editen screen uit SCR oehaald
  5 : E  EDIT DEPTH IF SCR ! ENDIF SETUP GET-SCR
  6      0 DISPL-SCR HOME COMMAND CLS FORTH ;
  7 \ EC  (( scr# ---  )) of (( ---  )) : als E. maar de screen
  8 \    wordt eerst schoon oemaakt
  9 : EC  EDIT DEPTH IF SCR ! ENDIF SETUP CLEAR
 10      COMMAND CLS FORTH ;
 11 ;S
 12
 13
 14
 15


SCR # 88
  0 \ Aandassinoen voor FORTH-svstemen met diskbuffers van 1Kbvte.
  1 \ ------------------------------------------------------------
  2 \ Nu is oeen extra buffer nodio: je kan rechtstreeks editen in
  3 \ je diskbuffers !
  4 \ 1. De variabele TEMP kan verdwijnen
  5 \ 2. Vervang overal TEMP door PREV @
  6 \ 3. De definitie van SAVE wordt:
  7 \    : SAVE  UPDATE FLUSH ;
  8 \ 4. De definitie van GET-SCR wordt:
  9 \    : GET-SCR SCR @ 16 0 DO I OVER (LINE) DROP DROP LOOP DROP ;
 10
 11
 12 \              ============================
 13
 14
 15

OK
```

DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65

RS232 ACIA (6551) instel programma.

Dit programma is speciaal geschreven voor de ACIA op de CPU-kaart van het DOS65 systeem. In de monitor bij de reset-routine worden twee waarden in het geheugen gezet die naar de ACIA gecopieerd worden op het moment dat in- of output device 3 aangezet wordt. Deze twee variabelen stellen het command en control register van de ACIA voor. Normaal moet altijd de data-sheet geraadpleegd worden om de ACIA in te stellen op mogelijkheden die niet standaard door de monitor worden ingesteld. Default staat de ACIA ingesteld op:

- twee stop bits
- 7 data bits
- interne baudrate generator
- 2400 baud

Wil men hier van afwijken, dan moeten de twee variabelen in het geheugen op de adressen $CE50 en $CE51 veranderd worden. Met dit programma is het mogelijk de waarden voor deze registers snel te bepalen en in te voeren. De waarden worden op de adressen $CE50 en $CE51 gezet. Desgewenst kan bij het verlaten van het programma de instelling in een file worden gezet die naar de systeemschijf wordt geschreven onder de naam RS232.DAT. Deze file kan in de LOGIN.COM file worden aangeroepen als altijd die instelling vereist is. Heeft men bijvoorbeeld een seriele printer van 4800 baud, die altijd op de ACIA is aangesloten, dan wordt een keer het programma RS232 aangeroepen en de ACIA goed ingesteld. Het resultaat wordt op schijf weggezet en in de LOGIN.COM file komt te staan ' LO RS232.DAT '. Zonder de aanhalingstekens natuurlijk. Wordt het LOGIN programma doorlopen, dan worden de waarden uit de file naar het geheugen gecopieerd en bij aanzetten van in- of output device 3 wordt de ACIA geinitialiseerd op de ingestelde manier.

Men moet er wel rekening mee houden dat na een reset de instelling weer default is zoals boven is genoemd.

Het programma RS232 wordt vanaf de systeemschijf aangeroepen als een commando. Na aanroepen wordt een overzicht gegeven op het beeldscherm van de default instelling van de ACIA. Alles kan nu gewijzigd worden door de 'cursor' te besturen met cursor besturings karakters. De 'cursor' staat op de plaats van de waarde die knippert. De invers op het beeldscherm staande waarden worden vastgehouden. Om een overzicht te krijgen van de geldige commando's en cursor besturings karakters wordt de toets H ingetypt voor Help.

Het programma RS232 zal vanaf DOS65 versie 2.01 standaard worden meegeleverd op de systeemschijf. Nu kan men in het bezit komen van dit programma door een geformatteerde diskette voorzien van retourenveloppe met postzegels en adres op te sturen naar het bekende DOS65 distributie adres. Op een landelijke clubbijeenkomst kan ook het programma gecopieerd worden.

DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65

```
13-Feb-86 20:53  RS232_LISTING  Page 1


                ;      File: RS232-1.TXT
                ;      Date: 25 Oct 85
                ;Programmer: A.S. Hankel
                ;
                ;Copyright (c): KIM Gebruikersclub Nederland
                ;
          8000          ORG     $8000
                ;**
                ;**
                ;
                ;********** Zero page locations **********
                ;
          0050  PNT     EQU     $50             ;common used pointer
          0052  INDPNT  EQU     PNT+2           ;indirect used pointer
          0054  ADPNT   EQU     PNT+4           ;addressing pointer
          0056  SPNT    EQU     PNT+6           ;'save' pointer
                ;
                ;********** Special characters **********
                ;
          0008  BS      EQU     $08             ;'left' command
          0009  HT      EQU     $09             ;'right' command
          000A  LF      EQU     $0A             ;'down' command
          000B  VT      EQU     $0B             ;'up' command
          000C  CLS     EQU     $0C             ;clear screen command
          0014  FB      EQU     $14             ;flag byte for X,Y coordinates
          0020  SP      EQU     $20             ;space-character
          001B  ESC     EQU     $1B             ;escape-character
                ;
                ;********** HaViSoft variables & entry's **********
                ;
          CE50  CTL     EQU     $CE50           ;acia control register
          CE51  CMD     EQU     $CE51           ;acia command register
          CEC7  SECONDS EQU     $CEC7           ;seconds register (clock)
          E00F  TEXT    EQU     $E00F           ;print following text-string
          E012  GET     EQU     $E012           ;fetch cha. if present else A=0
                ;
                ;********** DOS65 entry's **********
                ;
          A020  IN      EQU     $A020           ;fetch character from input
          A023  OUT     EQU     $A023           ;print character in accu
          A006  COMMAND1 EQU    $A006           ;execute command
                ;
                ;********** Utility start **********
                ;
8000 4C CC81    START   JMP     MAIN            ;continue behind sub's
                ;
                ;sub print string
8003 86 56      STRING  STX     SPNT            ;set pointer
8005 85 57              STA     SPNT+1
8007 A0 00              LDY     #0              ;index zero
8009 B1 56      STRING2 LDA     [SPNT],Y        ;fetch character from actual string
800B C9 FF              CMP     #$FF            ;end-of-string ?
800D F0 0B              BEQ     STRINGEND       ;if so, then exit
800F 20 23A0            JSR     OUT             ;else print the character
8012 E6 56              INC     SPNT            pointer:=pointer+1
8014 D0 F3              BNE     STRING2
8016 E6 57              INC     SPNT+1
8018 D0 EF              BNE     STRING2
801A 60         STRINGEND RTS                   ;return to caller
                ;
                ;sub compute MPOS table
801B A2 00      CMPOS   LDX     #0              ;reset pointer to table
801D AD 50CE            LDA     CTL             ;baudrate
8020 29 0F              AND     #$0F            ;make high nibble zero
8022 9D 3A88            STA     MPOS,X          ;save in tabel
8025 E8                 INX                     ;pointer=1
8026 AD 50CE            LDA     CTL             ;wordlenght
8029 29 7F              AND     #$7F            ;reset msb
802B 4A                 LSRA
802C 4A                 LSRA
```

```
802D 4A              LSRA
802E 4A              LSRA
802F 4A              LSRA
8030 8D 4D88         STA     TEMP
8033 A9 03           LDA     #3
8035 38              SEC
8036 ED 4D88         SBC     TEMP
8039 9D 3A88         STA     MPOS,X
803C E8              INX                     ;pointer=2
803D AD 51CE         LDA     CMD             ;parity
8040 29 20           AND     #%00100000      ;check b5 being 0
8042 D0 05           BNE     CMPOS2
8044 9D 3A88         STA     MPOS,X
8047 F0 0F           BEQ     CMPOS3          ;branch always
8049 AD 51CE CMPOS2  LDA     CMD
804C A0 06           LDY     #6              ;b6,7 to b0,1
804E 4A      MOVE    LSRA
804F 88              DEY
8050 D0 FC           BNE     MOVE
8052 A8              TAY
8053 C8              INY
8054 98              TYA
8055 9D 3A88         STA     MPOS,X
8058 E8      CMPOS3  INX                     ;pointer=3
8059 AD 50CE         LDA     CTL             ;stopbits
805C 0A              ASLA
805D B0 04           BCS     CMPOS4
805F A9 00           LDA     #0              ;1 stopbit
8061 F0 24           BEQ     CMPOS7
8063 AD 4D88 CMPOS4  LDA     TEMP
8066 C9 03           CMP     #3              ;check for 5 b. wordl.
8068 D0 0B           BNE     CMPOS5          ;no
806A AD 51CE         LDA     CMD             ;check no parity
806D 29 20           AND     #%00100000
806F F0 04           BEQ     CMPOS5
8071 A9 01           LDA     #1              ;1.5 stopbit
8073 D0 12           BNE     CMPOS7          ;branch always
8075 AD 4D88 CMPOS5  LDA     TEMP
8078 D0 0B           BNE     CMPOS6          ; no 8 bits wl.
807A AD 51CE         LDA     CMD             ;check parity
807D 29 20           AND     #%00100000
807F F0 04           BEQ     CMPOS6
8081 A9 00           LDA     #0
8083 F0 02           BEQ     CMPOS7          ;branch always
8085 A9 02   CMPOS6  LDA     #2              ;then 2 stopbits
8087 9D 3A88 CMPOS7  STA     MPOS,X
808A E8              INX                     ;pointer=4
808B AD 50CE         LDA     CTL             ;receiver clock source
808E 29 10           AND     #%00010000
8090 D0 04           BNE     CMPOS8
8092 A9 00           LDA     #0
8094 F0 02           BEQ     CMPOS9          ;branch always
8096 A9 01   CMPOS8  LDA     #1
8098 9D 3A88 CMPOS9  STA     MPOS,X
809B E8              INX                     ;pointer=5
809C AD 51CE         LDA     CMD             ;normal/echo mode
809F 29 10           AND     #%00010000
80A1 D0 02           BNE     CMPOS10
80A3 F0 02           BEQ     CMPOS11         ;branch always
80A5 A9 01   CMPOS10 LDA     #1
80A7 9D 3A88 CMPOS11 STA     MPOS,X
80AA E8              INX                     ;pointer=6
80AB AD 51CE         LDA     CMD             ;receiver interrupt
80AE 29 02           AND     #%00000010
80B0 F0 02           BEQ     CMPOS12         ;branch always
80B2 A9 01           LDA     #1
80B4 9D 3A88 CMPOS12 STA     MPOS,X
80B7 E8              INX                     ;pointer=7
80B8 AD 51CE         LDA     CMD             ;data terminal ready
80BB 29 01           AND     #%00000001
```

```
80BD 9D 3A88            STA     MPOS,X
80C0 E8                 INX                     ;pointer=8
80C1 AD 51CE            LDA     CMD             ;transmitter control
80C4 4A                 LSRA
80C5 4A                 LSRA
80C6 29 03              AND     #%00000011
80C8 9D 3A88            STA     MPOS,X
                ;
                ;compute index table
                ;
80CB A2 00      RCINDEX LDX     #0              ;set index zero
80CD 18         CMPOS13 CLC
80CE BD 3A88            LDA     MPOS,X          ;fetch value from table
80D1 7D 3A88            ADC     MPOS,X          ;double it
80D4 9D 4488            STA     INDEX,X         ;save in index table
80D7 E8                 INX                     ;increase pointer
80D8 E0 09              CPX     #9              ;until end of table
80DA D0 F1              BNE     CMPOS13
80DC 60                 RTS
                ;
                ;sub set inverse flag 'i' on computed locations
80DD A0 00      SETINV  LDY     #0              ;index Y zero
80DF A2 00              LDX     #0              ;index X zero
80E1 18         SETINV2 CLC                     ;prepare for add.
80E2 BD 0D88            LDA     OFFSET,X        ;fetch low byte offsettable
80E5 79 4488            ADC     INDEX,Y         ;add index to this value
80E8 85 52              STA     INDPNT          ;set low byte index pointer
80EA E8                 INX
80EB BD 0D88            LDA     OFFSET,X        ;fetch high byte offsettable
80EE 69 00              ADC     #$0             ;add carry
80F0 85 53              STA     INDPNT+1        ;set high byte index pointer
80F2 98                 TYA                     ;save Y-reg.
80F3 48                 PHA
80F4 A0 00              LDY     #$0
80F6 B1 52              LDA     [INDPNT],Y      ;fetch address pointer
80F8 85 54              STA     ADPNT
80FA C8                 INY
80FB B1 52              LDA     [INDPNT],Y
80FD 85 55              STA     ADPNT+1
80FF A0 04              LDY     #4              ;fourth character in string
8101 A9 69              LDA     #'i             ;inverse flag
8103 91 54              STA     [ADPNT],Y
8105 68                 PLA                     ;restore Y-reg
8106 A8                 TAY
8107 E8                 INX
8108 C8                 INY
8109 C0 0A              CPY     #10             ;done all offsets ?
810B D0 D4              BNE     SETINV2
810D 60                 RTS
                ;
                ;sub print string pointered by PNT
810E A0 00      PNTOUT  LDY     #0
8110 B1 50      PNTOUT2 LDA     [PNT],Y
8112 F0 06              BEQ     PNTOUT3         ;$00: end-ofstring
8114 2C 23A0            JSR     OUT
8117 C8                 INY
8118 D0 F6              BNE     PNTOUT2
811A 60         PNTOUT3 RTS
                ;
                ;next sub contains two functions:
                ;a-reset 'n' flags in BD00 file
                ;b-recompute acia CMD & CTL register
                ;  from MPOS variables
811B A9 7E      RECOM   LDA     #BD00&255
811D 85 56              STA     SPNT
811F A9 85              LDA     #BD00>>8
8121 85 57              STA     SPNT+1
8123 A2 04              LDX     #4
8125 A9 6E              LDA     #'n
8127 9D 7E85            STA     BD00,X          ;reset first flag
```

```
812A E6 56      RECOM2  INC     SPNT
812C D0 02              BNE     RECOM3
812E E6 57              INC     SPNT+1
8130 A0 00      RECOM3  LDY     #0
8132 B1 56              LDA     [SPNT],Y
8134 F0 02              BEQ     RECOM4          ;in case 00 found
8136 D0 F2              BNE     RECOM2          ;branch always
8138 A0 05      RECOM4  LDY     #5              ;index to 5th pos from 00
813A A9 6E              LDA     #'n             ;normal flag
813C 91 56              STA     [SPNT],Y
813E A5 57              LDA     SPNT+1          ;check for end
8140 C9 87              CMP     #TR03>>8        ;when not equal
8142 D0 E6              BNE     RECOM2          ;back to loop
8144 A5 56              LDA     SPNT            ;else check low byte
8146 C9 9C              CMP     #TR03-1&255
8148 D0 E0              BNE     RECOM2
                ;recompute CMD & CTL register
814A A2 01      RECOM5  LDX     #1              ;wordlenght
814C BD 3A88            LDA     MPOS,X
814F 8D 4D88            STA     TEMP
8152 A9 03              LDA     #3
8154 38                 SEC
8155 ED 4D88            SBC     TEMP
8158 A2 04              LDX     #4              ;rec. clock source
815A 48                 PHA
815B BD 3A88            LDA     MPOS,X
815E 4A                 LSRA
815F 68                 PLA
8160 6A                 RORA
8161 6A                 RORA
8162 6A                 RORA
8163 48                 PHA
8164 A2 03              LDX     #3              ;stopbits
8166 BD 3A88            LDA     MPOS,X
8169 F0 03              BEQ     RECOM6
816B 38                 SEC
816C B0 01              BCS     RECOM7          ;branch always
816E 18         RECOM6  CLC
816F 68         RECOM7  PLA
8170 6A                 RORA
8171 18                 CLC
8172 6D 3A88            ADC     MPOS            ;baudrate
8175 8D 50CE            STA     CTL
                ;
8178 A2 02              LDX     #2              ;parity
817A BD 3A88            LDA     MPOS,X          ;a special for parity
817D F0 16              BEQ     SPE
817F C9 01              CMP     #1
8181 F0 12              BEQ     SPE
8183 C9 02              CMP     #2
8185 D0 04              BNE     CC1
8187 A9 03              LDA     #3
8189 D0 0A              BNE     SPE
818B C9 03      CC1     CMP     #3
818D D0 04              BNE     CC2
818F A9 05              LDA     #5
8191 D0 02              BNE     SPE
8193 A9 07      CC2     LDA     #7
8195 48         SPE     PHA
8196 A2 05              LDX     #5              ;normal/echo mode
8198 BD 3A88            LDA     MPOS,X
819B 4A                 LSRA
819C 68                 PLA
819D 2A                 ROLA
819E 2A                 ROLA
819F 2A                 ROLA
81A0 A2 08              LDX     #8              ;transmitter
81A2 18                 CLC
81A3 7D 3A88            ADC     MPOS,X
81A6 2A                 ROLA
```

13-Feb-86 20:54 RS232_LISTING Page 5

```
81A7 2A              ROLA
81A8 48              PHA
81A9 A2 06           LDX    #6              ;receiver interrupt
81AB BD 3A88         LDA    MPOS,X
81AE F0 05           BEQ    RECOM8
81B0 68              PLA
81B1 09 02           ORA    #%00000010
81B3 D0 03           BNE    RECOM9
81B5 68      RECOM8  PLA
81B6 29 FD           AND    #%11111101
81B8 48      RECOM9  PHA
81B9 A2 07           LDX    #7              ;data terminal ready
81BB BD 3A88         LDA    MPOS,X
81BE F0 05           BEQ    RECOM10
81C0 68              PLA
81C1 09 01           ORA    #%00000001
81C3 D0 03           BNE    RECOM11
81C5 68      RECOM10 PLA
81C6 29 FE           AND    #%11111110
81C8 8D 51CE RECOM11 STA    CMD
81CB 60              RTS
                     ;
                     ;**************************************
                     ;
                     ;Main programm starts here
                     ;
81CC 20 1B80 MAIN    JSR    CMPOS           ;compute tables etc.
81CF 20 DD80         JSR    SETINV          ;reset inverse flag's
81D2 AE 4488         LDX    INDEX           ;compute string-address baudrate
81D5 BD BD87         LDA    V1,X
81D8 85 50           STA    PNT
81DA E8              INX
81DB BD BD87         LDA    V1,X
81DE 85 51           STA    PNT+1
81E0 A9 00           LDA    #0              ;reset VPOS
81E2 8D 4388         STA    VPOS
                     ;
                     ;master pgm loop
                     ;
81E5 A2 FB   MAIN1   LDX    #STARTSCR&255   ;set up startscreen
81E7 A9 82           LDA    #STARTSCR>>8
81E9 20 0380         JSR    STRING
81EC 20 1B80         JSR    CMPOS
81EF 20 DD80         JSR    SETINV
81F2 A2 7E           LDX    #BD00&255
81F4 A9 85           LDA    #BD00>>8
81F6 20 0380         JSR    STRING
81F9 20 0E81 MAIN2   JSR    PNTOUT
81FC AD C7CE         LDA    SECONDS
81FF 8D 4E88         STA    SEC2            ;set seconds flag
8202 20 12E0 MAIN3   JSR    GET             ;check pressed key
8205 D0 1A           BNE    COMHANDS        ;if so, exec cmd
8207 AD 4E88 MAIN4   LDA    SEC2            ;else ...
820A CD C7CE         CMP    SECONDS         ;wait for timeout
820D F0 F3           BEQ    MAIN3
820F A0 04           LDY    #4              ;toggle inverse flag
8211 B1 50           LDA    [PNT],Y
8213 C9 6E           CMP    #'n             ;normal
8215 F0 04           BEQ    MAIN5
8217 A9 6E           LDA    #'n             ;set normal
8219 D0 02           BNE    MAIN6
821B A9 69   MAIN5   LDA    #'i             ;set inverse
821D 91 50   MAIN6   STA    [PNT],Y
821F D0 D8           BNE    MAIN2           ;branch always
                     ;commandhandler; structure adapted
                     ;from HaViSoft's monitor pgm
8221 20 2782 COMHANDS JSR   COMHAND         ;exec commandhandler
8224 4C F981         JMP    MAIN2           ;continue in mainloop
                     ;
8227 AA      COMHAND TAX                    ;save command
```

```
8228 C9 08            CMP     #BS
822A F0 0B            BEQ     IVOUT
822C C9 09            CMP     #HT
822E F0 07            BEQ     IVOUT
8230 A9 69            LDA     #'i             ;set inverse
8232 8D 3F82          STA     IVMODE
8235 D0 05            BNE     SETM
8237 A9 6E      IVOUT LDA     #'n             ;set normal
8239 8D 3F82          STA     IVMODE
823C A0 04      SETM  LDY     #4
823E A9               FCC     $A9             ;code for LDA #
823F 00         IVMODE FCC    0
8240 91 50            STA     [PNT],Y         ;in inverse mode
8242 20 0E81          JSR     PNTOUT
8245 8A               TXA                     ;restore command
8246 A0 00            LDY     #0              ;set index zero
8248 D9 2888    COMHAND2 CMP  COMTABLE,Y      ;find command
824B F0 07            BEQ     COMHAND3        ;if found
824D C8               INY                     ;continue search
824E C0 06            CPY     #6              ;6 available commands
8250 D0 F6            BNE     COMHAND2        ;if not on end
8252 F0 0B            BEQ     COMHAND4        ;else return to mainloop
8254 98         COMHAND3 TYA
8255 0A               ASLA
8256 AA               TAX
8257 BD 2F88          LDA     COMADD+1,X
825A 48               PHA
825B BD 2E88          LDA     COMADD,X
825E 48               PHA
825F 60         COMHAND4 RTS                  ;exec command
                ;
                ;LF command
8260 AD 4388    COMLF LDA     VPOS            ;check possibility
8263 C9 08            CMP     #8              ;for move
8265 F0 25            BEQ     COMLFEND
8267 EE 4388          INC     VPOS
826A AD 4388    COMLFB LDA    VPOS
826D A8               TAY
826E 0A               ASLA                    ;*2
826F AA               TAX
8270 18               CLC
8271 BD 0D88          LDA     OFFSET,X
8274 79 4488          ADC     INDEX,Y
8277 85 52            STA     INDPNT
8279 E8               INX
827A BD 0D88          LDA     OFFSET,X
827D 69 00            ADC     #0
827F 85 53            STA     INDPNT+1
8281 A0 00            LDY     #0
8283 B1 52            LDA     [INDPNT],Y
8285 85 50            STA     PNT
8287 C8               INY
8288 B1 52            LDA     [INDPNT],Y
828A 85 51            STA     PNT+1
828C 60         COMLFEND RTS
                ;end LF command
                ;
                ;VT command
828D AD 4388    COMVT LDA     VPOS            ;check possibility
8290 F0 FA            BEQ     COMLFEND        ;when equal to zero
8292 CE 4388          DEC     VPOS
8295 4C 6A82          JMP     COMLFB          ;continue elsewhere
                ;end VT command
                ;
                ;BS command
8298 AD 4388    COMBS LDA     VPOS            ;fetch current V pos
829B AA               TAX                     ;use as index
829C BD 3A88          LDA     MPOS,X          ;fetch current H pos
829F F0 09            BEQ     COMBSEND        ;no movement possible
82A1 DE 3A88          DEC     MPOS,X          ;else decrease allowed
```

```
13-Feb-86 20:54  RS232_LISTING  Page 7


82A4 20 CB80    COMBSB  JSR     RCINDEX         ;recompute index
82A7 4C 6A82            JMP     COMLFB          ;continue elsewhere
82AA 60         COMBSEND RTS
                ;end BS command
                ;
                ;HT command
82AB AD 4388    COMHT   LDA     VPOS            ;fetch current V pos
82AE AA                 TAX                     ;use as index
82AF BD 3A88            LDA     MPOS,X          ;fetch current H pos
82B2 DD 1F88            CMP     MAX,X           ;compare with max value
82B5 F0 06              BEQ     COMHTEND        ;no movement possible
82B7 FE 3A88            INC     MPOS,X          ;else increase allowed
82BA 4C A482            JMP     COMBSB          ;continue elsewhere
82BD 60         COMHTEND RTS
                ;end HT command
                ;
                ;H command (help)
82BE A2 1D      COMH    LDX     #HELPSCR&255    ;print HELP-screen
82C0 A9 84              LDA     #HELPSCR>>8
82C2 20 0380            JSR     STRING
82C5 20 20A0            JSR     IN              ;wait for a key
82C8 20 1B81            JSR     RECOM           ;reset string's and reg.'s
82CB 68                 PLA                     ;reset stackpointer
82CC 68                 PLA
82CD 4C E581            JMP     MAIN1           ;continue in mainloop
                ;end H command
                ;
                ;Q command
82D0 20 1B81    COMQ    JSR     RECOM
82D3 A2 36              LDX     #END&255        ;ask for confirmation
82D5 A9 85              LDA     #END>>8
82D7 20 0380            JSR     STRING
82DA 20 20A0    COMQB   JSR     IN
82DD 48                 PHA
82DE 20 23A0            JSR     OUT
82E1 68                 PLA
82E2 C9 4E              CMP     #'N
82E4 F0 0B              BEQ     COMQEND
82E6 C9 59              CMP     #'Y
82E8 D0 07              BNE     COMQEND
82EA A0 5E              LDY     #SCOM&255
82EC A9 85              LDA     #SCOM>>8
82EE 20 06A0            JSR     COMMAND1
82F1 A2 79      COMQEND LDX     #END1&255       ;print END message
82F3 A9 85              LDA     #END1>>8
82F5 20 0380            JSR     STRING
82F8 68                 PLA                     ;reset stackpointer
82F9 68                 PLA
82FA 60                 RTS                     ;back to DOS65
                ;end Q command
                ;
                ;end availlable commands
                ;
                ;
                ;********** Tables etc. **********
                ;
                ;startscreen
82FB 0C1401011B STARTSCR FCC    CLS,FB,1,1,ESC,'i- RS232 -  '
8300 692D205253
8305 323332202D
830A 2020
830C 4143494120         FCC     'ACIA 6551 set & check utility '
8311 3635353120
8316 7365742026
831B 2063686563
8320 6B20757469
8325 6C69747920
832A 2020566572         FCC     '  Version 1.01   Oct. 1985  HaViSoft',ESC,'n'
832F 73696F6E20
8334 312E303120
```

13-Feb-86 20:54 RS232_LISTING Page 8

```
8339 20204F6374
833E 2E20313938
8343 3520204861
8348 5669536F66
834D 741B6E
8350 140C034261          FCC     FB,12,3,'Baudrate :'
8355 7564726174
835A 65203A
835D 140A05576F          FCC     FB,10,5,'Wordlength :'
8362 72646C656E
8367 677468203A
836C 140E065061          FCC     FB,14,6,'Parity :'
8371 7269747920
8376 3A
8377 140C075374          FCC     FB,12,7,'Stopbits :'
837C 6F70626974
8381 73203A
8384 1406095265          FCC     FB,6,9,'Receiver clock :'
8389 6365697665
838E 7220636C6F
8393 636B203A
8397 14070A5265          FCC     FB,7,10,'Receiver mode :'
839C 6365697665
83A1 72206D6F64
83A6 65203A
83A9 1402085265          FCC     FB,2,11,'Receiver interrupt :'
83AE 6365697665
83B3 7220696E74
83B8 6572727570
83BD 74203A
83C0 14010C4461          FCC     FB,1,12,'Data terminal ready :'
83C5 7461207465
83CA 726D696E61
83CF 6C20726561
83D4 6479203A
83D8 14230D5F5F          FCC     FB,35,13,'___'
83DD 5F
83DE 140C0E5472          FCC     FB,12,14,'Transmit : Interrupt:  RTS:  Transmitter:'
83E3 616E736D69
83E8 74203A2049
83ED 6E74657272
83F2 7570743A20
83F7 2052545333A
83FC 2020547261
8401 6E736D6974
8406 7465723A
840A 1401155479          FCC     FB,1,21,'Type H for help'
840F 7065204B20
8414 666F722068
8419 656C70
841C FF                  FCC     $FF             ;end-of-file marker
                         ;
                         ;helpscreen
841D 140102190A HELPSCR  FCC     FB,1,2,$19,10,10,10,SP,SP,SP,SP
8422 0A0A202020
8427 20
8428 417661696C          FCC     'Availlable commands :',13,10,10
842D 6C61626C65
8432 20636F6D6D
8437 616E647320
843C 3A0D0A0A
8440 2020202020          FCC     SP,SP,SP,SP,SP,SP,SP,SP,SP,SP,SP,SP
8445 2020202020
844A 20
844B 2020202020          FCC     SP,SP,SP,SP,SP,SP,SP,SP,SP,SP,SP,SP
8450 2020202020
8455 20
8456 48203A2070          FCC     'H : prints this message',13,10
845B 72696E7473
8460 2074686973
```

```
8465 206D657373
846A 6167650D0A
846F 2020202020         FCC     SP,SP,SP,SP,SP
8474 4261636B73         FCC     'Backspace or CTL H : left',13,10
8479 7061636520
847E 6F72204354
8483 4C2048203A
8488 206C656674
848D 0D0A
848F 486F72697A         FCC     'Horizontal tab or CTL I : right',13,10
8494 6F6E74616C
8499 2074616220
849E 6F72204354
84A3 4C2049203A
84A8 2072696768
84AD 740D0A
84B0 2020202020         FCC     SP,SP,SP,SP,SP,SP
84B5 20
84B6 4C696E6566         FCC     'Linefeed or CTL J : down',13,10
84BB 656564206F
84C0 722043544C
84C5 204A203A20
84CA 646F776E0D
84CF 0A
84D0 2020566572         FCC     SP,SP,'Vertical tab or CTL K : up',13,10
84D5 746963616C
84DA 2074616220
84DF 6F72204354
84E4 4C204B203A
84E9 2075700D0A
84EE 2020202020         FCC     SP,SP,SP,SP,SP,SP,SP,SP,SP,SP,SP
84F3 2020202020
84F8 20
84F9 2020202020         FCC     SP,SP,SP,SP,SP,SP,SP,SP,SP,SP,SP
84FE 2020202020
8503 20
8504 51203A2065         FCC     'Q : exit this utility',13,10,10
8509 7869742074
850E 6869732075
8513 74696C6974
8518 790D0A0A
851C 456E746572         FCC     'Enter any key to continue'
8521 20616E7920
8526 6B65792074
852B 6F20636F6E
8530 74696E7565
8535 FF                 FCC     $FF             ;end-of-file marker
                        ;
                        ;confirmation-message
8536 1401155361 END     FCC     FB,1,21,'Save current values on disk ? (Y/N) ',$FF
853B 7665206375
8540 7272656E74
8545 2076616C75
854A 6573206F6E
854F 206469736B
8554 203F202859
8559 2F4E2920FF
                        ;
                        ;DOS65 command
855E 5341564520 SCOM    FCC     'SAVE S:RS232.DAT CE50,CE51',0
8563 533A525332
8568 33322E4441
856D 5420434535
8572 302C434535
8577 3100
                        ;
                        ;cursor re-positioning
8579 1401151AFF END1    FCC     FB,1,21,$1A,$FF
                        ;
                        ;baudrate
```

```
857E 1417031B6E BD00    FCC     FB,23,3,ESC,'nExternal',ESC,'n',0
8583 4578746572
8588 6E616C1B6E
858D 00
858E 1420031B6E BD01    FCC     FB,32,3,ESC,'n50',ESC,'n',0
8593 35301B6E00
8598 1423031B6E BD02    FCC     FB,35,3,ESC,'n75',ESC,'n',0
859D 37351B6E00
85A2 1426031B6E BD03    FCC     FB,38,3,ESC,'n109.92',ESC,'n',0
85A7 3130392E39
85AC 321B6E00
85B0 142D031B6E BD04    FCC     FB,45,3,ESC,'n135.58',ESC,'n',0
85B5 3133352E35
85BA 381B6E00
85BE 1434031B6E BD05    FCC     FB,52,3,ESC,'n150',ESC,'n',0
85C3 3135301B6E
85C8 00
85C9 1438031B6E BD06    FCC     FB,56,3,ESC,'n300',ESC,'n',0
85CE 3330301B6E
85D3 00
85D4 143C031B6E BD07    FCC     FB,60,3,ESC,'n600',ESC,'n',0
85D9 3630301B6E
85DE 00
85DF 1417041B6E BD08    FCC     FB,23,4,ESC,'n1200',ESC,'n',0
85E4 313230301B
85E9 6E00
85EB 141C041B6E BD09    FCC     FB,28,4,ESC,'n1800',ESC,'n',0
85F0 313830301B
85F5 6E00
85F7 1421041B6E BD0A    FCC     FB,33,4,ESC,'n2400',ESC,'n',0
85FC 323430301B
8601 6E00
8603 1426041B6E BD0B    FCC     FB,38,4,ESC,'n3600',ESC,'n',0
8608 333630301B
860D 6E00
860F 142B041B6E BD0C    FCC     FB,43,4,ESC,'n4800',ESC,'n',0
8614 343830301B
8619 6E00
861B 1430041B6E BD0D    FCC     FB,48,4,ESC,'n7200',ESC,'n',0
8620 373230301B
8625 6E00
8627 1435041B6E BD0E    FCC     FB,53,4,ESC,'n9600',ESC,'n',0
862C 393630301B
8631 6E00
8633 143A041B6E BD0F    FCC     FB,58,4,ESC,'n19200',ESC,'n',0
8638 3139323030
863D 1B6E00
                ;wordlength
8640 1417051B6E WL05    FCC     FB,23,5,ESC,'n5',ESC,'n',0
8645 351B6E00
8649 1419051B6E WL06    FCC     FB,25,5,ESC,'n6',ESC,'n',0
864E 361B6E00
8652 141B051B6E WL07    FCC     FB,27,5,ESC,'n7',ESC,'n',0
8657 371B6E00
865B 141D051B6E WL08    FCC     FB,29,5,ESC,'n8',ESC,'n',0
8660 381B6E00
                ;parity
8664 1417061B6E PANO    FCC     FB,23,6,ESC,'nNo',ESC,'n',0
8669 4E6F1B6E00
866E 141A061B6E PAOD    FCC     FB,26,6,ESC,'nOdd',ESC,'n',0
8673 4F64641B6E
8678 00
8679 141E061B6E PAEV    FCC     FB,30,6,ESC,'nEven',ESC,'n',0
867E 4576656E1B
8683 6E00
8685 1423061B6E PAMA    FCC     FB,35,6,ESC,'nMark',ESC,'n',0
868A 4D61726B1B
868F 6E00
8691 1428061B6E PASP    FCC     FB,40,6,ESC,'nSpace',ESC,'n',0
8696 5370616365
```

13-Feb-86 20:54 RS232_LISTING Page 11

```
869B 1B6E00
                ;stopbits
869E 1417071B6E ST01    FCC     FB,23,7,ESC,'n1',ESC,'n',0
86A3 311B6E00
86A7 1419071B6E ST15    FCC     FB,25,7,ESC,'n1.5',ESC,'n',0
86AC 312E351B6E
86B1 00
86B2 141D071B6E ST02    FCC     FB,29,7,ESC,'n2',ESC,'n',0
86B7 321B6E00
                ;receiver clock
86BB 1417091B6E RCEX    FCC     FB,23,9,ESC,'nExternal',ESC,'n',0
86C0 4578746572
86C5 6E616C1B6E
86CA 00
86CB 1420091B6E RCIN    FCC     FB,32,9,ESC,'nInternal',ESC,'n',0
86D0 496E746572
86D5 6E616C1B6E
86DA 00
                ;receiver mode
86DB 14170A1B6E RMNO    FCC     FB,23,10,ESC,'nNormal',ESC,'n',0
86E0 4E6F726D61
86E5 6C1B6E00
86E9 141E0A1B6E RMEC    FCC     FB,30,10,ESC,'nEcho',ESC,'n',0
86EE 4563686F1B
86F3 6E00
                ;receiver interrupt
86F5 14170B1B6E RIEN    FCC     FB,23,11,ESC,'nEnabled',ESC,'n',0
86FA 456E61626C
86FF 65641B6E00
8704 141F0B1B6E RIDI    FCC     FB,31,11,ESC,'nDisabled',ESC,'n',0
8709 4469736162
870E 6C65641B6E
8713 00
                ;data terminal ready
8714 14170C1B6E DTRH    FCC     FB,23,12,ESC,'nDisabled (DTR high)',ESC,'n',0
8719 4469736162
871E 6C65642028
8723 4454522068
8728 696768291B
872D 6E00
872F 142B0C1B6E DTRL    FCC     FB,43,12,ESC,'nEnabled (DTR low)',ESC,'n',0
8734 456E61626C
8739 6564202844
873E 5452206C6F
8743 77291B6E00
                ;transmit
8748 14170F1B6E TR00    FCC     FB,23,15,ESC,'nDisabled    High  Off',ESC,'n',0
874D 4469736162
8752 6C65642020
8757 2020486967
875C 6820204F66
8761 661B6E00
8765 1417101B6E TR01    FCC     FB,23,16,ESC,'nEnabled     Low   On',ESC,'n',0
876A 456E61626C
876F 6564202020
8774 20204C6F77
8779 2020204F6E
877E 1B6E00
8781 1417111B6E TR02    FCC     FB,23,17,ESC,'nDisabled    Low   On',ESC,'n',0
8786 4469736162
878B 6C65642020
8790 20204C6F77
8795 2020204F6E
879A 1B6E00
879D 1417121B6E TR03    FCC     FB,23,18,ESC,'nDisabled    Low   Break',ESC,'n',0
87A2 4469736162
87A7 6C65642020
87AC 20204C6F77
87B1 2020204272
87B6 65616B1B6E
```

```
87BB 00
87BC FF                 FCC     $FF             ;end-of-file marker
                        ;
                        ;addresstable :
                        ;baudrate
87BD 7E85               V1      FDB     BD00
87BF 8E85                       FDB     BD01
87C1 9885                       FDB     BD02
87C3 A285                       FDB     BD03
87C5 B085                       FDB     BD04
87C7 BE85                       FDB     BD05
87C9 C985                       FDB     BD06
87CB D485                       FDB     BD07
87CD DF85                       FDB     BD08
87CF EB85                       FDB     BD09
87D1 F785                       FDB     BD0A
87D3 0386                       FDB     BD0B
87D5 0F86                       FDB     BD0C
87D7 1B86                       FDB     BD0D
87D9 2786                       FDB     BD0E
87DB 3386                       FDB     BD0F
                        ;wordlength
87DD 4086               V2      FDB     WL05
87DF 4986                       FDB     WL06
87E1 5286                       FDB     WL07
87E3 5B86                       FDB     WL08
                        ;parity
87E5 6486               V3      FDB     PAN0
87E7 6E86                       FDB     PA0D
87E9 7986                       FDB     PAEV
87EB 8586                       FDB     PAMA
87ED 9186                       FDB     PASP
                        ;stopbits
87EF 9E86               V4      FDB     ST01
87F1 A786                       FDB     ST15
87F3 B286                       FDB     ST02
                        ;receiver clock
87F5 BB86               V5      FDB     RCEX
87F7 CB86                       FDB     RCIN
                        ;receiver mode
87F9 DB86               V6      FDB     RMN0
87FB E986                       FDB     RMEC
                        ;receiver interrupt
87FD F586               V7      FDB     RIEN
87FF 0487                       FDB     RIDI
                        ;data terminal ready
8801 1487               V8      FDB     DTRH
8803 2F87                       FDB     DTRL
                        ;transmit
8805 4887               V9      FDB     TR00
8807 6587                       FDB     TR01
8809 8187                       FDB     TR02
880B 9D87                       FDB     TR03
                        ;
                        ;offset table for screen strings
880D BD87               OFFSET  FDB     V1
880F DD87                       FDB     V2
8811 E587                       FDB     V3
8813 EF87                       FDB     V4
8815 F587                       FDB     V5
8817 F987                       FDB     V6
8819 FD87                       FDB     V7
881B 0188                       FDB     V8
881D 0588                       FDB     V9
                        ;
                        ;table MAX; max movements par subcomm
881F 0F03040201 MAX     FCC     15,3,4,2,1,1,1,1,3
8824 01010103
                        ;
                        ;command table
```

```
13-Feb-86 20:54  RS232_LISTING  Page 13

8828 08090A0B48 COMTABLE FCC    BS,HT,LF,VT,'HQ'
882D 51
                ;
                ;command address table
882E 9782       COMADD  FDB     COMBS-1
8830 AA82               FDB     COMHT-1
8832 5F82               FDB     COMLF-1
8834 8C82               FDB     COMVT-1
8836 BD82               FDB     COMH-1
8838 CF82               FDB     COMQ-1
                ;
                ;in-programm variables
883A            MPOS    RES     9               ;actual (hor.) position in subcomm
8843            VPOS    RES     1               ;actual position (vertical)
8844            INDEX   RES     9               ;index table
884D            TEMP    RES     1               ;temporary use
884E            SEC2    RES     1               ;flagbyte seconds
                ;
                ;end RS232-SET
                ;
                        END
```

==> <==

VERVANGING 6502 DOOR 65C02 IN ELEKTUUR'S JUNIOR COMP.

==> <==

Jan Vernimmen
Telf: 02990 - 21739

De vervanging van de 6502-processor door een 65C02 is mij niet zonder meer mogelijk gebleken. Problemen traden op bij:
1 : De kloksignalen (0, 1 en 2)
2 : De fan-out
3 : Het RAM-RW signaal

1 : De klok 0 ingang van de ROCKWELL 65C02 blijkt niet geheel compatible met de 6502 versie. Daar van de interne oscilator gebruik gemaakt wordt (kristal direkt aan de klok 0 aangesloten) is het voltage van de 0 naar 1 overgang van deze ingang bepalend voor de hoge en lage fase van klok 2 en klok 1. Het gevolg was bij mij: klok 2 hoog 630 nanoseconden en (dus) klok 2 laag 370 nanoseconden welke waarden ver buiten de toleranties liggen (430 tot 570 nanoseconden). Dit had tot gevolg dat de 65C02 soms plotseling overschakelde op lagere of zelfs op hogere klokfrequenties, kennelijk met 370 nanoseconden als basis.
Door een weestand van 68 K Ohm van de ingang van klok 0 naar aarde te leggen (uitgezocht uiteraard met een scoop en een instelpotmeter) werden de tijden normaal: 520 nanoseconden hoog, 480 laag. Echter, nu weigerde soms (en misschien eerder ook al) de oscillator te starten. Dit werd verholpen door een weerstand van 1,5 M Ohm over het kristal te plaatsen. De faseverhouding van klok 2 werd hierdoor nauwelijk beinvloed (werd 518 om 482 nanoseconden). Een extra oscillator zou eenvoudiger geweest zijn.

2 : De fan-out van de CMOS processors bedraagt 1, die van de 6502 bedraagt 2. Dit probleem is simpelweg op te lossen door ervoor te zorgen dat alle TTL-IC's van het LS-type zijn. De geheugen-IC's en de PIA zijn MOS-IC's en doen verhoudingsgewijs niet mee aan stroomafname. Wilt U toch, net als ik, de PIA vervangen door zijn CMOS-versie, houdt dan rekening met extra problemen zoals een niet meer werkende (opnieuw af te stellen) cassette-interface.

3 : Het RAM-RW signaal.
Zorg dat de pull-up weerstand R5 op de hoofdprint 470 Ohm is zoals beschreven in deel 3 JUNIOR-computerboek, bladzijde 36.

Voor ik aan het RAM-RW signaal ging sleutelen heb ik de Grounds van de processor, de PIA en de Eprom op de basiskaart met aparte lijnen aan aarde verbonden (pin 16 a+c connector). Dit gaf een merkbare verbetering van de controle-signalen volgens waarneming met de oscilloscoop. Na wijziging R5 heb ik dit laten zitten. Of deze extra bedrading noodzakelijk geweest is heb ik verder niet nagegaan.

Conclusie:
Wilt U Uw 6502 vervangen door een (Rockwell) CMOS-versie en U heeft geen scoop om het kloksignaal te controleren: bouw een opsteekprintje met een externe oscillator en breng wijzigingen aan volgens punt 2 en eventueel 3.

```
**********************************
*                                *
*     D O S  6 5  C O R N E R    *
*                                *
**********************************
```

Author: drs Conrad H. Kleipool, Val de Perier, F-83310 Cogolin, France.
tel: (33) 94.54.43.82.

NEW: DOS-65 VERSION 2.

Let's start this Dos65-corner with some good news. The Brouwer-Hankel-Visschedyk team has not been wasting their time since they brought out Dos65. They have just announced an upgraded version which has some exciting new features:

- 80 instead of 40 tracks;
- Directory divided in 7 subdirectories;
- Memory extension to accomodate a virtual disk up to 1 Mbyte;
- Extra DD formatting writing 18 sectors per track instead of 16. This results in 720 Kbytes formatted for an 80-tracks floppy.
- Parameters can be incorporated in command files.

Some of these features I have already seen implemented at Ad Brouwer. A virtual disk (also called ramdisk) is a part of memory which is laid out as a disk.
The advantage is that you can download all of your systemdisk into this virtual diskmemory after startup. The diskmemory behaves exactly as a real disk, there are directories, tracks, sectors, etc. However, as the information now comes from memory, a virtual disk access has no delay and all utilities are available immediately. Also, your previous systemdrive is now free to do other work, so it will actually save a second drive.
Obviously, the extra memory can also be used as ordinary Ram. The Dos65 team is actually designing a PC-board for the virtual (ram) disk, which will hold 16 Drams of 256K for a total of 512 Mbyte. It will be possible to reduce this to 8 Drams for only 256 Kbyte or to extend the board to 1 Mbyte. Considering the give-away prices for Rams (about Dfl.10) a Ram disk will be a lot cheaper than an extra drive and faster too ! As Dos65 supports three drives you can work with the ramdisk and two diskdrives.
According to Erwin Visschedyk the bare pc board will cost about DFl.50, provided he can find sufficient clubmembers to justify a production order. So don't hesitate to call him if you want to participate.
By the way, there is an interesting article in the Review Section of the september 1985 Byte issue about a Ramdisk for the Commodore 64 produced by P-Technologies. For Commodore owners worth reading, it provides a good insight in the use of ramdisks.

NEW UTILITIES TOO !

Little by little Dos65 is becoming a very powerful professional tool. Those of my readers who work regularly with the Moser assembler know that this assembler, although rather powerful also tends to be very slow.
To assemble Dos65, for instance, takes at least an quarter of an hour, which is rather bothersome if one has made only a minor adjustment. Ad Brouwer has written a new assembler in C-language for the 6800, which he has now compiled for the 6502.
Although it is just as complete as Moser, its speed is absolutely amazing. Brouwer demonstrated AS65 for me and I estimate the speed at least five times faster than Moser. The source format has been revised and differs considerably from Moser, but everything is very logical.
It is possible to assemble for the 6502, the standard 65C02 and the Rockwell equivalent. In order to convert old Moser source files to the new AS65 format a new utility will be provided called MTOAS (Moser to As).
As65 does not store the object code it produces in memory but writes code directly on disk as binary files. To test the new code it is usually not convenient to load this in its own memory location because there may be eproms there or you do not wish to overwrite the old code yet. Normally, binary files cannot be loaded to a different starting address than the one provided in the file. Therefore a new utility has been written called OLOAD which allows you to load a binary file elsewhere in memory.
The third complement to AS65 will be a utility called MAP. This will show the begin and end addresses of the various parts of a binary file and also the main start address of this

6502-Tracer for the MON/DOS65 computer.

Author: Rene Hettfleisch, The Netherlands.
System: MON/DOS65 computer

While programming in the assembly-language, a tracer is a handy expedient to trace errors in a program. For this I use the "6502-tracer" of J. Ruppert (Elektor Holland nr. 244, Feb. 1984, page 2-66, 2-67), which I adjusted to MON/DOS65. The working of the tracer will be explained in this article. The original program is given in a hexdump in the article mentioned above. By this the following memory-locations have to be changed:

051C : old $7E, new $FE (change of interrupt-vector)
051D : old $1A, new $CE
0521 : old $7F, new $FF
0522 : old $1A, new $CE

06A1 : old $8F, new $27 (change of PRBYT)
06A2 : old $12, new $E0

06A6 : old $34, new $00 (change of PRCHA)
06A7 : old $13, new $E0

Furthermore there will be a little program added before and behind the original program; refer the source-listing. The program starts at $046B and ends at $072E. The program can be used as a utility by saving it on the system-disk and next you have to give the file the "command-mode" by using the SETMODE-RWDC filename.

After starting the tracer the program will ask the start-address to be traced. Before doing this the program to be traced has to be loaded into the memory. Watch the overlap of the tracerprogram by this. When the startaddress is entered the tracing will be started. It would be wise to send the output to the printer (output-device 2 and 3), because the tracing can't be interrupted in the meantime, after which the tracing can continue. The tracing can be stopped finally by depressing a random key (this causes an interruption). When at the end of the program which had to be traced, a RTS ($60) will be set, the tracing will be stopped here.

After the tracing has been stopped, a RESET has to be given, after which DOS65 has to be restarted. The reason for this is that the IRQ-pointer will be detoured and I won't be able to get it back normally, so the computer is blocked. However, you can live with this.
Good luck with the tracer!

```
046B           TRACER ORG    $046B

               TEMPORARIES AND BUFFERS

     ED 00     START  *      $00ED
     FE CE     IRQ    *      $CEFE
     69 04     IRQSAV *      $0469
     0E C1     VAIER  *      $C10E

               MON65 SUBROUTINES

     0F E0     STRING *      $E00F
     0C E0     INKEY  *      $E00C
     09 E0     OUT    *      $E009
     48 E0     CONVRT *      $E048


046B 20 0F E0  BEGIN  JSR    STRING
046E 1B               =      $1B
046F 69               =      'i
0470 0A               =      $0A
0471 20               =      '
0472 36               =      '6
0473 35               =      '5
0474 30               =      '0
0475 32               =      '2
0476 2D               =      '-
0477 74               =      't
0478 72               =      'r
0479 61               =      'a
047A 63               =      'c
047B 65               =      'e
047C 72               =      'r
047D 20               =      '
047E 1B               =      $1B
047F 6E               =      'n
0480 0D               =      $0D
0481 0A               =      $0A
0482 0A               =      $0A
0483 47               =      'G
0484 69               =      'i
0485 76               =      'v
0486 65               =      'e
0487 20               =      '
0488 73               =      's
0489 74               =      't
048A 61               =      'a
048B 72               =      'r
048C 74               =      't
048D 61               =      'a
048E 64               =      'd
048F 72               =      'r
0490 65               =      'e
0491 73               =      's
0492 73               =      's
0493 3A               =      ':
0494 20               =      '
0495 00               =      $00
0496 A9 00            LDAIM  $00
0498 85 ED            STA    START
049A 85 EE            STA    START  +01

049C 20 0C E0  GETCHR JSR    INKEY  GET CHARACTER FROM
049F AA               TAX           KEYBOARD
04A0 C9 0D            CMPIM  $0D    IF RETURN
04A2 D0 0E            BNE    GTCHRA
04A4 20 00 E0         JSR    OUT    PRINT CR
04A7 A9 0A            LDAIM  $0A
04A9 20 00 E0         JSR    OUT    PRINT LF TWO TIMES
04AC 20 00 E0         JSR    OUT
04AF 4C E6 04         JMP    BEGINA

04B2 C9 7F     GTCHRA CMPIM  $7F    IF DELETE
04B4 F0 1C            BEQ    DELETE GO TO DELETE
04B6 20 48 E0         JSR    CONVRT CONVERT ASCII TO
04B9 30 E1            BMI    GETCHR BINARY
04BB A8               TAY           IF VALID CHARACTER
04BC 8A               TXA
04BD 20 00 E0         JSR    OUT    PRINT CHARACTER
04C0 98               TYA
04C1 0A               ASLA          STORE CHAR IN START
04C2 0A               ASLA
04C3 0A               ASLA
04C4 0A               ASLA
04C5 A0 04            LDYIM  $04

04C7 0A        LOOPA  ASLA
04C8 26 ED            ROL    START
04CA 26 EE            ROL    START  +01
04CC 88               DEY
04CD D0 F8            BNE    LOOPA
04CF 4C 9C 04         JMP    GETCHR
```

MON/DOS65 CORNER

A MACROLOADER AND SAVER FOR ED

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

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

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

1A)

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

1B) ED (CR)

```
LOAD 'filenm_1.mcr'
LOAD 'filenm_2.mcr'
LOAD 'filenm_3.mcr'
LOAD 'filenm_4.mcr'
EX 'filenm_5.mcr'
```

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

Program 2:

a)

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

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

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

b)

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

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

load &1.mcr
```

PS: This works only with release 2.01 !!!

LOWER-CASE COMMENT IN MOSER's ASSEMBLER

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

```
;File COMMENT.MAC by B. de Bruine /E. Visschedijk
                9F00            org     $9f00

                0090    curpoi  equ     $90
                0092    endpoi  equ     $92

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

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

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

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

                9F00            end     comment

label table:

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

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

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

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

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

label table:    BRKTEST   90AC  CONT      90B8  KEYPORT    C111

Errors detected: 0
Back to DOS65 with SYS10*4096 (Cold) or ^J (Warm)
If you have any suggestions, corrections, extensions, a.s.o., please let me know.
```

B. de Bruine

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

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

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

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

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

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

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

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

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

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

Door : Phons Bloemen

Na de fantastisch mooie uitbreiding van de VDU-kaart door Janssen in DE 6502 KENNER tot een grafisch display van 640 * 200 pixels, nu een low-cost uitbreiding van de VDU-kaart tot kleurenkaart. Deze uitbreiding geeft de volgende mogelijkheden:

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

Beschrijving van de schema's.

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

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

De bouw.

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

De karaktergenerator Eprom.

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

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

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

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

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

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

Programmeermodel kleurenkaart.

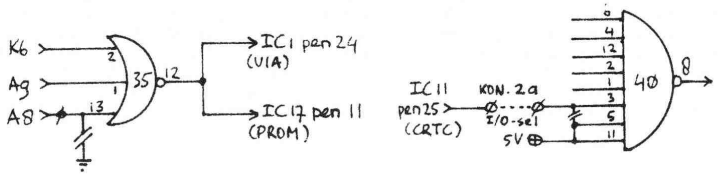
bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
kar. 0 adr.	kar. 1 adr.	kar. 2 adr.	kar. 3 adr.	kar. 4 adr.	kar. 5 adr.	kar. 6 adr.	kar. 7 adr.

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

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

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

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

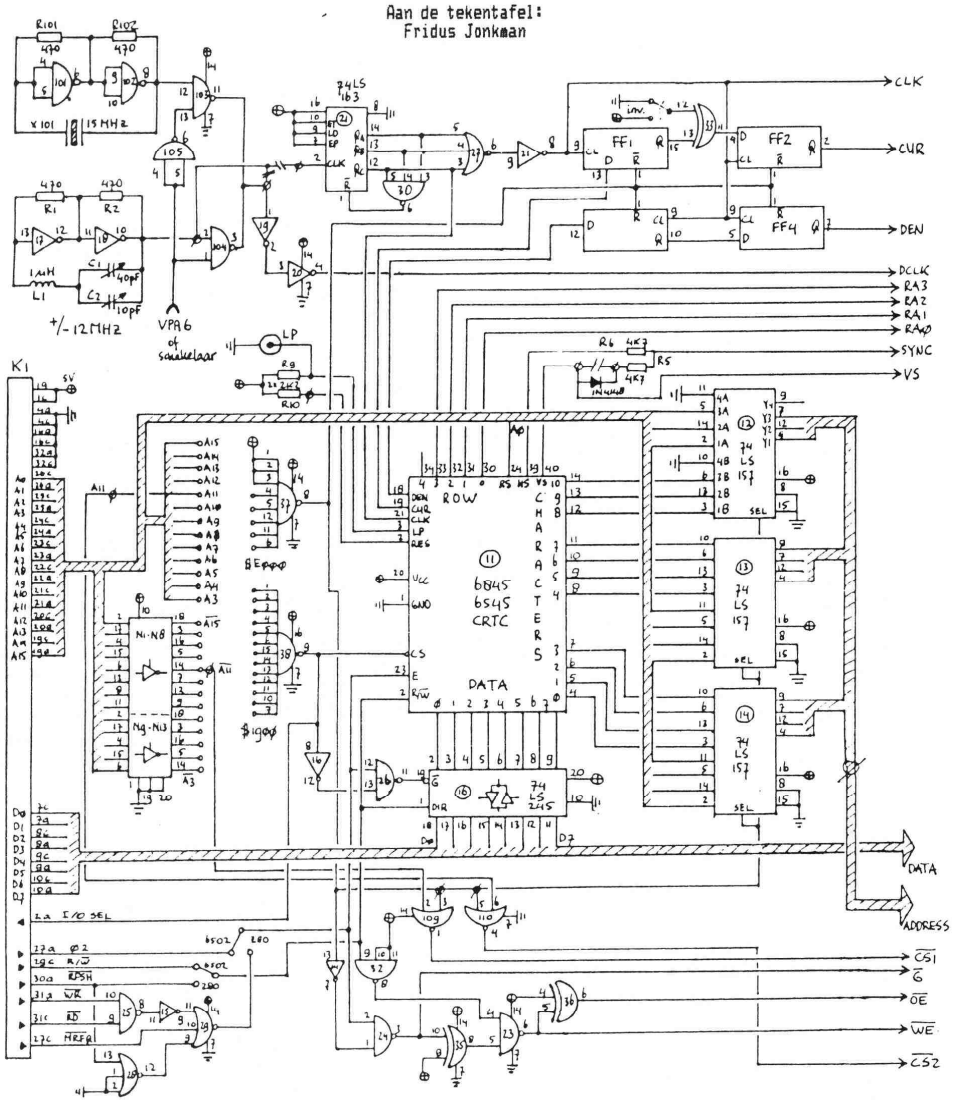


Benodigd: 1 x 6116 1 x 74LS02 1 x 10 uF 2 x 470 Ohm
2 x 74LS273 1 x 4024 1 x 33 Ohm 1 x 10 kOhm
3 x 74LS00 1 x 74LS245 1 x 68 Ohm 1 x 1N4148
2 x 74LS08 9 x 100 nF 1 x 120 Ohm
1 x kristal 12 of 15 MHz

Ombouwschema.

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

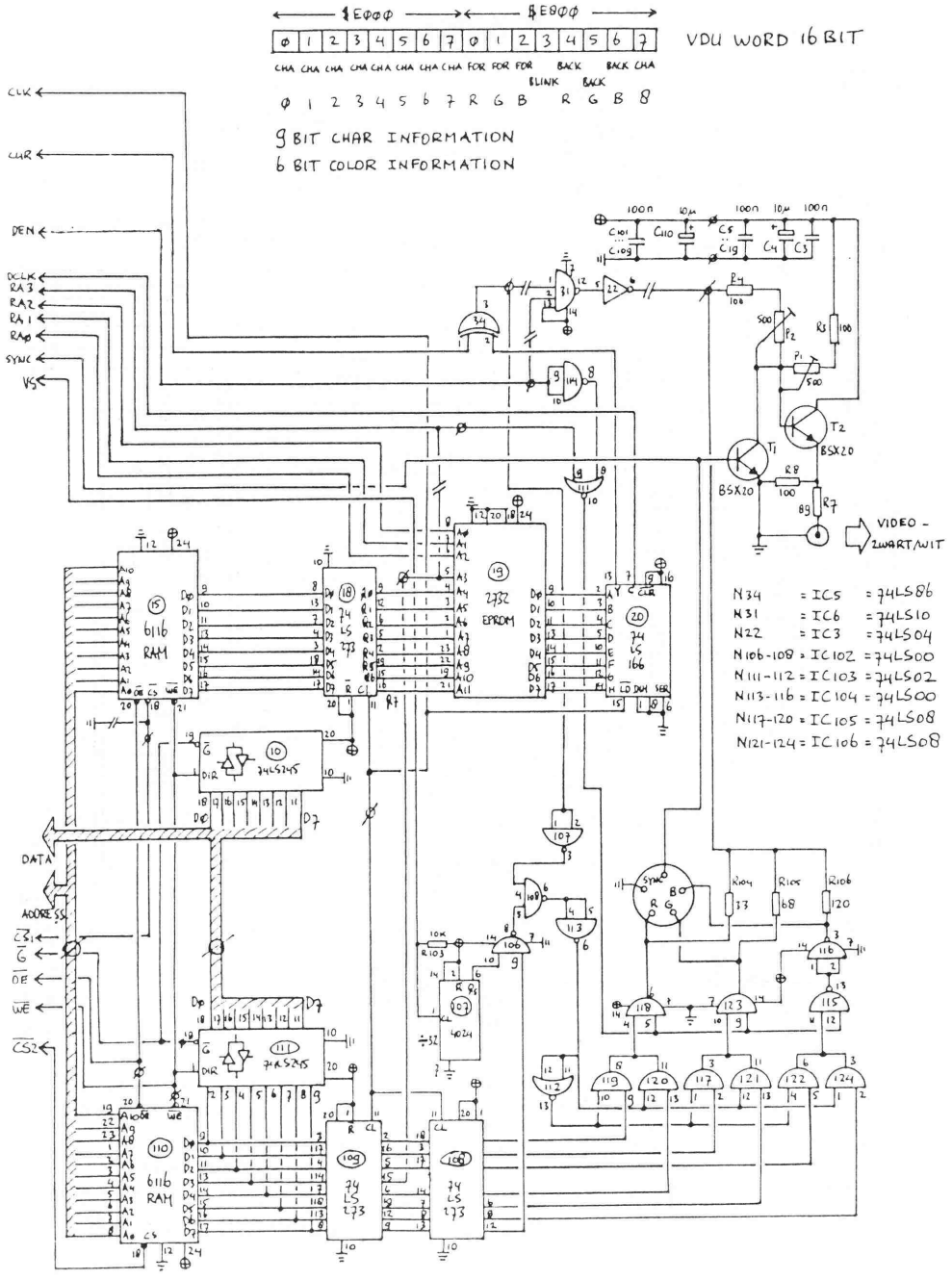
IC 3 pen 10 - IC 21 pen 2	Weerstand R2 - IC 3 pen 1
IC 5 pen 3 - IC 6 pen 1	IC 17 pen 7 - IC 6 pen 2
IC 3 pen 6 - Weerstand R4	IC 11 pen 35 - IC 19 pen 5
IC 15 pen 18 - IC 15 pen 12	IC 11 pen 40 - Weerstand R6



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

COLOR VDU BOARD
FOR JUNIOR COMPUTER

based on Elektuur VDU-board
83082 september 1983.
parts with no's 1xx are added.


$E000
$E800
VDU WORD 16 BIT
CHA CHA CHA CHA CHA CHA CHA CHA FOR FOR FOR BACK BACK CHA
$LINK
BACK
0 1 2 3 4 5 6 7 R G B R G B 8
9 BIT CHAR INFORMATION
6 BIT COLOR INFORMATION
CLK
CHR
DEN
DCLK
RA3
RA2
RA1
RA0
SYNC
VS
100n
10µ
C101 ... C109
C110
C5 ... C19
C4
C3
R4
100
500
P2
R5
100
P1
500
T2
BSX20
T1
BSX20
R8
100
R7
89
VIDEO - ZWART/WIT
6116 RAM
74 LS 273
2732 EPROM
74 LS 166
74LS245
DATA
ADDRESS
CS1
G
DE
WE
CS2
4024
÷32
R103
10K
R104
R105
R106
33
68
120
N34 = IC5 = 74LS86
N31 = IC6 = 74LS10
N22 = IC3 = 74LS04
N106-108 = IC102 = 74LS00
N111-112 = IC103 = 74LS02
N113-116 = IC104 = 74LS00
N117-120 = IC105 = 74LS08
N121-124 = IC106 = 74LS08

Nieuwe verbindingen.

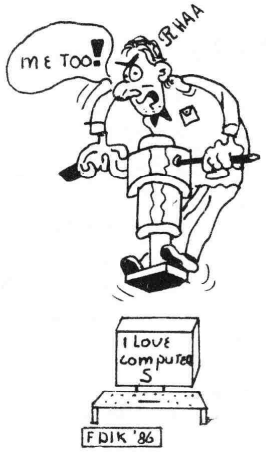
Op VDU-kaart zelf: IC 3 pen 10 - Weesrtand R2
Diode 101 : IC 11 pen 40 (anode) - Weerstand R6 (kathode)

Tussen VDU-kaart en nieuwe uitbreidingsprint:

IC 3 pen 10 - IC 101 pen 2 (X MHz)
IC 10/16 pen 11 - IC 111 pen 11 (D7)
IC 10/16 pen 12 - IC 111 pen 12 (D6)
IC 10/16 pen 13 - IC 111 pen 13 (D5)
IC 10/16 pen 14 - IC 111 pen 14 (D4)
IC 10/16 pen 15 - IC 111 pen 15 (D3)
IC 10/16 pen 16 - IC 111 pen 16 (D2)
IC 10/16 pen 17 - IC 111 pen 17 (D1)
IC 10/16 pen 18 - IC 111 pen 18 (D0)
IC 1 pen 14 - IC 103 pen 2 ($\overline{A11}$)
IC 12 pen 1 - IC 103 pen 3/5 ($\overline{VE}$)
IC 15 pen 18 - IC 103 pen 1 ($\overline{CS1}$)
IC 11 pen 35 - ic 103 pen 9 (RA3)
IC 17 pen 7 - IC 104 pen 9/10 (DEN)
IC 5 pen 3 - IC 102 pen 1/2 (CUR)
Weerstand R4 - Weerstand 104/105/106
IC 5 pen 10 / IC 4 pen 3 / IC 10 pen 19 - IC 111 pen 19 (C)
IC 15 pen 21 / IC 5 pen 5 / IC 4 pen 6 - IC 110 pen 21 ($\overline{WE}$)
IC 15 pen 20 / IC 5 pen 6 - IC 110 pen 20 ($\overline{OE}$)

IC 21 pen 2 - IC 101 pen 3 (DCL)
IC 12 pen 4 - IC 110 pen 19 (BA10)
IC 12 pen 12 - IC 110 pen 22 (BA9)
IC 12 pen 7 - IC 110 pen 23 (BA8)
IC 13 pen 9 - IC 110 pen 1 (BA7)
IC 13 pen 7 - IC 110 pen 2 (BA6)
IC 13 pen 12 - IC 110 pen 3 (BA5)
IC 13 pen 4 - IC 110 pen 4 (BA4)
IC 14 pen 9 - IC 110 pen 5 (BA3)
IC 14 pen 7 - IC 110 pen 6 (BA2)
IC 14 pen 12 - IC 110 pen 7 (BA1)
IC 14 pen 4 - IC 110 pen 8 (BA0)
IC 19 pen 5 - IC 109 pen 15 (CH8)
IC 17 pen 9 - IC 109 pen 11 (CLK)
IC 11 pen 40 - IC 107 pen 1 (VS)
Konnektor 21a- IC103 pen 6 (A11)

```
57 70 MLIST
SCR # 57
  0 ( FORTH 6502 assembler by W.F. Raosdale   July 1980          )
  1 ( Updated for 65C02 by Gert Klein         okt  1984          )
  2
  3 ( This version conforms to the CMOS 6502 CPU manufactured    )
  4 ( by ROCKWELL. You can however use this assembler for the    )
  5 ( SYNERTEK and GTE versions of the 65C02 as well as for the  )
  6 ( NMOS 6502 if you avoid using non existing instructions and)
  7 ( adressing modes.                                           )
  8
  9 ( This assembler is an updated version of the original 6502  )
 10 ( assembler provided through the courtesy of the             )
 11
 12 (           * Forth interest group     *                     )
 13 (           * PO box 1105              *                     )
 14 (           * San Carlos, CA 94070     *                     )
 15 --)


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


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


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



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


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


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


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

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


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


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


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

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

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

OK
```

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

A Cursor for GET

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

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

Fred Behringer, München

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


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


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


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

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


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


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


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

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


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


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


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

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

 OK
```

AVAILABLE FOR YOUR OCTOPUS/EC65 DISKETTE :

DISKETTE 11

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

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

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

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

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

35 entries free out of 64

Depress RETURN to continue ?

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

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

BASICODE-2

1 WAT IS BASICODE-2

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

2 SPECIFICATIES

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

- 5 seconden 2400 Hz,
- STX ($ 02),
- BASIC-info,
- ETX ($ 03),
- checksum,
- 5 seconden 2400 Hz.

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

2.1 PROTOCOL

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

2.1.1 ALGEMENE AFSPRAKEN

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

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

2.1.2 BASIC-STATEMENTS

De volgende commando's en operatoren mogen gebruikt worden:

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

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

2.1.3 KORTE BESCHRIJVING VAN DE STATEMENTS

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

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

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

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

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

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

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

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

END Hiermee wordt het programma gestopt.

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

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

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

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

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

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

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

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

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

LOG(X) Berekent de natuurlijke logaritme van X.

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

NEXT Zie FOR

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

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

OR Logische OR. Zie ook AND

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

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

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

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

RETURN Geeft het einde van een subroutine aan.

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

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

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

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

SQR(X) Berekent de wortel van X.

STEP Zie FOR.

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

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

THEN Zie IF.

TO Zie FOR.

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

2.1.4 STANDAARD ROUTINES

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

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

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

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

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

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

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

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

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

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

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

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

2.1.5 VARIABELEN

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

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

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

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

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

- Stringvariabelen mogen niet langer zijn dan 255 karakters.

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

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

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

2.1.6 DOCUMENTATIE

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

2.2 HOBBYSCOOP

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

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

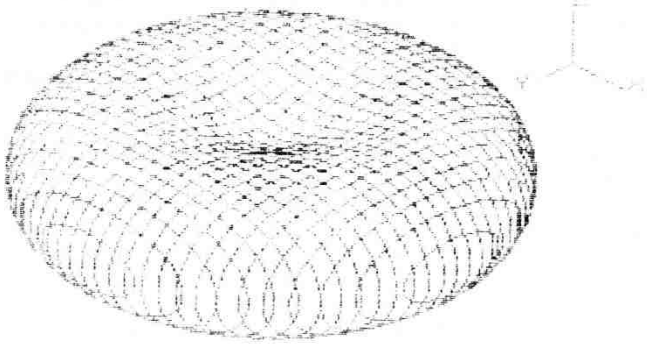
Naast BASICODE-programma's zendt HOBBYSCOOP ook de BASICODE BEELDKRANT uit met veel informatie over computers en de landelijke computer agenda.

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

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

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

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



HET LEZEN VAN BASICODE-2 VOOR DOS-65

1 INLEIDING

Nadat ik mijn JUNIOR omgebouwd had naar een DOS-65-computer, wilde ik nog steeds programma's van het NOS radio-programma HOBBYSCOOP ontvangen. Hiervoor heb ik het BASICODE READ PROGRAM (ELEKTUUR oktober 1983) herschreven. Als cassette-recorder interface gebruik ik het ELEKTUUR C64-cassette-interface. Het programma kan BASICODE pro-gramma's van max 32K ontvangen en opslaan op schijf.

2 GEBRUIK

Nadat de cassette in de recorder geplaatst is kan het programma gestart worden door 'RBC filenaam' in te typen. De computer meldt zich met 'READ BCODE-2'. Nu kan de recorder gestart worden. De data wordt nu ingelezen. Zodra dit gebeurd is, 'piept' de computer ten teken dat er weer gewerkt moet worden. Er kunnen twee fouten gemeld worden:

- OUT OF MEMORY = Het geheugen van de computer is te klein voor het programma,
- CHECKSUM ERROR = De ontvangen check sum is niet gelijk aan de berekende checksum.

Daarna wordt de ontvangen data weggeschreven (SAVE DATA) op de gebruikersschijf onder de opgegeven filenaam.

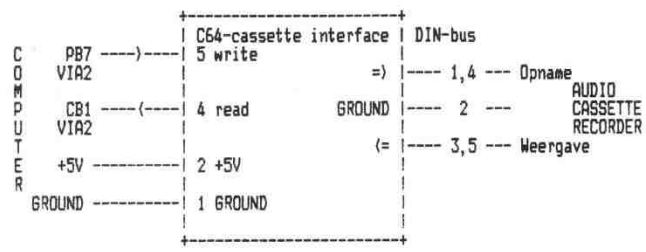
Met behulp van de EDITOR kan de ontvangen ASCII-file aangevuld worden met de standaard subroutines (BCSTRT). Deze nieuwe file kan door BASIC ingelezen worden middels het input redirect. Daarna kan de BASIC-file gesaved worden en kan het programma gerund worden.

3 CASSETTE-INTERFACE

De cassette-interface staat beschreven in de ELEKTUUR van januari 1985. Hiervan wordt alleen het gedeelte voor het lezen en schrijven van data gebruikt. De relaisschakeling voor de motor-sturing wordt niet gebruikt. Ik gebruik een PHILIPS cassette recorder type D 6410 (audio-recorder).

- R13 was 100 Ohm wordt 3300 Ohm,
- R14 was 1500 Ohm wordt 56000 Ohm,
- C6 was 100 nF wordt 56 nF.

Het interface heb ik als volgt met de computer verbonden:



4 ONTVANG-PROGRAMMA

Het READ-BASICODE-2 programma is afgeleid van het BASICODE READ PROGRAM van ELEKTUUR (oktober 1983).

Het oude programma maakte gebruik van de timer in de 6532. Aangezien deze niet standaard aanwezig is in de DOS-65 computer maakt het nieuwe programma gebruik van de timer in de 6522 (VIA-2). Omdat deze geen programmeerbare deelfactor heeft, heb ik het timing gedeelte van het programma zodanig aangepast dat de gemeten tijden door vier gedeeld worden. Hierdoor kan de periode-tijd weer in acht bits bewaard worden.

De machine-code van het programma moet op de systeem-schijf gezet worden onder de naam RBC. De mode van de file moet in de command mode (C) gezet worden, zodat het programma gerund kan worden door het intypen van RBC filenaam.

5 STANDAARD ROUTINES

Na het ontvangen van een Basicode-programma moet het programma nog uitgebreid worden met de standaard routines (BCSTRT). Deze routines maken gebruik van enkele subroutines die niet standaard aanwezig zijn in de DOS-65 computer. Daarom moet het programma BCSUBR ook op de systeemschijf gezet worden.
Om ervoor te zorgen dat de subroutines aanwezig zijn als BASIC geladen wordt, moet dit programma aan BASIC worden toegevoegd, of moet een commando gemaakt worden waarmee eerst BCSUBR geladen wordt en daarna BASIC wordt opgestart.

Voorbeeld: BASICODE (ASCII commando file)

```
LO 0:BCSUBR
BASIC
```

Hierna kan het BASICODE programma geladen en gerund worden.

6 LITERATUUR

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

Elektuur oktober 1983
Basicode-2 pag. 40-43
Basicode-2 voor de junior pag. 57-63

7 PETER LASKER, DRAKENSTEYN 291, 7608 TR ALMELO, 05490 - 72178

```
0005: F427                ORG   $F427
0010:
0015:                ##############################################################
0020:                ##             "B E L L"    for   E C 6 5                  ##
0025:                ##               by Lindstrøm & Rasmussen                 ##
0030:                ##               Parkvej 1, DK-4534 Hørve                 ##
0035:                ##               juli/86                                  ##
0040:                ##############################################################
0045:
0050:                This is a "bell"/chr$(7)/ for ec65. The routine is placed
0055:                in the 'boot-rom' instead of the different screen-parame-
0060:                ters not used in samson. To implement the routine, change
0065:                adrs. $F005..7 to 2027F4.
0070:                                                          +5v
0075:
0080:
0085:
0090:                          via2 / PB7 ->
0095:
0100:                                          2k2
0105:                                                           o
0110:
0115:                PB7 of via 2 is used for output of the tone.  Use a high-
0120:                ohm speaker or a mini-buzzer and add a transistor for lo-
0125:                uder sound (but not too loud! it will get on your nerves)
0130:
0135:                # via 2  adresses #
0140: F427           VBPBD  *      $E110
0145: F427           VBTACH *      VBPBD  +05 timer 1, counter-high
0150: F427           VBTALL *      VBPBD  +06 timer 1, latch low
0155: F427           VBTALH *      VBPBD  +07 timer 1, latch high
0160: F427           VBTBCL *      VBPBD  +08 timer 2, latch low, counter low
0165: F427           VBTBCH *      VBPBD  +09 timer 2, counter high
0170: F427           VBACR  *      VBPBD  +0B auxillary control register
0175: F427           VBIFR  *      VBPBD  +0D interrupt flag register
0180: F427           AHOLD  *      $2363
0185:
0190: F427 AD 63 23          LDA   AHOLD  get character
0195: F42A C9 07             CMPIM $07    is it CHR$(07)?
0200: F42C F0 01             BEQ   BELL   if yes, goto bell
0205: F42E 60                RTS          else return
0210: F42F 48          BELL  PHA          save A
0215: F430 A9 C0             LDAIM $C0    set bit 7,6 "1"->T1:en.PB7 out/fre run
0220: F432 8D 1B E1          STA   VBACR  bit 5 to "0"--> T2: one shot
0225: F435 A9 00             LDAIM $00    timer 1 generates the tone
0230: F437 8D 16 E1          STA   VBTALL
0235: F43A A9 02             LDAIM $02    set tone-frequency to approx. 1000 Hz
0240: F43C 8D 17 E1          STA   VBTALH
0245: F43F 8D 15 E1          STA   VBTACH start the tone
0250: F442 A9 00             LDAIM $00    timer 2 controls duration of the tone
0255: F444 8D 18 E1          STA   VBTBCL
0260: F447 A9 C3             LDAIM $C3    gives a duration of approx. 50 ms.
0265: F449 8D 19 E1          STA   VBTBCH
0270: F44C A9 20             LDAIM $20    timer 2 set bit 5 of IFR
0275: F44E 2C 1D E1    TIMOUT BIT   VBIFR  is time out ?
0280: F451 F0 FB             BEQ   TIMOUT if not wait,
0285: F453 AD 18 E1          LDA   VBTBCL time out, clear timer 2 interrupt flag
0290: F456 A9 00             LDAIM $00
0295: F458 8D 1B E1          STA   VBACR  and stop tone.
0300: F45B 68                PLA          restore A
0305: F45C 60                RTS          and return
```

```
Cassette routines              20-Sep-86 13:23              Page 0

                        ; ***** APPLE COMPATIBLE CASSETTE INTERFACE *****
                        ;
                        ; Pieter de Visser
                        ; Nijverheidslaan 6
                        ; 5506 EE Veldhoven
                        ;
                        ; The system I am using is a DOS65 computer with a couple of chánges; one of
                        ; these is the addition of a cassette interface. The hardware of it is quite
                        ; straightforward; it can also be used for Basicode. I use a 6522 VIA; input
                        ; is on PB7, outout on PB6.
                        ; The original Apple format has some disadvantages; I have improved it on two
                        ; points:
                        ; The header can now be varied in length; by default, I use one of 5 seconds
                        ; (that of the Apple is 10 seconds).
                        ; The read routine searches the start of a next transmission; it does this by
                        ; requiring 256 consecutive 'long 1's'.
                        ; Also, I have included a verify routine; it counts the number of differences
                        ; between a memory block and a cassette data block.
                        ; I have used some 65C02 instructions; it won't be easy to do without them.
                        ; The same applies to adapting the routines to a 2 MHz 65C02.

           00D2                 org     $00d2           8 zero page addresses used

00D2                    chksum  res     1               checksum in read routine
00D3                    lastin  res     1               bit 7 is a copy of last input state read
00D4                    begad   res     2               begin address of memory block
00D6                    endad   res     2               end address of memory block
00D8                    ercnt   res     2               error counter used in verify routine

           0200         header  equ     $0200           length of header must have been stored here
                        ; ($40 is 10 seconds, like Apple's)
           F780         orb     equ     $f780           VIA address; the port must have been
                        ; initialized properly

           2000                 org     $2000

                        ; Cassette write routine
                        ; Write memory block from begad until endad (inclusive) to cassette.

2000 08                 cassw   php
2001 78                         sei                     no interrupts
2002 AD 0002                    lda     header          write header
2005 20 2A20                    jsr     headr
2008 A0 26                      ldy     #$26
200A 52 D4              cassw1  eor     [begad]         update checksum
200C 48                         pha
200D B2 D4                      lda     [begad]         get byte
200F 20 2120                    jsr     wrbyte          write it
2012 20 5420                    jsr     next            next address
2015 A0 1C                      ldy     #$1c
2017 68                         pla
2018 B0 F0                      bcs     cassw1          not yet ready ?
201A A0 1F                      ldy     #$1f
201C 20 2120                    jsr     wrbyte          write checksum
201F 28                         plp
2020 60                         rts

2021 A2 10              wrbyte  ldx     #$10            write A as a byte
2023 0A                 1       asla
2024 20 3720                    jsr     wrbit           write one bit
2027 D0 FA                      bne     1.b             not yet written 8 bits ?
2029 60                         rts

202A A0 49              headr   ldy     #$49            write header
```

```
assette routines                20-Sep-86 13:23                Page 1

02C 20 3C20             jsr     zerdly          A * 256 long 1's
02F D0 F9               bne     headr
031 69 FE               adc     #$fe
033 B0 F5               bcs     headr
035 A0 1F               ldy     #$1f            and a short 0
037 20 3C20     wrbit   jsr     zerdly          write bit
03A C8                  iny
03B C8                  iny
03C 88          zerdly  dey                     write half bit
03D D0 FD               bne     zerdly
03F 90 05               bcc     2.f             zero bit ?
041 A0 32               ldy     #$32            delay for half '1' bit
043 88          1       dey
044 D0 FD               bne     1.b
046 A8          2       tay                     toggle output
047 AD 80F7             lda     orb
04A 49 40               eor     #$40
04C 8D 80F7             sta     orb
04F 98                  tya
050 A0 2A               ldy     #$2a
052 CA                  dex
053 60                  rts

054 E6 D4       next    inc     begad           increment begad
056 D0 02               bne     1.f
058 E6 D5               inc     begad+1
05A A5 D6       1       lda     endad           compare with endad
05C C5 D4               cmp     begad
05E A5 D7               lda     endad+1
060 E5 D5               sbc     begad+1
062 60                  rts                     C = (not ready)

                ; Cassette read routine
                ; Read into memory block from begad until endad (inclusive).
                ; Afterwards, C = 1 indicates error.

063 08          cassr   php
064 78                  sei                     no interruots
065 A9 FF               lda     #$ff            initialise checksum
067 85 D2               sta     chksum
069 20 A920             jsr     findst          find start of transmission
06C A0 3A               ldy     #$3a
06E 20 BF20     cassr1  jsr     rdbyte          read byte
071 92 D4               sta     [begad]
073 45 D2               eor     chksum          adapt chksum
075 85 D2               sta     chksum
077 A0 35               ldy     #$35
079 20 5420             jsr     next            next address
07C B0 F0               bcs     cassr1          not yet ready ?
07E 20 BF20             jsr     rdbyte          read checksum
081 38                  sec
082 E5 D2               sbc     chksum          A should be 0 now
084 28                  plp
085 C9 01               cmp     #$01            else C becomes 1
087 60                  rts

                ; Cassette verify routine
                ; Compare transmission with memory block from begad until endad (inclusive).
                ; Afterwards, ercnt holds number of differences.
                ; The checksum is not checked.

088 08          cassv   php
089 78                  sei                     no interrupts
08A 64 D8               stz     ercnt           no errors yet
08C 64 D9               stz     ercnt+1
08E 20 A920             jsr     findst          find start of transmission
```

```
0010 ;
0020 ;##########################################
0030 ;##########         RBC          ##########
0040 ;##########                      ##########
0050 ;##########    READ BASICODE-2   ##########
0060 ;##########    DATE 07-03-86     ##########
0070 ;##########                      ##########
0080 ;##########    P.LASKER          ##########
0090 ;##########    DRAKENSTEYN 291   ##########
0100 ;##########    7608 TR ALMELO    ##########
0110 ;##########    05494-72178       ##########
0120 ;##########                      ##########
0130 ;##########################################
0140 ;
0150 ;
0160 ; BASICODE READ PROGRAM FOR DOS-65 COMPUTER
0170 ; MAX 32K $2000 - $9FFF
0180 ;
0190 ; THE DATA FORMAT ON TAPE IS ASCII:
0200 ; 5 SEC 2400 HZ<>STX<>DATA<> / <>DATA<>ETX<>CHECKSUM<>5 SEC 2400 HZ
0210 ; MOST SIGNIFICANT ASCII-BIT IS "1".
0220 ; 1 = 2 * 2400 HZ
0230 ; 0 = 1 * 1200 HZ
0240 ; THE TRANSPORT RATE OF THE DATA IS 1200 BAUD
0250 ; BYTE: 1 STARTBIT (0) <> 8*DATA <> 2 STOPBITS (1)
0260 ;
0270           .OS
0280           .CE
0290           .ES
0300           .BA $C800
0310           .MC $C800
0320 ;
0330 ;
0340 ;**** PIA FLOPPY-DISK ****
0350 ;
0360 PBD       .DE $C002    ;DATA + DATADIRECTION B
0370 ;
0380 ;
0390 ;**** 6522 VIA-2 REGISTERS ****
0400 ;
0410 VIA2      .DE $C110
0420 T1CL      .DE VIA2+$04        ;T1, LATCH LOW, COUNTER LOW
0430 T1CH      .DE VIA2+$05        ;T1 COUNTER HIGH
0440 ACR       .DE VIA2+$0B        ;AUXILIARY CONTROL REGISTER
0450 PCR       .DE VIA2+$0C        ;PERIPHERAL CONTROL REGISTER
0460 IFR       .DE VIA2+$0D        ;INTERRUPT FLAG REGISTER
0470 IER       .DE VIA2+$0E        ;INTERRUPT ENABLE REGISTER
0480 ;
0490 ;
0500 ;**** TEMPORARY DATA BUFFERS ****
0510 ;
0520 CHSUM     .DE $CA00    ;CHECKSUM
0530 X         .DE CHSUM
0540 PRCNTL    .DE X+$01    ;PERIOD COUNTER
0550 PRCNTH    .DE X+$02
0560 ZERO      .DE X+$03    ;PERIOD-TIME
0570 HLFPTM    .DE X+$04    ;HALF PERIOD-TIME
0580 SIVL      .DE X+$05
0590 SIVH      .DE X+$06
0600 TIMECNTL  .DE X+$07    ;TIMER
0610 TIMECNTH  .DE X+$08
0620 SAVEACCU  .DE X+$09    ;SAVE ACCU
0630 SAVEX     .DE X+$0A    ;SAVE X-REG
0640 SAVEY     .DE X+$0B    ;SAVE Y-REG
0650 RAMSTARTL .DE X+$0C    ;STARTADDRESS RAM
0660 RAMSTARTH .DE X+$0D
0670 LENGTHL   .DE X+$0E    ;LENGTH OF FILE
0680 LENGTHH   .DE X+$0F
0690 ;
0700 ;
0710 ;**** EXTERNAL SUBROUTINE ****
0720 ;
0730 PRCHA     .DE $E000    ;PRINT CHARACTER ROUTINE
0740 KBIT      .DE $E144    ;INIT KEYBORD
0750 PIEPER    .DE $EC87    ;BUZZER
0760 ;
0770 ;
0780 ;**** DOS SUBROUTINES ****
0790 ;
0800 WRITE1    .DE $B027
0810 CREATE1   .DE $B039
0820 CLOSE1    .DE $B04B
0830 ERMES     .DE $B0B7
0840 ;
0850 ;
0860 ;##########################################################
0870 ;***** START *****
0880 ;##########################################################
0890           STA SAVEACCU        ;SAVE FILE-NAME POINTER
0900           STY SAVEY
0910 ;
0920 ;***** CREATE FILE *****
0930 ;
C806- A2 E1    0940           LDX #$E1     ;CREATE FILE R+W+D+ASCII
C808- 20 39 B0 0950           JSR CREATE1
C80B- B0 14    0960           BCS BUERA    ;ERROR
C80D- 8E 0A CA 0970           STX SAVEX    ;SAVE FILE NR
               0980 ;
C810- A9 17    0990           LDA #$17     ;DESELECT DRIVES
C812- 0D 02 C0 1000           ORA PBD
C815- 8D 02 C0 1010           STA PBD
               1020 ;
C818- 78       1030           SEI  ;NO INTERRUPTS
C819- A0 12    1040           LDY #$12
C81B- 20 1F C9 1050           JSR MESSY    ;PRINT 'READ BCODE-2'
C81E- 4C 24 C8 1060           JMP READ
               1070 ;
C821- 4C BE C9 1080 BUERA     JMP BUER
               1090 ;
               1100 ;##############################################
               1110 ;**** INITIALISE AND RECEIVE ****
               1120 ;##############################################
               1130 ;
C824- A9 7F    1140 READ      LDA #$7F
C826- 8D 1E C1 1150           STA IER      ;SET INTERRUPT DISABLE MODE
C829- A9 3F    1160           LDA #$3F     ;PB7 DISABLED, FREE-RUN DISABLED
C82B- 2D 1B C1 1170           AND ACR
C82E- 8D 1B C1 1180           STA ACR
C831- A9 00    1190           LDA #$00
C833- 8D 1C C1 1200           STA PCR      ;SET CB1 NEGTIVE EDGE DETECT
C836- 8D 00 CA 1210           STA CHSUM    ;CLEAR CHECKSUM
C839- 8D 0C CA 1220           STA RAMSTARTL       ;SET RAM START ADDRESS $2000
C83C- 8D 74 C8 1230           STA STAIND+$01      ;AND SET RAM POINTER
C83F- A9 20    1240           LDA #$20
C841- 8D 0D CA 1250           STA RAMSTARTH
C844- 8D 75 C8 1260           STA STAIND+$02
               1270 ;
               1280 ;**** HEADER ****
               1290 ;
C847- A9 10    1300 HEADER    LDA #$10
C849- 8D 02 CA 1310           STA PRCNTH   ;SET PERIOD COUNTER
C84C- 20 B8 C8 1320 HDR       JSR PERIOD
C84F- C9 40    1330           CMP #$40
C851- 90 F4    1340           BCC HEADER   ;PERIODE <<2400HZ
C853- C9 88    1350           CMP #$88
C855- B0 F0    1360           BCS HEADER   ;PERIODE >>2400HZ
C857- EE 01 CA 1370           INC PRCNTL
C85A- D0 F0    1380           BNE HDR      ;NOT 256 PERIODES
C85C- CE 02 CA 1390           DEC PRCNTH
C85F- D0 EB    1400           BNE HDR      ;NOT 16*256=PERIODES OF 2400HZ
C861- 0A       1410           ASL A        ;PERIODETIME *2 (=1200HZ)
C862- 38       1420           SEC
C863- E9 19    1430           SBC #$19     ;1200HZ TIME-100US=ZERO
C865- 8D 03 CA 1440           STA ZERO
               1450 ;
               1460 ;**** START-BIT ****
               1470 ;
C868- 20 BB C8 1480 STARTBIT  JSR HLFPER   ;FIND STARTBIT
C86B- CD 03 CA 1490           CMP ZERO
C86E- 90 F8    1500           BCC STARTBIT        ;NO 1200HZ THEN BRANCH
               1510 ;
               1520 ;**** READ BYTE ****
               1530 ;
C870- 20 FD C8 1540           JSR RBYT     ;READ ONE BYTE
C873- 8D FF FF 1550 STAIND    STA $FFFF    ;CHARACTER TO TABLE
C876- C9 03    1560           CMP #$03
C878- F0 1E    1570           BEQ EOT      ;END OF TEXT ?
C87A- EE 74 C8 1580           INC STAIND+$01      ;INCREMENT POINTER
C87D- D0 03    1590           BNE EORAM
C87F- EE 75 C8 1600           INC STAIND+$02
               1610 ;
               1620 ;**** CHECK END OF RAM ****
               1630 ;
C882- A9 9F    1640 EORAM     LDA #$9F     ;RAM MAX $9FFF
C884- CD 75 C8 1650           CMP STAIND+$02
C887- D0 DF    1660           BNE STARTBIT        ;NOT END OF RAM ?
C889- A9 FF    1670           LDA #$FF
C88B- CD 74 C8 1680           CMP STAIND+$01
C88E- D0 D8    1690           BNE STARTBIT
               1700 ;
               1710 ;**** END OF RAM ****
C890- A0 00    1720           LDY #$00
C892- 20 1F C9 1730           JSR MESSY    ;PRINT 'OUT OF MEMORY'
C895- 4C 84 C9 1740           JMP SAVERDATA
               1750 ;
               1760 ;**** END OF TEXT, GET CHEKSUM ****
               1770 ;
C898- 20 BB C8 1780 EOT       JSR HLFPER
C89B- CD 03 CA 1790           CMP ZERO
C89E- 90 F8    1800           BCC EOT
```

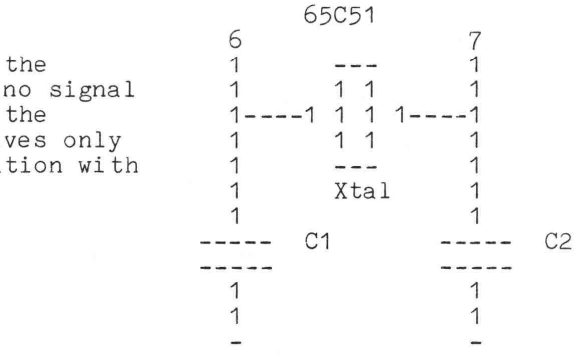
```
C8A0- 20 FD C8  1810          JSR RBYT    ;READ CHEKSUM
C8A3- AD 00 CA  1820          LDA CHSUM
C8A6- F0 08     1830          BEQ OKE
C8A8- A0 31     1840          LDY #$31
C8AA- 20 1F C9  1850          JSR MESSY   ;PRINT 'CHECKSUM ERROR'
C8AD- 4C 84 C9  1860          JMP SAVERDATA
C8B0- A0 44     1870 OKE      LDY #$44
C8B2- 20 1F C9  1880          JSR MESSY   ;PRINT 'DATA RECEIVED'
C8B5- 4C 84 C9  1890          JMP SAVERDATA
                1900 ;
                1910 ;*************************************************
                1920 ;***** MEASURE ONE PERIODTIME *****
                1930 ;*************************************************
                1940 ;
C8B8- 20 BB C8  1950 PERIOD   JSR HLFPER
C8BB- A9 10     1960 HLFPER   LDA #$10
C8BD- 2C 1D C1  1970 HLF      BIT IFR
C8C0- F0 FB     1980          BEQ HLF     ;NO ACTIVE EDGE THEN BRANCH
C8C2- 8D 1D C1  1990          STA IFR     ;CLEAR CB1 FLAG
C8C5- A9 10     2000          LDA #$10
C8C7- 4D 1C C1  2010          EOR PCR
C8CA- 8D 1C C1  2020          STA PCR     ;OPPOSITE ACTIVE CB1 EDGE DETECT
C8CD- A9 FF     2030          LDA #$FF
C8CF- AA        2040          TAX
C8D0- 4D 15 C1  2050          EOR T1CH    ;GET ELAPSED TIME
C8D3- 8D 08 CA  2060          STA TIMECNTH
C8D6- 8A        2070          TXA
C8D7- 4D 14 C1  2080          EOR T1CL
C8DA- 8D 07 CA  2090          STA TIMECNTL
C8DD- A9 FF     2100          LDA #$FF
C8DF- 8D 14 C1  2110          STA T1CL    ;RESET TIMER
C8E2- 8D 15 C1  2120          STA T1CH
C8E5- 4E 08 CA  2130          LSR TIMECNTH        ;DIVIDE TIME BY 4
C8E8- 6E 07 CA  2140          ROR TIMECNTL        ;AND PUT IT IN TIMECNTL
C8EB- 4E 08 CA  2150          LSR TIMECNTH
C8EE- 6E 07 CA  2160          ROR TIMECNTL
C8F1- AD 07 CA  2170          LDA TIMECNTL
C8F4- AA        2180          TAX
C8F5- 18        2190          CLC
C8F6- 6D 04 CA  2200          ADC HLFPTM  ;FULL PERIOD T IN ACCU
C8F9- 8E 04 CA  2210          STX HLFPTM  ;SAVE HALF PERIOD T
C8FC- 60        2220          RTS
                2230 ;
                2240 ;*************************************************
                2250 ;***** READ ONE BYTE *****
                2260 ;*************************************************
                2270 ;
C8FD- A0 08     2280 RBYT     LDY #$08    ;SET BIT COUNTER
C8FF- 48        2290 RB       PHA
C900- 20 B8 C8  2300          JSR PERIOD
C903- CD 03 CA  2310          CMP ZERO
C906- B0 06     2320          BCS FNDZRO  ;1200HZ?
C908- 20 B8 C8  2330          JSR PERIOD  ;SECOND 2400HZ PERIOD ?
C90B- 38        2340          SEC
C90C- B0 01     2350          BCS SHIFT   ;BRANCH ALWAYS
C90E- 18        2360 FNDZRO   CLC
C90F- 68        2370 SHIFT    PLA
C910- 6A        2380          ROR A       ;NEXT BIT TO ACCU
C911- 88        2390          DEY
C912- D0 EB     2400          BNE RB      ;NOT 8 BITS ?
C914- 48        2410          PHA
C915- 4D 00 CA  2420          EOR CHSUM
C918- 8D 00 CA  2430          STA CHSUM   ;UPDATE CHEKSUM
C91B- 68        2440          PLA
C91C- 29 7F     2450          AND #$7F    ;CLEAR BIT 7
C91E- 60        2460          RTS
                2470 ;
                2480 ;*************************************************
                2490 ;***** OUTPUT MESSAGE *****
                2500 ;*************************************************
                2510 ;
C91F- B9 2E C9  2520 MESSY    LDA MESS,Y  ;LOAD CHARACTER
C922- C9 03     2530          CMP #$03
C924- F0 07     2540          BEQ MESEND  ;END OF TEXT ?
C926- 20 00 E0  2550          JSR PRCHA
C929- C8        2560          INY
C92A- 4C 1F C9  2570          JMP MESSY
C92D- 60        2580 MESEND   RTS
                2590 ;
                2600 ;**** MESSAGES ****
                2610 ;
C92E- 0D 0A 4F  2620 MESS     .BY $0D $0A 'OUT OF MEMORY' $0A $0D $03
C931- 55 54 20
C934- 4F 46 20
C937- 4D 45 4D
C93A- 4F 52 59
C93D- 0A 0D 03
C940- 0D 0A 52  2630          .BY $0D $0A 'READ BCODE-2' $0A $0D $03
C943-
C946- 20 42 43
C949- 4F 44 45
C94C- 20 32 0A
C94F- 0D 03
C951- 0D 0A 53  2640          .BY $0D $0A 'SAVE DATA' $0A $0D $03
C954- 41 56 45
C957- 20 44 41
C95A- 54 41 0A
C95D- 0D 03
C95F- 0D 0A 43  2650          .BY $0D $0A 'CHECKSUM ERROR' $0A $0D $03
C962- 48 45 43
C965- 4B 53 55
C968- 4D 20 45
C96B- 52 52 4F
C96E- 52 0A 0D
C971- 03
C972- 0D 0A 44  2660          .BY $0D $0A 'DATA RECEIVED' $0A $0D $03
C975- 41 54 41
C978- 20 52 45
C97B- 43 45 49
C97E- 56 45 44
C981- 0A 0D 03
                2670 ;
                2680 ;*************************************************
                2690 ;***** SAVE RECEIVED DATA ****
                2700 ;*************************************************
                2710 ;
C984- 20 44 E1  2720 SAVERDATA JSR KBIT     ;INIT. KEYBORD
C987- 20 87 EC  2730          JSR PIEPER
C98A- A0 23     2740          LDY #$23
C98C- 20 1F C9  2750          JSR MESSY     ;PRINT 'SAVE DATA'
C98F- 58        2760          CLI  ;INTERRUPT ENABLE
                2770 ;
                2780 ;**** SET POINTERS TO DATA ****
                2790 ;
C990- 38        2800          SEC  ;CALCULATE LENGTH OF FILE
C991- AD 75 C8  2810          LDA STAIND+$02      ;END OF DATA HIGH
C994- ED 0D CA  2820          SBC RAMSTARTH
C997- 8D 0F CA  2830          STA LENGTHH  ;DATA LENGTHH VALUE
C99A- AD 74 C8  2840          LDA STAIND+$01      ;END OF DATA LOW
C99D- ED 0C CA  2850          SBC RAMSTARTL
C9A0- 8D 0E CA  2860          STA LENGTHL  ;DATA LENTHL VALUE
C9A3- A9 CA     2870          LDA #H,RAMSTARTH     ;SET FILE DATA POINTER
C9A5- A0 0C     2880          LDY #L,RAMSTARTL
                2890 ;
                2900 ;**** WRITE DATA ****
                2910 ;
C9A7- AE 0A CA  2920          LDX SAVEX   ;GET FILE NR
C9AA- 20 27 B0  2930          JSR WRITE1  ;WRITE DATA TO FILE
C9AD- B0 03     2940          BCS BUERC   ;ERROR
                2950 ;
C9AF- 4C 4B B0  2960 EOWRITE  JMP CLOSE1
                2970 ;
C9B2- AC 0B CA  2980 BUERC    LDY SAVEY   ;RESTORE FILE-NAME POINTER
C9B5- AD 09 CA  2990          LDA SAVEACCU
C9B8- 20 B7 B0  3000          JSR ERMES   ;WRITE ERROR
C9BB- 4C 4B B0  3010          JMP CLOSE1
                3020 ;
C9BE- 4C B7 B0  3030 BUER     JMP ERMES   ;CREATE ERROR
                3040 ;
                3050          .EN

                0010 ;
                0020 ;***********************************************
                0030 ;**********          BCSUBR           **********
                0040 ;**********                           **********
                0050 ;**********   BASICODE-2 SUBROUTINES  **********
                0060 ;**********   DATE 11-04-86           **********
                0070 ;**********                           **********
                0080 ;**********      P.LASKER             **********
                0090 ;**********      DRAKENSTEYN 291      **********
                0100 ;**********      7608 TR ALMELO       **********
                0110 ;**********      05490-72178          **********
                0120 ;**********                           **********
                0130 ;***********************************************
                0140 ;
                0150 ;
                0160          .OS
                0170          .CE
                0180          .BA $C800
                0190          .MC $C800
                0200 ;
                0210 ;**** MONITOR SUBROUTINES****
                0220 ;
                0230 GETASKEY  .DE $E00C   ;GET A KEY
                0240 PRTEXTSTR .DE $E00F   ;PRINT A TEXTSTRING, LAST CHAR. =$00
                0250 GETEVTAS  .DE $E012   ;GET A KEY IF THERE IS ONE, OTHERWISE A=$00
                0260 ;
                0270 ;
                0280 ;**** MONITOR VARIABLES ****
                0290 ;
                0300 DEVMODOUT .DE $CE12   ;OUTPUT DEVICE MODE BITS
                0310 ;
```

****
* **SUBJECT: ACIA 65c51 and MODEMS** *
* **To: all 6551 users** *
* **From: Bram de Bruine, The Netherlands. (6-1-87)** *
****

ACIA PROBLEM

I have had a problem with my Acia. The baudgenerator would n't run continuously. Sometimes he did not oscillate at all!

The problem of shut off of the internal oscillator 65C51 (no signal on pin 5 RXC) is solved by the scheme next. The problem lives only by CMOS versions in combination with a certain X-tal.
C1 from pin 6 to ground
C2 from pin 7 to ground
C1=6pF
C2=6pF

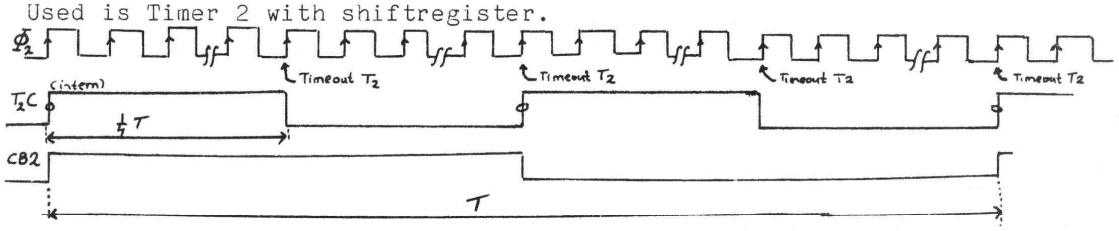


SPLIT SPEED WITH ACIA

Modems are getting more popular these days. Many databanks uses the split baudrate of 1200/75. The Acia can handle differend receive/transmit speeds, if bit 4 of the controlregister is zero, and a clocksignal for the receiver is present on pin 5 of the acia. This clocksignal is generated by timer2 of a VIA. (6522). Timer 2 is programmed in the free running mode. (oscillator)

PROGRAMMING A VIA AS AN OSCILLATOR

Used is Timer 2 with shiftregister.



↓ = decrement counter
ϕ = shift one bit out of SR to CB2

Every clockcycle (if systemclock = 1 Mhz, one clockcycle = 1 uS) the T2 counter is decreased by one. After each timeout of T2 the internal shiftclock (T2C) is inverted. On a rising edge of this shiftclock the programmed bitpattern is clocked out of the shiftregister. (CB2) After every shift b7 becomes b0. By use of a bitpattern of 01010101 ($55) in the shiftregister, the T2 counter must be loaded with T/4. (see diagram) Also the shiftregister can be used as a divider. (f.e. 00001111 [$0F] adds a divisionfactor of 4)

GENERATION 75 BAUD CLOCK

clock = 16 * 75 = 1200 Hz T = 1/f = 833 uS.
Mode ACR432 = 100 (free running)
T2CL = 833/4 = 208 uS
bitpattern shiftreg = $55 or $OF with T2CL = 208/4 = 52 uS.

```
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;Program name: SPLIT.MAC
;Date: 2 NOV 1986
;Programmer: Bram de Bruine
;Program requirements: $12 bytes
;Hardware: DOS 65
;Purpose: SPEEDCONVERSION 1200/75 - 75/1200 MODEM--)ACIA
;Program description:CB2 VIA2 IS CLOCKOUTPUT AND MUST BE
;CONNECTED WITH RXC PEN 5 ACIA
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  1000          ORG     $1000

  0010  VIAMODE EQU     %00010000       ;Free running t2
  000F  CLOUD1  EQU     %00001111       ;Symmetric output clock
  0055  CLOUD2  EQU     %01010101       ;Symmetric output clock div by 4

        ;Oscillatorfrequencies dependend on systemclock
  0032  T1200   EQU     52-2            ;52*4=208 us   1Mhz
  0008  T19K2   EQU     13-2            ;13 us
  0066  T12002  EQU     104-2           ;values for 2Mhz (double)
  0018  T19K22  EQU     26-2

        ;Via registers
  E11B  ACR2    EQU     $E11B
  E11A  SHIFT   EQU     $E11A
  E118  T2CL    EQU     $E118

1000 A9 10      OSC75   LDA     #VIAMODE
1002 8D 1BE1            STA     ACR2            ;T2 is osc.
1005 A5 32              LDA     T1200           ;75 baud (*16)
1007 8D 18E1            STA     T2CL            ;Load timer
100A A9 OF              LDA     #CLOUD1
100C 8D 1AE1            STA     SHIFT           ;Bit pattern/division factor
100F AD 1AE1            LDA     SHIFT           ;Start oscillator
1012 60                 RTS
Errors detected: 0
```

GENERATION 1200 BAUD CLOCK

clock = 16 * 1200 = 19.2 kc/s. --) T = 52 uS T2CL = 52/4 = 13.
OSC1200: Same as OSC75, but now T19K2 and CLOUD2 is programmed. Unfortunately DOS-65 used the same output for the beep! signal. So turn down volume, or disconnect the beep part. For systems with a systemclock of 2 Mhz, use the values T12002/T19k22.

The only hardware modification is a wire from acia to via. (use a switch or a (modem)connector as a switch: RS232 pin 17 RxC ACIA
pin 18 CB2 VIA2
In the modem-connector pin 17 must be connected with pin 18.

==========END==========

=+

SOLVED PROBLEMS ELEKTOR'S OCTOPUS/EC65 COMPUTER

By: Frank Vandekerkhove, Belgium.

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

PROBLEMS DOS65 COMPUTER

By: Andrew Gregory, England

I think I have discovered a bug in the DOS65 copy utility.
If you type:

```
ASN U=@
COPY -W FILENAME 0:
```

it works perfectly. The problem comes when copying from a subdirectory:

```
COPY -W F/FILENAME 0:
        ↑
        any sybdirectory
```

The file is copied correctly, but then it copies track 0 sector 1 from source to destination disc. The destination disc becomes corrupt. A DIR command shows that the destination disc has assumed the name of the source disc. I could not find the bug in the source listing.

=+

```
list video.h
/*************************************************************************/
/*                                                                       */
/*      File: video.h                                                    */
/*-----------------------------------------------------------------------*/
/*      Small C video control routines for I065/DOS65 system             */
/*              written by A.Megens  spring 1987                         */
/*                                                                       */
/*************************************************************************/

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

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

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

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

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

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

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

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

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


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

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

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

  #asm
        ldc.u 0          ;get first argument (small C macro)
        pha              ;save on stack
        ldc.u 2          ;get second argument
        tay              ;in Y-reg.
        pla              ;get stack-value
        tax              ;in X-reg.
        jsr $F024        ;I065 routine (goto screen coordinate X,Y)
  #endasm

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

```
/*************************************************************************/
/*                                                                       */
/* square(x,y,width,height)      - draw a square width,height with upper */
/*                                 left corner at coordinate x,y         */
/* The function is aborted with return value Ø when the square does not  */
/* fit on the screen. Else return value is 1.                            */
/*                                                                       */
/*************************************************************************/

square(x,y,width,height)
int x,y,width,height;
{ int i;
  if ((x<1)!(width<1)!((x+width)>79)!(y<1)!(height<1)!((y+height)>24))
        return(Ø);  /* return Ø if error in arguments, function aborted */

  grafon();gotoxy(x,y);
  printf("P");for (i=1;i<width;i++) printf("X");          /* draw topline */
  printf("V");
  for (i=1;i<height;i++)
  {     gotoxy(x,y+i);printf("Y");                       /* draw sides */
        gotoxy(x+width,y+i);printf("Y");
  }
  gotoxy(x,y+height);printf("R");
  for (i=1;i<width;i++) printf("X");                      /* draw bottom */
  printf("T");grafoff();

  return(1); /* no error, return 1 */
}
$
$
$ list demo.c
/*************************************************************/
/*                                                           */
/*      File: demo.c                                         */
/*-----------------------------------------------------------*/
/*              Demo program for video routines in video.h   */
/*                                                           */
/*************************************************************/

#include video.h        /* get library functions */

main()
{ int i;
  char c;

  cls();
  invers();printf("\n\t\t\tDEMO SMALL C VIDEO ROUTINES\n\n");normal();

  printf("\tNew routines: cls(), invers(), normal(), grafon(), grafoff(),");
  printf("home(),\n");
  printf("\tbell(), crlf(), xcrlf(n), gotoxy(x,y), square(x,y,width,height).");
  xcrlf(2);
  printf("\tnormal mode: @ A B C D E F ^ ? P Q R S T U V W X Y Z\n");
  printf("\tgrafic mode: ");
  grafon();printf("@ A B C D E F ^ ? P Q R S T U V W X Y Z\n");grafoff();

  square(1,1,78,23); /* maximum width and height for square function */
  square(65,7,1,1);  /* smallest possible square (width=1, height=1) */
  square(5,6,65,3);

  /* square is not drawn when it does not fit on screen */
  i=1;
  while (square(3Ø-3*i,1Ø+i,2*i+2,i+1)) i++; /* return value then Ø else 1 */
  i=1;
  while (square(3Ø+3*i,1Ø+i,2*i+2,i+1)) i++;

  grafon();gotoxy(75,21);printf("PWV"); /* grafic signature */
  gotoxy(75,22);printf("QUY");grafoff();

  gotoxy(45,11);printf("Press a key to exit.");
  bell();c=getchar();
  cls();
}
```

```
**************   D O S - 6 5   V2.01   ***************
*
* Zet printerlettermode commando voor de printer
* NMS-1431 van PHILIPS.
*
*****************************************************
```

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

Beschrijving van het printerinstelprogramma "SETPRINT"

Dit programma biedt de mogelijkheid om de printer NMS-1431 van PHILIPS een bepaalde voorinstelling te geven wat betreft:
- Letterkeuze.
- Kantlijn.
- Papierlengte.
- Opgaven printfile.

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

De commandoprocedure is opgezet volgens de DOS-65 stijl.

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

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

Letterkeuze.

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

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

Wanneer de letterkeuze niet opgegeven wordt dan is de defaultwaarde 'PICA' in de standaardmode.
Met de optie -K kan opgegeven worden of er over de perforatie van het papier heen geprint moet worden of niet. De defaultwaarde is: over de

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

Kantlijn instelling.

-m Met de eerste parameter kan de linkerkantlijn worden ingesteld. De defaultwaarde voor de kantlijn is 7. Deze waarde is aan te passen door in de printercommado TABEL1 het getal (L007) aan te passen.

-n De tweede parameter stelt de bladzijdelengte in. Default is deze instelling 12 insch. De opgave moet in insches geschieden.

-o De derde parameter geeft zoals gezegt aan hoeveel lege regels er gesprongen moet worden over een perforatie of bij een nieuwe bladzijde.

Filenaam.

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

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

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

Wanneer de optie -H wordt meegegeven met het commando geeft de computer een helpscherm weer. Hierna keert de computer terug in de commandomode op DOS niveau.
Alle printercommando-karakters staan als label gedeclareerd, zodat het gemakkelijker is om deze routine aan te passen voor een andere printer. In TABEL1 in de source staat de printer reset karakters.
De resetprocedure bestaat uit de volgende handelingen:

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

```
ijd  8-Apr-87 22:29  === file setprint.mac ===    pagina 1

                ;
                ; ***************   D O S - 6 5  V2.01   ****************
                ; *
                ; * Set printercharactermode command for the printer
                ; * NMS-1431 of PHILIPS.
                ; *
                ; ***********************************************************
                ;
                ;
                ;           H.A.J.Quast
                ;           Dekemastate 15
                ;           1275 CM Huizen N.H.
                ;           tel. 02152-54905
                ;
                ; Command: SETPRINT [-HECPDLIK +m,n,o] FILE
                ;
                ; Page zero address
                ;--------------------------------------
          00A0          ORG     $00A0
0A0             OPTMASK RES     1               Option mask
0A1             POINTER RES     2               Commandbuffer pointer
0A3             LEFTMAR RES     2               Pointer for left margin
0A5             PAGELEN RES     2               Pointer for page length
0A7             JUMPPER RES     2               Number of jump by skip
0A9             TEMPX   RES     1               Save address X-reg.
0AA             FILEIN  RES     1               Inputfilenumber
                ;
                ; System subroutine's
                ;--------------------------------------
                ;
          F006  OUTDEV  EQU     $F006           Outputdevice
          F00C  INITDEV EQU     $F00C           Init. outputdevice
          C03B  PRINT1  EQU     $C03B           Print text on screen
          C056  REDRIN  EQU     $C056           Redirect input
          C068  SOPT1   EQU     $C068           Get option
          C06B  SPAR1   EQU     $C06B           Get parameter
          D003  SREAD1  EQU     $D003           Single read out file
          D0B7  ERMES1  EQU     $D0B7           Error messages
                ;
                ; Mask for option-code
                ;
          0080  HELP    EQU     %10000000
          0040  ELITE   EQU     %01000000
          0020  CONDE   EQU     %00100000
          0010  PROPO   EQU     %00010000
          0008  DOUBL   EQU     %00001000
          0004  LETQU   EQU     %00000100
          0002  ITALI   EQU     %00000010
          0001  PERFO    EQU     %00000001
                ;
          0045  C.ELITE EQU     'E              Printercode elite charactertype
          0051  C.CONDE EQU     'Q              Printercode condensed charactertype
          0050  C.PROPO EQU     'P              Printercode proportional charactertype
          000E  C.DOUBL EQU     $0E             Printercode double charactertype
          0021  C.LETQU EQU     '!              Printercode letter Quality
          0043  C.ITAL1 EQU     'C              Printercode1 italic
          0049  C.ITAL2 EQU     'I              Printercode2 italic
                ;
          0002  PRINTER EQU     $02             centronics printerdevices
          0007  BELL    EQU     $07             Bell-code
          001B  ESC     EQU     $1B             ESC-code
          0040  APP     EQU     $40             @-code reset printer
          0018  CAN     EQU     $18             Clears printerbuffer
          004C  LMARGIN EQU     'L              Left margin-code
          004F  PAGINC1 EQU     'O              Printcode1 page length
          0049  PAGINC2 EQU     'I              Printcode2 page length
          004F  SKIPC1  EQU     'O              Skip-code 1
          0053  SKIPC2  EQU     'S              Skip-code 2
```

Tijd 8-Apr-87 22:29 === file setprint.mac === pagina 2

```
                        ;
                        ;
                        ;-------------------------------------
                        ; Mainprogram
                        ;-------------------------------------
          1000          ORG     $1000
                        ;
                        ; Initialisatie
                        ;
1000 20 68C0    SETPRINT JSR    SOPT1           Get option
1003 48                 FCC     'H'             Help
1004 45                 FCC     'E'             Elite 12 cpi
1005 43                 FCC     'C'             Condensed 17 cpi
1006 50                 FCC     'P'             Proportional
1007 44                 FCC     'D'             Double
1008 4C                 FCC     'L'             Letter Quality
1009 49                 FCC     'I'             Italic
100A 4B                 FCC     'K'             Perforation skip
100B 00                 FCB     $0              Option tabel end
100C 90 03              BCC     1.f             Branch if option are correct
100E 4C FA10            JMP     ERROR
                        ;
1011 86 A0      1       STX     OPTMASK         Save the option
1013 A2 40              LDX     #$40            dec. mode
1015 20 6BC0            JSR     SPAR1           Load parameters
1018 A3                 FCB     #LEFTMAR
1019 A5                 FCB     #PAGELEN
101A A7                 FCB     #JUMPPER
101B 00                 FCB     $00             page zero address for parameter
101C 90 03              BCC     2.f             Branch if par. are correct
101E 4C FA10            JMP     ERROR
                        ;
1021 84 A1      2       STY     POINTER         Save filepointer address
1023 85 A2              STA     POINTER+1
                        ;
                        ; Test the help command
                        ;
1025 A5 A0      TSHELP  LDA     OPTMASK         Load option masker
1027 29 80              AND     #HELP           Test help command
1029 F0 03              BEQ     INITPR
102B 4C 6211            JMP     INFO
                        ;
                        ; Initialisation printerdevice
                        ;
102E A2 02      INITPR  LDX     #PRINTER        Load device number
1030 20 0CF0            JSR     INITDEV         Init. device
1033 90 19              BCC     RESET           Branch if printer ready
1035 20 3BC0            JSR     PRINT1
1038 075072696E         FCC     7,'Printer not ready.\r',0
104D 60                 RTS
                        ;
                        ; Reset printer
                        ; Clears printerbuffer.
                        ; adjust delault printermode
                        ;
104E A0 00      RESET   LDY     #0
1050 B9 5911    2       LDA     TABEL1,Y        Load printer-code
1053 C0 09              CPY     #(TABEL2-TABEL1) Test end table 1
1055 F0 08              BEQ     TSCOND
1057 A2 02              LDX     #PRINTER        Load device number
1059 20 06F0            JSR     OUTDEV          Print code
105C C8                 INY
105D 10 F1              BPL     2.b
                        ;
                        ; Test condensed charactertype
                        ;
105F A5 A0      TSCOND  LDA     OPTMASK         Load option masker
1061 29 20              AND     #CONDE          Test condensed charactertype
1063 F0 0B              BEQ     TSELITE
1065 A9 51              LDA     #C.CONDE        Load charactercode
```

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

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

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

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

                ;
                ; Test lettermode option
                ;
1089 20 3211    TSLMOPT JSR     TSOPT           Test the charactertype-option

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

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

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

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

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

10DC B0 19              BCS     EXIT            Test if end of file
10DE 48                 PHA                     Save filedata
10DF A2 02              LDX     #PRINTER        Load device number
10E1 20 06F0            JSR     OUTDEV          Print filedata
10E4 68                 PLA                     Restore filedata for test CR
10E5 C9 0D              CMP     #$0D
10E7 D0 EE              BNE     READ            branch if not CR
10E9 A9 0A              LDA     #$0A
10EB A2 02              LDX     #PRINTER        Load device number
10ED 20 06F0            JSR     OUTDEV          print a linefeed
10F0 4C D710            JMP     READ
                ;
10F3 C9 12      NONAME  CMP     #$12            Test no filename
10F5 D0 03              BNE     ERROR
10F7 60         EXIT    RTS
                ;
10F8 A9 16      ERERIT  LDA     #$16            Errormessage
10FA 4C B7D0    ERROR   JMP     ERMES1          print error
                ;
                ; end main program
                ;
                ; Routine **SETDATA**
                ;
                ; This routine convert the decimal
                ; data into ascii data and send it
                ; to the printer.
                ; Start: Accu - data to printer
                ;        Y-reg. dec/ascii data
                ;
10FD 20 2C11    SETDATA JSR     SETP1           Asciidata to printer
1100 98                 TYA
1101 20 0F11            JSR     DECASC          dec. to ascii
1104 86 A9              STX     TEMPX           Save ascii low nibble
1106 20 2C11            JSR     SETP1           Asciidata to printer
1109 A5 A9              LDA     TEMPX           Restore ascii low nibble
110B 20 2C11            JSR     SETP1           Asciidata to printer
110E 60                 RTS
                ;
                ; Routine **DECASC**
                ;
                ; Convert decimal to ascii data
                ;
                ; Start: Accu decimal data
                ; Stop : X-reg lowbyte ascii
                ;        Accu  highbyte ascii
                ;
110F 48         DECASC  PHA                     Save databyte
1110 29 0F              AND     #$0F            Mask right nibble
1112 20 1B11            JSR     NASCII          Convert to ascii-value
1115 AA                 TAX                     Save asciidata
1116 68                 PLA                     Restore databyte
1117 4A                 LSRA                    Shift right 4
1118 4A                 LSRA
1119 4A                 LSRA
111A 4A                 LSRA
111B C9 0A      NASCII  CMP     #$0A            Test op errordata
111D 10 D9              BPL     ERERIT
111F 18                 CLC
1120 69 30              ADC     #'0'            Make a character of it
1122 60                 RTS
                ;
                ;
                ; Routine **SETPRIN**
                ;
                ; This routine sends a esc-code
                ; followed bij a printer-code to
                ; the printer.
                ; Start: Accu printcode
                ;
1123 48         SETPRIN PHA                     Save code
```

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

124 A2 02             LDX     #PRINTER        Load device number
126 A9 1B             LDA     #ESC
128 20 06F0           JSR     OUTDEV          Print-code
12B 68                PLA                     Restore code
12C A2 02       SETP1 LDX     #PRINTER        Load device number
12E 20 06F0           JSR     OUTDEV          Print-code
131 60                RTS
                ;
                ;
                ; Routine **TSOPT**
                ;
                ; This routine test the option
                ; for the differend charactertype
                ;
132 A5 A0       TSOPT LDA     OPTMASK         Load option masker
134 29 04             AND     #LETQU          Test if letter quality
136 F0 05             BEQ     TSPROP
138 A9 21             LDA     #C.LETQU        Load charactercode
13A 20 2311           JSR     SETPRIN         Set printer-code
13D A5 A0       TSPROP LDA    OPTMASK         Load option masker
13F 29 10             AND     #PROPO          Test if propartional printen
141 F0 05             BEQ     TSOPT1
143 A9 50             LDA     #C.PROPO        Load charactercode
145 20 2311           JSR     SETPRIN         Set printer-code
148 A5 A0       TSOPT1 LDA    OPTMASK         Load option masker
14A 29 02             AND     #ITALI          Test if italic
14C F0 0A             BEQ     1.f
14E A9 43             LDA     #C.ITAL1        Load charactercode
150 20 2311           JSR     SETPRIN         Set printer-code
153 A9 49             LDA     #C.ITAL2
155 20 2C11           JSR     SETP1           Asciidata to printer
158 60          1     RTS
                ;
                ;
                ; Printer command tabels
                ;
159 1B4007181B  TABEL1 FCC    ESC,APP,BELL,CAN,ESC,'L007'
162             TABEL2
                ;
                ;--------------------------------------
                ; Screentext
                ;--------------------------------------
                ;
162 20 3BC0     INFO  JSR     PRINT1          Print help-info
165 0C                FCC     '\f'
166 1B69205320        FCC     '\Ei S E T - P R I N T E R - M O D E \En\r\r'
18D 4974206973        FCC     'It is possible whit this program to set the printer\r'
1C1 616E642074        FCC     'and the character-mode.\r'
1D9 0D                FCC     '\r'
1DA 436F6D6D61        FCC     'Command: SETPRINT [-HECPDLIK +m,n,o] FILE\r\r'
205 1B69204F70        FCC     '\Ei Option tabel \En\r\r'
219 2020202020        FCC     '          Pica 10 cpi (default)\r'
237 202D482020        FCC     ' -H       Help\r'
244 202D452020        FCC     ' -E       Elite 12 cpi\r'
259 202D432020        FCC     ' -C       Condensed 17 cpi\r'
272 202D502020        FCC     ' -P       Proportional\r'
287 202D442020        FCC     ' -D       Double width\r'
29C 202D4C2020        FCC     ' -L       Letter Quality\r'
2B3 202D492020        FCC     ' -I       Italic\r'
2C2 202D4B2020        FCC     ' -K       Perforation skip (default jump)\r'
2EA 202B6D2C6E        FCC     ' +m,n,o m = Set left margin (default m=7)\r'
314 2020202020        FCC     '          n = Page length in inches  (default n=12)\r'
346 2020202020        FCC     '          o = Number of lines by a jump (default o=12)\r'
37B 2046494C45        FCC     ' FILE     Filename of the printfile\r'
39D 0D00              FCC     '\r',0
39F 60                RTS
                ;
                ;
       1000           END     SETPRINT
```

[DOS65 Hardware Bug.]

Door: Nico de Vries
By : Mari Andriessenrade 49
      NL-2907 MA Capelle aan den IJssel
      The Netherlands.

In de diskcontrollerkaart van DOS65 zit een kleine fout. Deze fout is er de oorzaak van, dat de kaart mogelijk niet goed werkt op hogere clockfrequenties dan 1 MHz. De fout bestaat uit het volgende.
Op de kaart wordt uit de signalen R/W, de adresbus, en phi2 een aantal nieuwe signalen voor de 1793 floppy-controller gegenereerd, waaronder de /RD, de /WR en de /CS. Verder maakt de schakeling een enable-signaal voor de 74LS245 databusbuffer aan. Al deze signalen zijn ge-AND met phi2, dat wil zeggen, dat ze pas in het tweede deel van de clock stabiel worden. Dit is geheel juist, op één ding na.
Bekijkt men het datasheet van de 1793, dan blijkt dat de data op de databus 50 ns stabiel moeten zijn, voordat de /CS en de /RD of /WR actief worden, wil de 1793 betrouwbaar lezen. Dit wordt in de schakeling op de controller-kaart voorkomen, doordat ook de enable voor de databusbuffer ge-AND is met phi2. De kaart stuurt de 1793 volgens specificaties aan, wanneer dit ongedaan gemaakt wordt.
De bug is verholpen wanneer pin 2 van IC1 (74LS00) uit de voet gehaald wordt en wordt verbonden met pin 1 van hetzelfde IC. De data wordt nu eerder in de clockcycle stabiel, en blijft langer staan. De kaart zal nu in vrijwel alle gevallen op 2 MHz willen werken, ook als de relatief langzame Siemens SAB1793 gebruikt wordt. Uiteraard dient de rest van het systeem ook voor 2 MHz geschikt te zijn.

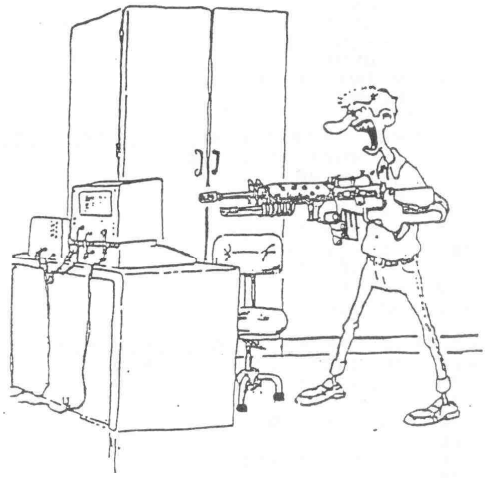
For the English speaking readers.

If you have a problem using the DOS65 diskcontrollercard with a 2 MHz clockrate, try this: remove pin 2 of IC1 (a 74LS00) from its socket, and connect it to pin 1 of the same IC. This will speed up the data transfer from/to the 1793. The rest of the system must be suitable for 2 MHz operation of course.

=+

```
;---------------------------------------------------------
;       Changes made in COMAL version 2.1 for DOS65
;=========================================================
;       Antoine Megens                         May 1987
;---------------------------------------------------------
;Directions:
;Load  old COMAL with LOAD S:COMAL,  then  enter  MONITOR.
;Change  the following addresses and save new COMAL  with:
;SAVE  S:NCOMAL  2000,4DFF,3000.  Then  enter  SETMODE  -C
;S:NCOMAL and test the changes with NCOMAL. The RUBOUT key
;should  work  on screen now and when the following  COMAL
;program is executed,  the file TEST.DAT should be  closed
;with  a  $00 byte instead of $1C (check this  with  DOS65
;command DUMP TEST.DAT).
;
; 100 CREATE "TEST.DAT"
; 110 OPEN #1;"TEST.DAT"
; 120 PRINT #1;"Testing EOF change"
; 130 CLOSE #1
;
; If this works you may rename the file NCOMAL to COMAL.
;---------------------------------------------------------
28D2  4C D0 4D          JMP  $4DD0     ;was JMP $C023,
;                                      ;now check RUBOUT
;                                      ;first
4C94  A9 00             LDA  #$00      ;was LDA #$1C
;
; RUBOUT check routine (unused space in old COMAL)
;
4DD0  C5 2D             CMP  $2D       ;RUBOUT character?
4DD2  F0 03             BEQ  $4DD7     ;yes! else
4DD4  4C 23 C0          JMP  $C023     ;just print char.
4DD7  8A                TXA            ;at zero position
                                       ;of input?
4DD8  10 03             BPL  $4DDD     ;no, continue
4DDA  A5 2D             LDA  $2D       ;else exit with
                                       ;RUBOUT char.
4DDC  60                RTS            ;(do nothing)
4DDD  A9 08             LDA  #$08      ;print <BS><space>
                                       ;<BS>
4DDF  20 23 C0          JSR  $C023     ;to simulate
                                       ;RUBOUT on screen
4DE2  A9 20             LDA  #$20
4DE4  20 23 C0          JSR  $C023
4DE7  A9 08             LDA  #$08
4DE9  20 23 C0          JSR  $C023
4DEC  A5 2D             LDA  $2D       ;exit with RUBOUT
                                       ;char. in A
4DEE  60                RTS
```

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



Waddaya mean, user error!?

```
1780  FLAG:=0
1790  REPEAT
1800   REPEAT
1810    Q:=3*P+INT(RND(1)*A+1)
1820   UNTIL MID$(A$,Q,1)<>"0"
1830   C$:="FUN"+MID$(A$,Q,1)
1840   EXEC: C$,FLAG,X,Y,D(X,Y),C2
1850  UNTIL FLAG<>0
1860  PRINT
1870  T:=T+1
1880  IF RND(1)<C1 THEN
1890   X:=INT(RND(1)*B)
1900   Y:=INT(RND(1)*H)
1910  ENDIF
1920 UNTIL T>=E
1930 CLS
1940 D(B-I,H):=D(B-I,H)+4
1950 EXEC: "MAZE"
1960 END.
1970 //
1980 PROC "MAZE"
1990  PRINT GRAF$;
2000  Y:=0
2010  FOR X:=0 TO B
2020   IF X=I THEN
2030    PRINT "Z  ";
2040   ELSE
2050    PRINT "ZXX";
2060   ENDIF
2070  ENDFOR
2080  PRINT "Z"
2090  FOR Y:=0 TO H
2100   FOR X:=0 TO B
2110    IN:=D(X,Y)
2120    EXEC: "BINARY",IN,B(1),B(2)
2130    IF B(1)=1 THEN
2140     PRINT "   ";
2150    ELSE
2160     PRINT "Y  ";
2170    ENDIF
2180   ENDFOR
2190   PRINT "Y"
2200   FOR X:=0 TO B
2210    IN:=D(X,Y)
2220    EXEC: "BINARY",IN,B(1),B(2)
2230    IF B(2)=1 THEN
2240     PRINT "Z  ";
2250    ELSE
2260     PRINT "ZXX";
2270    ENDIF
2280   ENDFOR
2290   PRINT "Z"
2300  ENDFOR
2310  PRINT TEXT$
2320 ENDPROC
2330 //
2340 PROC "FUN1",FLAG,X,Y,D(X,Y),C2
2350  IF RND(1)<C2 THEN
2360   FLAG:=0
2370  ELSE
2380   X:=X+1
2390   PRINT "+X";
2400   D(X,Y):=2
2410   FLAG:=1
2420  ENDIF
2430 ENDPROC
2440 //
2450 PROC "FUN2",FLAG,X,Y,D(X,Y),C2
2460  IF RND(1)<(1-C2) THEN
2470   FLAG:=0
2480  ELSE
2490   D(X,Y):=D(X,Y)+2
2500   X:=X-1
2510   PRINT "-X";
2520   D(X,Y):=1
2530   FLAG:=1
2540  ENDIF
2550 ENDPROC
2560 //
2570 PROC "FUN3",FLAG,X,Y,D(X,Y),C2
2580  IF RND(1)<C3 THEN
2590   FLAG:=0
2600  ELSE
2610   D(X,Y):=D(X,Y)+4
2620   Y:=Y+1
2630   PRINT "+Y";
2640   D(X,Y):=1
2650   FLAG:=1
2660  ENDIF
2670 ENDPROC
2680 //
2690 PROC "FUN4",FLAG,X,Y,D(X,Y),C2
2700  Y:=Y-1
2710  PRINT "-Y";
2720  D(X,Y):=4
2730  FLAG:=1
2740  IF RND(1)<C4 THEN
2750   C2:=1-C2
2760  ENDIF
2770 ENDPROC
2780 //
2790 PROC "BINARY",IN,B(1),B(2)
2800  NUM:=IN
2810  FOR I:=3 TO 0 STEP -1
2820   IF NUM-2^I>=0 THEN
2830    B(I):=1
2840    NUM:=NUM-2^I
2850   ELSE
2860    B(I):=0
2870   ENDIF
2880  ENDFOR
2890 ENDPROC
2900 //
```

18-Mar-87 22:55 timedate.mac Page 1

```
;*************************************************************************
; SYSTEM: DOS65 2.01
;
; PROGRAMMA TIMEDATE
; (simpel alternatief voor een real time clock kaart)
; Dit programma vraagt de tijd en de datum af. Geldige invoer wordt
; in de monitor variabelen voor deze parameters gezet. Wanneer alleen
; een CR wordt ingegeven wordt niets veranderd. De routine kan in de
; LOGIN.COM file worden aangeroepen.
;
; PROGRAM TIMEDATE
; (Simple alternative for real time clock cart)
; This routine prompts for time and date. If valid data is entered
; the monitor variables for these parameters are updated. If CR is
; entered no changes are made. The routine can be called from the
; LOGIN.COM file.
;
; Peter Roessingh, March 1986
; Changed to DOS65 2.01 December 1986
;*************************************************************************
        ORG $2000
;
        *** External  adrresses****
;
DATUPD  equ     $E77E           ; flag for date update
DAY     equ     $E77F           ; day register
MONTH   equ     $E780           ; month register
YEAR    equ     $E781           ; year register
HOURS   equ     $E782           ; hour register
MINUTES equ     $E783           ; minute register
SECONDS equ     $E784           ; seconds register
;
        *** Temporary storage locations ***
;
SVSTCK  res     1               ; temp for stackpointer
INDEX   res     1               ; temp for bufferindex
;
;
        *** Library files ***
;
        lib     INOUT           ; DOS65 I/O entry's
        lib     GETPAR          ; read decimal parameters
        lib     BOUND8          ; check boundaries, 8 bit
;
;*************************************************************
;
        *** Main program starts here ***
;
TIMDAT  JSR     DATLOOP         ; get date and update parameters
        JSR     TIMLOOP         ; get time and update parameters
;
EXIT    JSR     CRLF            ; print crlf
        JSR     CRLF            ; and another one
        RTS                     ; end of program timedate
;
        *** subroutine DATLOOP, get new date, loop until valid ***
;
DATLOOP JSR     PRINT           ; start loop: ask for new date
;
        fcc     $0D,$0A,$0A,'Today\'s date:  ',0
;
        JSR     BUFIN           ;   get new date in buffer
        JSR     DODATE          ;   try to set date
        BCS     ERRDAT          ;   on error continue in loop
```

```
;
        RTS                     ; exit loop if date valid
;
ERRDAT  JSR     PRINT           ;   print errormessage
;
        fcc     $0D,$0A,'Invalid date!    '
        fcc     'Format shoud be : dd-mm-yy',$0D,$0A,$0A
        fcc     'dd between  1 and 31',$0D
        fcc     'mm between  1 and 12',$0D
        fcc     'yy between 87 and 99',$0D,$0A,$0A
        fcc     'Try again please',$0D,$0A,0
;
        JMP     DATLOOP         ; loop until valid exit
;
;       *** Subroutine TIMLOOP, get new time, loop until valid ***
;
TIMLOOP JSR     PRINT           ; start loop: ask for new time
;
        fcc     'Current time:  ',0
;
        JSR     BUFIN           ;   get new time
        JSR     DOTIME          ;   try to set time
        BCS     ERRTIM          ;   on error continue in loop
;
        RTS                     ; exit loop if time valid
;
ERRTIM  JSR     PRINT           ;   print error message
;
        fcc     $0D,$0A,$0A,'Invalid time!    '
        fcc     'Format should be: hh:mm',$0D,$0A,$0A
        fcc     'hh between  0 and 23',$0D,$0A
        fcc     'mm between  0 and 59',$0D,$0A,$0A
        fcc     'Try again please',$0D,$0A,0
;
        JMP     TIMLOOP         ; loop until valid exit
;
;       ****************************************************
;
;       Routine to get and set the date
;
;
DODATE  JSR     BEGIN           ; init bufferpointer, test on CR
        BEQ     DODEND          ; CR found, immidiate exit
;
        TSX                     ; get stackpointer
        STX     SVSTCK          ; and save it
;
;       *** get day
;
        LDY     INDEX           ; pass index in Y index
        LDA     #'-'            ; pass delimiter in A
        JSR     GETPAR          ; get parameter
        BCS     DODERR          ; error exit
        CPX     #$00            ; test high byte
        BNE     DODERR          ; error exit if not zero
;
        STY     INDEX           ; save index
        LDY     #$01            ; lower limit 1
        LDX     #$1F            ; higher limit 31
        JSR     BOUND8          ; check boundaries
        BCS     DODERR          ; error exit
;
        PHA                     ; result on stack
;
;       *** get month
;
        LDY     INDEX           ; pass index in Y index
        LDA     #'-'            ; pass delimiter in A
        JSR     GETPAR          ; get parameter
        BCS     DODERR          ; error exit
        CPX     #$00            ; test high byte
```

```
        BNE     DODERR          ; error exit if not zero
;
        STY     INDEX           ; save index
        LDY     #$01            ; lower limit 1
        LDX     #$0C            ; higher limit 12
        JSR     BOUND8          ; check boundaries
        BCS     DODERR          ; error exit
;
        PHA                     ; result on stack
;
;       *** get year
;
        LDY     INDEX           ; pass index in Y index
        LDA     #$0D            ; pass delimiter in A
        JSR     GETPAR          ; get parameter
        BCS     DODERR          ; error exit
        CPX     #$00            ; test high byte
        BNE     DODERR          ; error exit if not zero
;
        STY     INDEX           ; save index
        LDY     #$57            ; lower limit 1987
        LDX     #$63            ; higher limit 1999
        JSR     BOUND8          ; check boundaries
        BCS     DODERR          ; error exit
;
        PHA                     ; result on stack
;
;       *** set the date
        PLA                     ; get year back
        STA     YEAR            ; set year
        PLA                     ; get month back
        STA     MONTH           ; set month
        PLA                     ; get day back
        STA     DAY             ; set day
        LDA     #$FF            ; get flag
        STA     DATUPD          ; and set it
;
;       *** update the date in the statusline
;
        BCC     DODEND          ; branch always
;
DODERR  LDX     SVSTCK          ; get saved stackpointer
        TXS                     ; restore stackpointer
        SEC                     ; indicate error
;
DODEND  RTS                     ; return from dodate
;
;       ****************************************************
;
;       Routine to get and set the time
;
DOTIME  JSR     BEGIN           ; init bufferpointer, test on CR
        BEQ     DOTEND          ; CR found, immidiate exit
;
        TSX                     ; get stackpointer
        STX     SVSTCK          ; and save it
;
;       *** get hours
;
        LDY     INDEX           ; pass index in Y index
        LDA     #':'            ; pass delimiter in A
        JSR     GETPAR          ; get parameter
        BCS     DOTERR          ; error exit
        CPX     #$00            ; test high byte
        BNE     DOTERR          ; error exit if not zero
;
        STY     INDEX           ; save index
        LDY     #$00            ; lower limit 0
        LDX     #$17            ; higher limit 23
        JSR     BOUND8          ; check boundaries
        BCS     DOTERR          ; error exit
;
```

18-Mar-87 22:55 timedate.mac Page 4

```
        PHA                     ; result on stack
;
;       *** get minutes
;
        LDY     INDEX           ; pass index in Y index
        LDA     #$0D            ; pass delimiter in A
        JSR     GETPAR          ; get parameter
        BCS     DOTERR          ; error exit
        CPX     #$00            ; test high byte
        BNE     DOTERR          ; error exit if not zero
;
        STY     INDEX           ; save index
        LDY     #$00            ; lower limit 00
        LDX     #$3B            ; higher limit 59
        JSR     BOUND8          ; check boundaries
        BCS     DOTERR          ; error exit
;
        PHA                     ; result on stack
;
;       *** set time ***
;
        PLA                     ; get minutes back
        STA     MINUTES         ; set minutes
        PLA                     ; get hours back
        STA     HOURS           ; set hours
        LDA     #$00            ; seconds are zero
        STA     SECONDS         ; set seconds
;
        BCC     DOTEND          ; branch always
;
DOTERR  LDX     SVSTCK          ; get saved stackpointer
        TXS                     ; restore stackpointer
        SEC                     ; indicate error
;
DOTEND  RTS                     ; return from dotime
;
;       ********************************************************
;
;       *** routine to init bufferpointer
;
BEGIN   LDA     #$00
        STA     INDEX           ; init index to buffer
;
        TAY                     ; point to begin buffer
        JSR     GETBUF          ; get first char
        BEQ     1.f             ; CR found, immidiate exit
;
        LDA     #$FF            ; signal normal exit
1       RTS                     ; and return
;
;       ********************************************************
        END     TIMDAT          ; end of program
```

2-Apr-87 23:08 inout.mac Page 1

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

```
2-Apr-87 23:00  getpar.mac  Page 1

;*****************************************************************
;
;       Library file GETPAR.MAC          Subroutine getpar
;
;       This routine gets a decimal parameter from the DOS65
;       inputbuffer and converts it to hex. The delimiter for
;       the parameter should be in A, the startposition in the
;       buffer in Y. The hex result is given back in A and X.
;       For the algorithm, see Leventhal and Saville (1982).
;
;       The library file INOUT.MAC is expected to be present in
;       the calling program.
;
;       Call:   A  delimiter
;               Y  buffer index
;
;       Return  A  low byte result (hex)
;               X  high byte result (hex)
;               Y  buffer index
;               C  error flag
;
;*****************************************************************
;
;       *** temporarys ***
;
RETADR  res     2               ; return adress
DLMTR   res     1               ; delimiter
ACCUM   res     2               ; result (2 bytes)
NGFLAG  res     1               ; flag for negative number
;
;*****************************************************************
;
GETPAR  STA     DLMTR           ; save delimiter
        LDA     #$00
        STA     ACCUM           ; init result
        STA     ACCUM+1         ; high byte also
        STA     NGFLAG          ; default positive
;
        PLA                     ; get return adress
        STA     RETADR          ; save it
        PLA
        STA     RETADR+1
;
        JSR     GETBUF          ; get character
        CMP     #'-'            ; test  on minus
        BNE     PLUS            ; branch if not minus
        LDA     #$FF            ; get flagbyte
        STA     NGFLAG          ; indicate negative number
        INY                     ; skip past minus sign
        JMP     CNVERT          ; start conversion
;
PLUS    CMP     #'+'            ; test on plus sign
        BNE     CNVERT          ; branch if not plus
        INY                     ; skip past plus sign
;
CNVERT  JSR     GETBUF          ; get character
        CMP     DLMTR           ; test if delimiter
        BEQ     2.f             ; exit if found
        CMP     #'0'            ; test if smaller then 0
        BMI     4.f             ; error, < 0 (not a digit)
        CMP     #'9'+1          ; test if greater then 9
        BPL     4.f             ; error, > 9 (not a digit)
        PHA                     ; save digit on stack
;
;       *** Valid decimal digit, convert to hex ***
;
        ASL     ACCUM
        ROL     ACCUM+1 ; * 2
        LDA     ACCUM
        LDX     ACCUM+1         ; save result * 2
        ASL     ACCUM
        ROL     ACCUM+1
        ASL     ACCUM
        ROL     ACCUM+1 ; * 8
        CLC
```

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

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

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

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

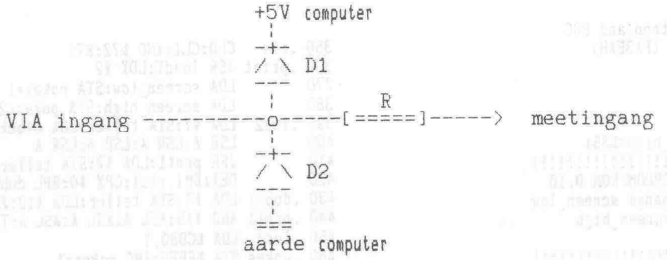
```
;
BOUND8  STA     PARAM           ; save parameter
;
        CPY     PARAM           ; test lower limit
        BEQ     1.f             ; equal is o.k.
        BCS     2.f             ; par too low, error
;
1       CPX     PARAM           ; test high limit
        BCC     2.f             ; par too high, error
;
        CLC                     ; flag no error
        BCC     3.f             ; branch always, normal exit
;
2       SEC                     ; flag error
3       RTS                     ; and return
;
        END     BOUND8          ; end of routine
;*****************************************************************************
```

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

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

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

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



In deze schakeling beschermt de diode D1 de ingang tegen spanningen die hoger zijn dan ≈ +5,6V omdat boven deze spanning de diode in geleiding komt.
Diode D2 beschermt de ingang tegen spanningen die lager zijn dan≈-0,6V. Voor diode D2 zou eigenlijk een Schottky of een germanium diode gebruikt moeten worden aangezien deze een lagere doorlaat- spanning hebben. De serieweerstand dient als stroombegrensings- weerstand. Hoe groter de weerstand R is des te lagen is de belasting op de te meten schakeling. Met één ding moet echter wel rekening worden gehouden: aangezien de ingang van de VIA en de diode een bepaalde capaciteit hebben zal er bij een signaal met een hoge frequentie vervorming van de flanken ontstaan.

Een bruikbare waarde voor R = 5 kΩ tot 10 kΩ
Voor de diode D1 en D2 kan één van de onderstaande type gebruikt worden.

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