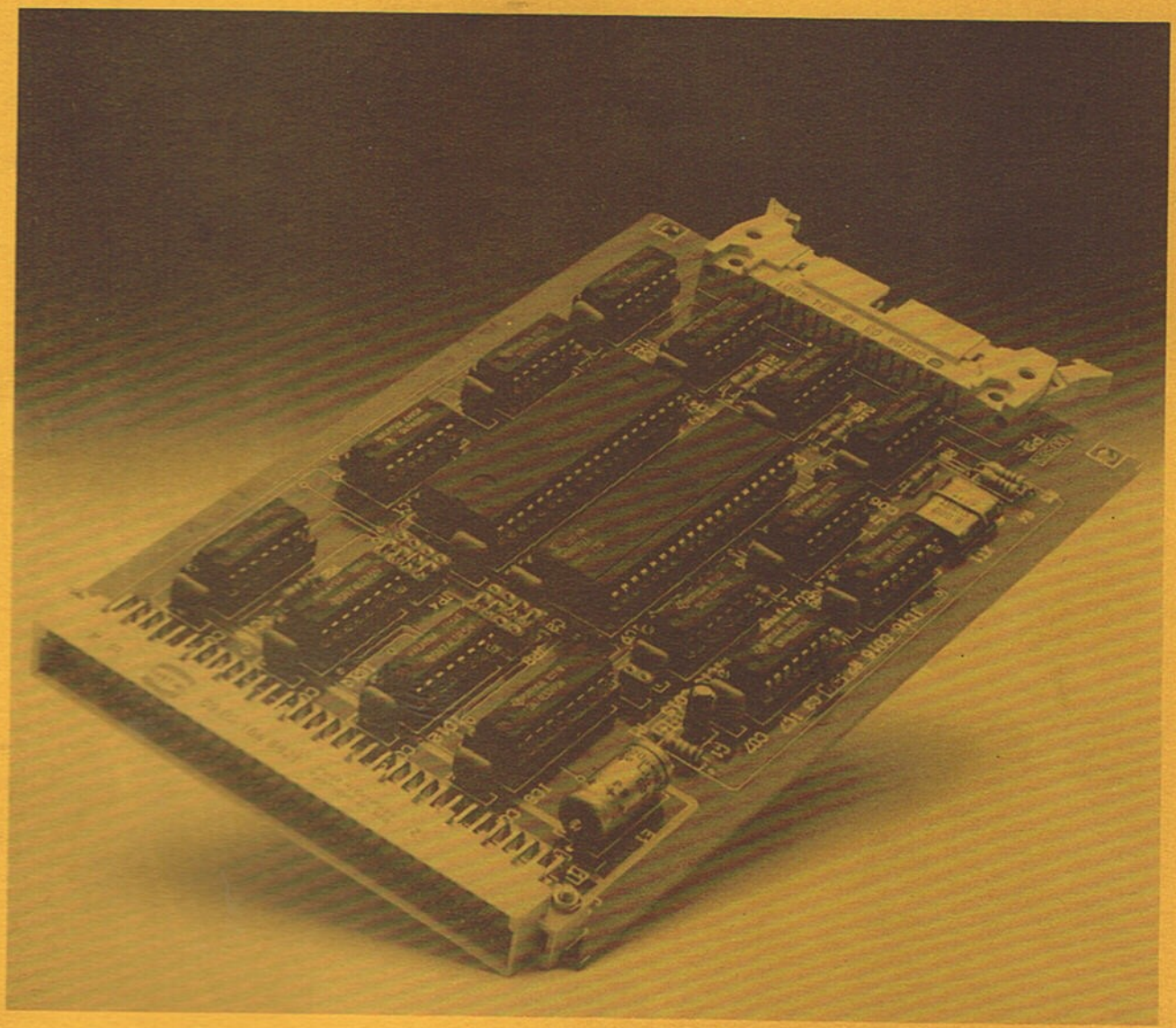
DE 6502 KENNER 49



Elfde Jaargang, Nr. 2
April 1987

DE 6502 KENNER is published by the KIM Users Club The Netherlands.

Address all editorial, advertising and subscription inquiries to :
DE 6502 KENNER
c/o Willem L. van Pelt
Jacob Jordaensstraat 15
NL-2923 CK Krimpen/IJssel
The Netherlands.

Editorial staff:
Willem L. van Pelt
Gerard van Roekel
Frans Smeehuijzen
Coen Boltjes

Freelancers:
Fred Behringer (Germany)
Andrew Gregory (England)
Marc Lachaert (Belgium)
Fernando Lopez (Portugal)
Gert van Opbroek
Leif Rasmussen (Denmark)
Ruud Uphoff
Frans Verberkt
Simon Voortman
Herman Zondag
and many others.

Translations:
Fred Behringer (Germany)
Willem van Asperen
Frank Bens
Albert v.d. Beukel
Rene Hettfleisch
Coen Kleipool (France)
Maarten van Lieshout
Antoine Megens
Piet K. de Vries
and many others.

Illustrations/cartoons:
Antoine Megens
Piet K. de Vries
Frank Vergoossen
Gonda Engel

The articles to be published have to be written by the sender.
DE 6502 KENNER appears in Febr, Apr, May, July, Sept, Oct, and Dec.

On frontpage is the DOS65 controllercard, developed by our member Ad Brouwer.
CAD/CAM: E. Visschedijk
Coöp: A. Hankel
Photo : Fr. Visschedijk

CONTENTS OF DE 6502 KENNER NO. 49, APRIL 1987 VOL. 11 NO. 2.

Print your articles, programs etc. with a new ribbon and use 8 lines/inch by 73 lines/page max.
Write your articles, programs etc. in English unless you need help to do it.

We need more members to do more for our members. Look around in your family, among your friends and on your job. Send the names and addresses of 6xxxx-users to the editorial office. They will be sent information about our club, to let them join our club.

PLEASE PAY YOUR 1988 SUBSCRIPTION BEFORE DECEMBER 1987.

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

Datum : zaterdag 16 mei 1987
Lokatie : GEWIJZIGDE LOKATIE: !!!
Wijkcentrum 't Veurbrook
Jan Tooropstraat 27
7606 JS ALMELO
Tel.: 05490 - 10353

Routebeschrijving:

Voor degenen die al eerder op bijeenkomsten in Almelo waren, is het eenvoudig: U rijdt naar de U bekende lokatie aan de Jan Steenstraat. Daar aangekomen gaat U steeds rechtdoor, tot U niet verder kunt. Hier gaat U linksaf. Dit is de Jan Tooropstraat. Met de bocht mee naar rechts. Na plm. 20 meter links, 't Veurbrook.

Vanuit het westen en het zuiden (via A1/A35):
1. Aan het einde van de snelweg rechtsaf. Bij het eerstvolgende kruispunt MET STOPLICHTEN linksaf, richting Wierden/Zwolle. Bij de eerstvolgende stoplichten rechtdoor. Bij de volgende stoplichten (links BP tankstation en Opel garage Kamp) gaat U rechtsaf.
2. U rijdt nu op de Windmolenbroeksweg. Doorrijden tot over de brug, dan de eerste straat rechts. Dit is de W. van Konijnenburgstraat. Na plm. 50 meter rechtsaf. Dit is de Jan Tooropstraat. Met de bocht mee naar links. Na plm. 50 meter aan de rechterkant 't Veurbrook.

Vanuit het noorden (via de N36):
1. Bij de eerste stoplichten rechtsaf, richting streekziekenhuis. U bevindt zich nu op de rondweg om Almelo. Deze weg blijven volgen tot U het BP tanstation ziet. Bij dit kruispunt linksaf. Zie verder punt 2.

Met het openbaar vervoer
Vanaf NS station Almelo met de stadsbus naar de wijk Molenbroek. Uitstappen bij de halte Windmolenbroeksweg. Schuin tegenoer de bushalte staat een wegwijzer. Daarop staat ook 't Veurbrook vermeld.

TOEGANGSPRIJS : FL. 10,==

PROGRAMMA

09.30 Zaal open.
10.15 Opening door de gastheren Adri Hankel en Erwin Visschedijk.
10.30 COMMUNICATIEDAG.
De nadruk zal liggen op het werken met computer en modem.
11.30 Koffiepauze.
11.45 Forum. Aan het forum kunnen vragen gesteld worden van allerlei aard.
12.00 Lunchpauze.
13.00 INFORMEEL GEDEELTE.
Tijdens het informeel gedeelte kunnen leden vrij met elkaars ervaringen kennis maken. Leden brengen hun systemen mee en demonstreren dit aan de aanwezigen.
NEEM DAAROM UW COMPUTER MEE !!!
Het verdient aanbeveling ook een of meerdere verlengsnoeren mede te nemen.
MARKT. Op eigen tafel(s) te regelen.
17.00 Sluiting.

==========

Albert v.d. Beukel, Van Slingerlandtstraat 19, NL-2623 TT Delft, The Netherlands.

I like to know whether there are members of the club that typed in the ADDRESS PROGRAMME of the Elektor's Computer Special nr. 1. The programme will not run on my computer, so I like to know what's wrong with it.

==========

REDAKTIONEEL

Deze editie is voor Uw redaktie opnieuw van bijzondere betekenis. U zult merken dat het aantal artikelen opvallend veel is. Dat is geen toeval. U zult ook merken dat in veel gevallen de teksten duidelijker leesbaar zijn dan we gewend waren. Het is nog even afwachten wat het effekt zal zijn wanneer de editie daadwerkelijk op de deurmat ligt, maar we hebben goede hoop dat ook de drukker weer wat betere resultaten weet te bereiken nu deze nogal wat nieuwe apparatuur heeft aangeschaft. Wel zullen de teksten kleinere letterformaten bevatten, maar door de verhoogde leesbaarheid hopen we het storende effekt van condensed printen met de matrixprinter zoveel mogelijk uit te gaan schakelen om tegemoet te komen aan die enkeling in de club die toch moeite ermee had om dat te lezen. Een voordeel dat hieruit voortvloeit is het feit dat er wederom meer artikelen in een editie terecht kunnen. Dat feit moet echter vergezeld gaan, willen we niet met een lege copy-buffer komen te zitten op de lange termijn, met een verhoogd enthousiasme om eigen programma's in te sturen ter publikatie.

In de resultaten van de enquete die door het bestuur in 1985 werd gehouden komt o.a. de wens voor om een rubriek te openen waarin problemen met de Octopus aan de orde kunnen komen. Hierop wil ik graag even reageren.
Als het zo is dat een dergelijke rubriek niet bestaat, dan zou men positief kunnen denken dat het met de Octopus prima gesteld is, er derhalve geen problemen bestaan. Men kan ook denken aan het feit dat het wat gek zou klinken als deze computer geen problemen kent, maar dat de leden die problemen niet kenbaar maken via DE 6502 KENNER. Noch het een, noch het ander is het geval, voor zover ik dat kan overzien. De Octopus heeft m.i. zowel aan de hardware- als aan de softwarezijde nog heel wat te wensen over gelaten. Voordeel voor de club is dat er dan heel wat te doen valt, nadeel is dat men nogal eens denkt dat de redaktie al die problemen zelf kan konstateren en daarover publiceren. Niets is minder waar. Bovendien is het zo dat leden allang van alles en nog wat over de Octopus naar buiten brengen, gelukkig meestal via het clubblad. Echter het is niet zo dat daaruit de noodzaak van een vaste rubriek valt af te lijden; het is daarenboven te weinig om er vaste ruimte voor te reserveren. Wat wel uit deze kan worden geleerd is dit: de leden doen er goed aan deze behoefte onder ogen te zien en er zo mogelijk aan mee te werken dat problemen met de Octopus in alle gevallen aan de redaktie worden toegezonden.
Zo was er in de enquete ook een opmerking dat in de edities "niet helemaal uitgewerkte ideeën" voorkwamen. Het is mij niet geheel duidelijk geworden wat hier precies mee bedoeld werd. Wat dat betreft is de probleemstelling niet helemaal uitgewerkt door de inzender, maar dat klinkt als terugpesten, terwijl ik het hier juist aanroer omdat ik er serieus mee om wil springen. Het zou mij een troost zijn als hier wat meer over gezegd zou kunnen worden. De inzender moet dit nog wel herkennen en die verzoek ik daaromtrent met mij kontakt op te nemen, zodat er aandacht aan besteed kan worden, en -zo dat mogelijk is- oplossingen voor bedacht.
Er kwam ook de wens dat er meer kleine artikelen in de edities moesten komen. Met deze editie is die wens, althans voor dit moment, in vervulling gegaan. Of aan die wens voortaan gevolg kan worden gegeven is niet aan de redaktie maar aan de leden zelf om daarvoor te zorgen. Daar komen immers de inzendingen vandaan. Voor de redaktie is het in elk geval een plezierige bijkomstigheid als er veel niet te omvangrijk materiaal in de copy-buffer zit. Anderzijds is de redaktie ook weer heel blij met grote inzendingen die kwalitatief van hoog nivo zijn. Dat is voor de hele club meer dan alleen een visitekaartje en aandachttrekker tot over de landsgrenzen.

==========

Tony Lehaen, Kloosterstr. 24, B-3580 Neerpelt, België.

Ik heb de oorspronkelijke software van mijn JUNIOR veranderd. Zo zit bijv. de basismonitor en videosoftware in Eprom vanaf adres $F2C0. Ik ben nu bezig de cassetteroutines van Ad Brouwer uit edities 30 en 31 van DE 6502 KENNER aan mijn systeem aan te passen, doch dit wil nog niet zo best lukken.
Vraag: Heeft iemand deze cassetteroutines ook op z'n systeem draaien? Wil deze zich dan met mij in verbinding stellen?
I changed the original systemsoftware of my JUNIOR. For instance, the base-monitor and videosoftware are in Eprom now from address $FC20. I am adapting now the cassetteroutines of Ad Brouwer as published in issues 30 and 31 of DE 6502 KENNER, but it won't work until now. Has anyone good running cassetteroutines as mentioned on his system? please contact with me.

==========

KUNT U EN WILT U CARTOONS TEKENEN VOOR DE 6502 KENNER ?
Stuur het naar de redaktie, U doet er velen plezier mee.

= A DIGITAL VOLTMETER FOR THE COMMODORE 64 =

DVM V1.0

Author: Jaap de Hoop, The Netherlands.
Transl: Piet de Vries, The Netherlands.

EXPLANATORY NOTE

The design of this "low cost" digital voltmeter is made especially for the Commodore 64.
The heart of the circuit is IC4, an AD 2020 made by Analog Devices or a CA 3162 produced by RCA. The latter is the cheapest (Hfl. 18,98). The circuit can be split up into two parts, the read out part and the pre-amplifier stage. Without the amplifier the circuit has a range from -99mV to 999 mV, with the amplifier from -9.99V to 99.9V. The internal resistance is in all measureranges 1 MegaOhm. The sample speed is determined at 96 Hz.

The read out part

The heart of this is the earlier mentioned CA 3162. This chip is designed to drive a BCD/7 SEGMENTS DECODER. This driving is done in a multiplexed way, see the timing diagrams.

The program is so designed that at the at the negative trailing edge of MSD, NSD and LSD the at that moment valid BCD word is taken over by the computer. Also information about the state of the RANGE SELECT switch is taken over, so the program 'knows' in which state the circuit is.

°The pre-amplifier stage

The heart of the pre-amplifier stage is formed by three J-FET OPAMPs LM 356 (Hfl. 3,06 a piece). The high impedance is gained by using these OPAMPs. As protection diodes I used transistors which I switched as diodes. This because of the fact that transistors have a smaller leakcurrent in backward direction (1 nA instead of 20 nA). The input can be used floating as well as not floating (signal to mass).

The construction

- The best way is to make a print, but it can also be done on a hole-board.
- When you don't use IC sockets you should instal IC 3 during the tuning procedure.
- The pre-amplifier stage needs some more attention. Make good mass-connections, use short signalwires and coax cable to avoid jammer.
- To be totally independent of the C64, the most simple way is to use a plug that fits into the userport. There are 9 connections to be made, PB0...PB7 and a mass (don't forget!). Check the connections made and be carefull when switching on the power (PIA's don't live forever when a short circuit occurs).
- After building and checking the circuit, the test-program has to be loaded. In this article is an example.
- The big moment, switch on your Commodore, circuit on, run the testprogram.
- When the circuit and the program are correct, data will flow over the screen, when not something is wrong.
- The program as well as the circuit are build and tested in practice by myself and worked correctly.

Applications

There are according to me a number of nice applications. E.g. a slow A/D converter. By writing a program in machine-code that reads the DVM at maximum speed and stores the data in the internal memory of the Commodore, it is possible to scan time-varying signals and to plot curves of these signals on the screen.

By using a different converter, there are other magnitudes you can measure, e.g. resistance, current, frequency, temperature and light intensity.

It is also possible to measure at a longer time-scale, by referencing to the systemcode of the Commodore. E.g. the conduct of the temperature during a day with every half an hour a sample. The measured signals can be transformed well in visual information with aid of the graphical possibilities of the Commodore.

The circuit is made for the Commodore 64, but can easily be adapted to all other computers with 8 free wires. The number of 8 can be reduced to 6 (with range select) or 5 (without range select). It is possible to use only the MSD digit select and to use internal delays to determine NSD and LSD.

I think that there are lots of possibilities for an inventive programmer.

Program for reading the DVM-module

```
49152 0          VAR    BCDM          DATA WORD DURING MSD
49153 0          VAR    BCDN          DATA WORD DURING MSD
49154 0          VAR    BCDL          DATA WORD DURING LSD
49155 0          VAR    RANGE         RANGE CODE
---------------- START                ---------------------
49156 169,0      LDA # 0000 0000
49158 141,3,221  STA $ DATA DIR REG   PB0..PB7 INPUT
---------------- BCDM                 ---------------------
49161 173,1,221  LDA $ DATA REG
49164 41,64      AND # 0100 0000
49166 208,249    BNE   BCDM           MSD ACTIVE ?
49168 173,1,221  LDA $ DATA REG
49171 141,0,192  STA $ BCDM           IF YES: SAVE DATA WORD
49174 41,128     AND # 1000 0000
49176 208,5      BNE   BCDL           RANGE ACTIVE ?
49178 169,1      LDA # 0000 0001
49180 141,3,192  STA $ RANGE          IF YES: RANGE CODE = 1
---------------- BCDL                 ---------------------
49183 173,1,221  LDA $ DATA REG
49186 41,16      AND # 0001 0000
49188 208,249    BNE   BCDL           LSD ACTIVE ?
49190 173,1,221  LDA $ DATA REG
49193 141,2,192  STA $ BCDL           IF YES: SAVE DATA WORD
49196 41,128     AND # 1000 0000
49198 208,5      BNE   BCDN           RANGE ACTIVE ?
49200 169,3      LDA # 0000 0011
49202 141,3,192  STA $ RANGE          IF YES: RANGE CODE = 3
---------------- BCDN                 ---------------------
49205 173,1,221  LDA $ DATA REG
49208 41,32      AND # 0010 0000
49210 208,249    BNE   BCDN           NSD ACTIVE ?
49212 173,1,221  LDA $ DATA REG
49215 141,1,192  STA $ BCDN           IF YES: SAVE DATA WORD
49218 41,128     AND # 1000 0000
49220 208,5      BNE   END            RANGE ACTIVE ?
49222 169,2      LDA # 0000 0010
49224 141,3,192  STA $ RANGE          IF YES: RANGE CODE = 2
---------------- END                  ---------------------
49227 96         RTS                  OUTPUT CYCLE COMPLETED
```

Hardware specifications of the digital voltmeter

Data Format

PB0	BCD BIT 1	POS LOGIC MULTIPLEXED
PB1	BCD BIT 2	POS LOGIC MULTIPLEXED
PB2	BCD BIT 3	POS LOGIC MULTIPLEXED
PB3	BCD BIT 4	POS LOGIC MULTIPLEXED
PB4	LSD DIGIT SELECT	NEG LOGIC
PB5	NSD DIGIT SELECT	NEG LOGIC
PB6	MSD DIGIT SELEXT	NEG LOGIC
PB7	RANGE SELECT	NEG LOGIC MULTIPLEXED

PB0-PB1-PB2-PB3

These four bits form together a BCD data word. Three BCD data words form together the sample-value. The three BCD are named MSD, NSD and LSD (Most, Next and Least Significant Digit). Which of the three BCD words is at the output is indicated with the DIGIT SELECT lines.

PB4-PB5-PB6

These wires indicate which BCD word is available at the output. One of the three wires is low, the others are high. The line that is low indicates that the related digit is active.

PB7

This wire indicates at which range the digital voltmeter is set.

PB7	low during MSD	inputrange	1V
PB7	low during NSD	inputrange	10V
PB7	low during LSD	inputrange	100V

Special sample values

BCDM	BCDN	BCDL	
1011	1011	1011	overrange positive
1010	1010	1010	overrange negative
1010	XXXX	XXXX	indicate negative

Power Supply

5V POS - 20 mA
5V NEG - nil
12V POS - 14 mA
12V NEG - 14 mA

Software specifications of the digital voltmeter

Variable BCDM

Memory location: 49152
Contents : see hardware specifications

Variable BCDN

Memory location: 49153
Contents : see hardware specifications

Variable BCDL

Memory location: 49154
Contents : see hardware specifications

Variable RANGE

Memory location: 49155
Contents : 0 - after loading the program
1 - 1 Volt RANGE
2 - 10 Volt RANGE
3 - 100 Volt RANGE
all other values are not valid

memory organisation

First address : 49152
Last address : 49227
Start address : 49156
Positions : 75

Example BASIC program for test of the DVM

```
100 PRINT "<FF>"
110 PRINT " "
120 PRINT "DIGITAL VOLTMETER V1.0"
130 PRINT " "
140 PRINT "LOADING MEMORY"
150 FOR IN = 1 TO 76
160 READ DA
170 POKE (49151+IN),DA
180 NEXT IN
190 PRINT " "
200 PRINT "WHEN YOU DON'T SEE THIS TEXT DISAPPEAR"
210 PRINT "THERE IS A HARDWARE FAULT !"
220 SYS(49156)
230 M=PEEK(49152) AND 15
240 N=PEEK(49153) AND 15
250 L=PEEK(49154) AND 15
260 R=PEEK(49155)
270 IF M=11 THEN PRINT "+++++++++++++++": GOTO 220
280 IF N=10 THEN PRINT "---------------": GOTO 220
290 IF R=1 THEN T$="1 V MAX"
300 IF R=2 THEN T$="10 V MAX"
310 IF R=3 THEN T$="100 V MAX"
320 ZE=0
330 IF M=10 AND N<>10 THEN ZE=1
340 IF ZE=1 THEN PRINT "-";(N*10)+L,T$:GOTO 220
350 IF ZE=0 THEN PRINT "+";(M*100)+(N*10)+L,T$:GOTO 220

1000 REM DATA VELD
1010 DATA 0
1020 DATA 0
1030 DATA 0
1040 DATA 0
1050 DATA 169,0
1060 DATA 141,3,221
1070 DATA 173,1,221
1080 DATA 41,64
1090 DATA 208,249
1100 DATA 173,1,221
1110 DATA 141,0,192
1120 DATA 41,128
1130 DATA 208,5
1140 DATA 169,1
1150 DATA 141,3,192
1160 DATA 173,1,221
1170 DATA 41,16
1180 DATA 208,249
1190 DATA 173,1,221
1200 DATA 141,2,192
1210 DATA 41,128
1220 DATA 208,5
1230 DATA 169,3
1240 DATA 141,3,192
1250 DATA 173,1,221
1260 DATA 41,32
1270 DATA 208,249
1280 DATA 173,1,221
1290 DATA 141,1,192
1300 DATA 41,128
1310 DATA 208,5
1320 DATA 169,2
1330 DATA 141,3,192
1340 DATA 96
```

Tuningprocedure for the DVM

1. IC3 is removed or not installed if it's not on a socket.
2. On the print pin 6 of IC3 is connected to the mass.
3. Start the testprogram.
4. With PB3 (zero adjust), the readout is set to 000.
5. Stop the testprogram.
6. Remove the massconnection of step 2.
7. Install IC3.
8. Both inputs (signal+, signal-) are connected to the mass.
9. Start the testprogram.
10. With P2 (zero) the readout is set to 000 again.
11. Both inputs are disconnected from the mass and they are both connected to a voltage of about 3V with regard to the mass.
12. With P1 (common mode rejection) the readout is set to 000 again.
13. Both inputs are disconnected. Signal + has to be connected to a well known voltage, e.g. 800 mV.
14. With P4 (gain adjust) the readout is set to the known sample value; so in this example 800 mV.

List of components for the DVM

R1	10K
R2-R3	180K
R4-R5	1M8
R6-R8	10K
R9-R16	100K
R17	1M
R18-R25	100K
D1-D4	as a diode connected transistor BC557
S1	dubbeldeck 3-state switch
IC1-IC3	LM356
IC4	CA3162 of AD2020
IC5	74LS27
IC6	74LS02
P1-P2	10 turns tuningresistor 25K
P3	10 turns tuningresistor 47K
P4	10 turns tuningresistor 10K
C1	MKH 1uF
C2	MKH 220nF

➔

BOEKINFORMATIE

FORTH een praktische introductie.
Auteur: Leo Brodie (Vert: Luk van Loock)
Uitg. : Maarten Kluwer, Antwerpen/Apeldoorn,
1985, 235 p., Hfl. 59,50
ISBN 90 6215 1167

De laatste jaren geniet de programmeertaal FORTH een sterk toenemende populariteit. FORTH dankt dit ten eerste aan het feit, dat het een veelzijdige taal is. Het is gelijkertijd:
- een hogere-ordetaal
- een assembly-taal
- een operating-system
- een set development tools
- een filosofie m.b.t. het ontwerpen van software.

FORTH beschikt daarbij over een groep van zeer krachtige standaard commando's gekoppeld aan een mechanisme, waarmee u uw eigen commando's kunt definieren aan de hand van vorige definities. Een en ander brengt met zich mee, dat FORTH snel, compact, flexibel en overdraagbaar is. Kortom FORTH wordt daarom met name toegepast in:
- de wetenschap
- de procescontrole
- de data acquisitie en analyse
- draagbare intelligente apparatuur

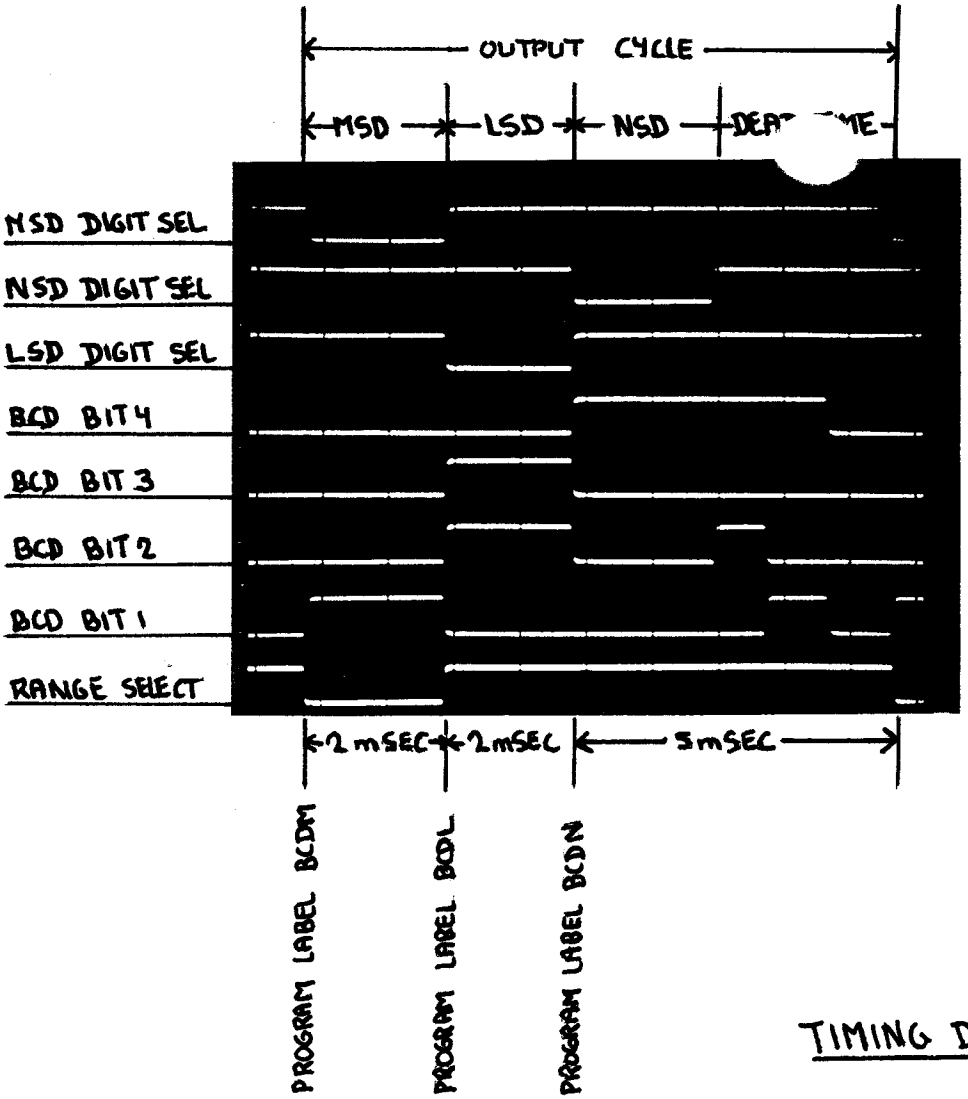
overigens: de leertijd van FORTH is relatief veel korter dan bij andere programmeertalen. Overtuig uzelf met dit boek, dat momenteel als het beste op dit gebied wordt beschouwd.
De engelstalige uitgave is getiteld: Starting FORTH.

BRIEF AAN DE REDAKTIE

Maarten van Lieshout, Lambertushof 72, 5667 SG Geldrop roept op tot het schrijven van een programma voor de "Computerskoop" van Elektuur ten behoeve van DOS65 + grafische kaart.

OPROEP

WILLEN ZIJ DIE ZENDAMATEUR ZIJN DIT OPGEVEN BIJ DE REDAKTIE

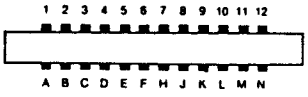


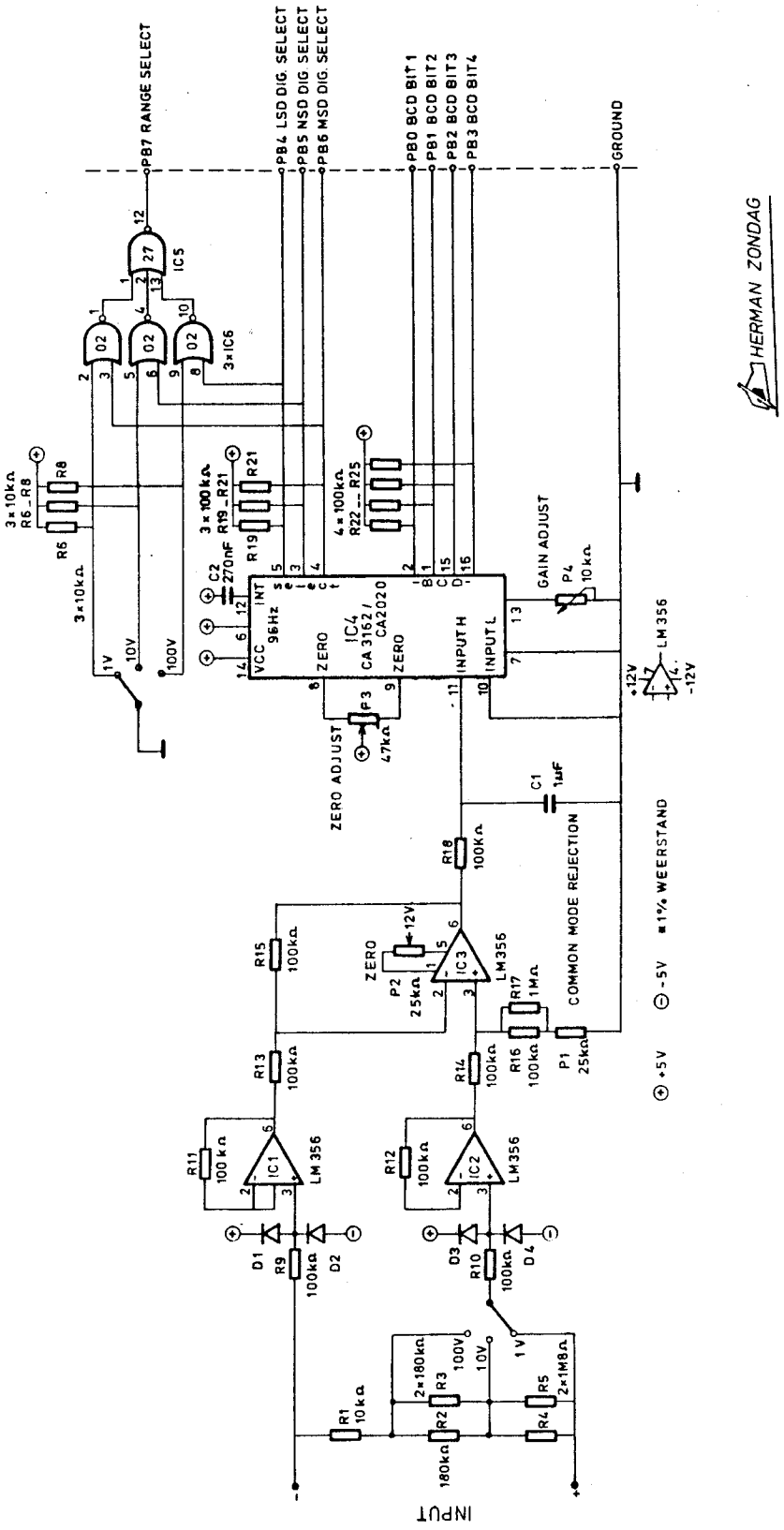
TIMING DIAGRAM

User I/O

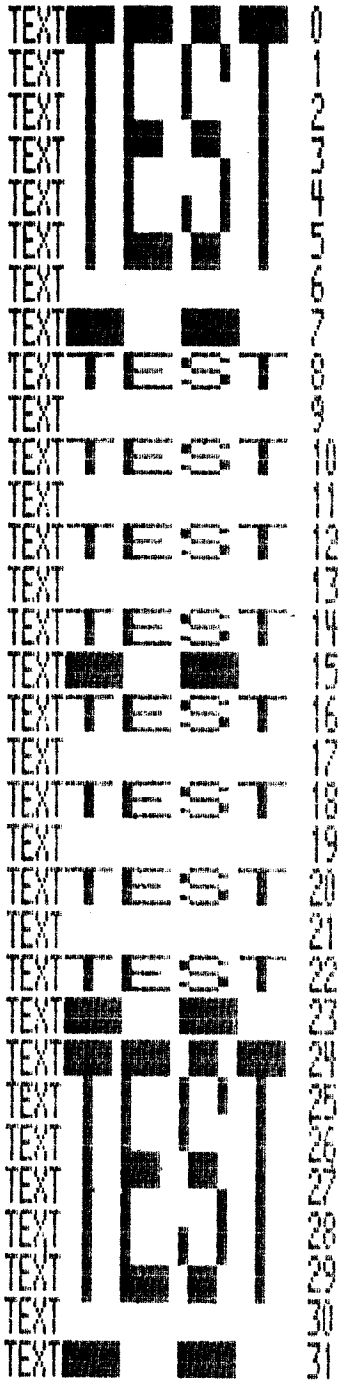
Pin	Type	Note
1	GND	
2	+5V	MAX. 100 mA
3	$\overline{\text{RESET}}$	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	$\overline{\text{PC2}}$	
9	SER. ATN IN	
10	9 VAC	MAX. 100 mA
11	9 VAC	MAX. 100 mA
12	GND	

Pin	Type	Note
A	GND	
B	$\overline{\text{FLAG2}}$	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	

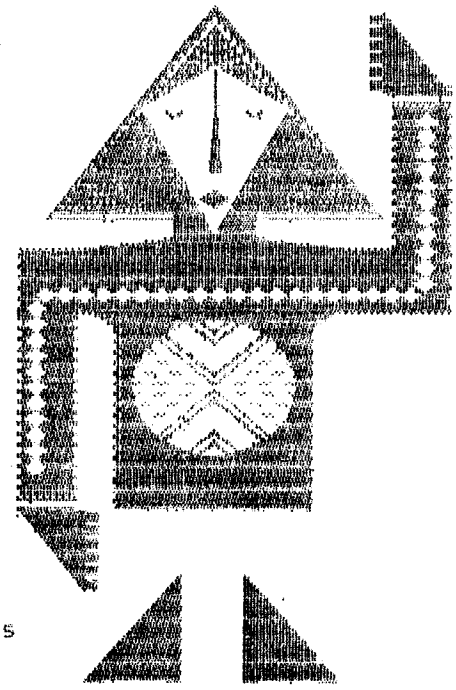



PB7 RANGE SELECT
PB4 LSD DIG. SELECT
PB5 NSD DIG. SELECT
PB6 MSD DIG. SELECT
PB0 BCD BIT 1
PB1 BCD BIT 2
PB2 BCD BIT 3
PB3 BCD BIT 4
GROUND
IC5
3×IC6
IC4 CA 3162 / CA2020
96Hz INT
VCC
ZERO
INPUT H
INPUT L
GAIN ADJUST
P4 10kΩ
ZERO ADJUST
P3 47kΩ
C1 1µF
C2 270nF
R18 100kΩ
R15 100kΩ
R13 100kΩ
R14 100kΩ
R16 100kΩ
P1 25kΩ
COMMON MODE REJECTION
P2 25kΩ
R17 1MΩ
IC3 LM356
IC1 LM356
IC2 LM356
R11 100kΩ
R12 100kΩ
R9 100kΩ
R10 100kΩ
D1
D2
D3
D4
R1 10kΩ
R2 180kΩ
R3 2×180kΩ
R4
R5 2×1M8Ω
INPUT
+12V
-12V
+5V
-5V
1% WEERSTAND
HERMAN ZONDAG

```
4720:
0005: CD70                      ORG   $CD70
0010:                   ************************************
0015:                   #                                  #
0020:                   #     SCREEN DUMP FOR KOLORATOR    #
0025:                   #         on EC65 and Epson        #
0030:                   #           .                      #
0035:                   #   by Leif Rasmussen, Parkvej 1   #
0040:                   #   DK 4534 Hørve.       jun'86    #
0045:                   #                                  #
0050:                   ************************************
0055:
0060:                   This routine for kolorator (lavigne-
0065:                   Meyer) enables a hardcopy of screen.
0070:                   Prt.ctrl codes are in Epson standard
0075:                   All the different  graphic modes can
0080:                   be used, and a  fraction  of  screen
0085:                   can be printed. To implement the ro-
0090:                   utine put the adrs. for F and  K  in
0095:                   the 'graphics' jump-table.
0100:
0105: E150              CMD     EQU   $E150   gdp regs
0110: E158              MSBX    EQU   CMD     +08
0115: E159              LSBX    EQU   CMD     +09
0120: E15A              MSBY    EQU   CMD     +0A
0125: E15B              LSBY    EQU   CMD     +0B
0130: E164              COLOR   EQU   CMD     +14
0135: C08D              READY   EQU   $C08D   wait for gdp
0140: C7D3              STCNT1  EQU   $C7D3   two data cmd
0145: C949              STCNT2  EQU   $C949   three data cmd
0150: C66A              SETMOD  EQU   $C66A   single cmd
0155: C655              ENDMOD  EQU   $C655   mult. cmd
0160: CA2D              ENDRW   EQU   $CA2D   restore coords
0165: CF80              MIRHX   EQU   $CF80   software regs
0170: CF8A              PIXBUF  EQU   MIRHX   +0A
0175: CF94              MODE    EQU   MIRHX   +14
0180: CF9E              DATA2   EQU   MIRHX   +1E
0185: CFA0              DATA1   EQU   MIRHX   +20
0190: CFA2              DATA0   EQU   MIRHX   +22
0195: 2322              OUTABL  EQU   $2322   dos outputtable
0200: 2343              PRINT   EQU   $2343   print chr in (A)
0205: 25A9              SKIPOC  EQU   $25A9   test spc. chrs
0210: 2D73              STROUT  EQU   $2D73   output string
0215:
0220:                   Fraction, define a window
0225:                   "Fx,x,y,y"
0230:
0235: CD70 AD 94 CF     F       LDA   MODE    "Fxfrom,xto,yfrom,yto"
0240: CD73 30 04                BMI   X.YFLAG
0245: CD75 4C D3 C7             JMP   STCNT1
0250:
0255: CD78 20           LINEBUF HEX   20 nmbr 8pix-lines to print
0260:
0265: CD79 A9 00        X.YFLAG LDAIM $00     flipflop for first
0270: CD7B 49 01                EORIM $01     and second set of cmd
0275: CD7D 8D 7A CD             STA   X.YFLAG +01
0280: CD80 F0 28                BEQ   YFORMAT
0285: CD82 38           XFORMAT SEC           first set
```



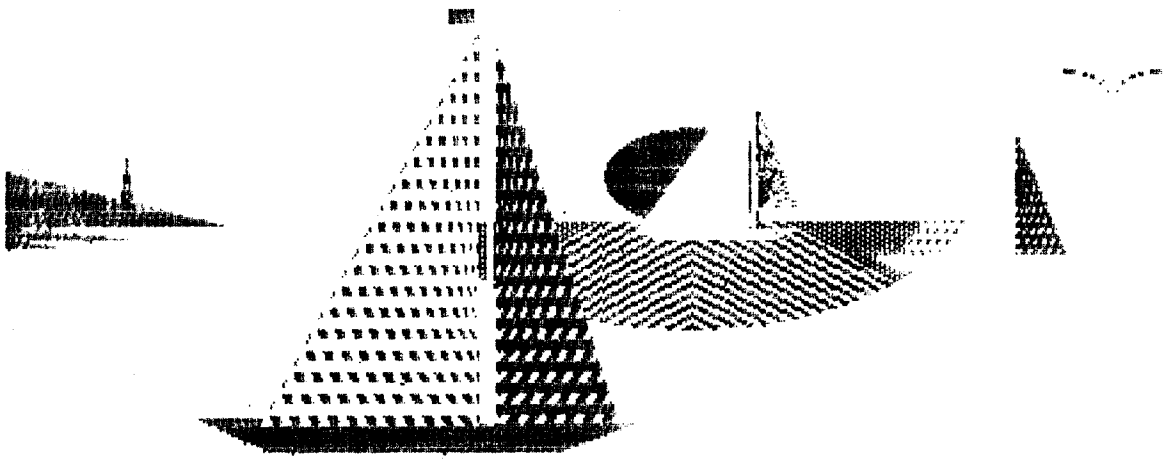
```
0290: CD83 AD A3 CF          LDA   DATA0   +01 lsbx to
0295: CD86 8D A5 CE          STA   XTILLSB +01
0300: CD89 ED A1 CF          SBC   DATA1   +01 lsbx from
0305: CD8C 8D 4B CE          STA   INILIN  +05 lsb of chrs to print
0310: CD8F AD A2 CF          LDA   DATA0   msbx to
0315: CD92 8D 9B CE          STA   XTILMSB +01
0320: CD95 ED A0 CF          SBC   DATA1   msbx from
0325: CD98 8D 4C CE          STA   INILIN  +06 msb of chrs to print
0330: CD9B AD A1 CF          LDA   DATA1   +01
0335: CD9E 8D 4E CE          STA   XSTART  +01
0340: CDA1 AD A0 CF          LDA   DATA1
0345: CDA4 8D 50 CE          STA   XSTART  +03
0350: CDA7 4C 55 C6          JMP   ENDMOD  get next set of cmd
0355:
0360: CDAA AD A3 CF  YFORMAT LDA   DATA0   +01 second set, lsby from
0365: CDAD 8D 33 CE          STA   YSTART  +01
0370: CDB0 ED A1 CF          SBC   DATA1   +01 lsby to
0375: CDB3 4A                LSRA          div. 8
0380: CDB4 4A                LSRA
0385: CDB5 4A                LSRA
0390: CDB6 A8                TAY
0395: CDB7 C8                INY           minimum one line
0400: CDB8 8C 78 CD          STY   LINEBUF
0405: CDBB 4C 6A C6          JMP   SETMOD  end of cmd
0410:
0415:                Kopi, screen dump
0420:
0425:                "K-t,m,y"
0430:                "- : -= pix on -> dot off
0435:                     += pix on -> dot on
0440:                "t : tabulator
0445:                "m : gra. mode 0...5
0450:                "y : 1= 1pix->2dots,
0455:                     0= 1pix->1dot y-direct.
0460:
0465: CDBE AD 94 CF  K       LDA   MODE
0470: CDC1 30 04             BMI   HARDCOP
0475: CDC3 4C 49 C9          JMP   STCNT2
0480:
0485: CDC6 20        LINECNT HEX   20 nmbr lines
0490:
0495: CDC7 A2 03     HARDCOP LDXIM $03     save coords
0500: CDC9 BD 58 E1  SAVEREG LDAX  MSBX
0505: CDCC 48                PHA
0510: CDCD CA                DEX
0515: CDCE 10 F9             BPL   SAVEREG
0520: CDD0 AD 78 CD          LDA   LINEBUF init. nmbr. of lines
0525: CDD3 8D C6 CD          STA   LINECNT
0530: CDD6 AD 9E CF          LDA   DATA2   if 0 then positiv pict.
0535: CDD9 F0 04             BEQ   POSITIV if <>0 then negativ
0540: CDDB A9 D0             LDAIM $D0     for bne
0545: CDDD D0 02             BNE   GODN
0550: CDDF A9 F0     POSITIV LDAIM $F0     for beq
0555: CDE1 8D 72 CE  GODN    STA   REVERSE
0560: CDE4 AD 9F CF          LDA   DATA2   +01 tabulator
0565: CDE7 8D 2B CE          STA   INITPRI +05
0570: CDEA AD A1 CF          LDA   DATA1   +01 graphic mode 0...5
0575: CDED 8D 4A CE          STA   INILIN  +04
```



```
0580: CDF0 AD A3 CF          LDA   DATA0   +01 if 0 then y-dimension
0585: CDF3 F0 10             BEQ   YDIM    is 1/1
0590: CDF5 A9 C0             LDAIM $C0     else 2/1
0595: CDF7 8D 60 CE          STA   ENLARGE +01
0600: CDFA 0E C6 CD          ASL   LINECNT double nmbr of lines
0605: CDFD A9 4E             LDAIM $4E     for lsr mult +01
0610: CDFF A2 75             LDXIM MULT    +01
0615: CE01 A0 CE             LDYIM MULT    /
0620: CE03 D0 0B             BNE   GOON1
0625: CE05 A9 80     YDIM    LDAIM $80     no enlargement
0630: CE07 8D 60 CE          STA   ENLARGE +01
0635: CE0A A9 EA             LDAIM $EA     for nop, nop, nop
0640: CE0C A2 EA             LDXIM $EA
0645: CE0E A0 EA             LDYIM $EA
0650: CE10 8D 86 CE  GOON1   STA   ENLAR0
0655: CE13 8E 87 CE          STX   ENLAR0  +01
0660: CE16 8C 88 CE          STY   ENLAR0  +02
0665:
0670: CE19 A9 60             LDAIM $60     for rts
0675: CE1B 8D A9 25          STA   SKIPOC  we dont want extra lfs
0680: CE1E A9 08             LDAIM $08     turn on printer
0685: CE20 8D 22 23          STA   OUTABL
0690: CE23 20 73 2D          JSR   S1ROUT  initiate printer
0695: CE26 1B 33 18  INITPRI HEX   1B33181B6C0600
      CE29 1B 6C 06
      CE2C 00
0700: CE2D A9 00             LDAIM $00     we work with 512*256 pix.
0705: CE2F 8D 5A E1          STA   MSBY
0710: CE32 A0 FF     YSTART  LDYIM $FF     starting row
0715:
0720: CE34 8C A3 CF  PRINIT  STY   DATA0   +01 outer loop start
0725: CE37 A2 00             LDXIM $00
0730: CE39 BD 46 CE  IGEN    LDAX  INILIN  header for each line:
0735: CE3C 20 43 23          JSR   PRINT   cr,lf,gra.mod,chr.nrs
0740: CE3F E8                INX
0745: CE40 E0 07             CPXIM $07
0750: CE42 D0 F5             BNE   IGEN
0755: CE44 F0 07             BEQ   XSTART  allways
0760: CE46 0A 0D 1B  INILIN  HEX   0A0D1B2A050002
      CE49 2A 05 00
      CE4C 02
0765: CE4D A2 00     XSTART  LDXIM $00     starting collum
0770: CE4F A9 00             LDAIM $00
0775:
0780: CE51 8E 59 E1  NEXTCHR STX   LSBX    middle loop start
0785: CE54 8D 58 E1          STA   MSBX
0790: CE57 8C 5B E1          STY   LSBY
0795: CE5A A9 00             LDAIM $00     clear pixel buffer
0800: CE5C 8D 8A CF          STA   PIXBUF
0805: CE5F A9 80     ENLARGE LDAIM $80     is normal, $C0 is enlarged
0810: CE61 8D 75 CE          STA   MULT    +01
0815: CE64 18                CLC
0820:
0825: CE65 A9 0F     GETPIX  LDAIM $0F     inner loop start
0830: CE67 8D 50 E1          STA   CMD     enter mfree mode
0835: CE6A 20 8D C0          JSR   READY   wait for gdp
0840: CE6D AD 64 E1          LDA   COLOR   pixels are in b0..b3
0845: CE70 29 01             ANDIM $01     mask color
```

```
0850: CE72 F0 04     REVERSE BEQ   NODOT   beq = pos, bne = reversed
0855: CE74 A9 80     MULT    LDAIM $80     or 40....01/C0....03
0860: CE76 D0 02             BNE   ADD     allways
0865: CE78 A9 00     NODOT   LDAIM $00
0870: CE7A 6D 8A CF  ADD     ADC   PIXBUF  accumulate 8/4 pix to a chr
0875: CE7D 8D 8A CF          STA   PIXBUF
0880: CE80 CE 5B E1          DEC   LSBY
0885: CE83 4E 75 CE          LSR   MULT    +01
0890: CE86 4E 75 CE  ENLARG  LSR   MULT    +01 = double-y, nop =single-y
0895: CE89 90 DA             BCC   GETPIX  if 8/4 pixels are read,
0900:                                      inner loop end
0905: CE8B 20 43 23          JSR   PRINT   then send the byte to printer
0910: CE8E 18                CLC
0915: CE8F AD 59 E1          LDA   LSBX    increase the x-direction
0920: CE92 69 01             ADCIM $01
0925: CE94 AA                TAX
0930: CE95 AD 58 E1          LDA   MSBX
0935: CE98 69 00             ADCIM $00
0940: CE9A C9 02     XTILMSB CMPIM $02     if 512 chr.s are printed
0945: CE9C F0 06             BEQ   XTILLSB then goto the next line
0950: CE9E AC A3 CF  GOON2   LDY   DATA0   +01 else restore lsby
0955: CEA1 4C 51 CE          JMP   NEXTCHR msbx (A), lsbx (X), lsby (Y)
0960: CEA4 E0 00     XTILLSB CPXIM $00
0965: CEA6 D0 F6             BNE   GOON2   middle loop end
0970:
0975: CEA8 CE C6 CD  NEWLINE DEC   LINECNT if all lines are printed
0980: CEAB F0 06             BEQ   SLUT    then terminate
0985: CEAD AC 5B E1          LDY   LSBY    else start a new line
0990: CEB0 4C 34 CE          JMP   PRINIT  with lsby reduced with 9
0995:                                      outer loop end
1000: CEB3 A9 01     SLUT    LDAIM $01     turn off printer
1005: CEB5 8D 22 23          STA   OUTABI
1010: CEB8 A9 48             LDAIM $48     restore dos
1015: CEBA 8D A9 25          STA   SKIPOL
1020: CEBD 4C 2D CA          JMP   ENDRW   restore coords and end
```

```
>>> Error in 0000 statement(s)

>>> Op-Code: $B000 - $B14F / $CD70 - $CEBF / 0336 Bytes / 02 Page(s)

>>> Assembled by ASS114 / 3.4
```



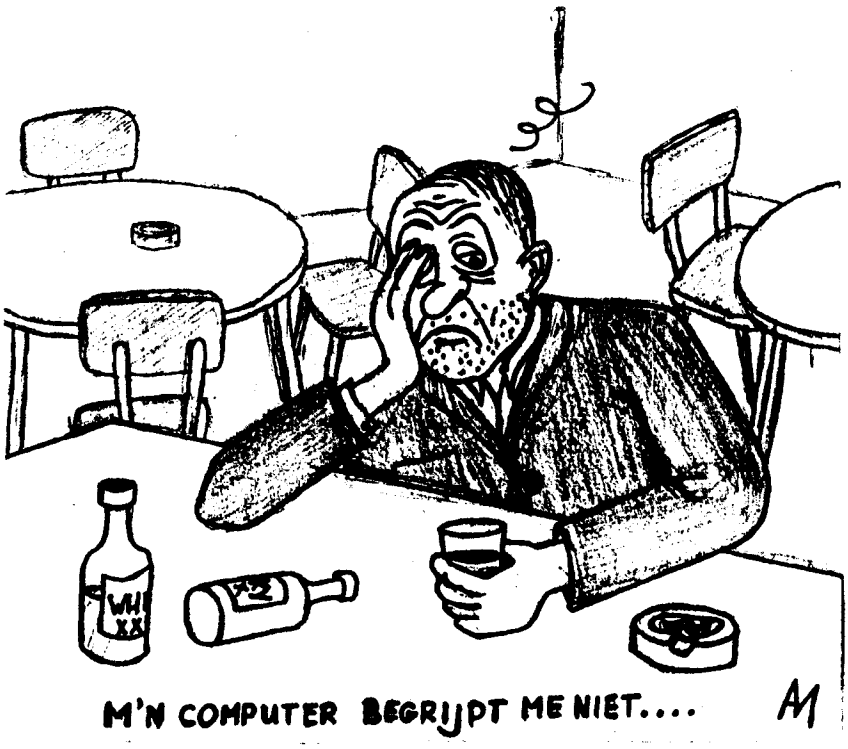
```
/*=========================================================================*/
/*                                                                         */
/*                              PLOT_POINTS                                */
/*                                                        versnr 86112401  */
/*                             J.H.Vernimmen                               */
/*                             v.IJsendijkstr 128                          */
/*                             1442 CS  PURMEREND                          */
/*                             ==================                          */
/*                                                                         */
/*  This program is made to get experience with LATTICE - C on the         */
/*  ATARI 520 ST. ( 68000 processor )                                      */
/*  It plots a figure on the screen of which the coordinates are given in  */
/*  the array 'points' ; in this this case a small aeroplane.              */
/*  The plotting takes place on an invisible screen pointed by 'beeld';    */
/*  after drawing the visible and invisible screen are changed.            */
/*  After plotting the figure again with another offset it gives the       */
/*  impression to move over the screen.                                    */
/*  The screen must be seen as an array of :                               */
/*                  - Horisontal 40 words of 16 bits ( 560 pixels )        */
/*                  - Vertical  400 lines            ( 400 pixels )        */
/*  So this needs a continuous piece of 32K bytes memory.                  */
/*                                                                         */
/*=========================================================================*/

#include "stdio.h"              /* include the standard C  I/O - routines */
#include "osbind.h"
#define msbit  0X8000L  /* set hi-order bit in word marked as a long word */
#define bmask  0X0F                    /* masker to get the 4 lo-order bits */
#define strip  0X04   /* ROR 4 times = devide by 16 to get the wordnumber */
#define streep 0XFFFF                                /* 1 word all bits set */
#define x_fact 0X0A       /* the constant '10' used in combination with x */
#define y_fact 0X09        /* the constant '9' used in combination with y */
#define cor_fc 1<<y_fact                          /* geeft 2^y_fact = 512 */


/*=========================================================================*/
/*  The array 'points' contains the coordinates to be plotted.             */
/*  Single dots and connected lines are possible.                          */
/*  Each group of connected lines starts wich a move ('DOT'). Therefore    */
/*  each group has a pointer to the next not direct connected line.        */
/*  The grouping of points is as follows :                                 */
/*  1' position : pointer to next group of points giving connected lines   */
/*                ( the pointer is always pointing to the next pointer )   */
/*  2'..k' position : pairs of x - y coordinates                           */
/*  k+1' position : next pointer                                           */
/*  The last pointer is a '0' followed bij two '0''s : x=0 ; y = 0 ;       */
/*=========================================================================*/


short points[80] = {
  19, -59, 5,  30, 5, 43, 3, 43, -2, 10, -4, -25,-4, -52, 0, -52, 3, -38,5,
  15, -44,-1, -32,-8, 23,-8, 42,-19, 49,-19,  38,-8,  38,-3,
  13,  -1, 5, -16, 3,  0, 1,  9,  1, 28,  3,  12, 5,
   5, -25,-4, -21,-8,
   5,  32,-5,  48,-5,
   0,   0, 0        } ;

int aantal = 57;                          /* number of elements in points */
                                  /* mind the first element is points[0] */

int offs_x, offs_y ;
struct FIG { unsigned short beeld_arr[400][40] ; } *beeld, *fysbld, *logbld;
/* Create a type of variable FIG as big as one screen-map and init        */
/* 3 pointers pointing to its beginning                                   */


/*=========================================================================*/
/* Function DOT                                                            */
/* Plot one pixel in 'beeld_arr' : The screen pointed by 'beeld'           */
/*=========================================================================*/

DOT ( x , y )
int x, y;

{
 beeld->beeld_arr[ y ][ x >> strip ] |= msbit >> ( x & bmask ) ;
/* Find the word by stripping x  and the bit inside the word by masking x */
}
```

```
/*=========================================================================*/
/* Function HORIZONTAL                                                     */
/* Plot a line from xw to x2 with on line y = y1w                          */
/* Calculate first the number of words in which all bits have to be set    */
/*=========================================================================*/

HORIZONTAL ( xw, y1w, x2 )
int xw, y1w, x2 ;
{
 int x1, sgn_delta ;
 unsigned lo_x1, lo_x2 ;
 register unsigned xchar1, xchar2, x, xm ;

 x1 = xw + ( x2 > xw ? 1 : -1 );        /* first pixel is already plotted */
 xchar1 = x1 >> strip ;                    /* find the wordnumber start */
 xchar2 = x2 >> strip ;                   /* find the wordcount end line */
 lo_x1  = msbit >> ( x1 & bmask ) ;     /* mask startbit inside first word */

 lo_x2  = msbit >> ( x2 & bmask ) ;      /* mask last bit inside last word */
 sgn_delta = xchar2 > xchar1 ? 1 : -1 ; /* find the direction of the line */

 if ( xchar1 != xchar2 )           /* there may be words with all bits set */
    {
     for ( x = xchar1 + sgn_delta; x != xchar2 ; x += sgn_delta )
        beeld->beeld_arr[ y1w ][ x ] = streep ;      /* plot all bits set */

    if ( xchar1 < xchar2 )                       /* plot from left to right */
      {
       xm = 0 ;
       for ( x = lo_x1 ;
             x != 0;  x >>= 1 ) xm |= x ; /* sample lo bits to be plotted */
       beeld->beeld_arr[ y1w ][ xchar1 ] |= xm ; /* plot them all at once */

       xm = 0 ;                /* same procedure for bits in hi-order word */
       for (x = lo_x2; x != 0;  x <<= 1 ) xm |= x ;
       beeld->beeld_arr[ y1w ][ xchar2 ] |= xm ;
      }
    else
      {
       xm = 0 ;                               /* plot goes from right to left */
       for ( x = lo_x1; x != 0;  x <<= 1 ) xm |= x ; /* hi-order sampling */
       beeld->beeld_arr[ y1w ][ xchar1 ] |= xm ;                  /* plot */

       xm = 0 ;
       for ( x = lo_x2 ; x != 0; x >>= 1 ) xm |= x ; /* lo-order sampling */
       beeld->beeld_arr[ y1w ][ xchar2 ] |= xm ;                  /* plot */
      }
    }
 else                  /* xchar1 = xchar2 : the first word = the last word */
    {
     xm = 0 ;
     if ( x1 <= x2 )                              /* plot from left to right */
       for ( x = lo_x1 ; ( x >= lo_x2 ) && x ;  x >>= 1 ) xm |= x ;
     else                                                 /* right to left */
       for ( x = lo_x1 ; ( x <= lo_x2 ) && x ;  x <<= 1 ) xm |= x ;
     beeld->beeld_arr[ y1w ][ xchar1 ] |= xm ;                    /* plot */
    }
}


/*=========================================================================*/
/* Function VERTICAL                                                       */
/* Plot a vertical line from y1w to y2w at x = x1                          */
/*=========================================================================*/

VERTICAL ( x1, y1w, y2w )
```

```
int x1, y1w, y2w;
{
 int sgn_delta ;
 unsigned xchar1 ;
 register unsigned x, yw ;

 xchar1 = x1 >> strip ;           /* split x in wordnumber and           */
 x = msbit >> ( x1 & bmask ) ;    /*                          x bitnumber */
 sgn_delta = y2w > y1w ? 1 : -1 ; /* low to high = downward is positive  */
 for ( yw = y1w + sgn_delta; yw != y2w + sgn_delta ; yw += sgn_delta )
       beeld->beeld_arr[ yw ][ xchar1 ] |= x ;/* plot every point separate*/
}

/*=========================================================================*/
/* function DIAGONAL                                                       */
/* Plot a line with an angle of 45 degr with respect to the X and Y- axis */
/* ( positive or negative ) from ( x1,y1w ) to ( x2,y2w ).                 */
/* Mind  x2 - x1 == y2w - y1w                                              */
/*=========================================================================*/

DIAGONAL ( x1, y1w, x2, y2w )
int x1, y1w, x2, y2w;

{
 int sgn_x, sgn_y ;
 register unsigned x, yw ;

 sgn_x = x2 > x1 ? 1 : -1 ;            /* calculate counter x as 1 or -1 */
 sgn_y = y2w > y1w ? 1 : -1 ;          /* calculate counter y as 1 or -1 */
 for ( yw = y1w, x = x1 ; yw != y2w + sgn_y; yw += sgn_y, x += sgn_x )
       beeld->beeld_arr [yw][ x >> strip ] |= msbit >> ( x & bmask ) ;
                                        /* plot every point separate */
}


/*=========================================================================*/
/*  function HELLING                                                       */
/* Plot a line with an angle <> +- 45 and <> 0  degrees with respect to    */
/* the X and Y - axis.                                                     */
/* Divide the line in different HORIZONTAL, VERTICAL or DIAGONAL lines     */
/* and call this routines for every piece of line                          */
/*=========================================================================*/

HELLING ( x1, y1w, x2, y2w )
int x1, y1w, x2, y2w;

{
 register int x, yw ;
 unsigned delta_x, delta_y, hd_points ;
 int sgn_x, sgn_y, h_points, hx_points, hy_points, aant_str ;

 x = x1 ;
 sgn_x = x2 > x ? 1 : -1 ;                              /* direction x */
 sgn_y = y2w > y1w ? 1 : -1 ;                           /* direction y */
 delta_x = x2 > x ? x2 - x : x - x2 ;          /* absolute distance x2 - x */
 delta_y =  y2w > y1w ? y2w - y1w : y1w - y2w ;      /* absolute y2w - y1w */
 if ( delta_x > delta_y )                             /* angle < 45 degrees */
  {
   if ( delta_x > delta_y << 1 )                          /* dx/dy > 2 */
    {                                    /* plot dy+1 horizontal lines */
     hd_points = ( delta_x << x_fact ) / ++delta_y ;
       /* expand x, ( Mind no REALS! ) calc length of the horizontal part */
     h_points = sgn_x == 1 ? hd_points : -hd_points ;
                                        /* find x - direction pos or neg */
     for ( yw = y1w, x = x1 << x_fact ;
           yw != y2w ;
           yw += sgn_y, x += h_points )
        HORIZONTAL ( x >> x_fact, yw, ( x + h_points ) >> x_fact ) ;
     HORIZONTAL ( x >> x_fact, y2w, x2 ) ;      /* compres again and plot */
    }
   else    /* delta_x / delta_y = 1: INTEGER !  maybe 1.99999 as a REAL ! */
    {                                    /* plot as diagonal lines */
     aant_str = delta_x - delta_y + 1 ;        /* number of DIAGONAL lines */
     h_points = (( delta_x << y_fact ) / aant_str-- ) - ( cor_fc ) ;
     hx_points = sgn_x == 1 ? h_points : -h_points ;       /* direction x */
     hy_points = sgn_y == 1 ? h_points : -h_points ;       /* direction y */
     x = x1 << y_fact ;                                  /* expand factors */
     yw = y1w << y_fact ;
     sgn_x <<= y_fact ;
     while ( aant_str )
```

```
      {
       --aant_str ;                    /* plot delta_x - delta_y diagonals */
       DIAGONAL ( x >> y_fact , yw >> y_fact,
                 x + hx_points >> y_fact, yw + hy_points >> y_fact ) ;
       x += hx_points + sgn_x ;
       yw += hy_points ;
      }
     DIAGONAL ( x >> y_fact, yw >> y_fact, x2, y2w ) ;  /* plot last one */
    }
 }
else  /* delta_x < delta_y */
                          /* the same whole story for delta_y > delta_x */
 {
  if ( delta_y > delta_x << 1 )
   {
    hd_points = ( delta_y << y_fact ) / ++delta_x ;
    h_points = sgn_y == 1 ? hd_points : -hd_points ;
      for ( x = x1, yw = y1w << y_fact ;
            x != x2;
            x += sgn_x, yw += h_points )
         VERTICAL ( x, yw >> y_fact, ( yw + h_points ) >> y_fact ) ;
      VERTICAL ( x, yw >> y_fact, y2w ) ;
    }
   else  /* delta_y > delta_x ; delta_y / delta_x = 1 */
    {
     aant_str = 1 + delta_y - delta_x ;
     h_points = (( delta_y << y_fact ) / aant_str-- ) - ( cor_fc ) ;
     hx_points = sgn_x == 1 ? h_points : -h_points ;
     hy_points = sgn_y == 1 ? h_points : -h_points ;
     x = x1 << y_fact ;
     yw = y1w << y_fact ;
     sgn_y <<= y_fact ;
     while ( aant_str )
       {
        --aant_str ;
        DIAGONAL ( x >> y_fact , yw >> y_fact,
                  x + hx_points >> y_fact, yw + hy_points >> y_fact ) ;
        x += hx_points ;
        yw += hy_points + sgn_y ;
       }
      DIAGONAL ( x >> y_fact, yw >> y_fact, x2, y2w ) ;
     }
  }
}

/*=========================================================================*/
/* function PLOT_POINTS                                                    */
/* Take the coordinates from the array 'points' and calculate the screen-  */
/* coordinates with respect to the x- and y-offset of the figure on the    */
/* screen ( offs_x and offs_y ) and call the suitable function.            */
/*=========================================================================*/

PLOT_POINTS (aant)
int aant ;
{
 register int  x1, y1, x2, y2 ;
 int mp, mv_ind ;

 mp = 0 ;
 mv_ind = 0 ;
 do
 {
  mp += points[ mv_ind++ ] ;                  /* pointer to next offset */
  x1 = offs_x + points[ mv_ind++ ] ;          /* startpoint x           */
  y1 = offs_y + points[ mv_ind++ ] ;          /* startpoint y           */

  DOT ( x1, y1 ) ;                            /* move to this point x,y */
  while ( mv_ind < mp )            /* while index < next pointerposition */
  {
   x2 = offs_x + points [ mv_ind ] ;                          /* next x */
   y2 = offs_y + points [ mv_ind + 1 ] ;                      /* next y */
   if ( x2 == x1 )
     {
     if ( y2 == y1 )
        DOT ( x1, y1 ) ;                                /* dx == dy == 0 */
     else
        VERTICAL ( x1, y1, y2 ) ;
     }
   else
     {
     if ( y2 == y1 )
        HORIZONTAL ( x1, y1, x2 ) ;                  /* dy == 0; dx <> 0 */
```

```
      else
        {
        if ( abs ( x2 - x1 ) == abs ( y2 - y1 ) )
           DIAGONAL ( x1, y1, x2, y2 ) ;          /* abs(dx) == abs(dy) <> 0 */
        else
           HELLING ( x1, y1, x2, y2 ) ;           /* abs(dx) <> abs(dy) <> 0 */
        }
      }
    x1 = offs_x + points [ mv_ind++ ] ;             /* after plot : x1 = x2 */
    y1 = offs_y + points [ mv_ind++ ] ;             /* and y1 = y2          */
   }
  }
while ( points [ mv_ind ] && ( mv_ind < aant ) ) ;  /* while pointer <> 0 */
}


/*=========================================================================*/
/*                            MAIN LOOP                                    */
/*=========================================================================*/

main ( )

{
char *malloc() ;  /* GEMDOS function memory allocate number of characters */
long htmp ;                      /* pointer-location to allocated memeory */

offs_x = 510 ;         /* startoffset x: pointing to the figurepoint (0,0) */
offs_y = 300 ;                                                  /* idem y */
fysbld=(struct FIG *)Physbase(); /* BIOSfunction fysical RAM screen adres */
htmp  =  0X100 + malloc ( 0X8100 ) ;       /* reserve room for one screen */

  /* Mind a screen-image has to start on an address xxxxxx00 so some more */

  /* memory is allocated; the pointer is first set 0X100 inside the       */
  /* allocated memory after which,in the next line,the address is cut-off.*/

logbld = ( struct FIG *) ( htmp & 0XFFFFFF00 ) ;   /* cut-off at xxxxxx00 */
/* this method is not correct; gives a compilation-warning but works */
do
{
  beeld = beeld == fysbld ? logbld : fysbld ;/* switch the screen-pointer */
  Setscreen ( beeld, beeld == fysbld ? logbld : fysbld, -1 );
           /* switch screen                                               */
           /* screen-instructions will go to the invisible screen 'beeld' */

  printf ("%c%c",27,'E');                          /* clear screen 'beeld' */
  PLOT_POINTS ( aantal ) ;          /* plot the figure; always on 'beeld' */
  Setscreen ( beeld, beeld, -1 ) ;                 /* make 'beeld' visible */
  offs_x -= 25 ;                                  /* decrement the offset x */
  offs_y -= 4 ;                                   /* decrement the offset y */
}
while ( offs_x >= 60 ) ;                        /* go on while offset x > 60 */
do { } while ( Cconis( ) == 0 ) ;                              /* wait fot a key */
Setscreen ( fysbld, fysbld, -1 ) ;   /* leave screen in a normal position */
}

/*========================== END PROGRAM ==================================*/
```



Expansion of OHIO-DOS Extentions

by: Coen Boltjes vidibus: 400029830
Nw. Plantage 9
2611 XH Delft
The Netherlands

Translated by: Elja van der Veer

In the beginning one had to manage when a new file name had to be put into the directory. This had to take place with the help of the basic programme BEXEC*, which, however, could only be run if the BASIC-interpeter was loaded. With the extentions from Elektuur the DOS could take care of creating a file name in the directory if it didn't exist. This was quite an advancement, but it also entailed some drawbacks:

-When a wrong disk is put in the drive, the file can be stored on the wrong disk without warning.
-The number of tracks for the file is chosen to such an extent that it fits the file exactly. In extending files this may be very unpractical.

Fortunately our secretary Gert Klein also noticed this and he extended the DOS-commands with a create command. When adding a filename in the directory the first and the last track must be given up. CR NAME=beg,end (See issue 39).
The user of this command must take care that the various files do not overlap sothat searching for a free place may take a while. Searching for a place is simplified with the programme described below, as it sorts out the filenames in the order of increasing track number. It can either be implemented as a new DOS-Command (i.e. with RS of ReSequence), or as executable programme.

Looking at the main routine the operation will be clear at once. Reading, sorting, writing back the directory and additional: printing. The latter can also be left out. At the same time it becomes clear that the use of subroutines has advantages: easy to read, without getting lost in the how and why of tests and branches. Subroutines can also be used for other commands. In the sorting procedure a bubble-sort algorithm is used. All entries are compared with their successor and if this successor has a lower last-track number the two entries are exchanged. The whole directory is run through in this way and if no exchange has taken place the directory is sorted out.
The fact that the filenames are ordered does not necessarily mean that they appear in the directory successively. It is possible that empty entries occur, but this is no problem for our purpose. In case the sorting routine ALFA is used for other extentions this should be born in mind.

Extentions on OHIO-DOS V3.3 by Gert Klein which enables you to use new DOS-Commands as: Directory, Create and Delete files, Change filenames etc.
Complete english assembly source with introduction. Original for the Junior, but adjustment procedure for EC65(K) is included. Send cheque of Hfl 17.50 to the editoral office. (Euroceque 8.00)

I intend to write one ore more articles about the subject "Datacommunication". If there are members with specific questions dealing with this subject, please send these to the editoral office, so I can answer them in my article.
This is THE opportunity to ask for the diference between Bit- and Baudrate or Arithmetic- and Cyclic redundancy check.

Coen Boltjes

```
6850 00FE=       ENT1  =VEC
6860 00E4=       ENT2  =TEMP2
6870 00E0=       SAVEX =TEMP1
6880 00E6=       MOVCNT=TEMP4
6890             ;
6900 D699 2086D4 RESEQ   JSR TR12       ;SET HEAD ON TRACK 12
6910 D69C 20F8D1         JSR READDI     ;READ DIRECTORY
6920 D69F AE0023         LDX RAMLOC
6930 D6A2 CA             DEX
6940 D6A3 86E0           STX SAVEX
6950 D6A5 20AFD6         JSR ALFA       ;SEQUENCE DIRECTORY
6960 D6A8 2005D7         JSR DUMPDI     ;WRITE DIRECTORY ON DISK
6970 D6AB 20A3D1         JSR DIRECT+$B ;PRINT THE DIRECTORY
6980 D6AE 60             RTS
6990             ;
7000 D6AF A6E0   ALFA    LDX SAVEX      ;LOAD START OF DIRECT
7010 D6B1 86FF           STX ENT1+1
7020 D6B3 86E5           STX ENT2+1
7030 D6B5 A980           LDA #$80
7040 D6B7 85E6           STA MOVCNT     ;MOVE COUNTER
7050 D6B9 A900           LDA #$00
7060 D6BB 85FE           STA ENT1
7070 D6BD A908           LDA #$08
7080 D6BF 85E4           STA ENT2       ;INIT ENTITY POINTERS
7090 D6C1 A007   ALFA1   LDY #$07       ;LOADPOINTER TO LAST TRACKNR
7100 D6C3 B1E4           LDA (ENT2),Y   ;LOAD
7110 D6C5 F026           BEQ ALFA6      ;=>ENT2 EMPTY
7120 D6C7 B1FE           LDA (ENT1),Y
7130 D6C9 F006           BEQ ALFA2      ;=>ENT1 EMPTY
7140 D6CB D1E4           CMP (ENT2),Y   ;COMPARE TRACKS
7150 D6CD 3013           BMI ALFA5      ;=>TRACK1<TRACK2
7160 D6CF F011           BEQ ALFA5      ;=>TRACK1=TRACK2
7170 D6D1 A007   ALFA2   LDY #$07
7180 D6D3 B1E4   ALFA3   LDA (ENT2),Y   ;LOAD CHARACTER
7190 D6D5 AA             TAX            ;SAVE IT
7200 D6D6 B1FE           LDA (ENT1),Y
7210 D6D8 91E4           STA (ENT2),Y
7220 D6DA 8A             TXA
7230 D6DB 91FE           STA (ENT1),Y   ;EXCHANGE CHARACTERS
7240 D6DD 88             DEY
7250 D6DE 10F3           BPL ALFA3      ;=>EXCHANGE NOT COMPLETED
7260 D6E0 E6E6   ALFA4   INC MOVCNT     ;
7270 D6E2 18     ALFA5   CLC
7280 D6E3 A5FE           LDA ENT1
7290 D6E5 6908           ADC #$08
7300 D6E7 85FE           STA ENT1       ;ENT1 TO NEXT ENTRY
7310 D6E9 D002           BNE ALFA6      ;=>NO PART BOUNDARY
7320 D6EB E6FF           INC ENT1+1
7330 D6ED 18     ALFA6   CLC
7340 D6EE A5E4           LDA ENT2
7350 D6F0 6908           ADC #$08
7360 D6F2 85E4           STA ENT2       ;ENT2 TO NEXT ENTRY
7370 D6F4 D0CB           BNE ALFA1      ;=>PART NOT FINISHED
7380 D6F6 A5E6           LDA MOVCNT
7390 D6F8 1008           BPL ALFA7      ;=>BOTH PART FINISCHED
7400 D6FA 297F           AND #$7F       ;
7410 D6FC 85E6           STA MOVCNT
7420 D6FE E6E5           INC ENT2+1     ;
7430 D700 D0BF           BNE ALFA1      ;=>ALLWAYS (DO SECOND PART)
7440 D702 D0AB   ALFA7   BNE ALFA       ;=>NOT IN RIGHT SEQUENCE
7450 D704 60             RTS
7460             ;
7470 D705 AE0023 DUMPDI  LDX RAMLOC
7480 D708 CA             DEX
7490 D709 86FF           STX VEC+1      ;SET LOADVEC TO FIRST PART
7500 D70B A000           LDY #$00
7510 D70D 84FE           STY VEC        ;
7520 D70F C8             INY
7530 D710 8C5E26         STY SECTNU     ;SECTOR 01
7540 D713 8C5F26         STY PAGENU     ;1 PAGE
7550 D716 20E127         JSR DUMPSE     ;DUMP 12,1
7560 D719 EE5E26         INC SECTNU     ;SECTOR 02
7570 D71C 20E127         JSR DUMPSE     ;DUMP 12,2
7580 D71F 60             RTS
7590             ;
7600             ;
```

TR12, READDI en DIRECT are routines described in issue 39. The locations of this routines deviates.

```
: A S M
                    1   ****************************************************************
                    2   **                                                            **
                    3   **  COMPUTER: APPLE II WITH DOS 3.3 OR DIVERSI-DOS             **
                    4   **                                                            **
                    5   **  AUTHOR: M.J. VISSER                                       **
                    6   **          PASTOOR KONIJNSTRAAT 48                           **
                    7   **          1616 BX HOOGKARSPEL                               **
                    8   **                                                            **
                    9   ****************************************************************
                   10   *
                   11   * GLOBAL ADDRESSES
                   12   *
                   13   BEGIN    EQU  $1000      ;START OF ROUTINES
                   14   *
                   15   ****************************************************************
                   16   **                                                            **
                   17   **  LAUNCHER:                                                 **
                   18   **                                                            **
                   19   **    - CHECK FOR RIGHT DOS (DOS 3.3 OR DIVERSI-DOS)          **
                   20   **    - MAKE ROOM BETWEEN DOS AND ITS BUFFERS                 **
                   21   **    - RELOCATE THE ROUTINES INTO THIS NEW SPACE             **
                   22   **    - LINK THE USER VECTOR                                  **
                   23   **                                                            **
                   24   ****************************************************************
                   25   *
                   26   * LOCAL ADDRESSES
                   27   *
                   28   LENGTH   EQU  $2F        ;LENGTH OPCODE
                   29   TEMP     EQU  $3C
                   30   SRCPTR   EQU  $3C        ;SOURCE POINTER
                   31   SRCEND   EQU  $3E        ;END OF SOURCE POINTER
                   32   RELPOS   EQU  $40        ;REL. POSITION OF ADDRESS
                   33   TRGTPTR  EQU  $42        ;TARGET POINTER
                   34   TARGET   EQU  $44        ;ABSOLUTE TARGET ADDRESS
                   35   BYTES    EQU  $46        ;BUFFER FOR OPCODE
                   36   STACK    EQU  $100
                   37   WRMSTRT  EQU  $3D0       ;DOS WARMSTART ADDRESS
                   38   USR      EQU  $3F8
                   39   MAKEBUF  EQU  $A7D4      ;BUILDS DOS BUFFERS
                   40   INSDS2   EQU  $F88E      ;DETERMINES LENGTH OF OPCODE
                   41   NXTA4    EQU  $FCB4      ;INC SRCPTR & TRGTPTR AND CHECK END
                   42   *
                   43   * CHECK FOR DIVERSI-DOS OR DOS 3.3
                   44   * IF NOT FOUND THEN EXIT LAUNCHER
                   45   *
8000: 18           46   ENTRY    CLC             ;VERSION # IS AT OFFSET $16BE
8001: A9 BE        47            LDA  #$BE
8003: 85 3C        48            STA  TEMP
8005: AD D2 03     49            LDA  WRMSTRT+2
8008: 69 16        50            ADC  #$16
800A: 85 3D        51            STA  TEMP+1
800C: A0 00        52            LDY  #$00
800E: B1 3C        53            LDA  (TEMP),Y   ;VERSION
8010: C9 03        54            CMP  #3         ;IF DOS 3.3 THEN CONTINU
8012: F0 01        55            BEQ  MAKEROOM
8014: 60           56            RTS             ; ELSE EXIT
                   57   *
                   58   * MAKE ROOM FOR THE ROUTINE BETWEEN
                   59   * DOS AND ITS BUFFERS AND INITIALIZE
                   60   * THE TARGET POINTER.
                   61   *
8015: 98           62   MAKEROOM TYA             ;Y=0
8016: 85 3C        63            STA  TEMP       ;GET POINTER TO FIRST BUFFER
8018: AD D2 03     64            LDA  WRMSTRT+2  ;THIS POINTER IS LOCATED AT $9D00
801B: 85 3D        65            STA  TEMP+1     ; (48K APPLE)
801D: 38           66            SEC             ;POINTER := POINTER - LEN ROUTINE
801E: B1 3C        67            LDA  (TEMP),Y
8020: E9 B1        68            SBC  #(ENDDMP-BEGIN
8022: 91 3C        69            STA  (TEMP),Y   ;PUT THE NEW POINTER ON $9D00
8024: AA           70            TAX             ;AND SAVE IT IN THE X & Y REGISTERS
8025: C8           71            INY
8026: B1 3C        72            LDA  (TEMP),Y
8028: E9 00        73            SBC  #)ENDDMP-BEGIN
```

```
802A: 91 3C      74            STA  (TEMP),Y
802C: A8         75            TAY
802D: 18         76            CLC              ;INITIALIZE THE ABSOLUTE TARGET
802E: 8A         77            TXA              ; ADDRESS AND THE TARGET POINTER
802F: 69 26      78            ADC  #38         ; 38 BYTES ABOVE THE FIRST DOS
8031: 85 44      79            STA  TARGET      ; BUFFER.
8033: 85 42      80            STA  TRGTPTR
8035: 98         81            TYA
8036: 69 00      82            ADC  #00
8038: 85 45      83            STA  TARGET+1
803A: 85 43      84            STA  TRGTPTR+1
803C: 20 D4 A7   85            JSR  MAKEBUF     ;REBUILD THE DOS BUFFERS
                 86  *
                 87  * RELOCATE THE ROUTINE INTO THE
                 88  * NEWLY CREATED SPACE.
                 89  *
803F: 18         90  NWBUF     CLC              ;WHERE AM I?
8040: BA         91            TSX              ; RETURN ADDRESS STILL ON STACK
8041: BD         92            DFB  $BD         ;LDA STACK-1,X
8042: FF 00      93            DFB  $FF,$00
8044: 69 7E      94            ADC  #(ENDIT-NWBUF+1 ;CALC THE POSITION OF THE
8046: 85 3C      95            STA  SRCPTR      ; ROUTINE AND STORE THIS ADDRESS
8048: BD 00 01   96            LDA  STACK,X     ; IN SRCPTR
804B: 69 00      97            ADC  #)ENDIT-NWBUF+1
804D: 85 3D      98            STA  SRCPTR+1
804F: 18         99            CLC              ;CALCULATE THE END POSITION OF THE
8050: A5 3C      100           LDA  SRCPTR      ;ROUTINES BY ADDING THE LENGTH TO
8052: 69 B1      101           ADC  #(ENDDMP-BEGIN ;SRCPTR AND STORE THIS ADDRESS
8054: 85 3E      102           STA  SRCEND      ;IN SRCEND
8056: A5 3D      103           LDA  SRCPTR+1
8058: 69 00      104           ADC  #)ENDDMP-BEGIN
805A: 85 3F      105           STA  SRCEND+1
805C: A0 02      106 RELOCATE  LDY  #$02        ;MOVE 3 BYTES FROM SOURCE INTO
805E: B1 3C      107 TAKE3BYT  LDA  (SRCPTR),Y ;BUFFER
8060: 99 46 00   108           STA  BYTES,Y
8063: 88         109           DEY
8064: 10 F8      110           BPL  TAKE3BYT
8066: 20 8E F8   111           JSR  INSDS2      ;DETERMINE THE LENGTH OF THE OPCODE
8069: A6 2F      112           LDX  LENGTH      ;LENGTH = LENGTH -1
806B: E0 02      113           CPX  #$02        ;IF ABSOLUTE ADDRESSING THEN
806D: D0 25      114           BNE  MOVEBYTS
806F: A9 B1      115           LDA  #(ENDDMP    ; IF ADDRESS ) END THEN MOVE
8071: C5 47      116           CMP  BYTES+1
8073: A9 10      117           LDA  #)ENDDMP
8075: E5 48      118           SBC  BYTES+2
8077: 90 1B      119           BCC  MOVEBYTS
8079: A5 47      120           LDA  BYTES+1     ; IF ADDRESS ( BEGIN THEN MOVE
807B: E9 00      121           SBC  #(BEGIN
807D: 85 40      122           STA  RELPOS
807F: A5 48      123           LDA  BYTES+2
8081: E9 10      124           SBC  #)BEGIN
8083: 85 41      125           STA  RELPOS+1
8085: 90 0D      126           BCC  MOVEBYTS
8087: 18         127           CLC              ; ELSE
8088: A5 40      128           LDA  RELPOS      ;  RELOCATE THE ABSOLUTE ADDRESS
808A: 65 44      129           ADC  TARGET
808C: 85 47      130           STA  BYTES+1
808E: A5 41      131           LDA  RELPOS+1
8090: 65 45      132           ADC  TARGET+1
8092: 85 48      133           STA  BYTES+2
8094: A2 00      134 MOVEBYTS  LDX  #$00        ;MOVE LENGTH BYTES TO TARGET
8096: B5 46      135 MOVE      LDA  BYTES,X
8098: 91 42      136           STA  (TRGTPTR),Y
809A: E8         137           INX
809B: 20 B4 FC   138           JSR  NXTA4       ;INC SOURCE AND TARGET POINTERS,
809E: C6 2F      139           DEC  LENGTH      ; AT END CARRY IS SET
80A0: 10 F4      140           BPL  MOVE
80A2: 90 B8      141           BCC  RELOCATE    ;UNTIL AT END
                 142 *
                 143 * LINK THE USER ROUTINE TO THE
                 144 * USER VECTOR
                 145 *
80A4: A0 0A      146           LDY  #$0A        ;LINK THE PREVIOUS ROUTINE
80A6: AD F9 03   147           LDA  USR+1       ; TO THE CURRENT ONE
80A9: 91 44      148           STA  (TARGET),Y
80AB: C8         149           INY
```

```
80AC: AD FA 03  150           LDA  USR+2
80AF: 91 44     151           STA  (TARGET),Y
80B1: A5 44     152           LDA  TARGET     ;REROUTE THE USER VECTOR
80B3: 8D F9 03  153           STA  USR+1
80B6: A5 45     154           LDA  TARGET+1
80B8: 8D FA 03  155           STA  USR+2
80BB: 60        156           RTS
                157  ENDIT    EQU  *
                158  *
                159  ************************************************************
                160  **                                                        **
                161  **  HEX/ASCII DUMP:                                       **
                162  **                                                        **
                163  **    - CHECK USER CALL ('H')                             **
                164  **    - CHECK BEGIN ( END                                 **
                165  **    - GET THE SCREEN LENGTH                             **
                166  **    - DUMP THE MEMORY RANGE                             **
                167  **                                                        **
                168  ************************************************************
                169  *
                170  * LOCAL ADDRESSES
                171  *
                172  LEN       EQU  $2F          ;BYTES PER LINE
                173  YSAVE     EQU  $34          ;POSITION ON INPUT BUFFER
                174  START     EQU  $3C          ;START ADDRESS
                175  END       EQU  $3E          ;END ADDRESS
                176  SKIPSTRT  EQU  $40          ;SKIP # BYTES
                177  SKIPEND   EQU  $41          ;SKIP # BYTES
                178  IN        EQU  $200         ;INPUT BUFFER
                179  TRUEOUT   EQU  $AA53        ;TRUE OUTPUT VECTOR
                180  PR3BLANK  EQU  $F948        ;PRINT 3 BLANKS
                181  PRBLANK   EQU  $F94A        ;PRINT X BLANKS
                182  PRBLANK2  EQU  $F94C        ;PRINT A, FOLOWED BY X BLANKS
                183  PRSTART   EQU  $FD92        ;PRINT START ADDRESS
                184  PRBYTE    EQU  $FDDA        ;PRINT A AS HEX
                185  COUT      EQU  $FDED        ;OUTPUT ROUTINE
                186  RTS1      EQU  $FE17        ;FIXED RTS INSTRUCTION
                187  *
                188           ORG  BEGIN
                189  *
                190  * CHECK THE CHARACTER FOLOWING THE ^Y
                191  * IF IT IS A 'H' THEN THIS ROUTINE IS REQUESTED
                192  *
1000: A4 34     193  HEXDUMP  LDY  YSAVE        ;GET THE NEXT CHAR
1002: B9 00 02  194           LDA  IN,Y
1005: C9 C8     195           CMP  #"H          ;IF IT ISN'T A 'H' THEN
1007: F0 03     196           BEQ  HEXDMP2      ; TRY NEXT ROUTINE
1009: 4C 17 FE  197           JMP  RTS1
100C: C8        198  HEXDMP2  INY               ;ADJUST INPUT BUFFER
100D: 84 34     199           STY  YSAVE
                200  *
                201  * CHECK BEGIN <= END
                202  *
100F: A5 3E     203           LDA  END          ;IF END ( BEGIN THEN
1011: C5 3C     204           CMP  START        ; STOP
1013: A5 3F     205           LDA  END+1
1015: E5 3D     206           SBC  START+1
1017: B0 01     207           BCS  CHKSCRN
1019: 60        208  ATEND    RTS
                209  *
                210  * CHECK WHETHER 40 OR 80 COL.
                211  * AND INITIALIZE POINTERS
                212  *
101A: A9 0F     213  CHKSCRN  LDA  #$0F         ;IF OUTPUT VECTOR = $FDXX THEN
101C: AC 54 AA  214           LDY  TRUEOUT+1    ; LEN = 8
101F: C0 FD     215           CPY  #$FD         ; ELSE
1021: D0 01     216           BNE  EIGHTY       ;   LEN = 16
1023: 4A        217           LSR
1024: 85 2F     218  EIGHTY   STA  LEN
1026: E6 2F     219           INC  LEN
1028: 25 3C     220           AND  START        ;BYTES TO SKIP ON FIRST LINE
102A: 85 40     221           STA  SKIPSTRT     ; (START MOD LEN)
102C: 38        222           SEC               ;ROUND STARTING ADDRESS
102D: A5 3C     223           LDA  START        ; (START-START MOD LEN)
102F: E5 40     224           SBC  SKIPSTRT
1031: 85 3C     225           STA  START
                226  *
```

```
                        227  * DUMP MEMORY RANGE
                        228  *
1033: 38                229  LOOP     SEC              ;IF START ) END THEN
1034: A5 3E             230  LOOP2    LDA  END         ; AT END
1036: F5 3C             231           SBC  START
1038: AA                232           TAX
1039: A5 3F             233           LDA  END+1
103B: E5 3D             234           SBC  START+1
103D: 90 DA             235           BCC  ATEND
103F: D0 05             236           BNE  NORMEND
1041: E4 2F             237           CPX  LEN         ;IF END-START ( LEN THEN
1043: E8                238           INX              ;  X := END - START
1044: 90 02             239           BCC  SPECEND     ; ELSE
1046: A6 2F             240  NORMEND  LDX  LEN         ;  X := LEN
1048: 86 41             241  SPECEND  STX  SKIPEND
104A: 20 92 FD          242           JSR  PRSTART     ;PRINT START ADDRESS
104D: A4 40             243           LDY  SKIPSTRT    ;SKIP BYTES
104F: F0 08             244           BEQ  PRBYTES
1051: 98                245           TYA              ; (3 SPACES PER BYTE)
1052: 0A                246           ASL
1053: 65 40             247           ADC  SKIPSTRT
1055: AA                248           TAX
1056: 20 4A F9          249           JSR  PRBLANK     ;PRINT THE SPACES
1059: A9 A0             250  PRBYTES  LDA  #"          ;PRINT THE BYTES
105B: 20 ED FD          251           JSR  COUT
105E: B1 3C             252           LDA  (START),Y
1060: 20 DA FD          253           JSR  PRBYTE
1063: C8                254           INY
1064: C4 41             255           CPY  SKIPEND     ;UNTIL AT END OF LINE
1066: 90 F1             256           BCC  PRBYTES
1068: B0 04             257           BCS  ENDSPC
106A: 20 48 F9          258  SPC2     JSR  PR3BLANK    ;SKIP THE LAST BYTES
106D: C8                259           INY              ; (LAST LINE ONLY)
106E: C4 2F             260  ENDSPC   CPY  LEN
1070: 90 F8             261           BCC  SPC2
1072: A9 A0             262           LDA  #"          ;PRINT ' : '
1074: 20 ED FD          263           JSR  COUT
1077: A9 BA             264           LDA  #":
1079: A6 40             265           LDX  SKIPSTRT    ;AND SKIP START BYTES
107B: E8                266           INX
107C: 20 4C F9          267           JSR  PRBLANK2
107F: A4 40             268           LDY  SKIPSTRT    ;PRINT THE ASCII VALUES
1081: B1 3C             269  PRASCII  LDA  (START),Y
1083: 09 80             270           ORA  #%10000000
1085: C9 A0             271           CMP  #"          ;CTRL CHARS ARE REPRESENTED
1087: B0 02             272           BCS  PRASC       ; AS '.'
1089: A9 AE             273           LDA  #".
108B: 20 ED FD          274  PRASC    JSR  COUT
108E: C8                275           INY
108F: C4 41             276           CPY  SKIPEND     ;UNTIL AT END OF LINE
1091: 90 EE             277           BCC  PRASCII
1093: B0 06             278           BCS  SPC4
1095: A9 A0             279  SPC5     LDA  #"          ;SKIP THE LAST BYTES
1097: 20 ED FD          280           JSR  COUT        ; (LAST LINE ONLY)
109A: C8                281           INY
109B: C4 2F             282  SPC4     CPY  LEN
109D: 90 F6             283           BCC  SPC5
109F: A9 00             284           LDA  #$00        ;CLEAR SKIPSTRT
10A1: 85 40             285           STA  SKIPSTRT
10A3: 18                286           CLC              ;CALC NEXT START ADDRESS
10A4: A5 3C             287           LDA  START
10A6: 65 2F             288           ADC  LEN
10A8: 85 3C             289           STA  START
10AA: 90 87             290           BCC  LOOP
10AC: E6 3D             291           INC  START+1
10AE: B0 84             292           BCS  LOOP2
10B0: 60                293           RTS
                        294  ENDDMP   EQU  *
```


```
--End assembly--

  365 bytes

  Errors: 0
```

```
]LIST

0  GOTO 100
7  REM
8  REM *** SCREEN CONTROL ROUTINE
     S ***
9  REM
10  PRINT  CHR$ (12);: HOME : RETURN
     : REM CLEAR SCREEN
20  PRINT  CHR$ (29);: CALL  - 86
     8: RETURN : REM CLREOL
30  PRINT  CHR$ (25);: VTAB VT: HTAB
     HT: RETURN : REM POSITION CU
     RSOR
40  PRINT  CHR$ (25);: VTAB 1: HTAB
     1: GOSUB 20: RETURN : REM CL
     EAR TOP LINE
50  PRINT  CHR$ (15);: INVERSE : RETURN
60  PRINT  CHR$ (14);: NORMAL : RETURN

70 VT = 24:HT = 1: GOSUB 30: GOSUB
     20: PRINT O$;: REM COMMAND-
     + BOTTOMLINE
80  GOSUB 40: PRINT CL$;: RETURN
     : REM SHOW COMMANDLINE
97  REM
98  REM * * * INITIALIZE * * *
99  REM
100  GOSUB 20000: REM INITIALIZE
110 C = 2: GOSUB 2010: REM GET AD
     DRESSES
197  REM
198  REM * * * MAIN PROGRAM * * *

199  REM
200  FOR M = FALSE TO TRUE: REM
     REPEAT
220 CL$ = "HEXDUMP: D(UMP N(EW O(
     UTPUT Q(UIT?"
230  GOSUB 70: REM COMMANDLINE +
     BOTTOMLINE
235 C = 0
240  FOR I = FALSE TO TRUE
250  GET R$: REM  GET COMMAND
260  FOR J = 1 TO 5: REM  CORRECT
      COMMAND?
270  IF R$ =  MID$ ("DNOQ?",J,1) THEN
     C = J:J = 5
280  NEXT J
290 I = (C ) 0): REM  UNTIL CORRE
     CT COMMAND
300  NEXT I
310  ON C GOSUB 1000,2000,3000,40
     00,5000: REM  PROCESS COMMAN
     D
320 M = FALSE: REM UNTIL FALSE
330  NEXT M
997  REM
998  REM * * * DUMP * * *
999  REM
1000 CL$ = "DUMP: ": IF  NOT PR THEN
     CL$ = CL$ + "(ARROWS) MOVE P
     AGE "
1010 CL$ = CL$ + "(ESC) ESCAPES?"

1020  GOSUB 10: REM HOME
1030  GOSUB 70: REM COMMANDLINE +
     BOTTOMLINE
1040 PL = 160: REM PAGE LENGTH
1050 LL = 8: REM LINE LENGTH
1060  IF S80 THEN PL = 320:LL = 1
     6: REM 80-COLUMN
1070  IF PR THEN  PRINT PR$:PL =
     16:LL = 16: REM PRINTER
1080 STRT = B
1090 EN =  INT ((STRT + PL) / LL)
      * LL - 1
1100  IF EN ) E THEN EN = E
1110  FOR D = FALSE TO TRUE
1120  GOSUB 12000: REM DUMP
1130  IF PR THEN R$ = " ": IF  PEEK
     ( - 16384) = 155 THEN R$ =  CHR$
     (27)
1140  IF PR AND (EN = E) THEN R$ =
      CHR$ (27)
1150  IF  NOT PR THEN  GOSUB 70: FOR
     J = FALSE TO TRUE: GET R$:J =
     (R$ =  CHR$ (8)) OR (R$ =  CHR$
     (21)) OR (R$ =  CHR$ (27)) OR
     (R$ = " ") OR (R$ = "?"): NEXT
     J
1160  IF R$ = " " OR R$ =  CHR$ (
     21) AND  NOT (EN = E) THEN S
     TRT = EN + 1
1170  IF R$ =  CHR$ (8) AND  NOT
     (STRT = B) THEN STRT = STRT -
     PL: IF STRT ( B THEN STRT =
     B
1180  IF R$ = "?" THEN  GOSUB 500
     0: REM SHOW HELP PAGE
1190 EN =  INT ((STRT + PL) / LL)
      * LL - 1: IF EN ) E THEN EN
      = E
1200 D = (R$ =  CHR$ (27)): REM U
     NTIL ESC
1210  NEXT D
1220  PRINT SC$: REM RETURN TO SC
     REEN OUTPUT
1230  RETURN
1997  REM
1998  REM * * * NEW ADDRESS * * *

1999  REM
2000  GOSUB 10: REM HOME
2010 CL$ = "NEW: $HEX, DEC (ESC)
     ESCAPES?"
2020  GOSUB 70: REM COMMANDLINE +
      BOTTOMLINE
2030  FOR N = FALSE TO TRUE
2040  FOR NN = FALSE TO TRUE
2050 VT = 3:HT = 1: GOSUB 30: REM
     POSITION CURSOR
2060 NM = B: GOSUB 11000: REM DEC
     -HEX CONV.
2070  PRINT "BEGIN ADDRESS (";S$;
     "): ";
2080  GOSUB 14000: REM INPUT LINE

2090 NN =  NOT HLP: IF HLP THEN  GOSUB
     5000: GOSUB 10: GOSUB 70: REM
     SHOW HELP PAGE
2100  NEXT NN
2110 N = ESC: IF N THEN  NEXT N: RETURN
2120 B2 = AD
2130  IF J = 1 THEN B2 = B: PRINT
     S$;: REM DEFAULT VALUE
2140  FOR NN = FALSE TO TRUE
2150  PRINT :NM = E: GOSUB 11000:
      REM DEC-HEX CONV.
2160  PRINT "END ADDRESS (";S$;")
     :  ";
2170 VT = 4: GOSUB 14000: REM INP
     UT LINE
2180 NN =  NOT HLP: IF HLP THEN  GOSUB
     5000: GOSUB 10: GOSUB 70:VT =
     3:HT = 1: GOSUB 30:NM = B: GOSUB
     11000: PRINT "BEGIN ADDRESS
```

```
(";S$;"): ";:NM = B2: GOSUB
11000: PRINT S$;: REM SHOW H
ELP PAGE
2190  NEXT NN
2200 N = ESC: IF N THEN  NEXT N: RETURN
2210 E2 = AD: IF J = 1 THEN E2 =
E: PRINT S$;: REM DEFAULT VA
LUE
2220 N = (E2 )  = B2): IF  NOT N THEN
ERR = 3: GOSUB 13000: REM RA
NGE ERROR
2230  NEXT N
2240 B = B2:E = E2
2250  GOSUB 10000: REM MAKE BOTTO
MLINE
2260  RETURN
2997  REM
2998  REM * * * OUTPUT * * *
2999  REM
3000 CL$ = "OUTPUT: S(CREEN P(RIN
TER (ESC) ESCAPES?"
3010  GOSUB 10: REM HOME
3020  GOSUB 70: REM COMMANDLINE +
 BOTTOMLINE
3030  FOR O = FALSE TO TRUE
3040  GET R$
3050  IF R$ = "?" THEN  GOSUB 500
0: GOSUB 80: REM SHOW HELP P
AGE
3060 O = (R$ = "S") OR (R$ = "P")
 OR (R$ =  CHR$ (27))
3070  NEXT O
3080  IF R$ =  CHR$ (27) THEN  RETURN

3090 PR = (R$ = "P")
3100  GOSUB 10000: REM MAKE BOTTO
MLINE
3110  RETURN
3997  REM
3998  REM * * * QUIT * * *
3999  REM
4000 CL$ = "QUIT: B(ASIC M(ONITOR
 (ESC) ESCAPES?"
4010  GOSUB 10: REM HOME
4020  GOSUB 70: REM COMMANDLINE +
 BOTTOMLINE
4030  FOR Q = FALSE TO TRUE
4040  GET R$
4050  IF R$ = "?" THEN  GOSUB 500
0: GOSUB 80: REM SHOW HELP P
AGE
4060 Q = (R$ = "B") OR (R$ = "M")
 OR (R$ =  CHR$ (27))
4070  NEXT Q
4080  IF R$ =  CHR$ (27) THEN  RETURN

4090  POP
4100  IF R$ = "M" THEN  CALL  - 1
51: REM GOTO MONITOR
4110  END : REM  GOTO BASIC
4997  REM
4998  REM * * * HELP SCREENS * *
*
4999  REM
5000  GOSUB 10: REM HOME
5010  GOSUB 70: REM COMMANDLINE +
 BOTTOMLINE
5020 VT = 3:HT = 1: GOSUB 30: REM
POSITION CURSOR
5030  ON C GOSUB 5100,5200,5300,5
400,5500
5040  IF C = 1 OR C = 2 THEN  PRINT
 TAB( 5);"PRESS (SPACEBAR) T
O CONTINUE ";: FOR H = FALSE
 TO TRUE: GET R$:H = (R$ = "
"): NEXT H
5050  RETURN
5099  REM *** DUMP HELP SCREEN **
*
5100  PRINT  TAB( 11);"* * * DUMP
 * * *"
5105  PRINT
5110  PRINT "THIS COMMAND DUMPS T
HE CONTENTS OF"
5115  PRINT " A MEMORY RANGE TO T
HE SCREEN OR"
5120  PRINT " PRINTER."
5125  PRINT
5130  PRINT "WHEN SCREEN OUTPUT I
S SELECTED, YOU"
5135  PRINT " CAN DUMP THE CONTEN
TS BY PAGE."
5140  PRINT " A PAGE IS ";PL;" BY
TES LONG."
5145  PRINT
5150  PRINT "YOU CAN GO FORWARD B
Y PRESSING THE"
5155  PRINT " FORWARD-ARROW OR TH
E SPACEBAR. THE"
5160  PRINT " BACKWARD-ARROW MOVE
S YOU BACKWARDS."
5165  PRINT
5170  PRINT "PRESSING THE (ESC)-K
EY RETURNS YOU"
5175  PRINT " TO THE MAIN COMMAND
-LEVEL."
5180  PRINT : PRINT : PRINT
5185  RETURN
5199  REM *** NEW HELP SCREEN ***

5200  PRINT  TAB( 7);"* * * NEW A
DDRESSES * * *"
5205  PRINT
5210  PRINT "THIS COMMAND LETS YO
U CHANGE THE"
5215  PRINT " MEMORY-RANGE."
5220  PRINT
5225  PRINT "YOU CAN ENTER THE ST
ARTING AND ENDING"
5230  PRINT " ADDRESSES IN EITHER
 HEXADECIMAL OR"
5235  PRINT " DECIMAL NOTATION. H
EXADECIMAL"
5240  PRINT " NOTATION MUST BE PR
ECEEDED BY A"
5245  PRINT " DOLLAR-SIGN ($).
5250  PRINT
5255  PRINT "YOU CAN EDIT YOUR IN
PUT WITH THE"
5260  PRINT " BACKWARD-ARROW."
5265  PRINT
5270  PRINT "PRESSING THE (ESC)-K
EY RETURNS YOU"
5275  PRINT " TO THE MAIN COMMAND
-LEVEL."
5280  PRINT : PRINT : PRINT
5285  RETURN
5300  PRINT  TAB( 8);"* * * OUTPU
T SLOT * * *"
5305  PRINT
5310  PRINT "THIS COMMAND LETS YO
U CHANGE THE OUTPUT"
5315  PRINT " SLOT. YOU CAN CHOOS
E BETWEEN THE"
5320  PRINT " SCREEN OR THE PRINT
ER, BY PRESSING"
5325  PRINT " 'S' OR 'P'."
5330  PRINT
5335  PRINT "PRESSING THE (ESC)-K
EY RETURNS YOU"
5340  PRINT " TO THE MAIN COMMAND
-LEVEL."
```

```
5345  RETURN
5400  PRINT  TAB( 11);"* * * QUIT
    * * *"
5405  PRINT
5410  PRINT "THIS COMMAND LETS YO
    U QUIT THE PROGRAM."
5415  PRINT " YOU CAN EXIT INTO B
    ASIC OR MONITOR"
5420  PRINT " BY PRESSING 'B' OR
    'M'."
5425  PRINT
5430  PRINT "PRESSING THE (ESC)-K
    EY RETURNS YOU"
5435  PRINT " TO THE MAIN COMMAND
    -LEVEL."
5440  RETURN
5499  REM *** MAIN HELP SCREEN **
    *
5500  PRINT  TAB( 6);"* * *  HEX/
    ASCII DUMP  * * *"
5505  PRINT
5510  PRINT "D(UMP THE HEXADECIMA
    L CONTENTS OF THE"
5515  PRINT "  MEMORY RANGE WITH
    THEIR ASCII-VALUES"
5520  PRINT "  TO THE SCREEN OR P
    RINTER"
5525  PRINT
5530  PRINT "N(EW MEMORY RANGE (D
    ECIMAL OR HEX"
5535  PRINT "  NOTATION)"
5540  PRINT
5545  PRINT "O(UTPUT SLOT, EITHER
    SCREEN OR PRINTER"
5550  PRINT
5555  PRINT "Q(UIT PROGRAM, EXITI
    NG IN BASIC OR"
5560  PRINT "  MONITOR"
5565  RETURN
9999  REM * * * MAKE BOTTOMLINE *
    * *
10000 O$ = "BEGIN:"
10010 NM = B: GOSUB 11000:O$ = O$
    + S$
10020  IF S80 THEN O$ = O$ + " ("
    +  STR$ (B) + ")"
10030 O$ = O$ + "  END:"
10040 NM = E: GOSUB 11000:O$ = O$
    + S$
10050  IF S80 THEN O$ = O$ + " ("
    +  STR$ (E) + ")"
10060 O$ = O$ + "  OUTPUT:" + OP$
    (PR)
10070  RETURN
10999  REM * * * DEC/HEX CONVERSI
    ON * * *
11000 S$ = "$": IF NM < 0 THEN NM
    = 65536 + NM
11020 J = 4096: FOR H = 0 TO 3:D =
     INT (NM / J):S$ = S$ +  MID$
    ("0123456789ABCDEF",D + 1,1)
    :NM = NM - D * J:J = J / 16:
     NEXT H: RETURN
11999  REM * * * DUMP CONTENTS *
    * *
12000 VT = 2:HT = 1: GOSUB 30: REM
    POSITION CURSOR
12010  POKE 60,STRT -  INT (STRT /
    256) * 256: POKE 61, INT (ST
    RT / 256)
12020  POKE 62,EN -  INT (EN / 25
    6) * 256: POKE 63, INT (EN /
    256)
12030  POKE 52,0: POKE 512,200: REM
    CTRL-Y AS CHAR IN INPUT BUFF
    ER. BUFFERCOUNT IS ZERO
12040  CALL 1016: REM USER-VECTOR
    JUMP ADDRESS
12050  IF  NOT PR THEN  FOR JJ =
    1 TO (21 - (EN - STRT) / LL)
    : GOSUB 20: PRINT : NEXT : REM
    CLEAR TO END OF SCREEN
12060  RETURN
12999  REM * * * SHOW ERROR MESSA
    GE * * *
13000  GOSUB 40: REM CLEAR TOP LI
    NE
13010  IF ERR = 3 THEN  PRINT "ER
    R. IN RANGE SPECIFICATION. P
    RESS (SP)";
13020  IF ERR = 53 THEN  PRINT "I
    LLEGAL QUANTITY. PRESS (SPAC
    EBAR) ";
13030  IF ERR = 254 THEN  PRINT "
    ILL CHAR IN INPUT RESPONSE.
    PRESS (SP) ";
13040  FOR JJ = FALSE TO TRUE: GET
    R$:JJ = (R$ = " "): NEXT JJ
13050  GOSUB 80: REM SHOW COMMAND
    LINE
13060  RETURN
13999  REM * * * INPUT LINE * * *

14000 AD = 0:CR = FALSE:ESC = FAL
    SE:HLP = FALSE:J = 1
14010  FOR IN = FALSE TO TRUE
14020  POKE  - 16384,0: GET R$
14030  GOSUB 14100
14040 IN = CR OR ESC OR HLP
14050  NEXT IN
14060  RETURN
14100  IF J = 1 THEN HX = (R$ = "
    $"):NEG = (R$ = "-"): IF HX THEN
     PRINT "$";:J = 2: RETURN
14110  IF J = 1 AND NEG THEN  PRINT
    "-";:J = 2: RETURN
14120  IF J = 1 AND R$ = "+" THEN
     PRINT "+";:J = 2: RETURN
14130  IF R$ =  CHR$ (8) AND J >
    1 THEN J = J - 1: PRINT  CHR$
    (8);" "; CHR$ (8);: ON HX +
    1 GOSUB 14600,14650: RETURN

14140  IF R$ =  CHR$ (13) THEN CR
     = TRUE: ON NEG GOSUB 14700:
     RETURN
14150  IF R$ =  CHR$ (27) THEN ES
    C = TRUE: RETURN
14160  IF R$ = "?" THEN HLP = TRU
    E: RETURN
14170  ON HX + 1 GOSUB 14500,1455
    0
14180  IF AD > 65535 THEN  ON HX +
    1 GOSUB 14600,14650:ERR = 53
    : GOSUB 13000:HT = 23 + J: GOSUB
    30: PRINT  CHR$ (8);" "; CHR$
    (8);:J = J - 1: REM ILL. QUA
    NTITY
14190  RETURN
14500  IF (R$ >  = "0" AND R$ <  =
    "9") THEN  PRINT R$;:AD = AD
     * 10 +  VAL (R$):J = J + 1:
     RETURN
14510 ERR = 254: GOSUB 13000:HT =
    23 + J: GOSUB 30: PRINT  CHR$
    (0);: RETURN
14550  IF (R$ >  = "0" AND R$ <  =
    "9") OR (R$ >  = "A" AND R$ <
     = "F") THEN  PRINT R$;:AD =
    AD * 16 +  VAL (R$) + (R$ >
     = "A" AND R$ <  = "F") * ( ASC
```

```
     (R$) - 55):J = J + 1: RETURN

14560 ERR = 254: GOSUB 13000:HT =
     23 + J: GOSUB 30: PRINT  CHR$
     (0);: RETURN : REM ILL. CHAR

14600 AD =  INT (AD / 10): RETURN

14650 AD =  INT (AD / 16): RETURN

14700 AD = 65536 - AD: RETURN
19999  REM  * * * INITIALIZE * *
     *
20000  GOSUB 10
20010  VTAB 6
20020  PRINT  TAB( 5);"##########
     ####################"
20030  PRINT  TAB( 5);"#"; SPC( 2
     8);"#"
20040  PRINT  TAB( 5);"#       HE
     X/ASCII DUMP         #"
20050  PRINT  TAB( 5);"#"; SPC( 2
     8);"#"
20060  PRINT  TAB( 5);"#        BY
      M.J. VISSER         #"
20070  PRINT  TAB( 5);"#"; SPC( 2
     8);"#"
20080  PRINT  TAB( 5);"##########
     ####################"
20090  VTAB 20
20100  PRINT  TAB( 14);: GOSUB 50
     : PRINT "INITIALIZING": GOSUB
     60
20190  REM
20191  REM SURCH HEXDUMP.OBJ
20192  REM
20193  REM HEXDUMP.OBJ IS A 'LINK
     ED LIST' ROUTINE. ITS CODE S
     TARTS AS FOLLOWS:
20194  REM
20195  REM  LDY $34    ;BUFFER COU
     NT
20196  REM  LDA $200,Y;FETCH CHAR

20197  REM  CMP #$C8  ;"H"?
20198  REM  BEQ START ;YES -) DUM
     P
20199  REM  JMP NEXT  ;CHECK NEXT

20200 STRT = 1007: REM  USER VECT
     OR -10
20210  FOR I = 0 TO 1: REM  REPEA
     T
20220 STRT =  PEEK (STRT + 10) +
      PEEK (STRT + 11) * 256: REM
      NEXT ROUTINE
20230 I = ( PEEK (STRT + 6) = 200
     ) OR (STRT = 65381): REM  UN
     TIL "H" OR NO MORE ROUTINES
20240  NEXT I
20250  IF STRT = 65381 THEN  PRINT
      CHR$ (13); CHR$ (4);"BRUN H
     EXDUMP.OBJ": REM LOAD OBJ CO
     DE
20297  REM
20298  REM INITIALIZE CONSTANTS
20299  REM
20300 FALSE = 0:TRUE = 1
20310  DIM OP$(1): REM OUTPUT POR
     T
20320 OP$(0) = "SCREEN":SC$ =  CHR$
     (13) +  CHR$ (4) + "PR#3": REM
     SCREEN OUTPUT
20330 OP$(1) = "PRINTER":PR$ =  CHR$
     (13) +  CHR$ (4) + "PR#1": REM
     PRINTER OUTPUT
20340 PR = FALSE: REM PRINTER OFF

20350 S80 = ( PEEK (43604) <  > 2
     53): REM 80-COL OUTPUT
20360  IF  NOT S80 THEN SC$ =  CHR$
     (13) +  CHR$ (4) + "PR#0"
20370  GOSUB 10000: REM INIT BOTT
     OMLINE
20400  VTAB 20
20410  PRINT  TAB( 11);: GOSUB 50
     : PRINT "PRESS (?) FOR HELP"
     : GOSUB 60
20420  RETURN
```

THIS ROUTINES ARE ALSO AVAILABLE ON DISKETTE. FOR THOSE WHO WANT TO SAVE TIME OR TO AVOID TYPE ERRORS. SEND YOUR EUROCHEQUE TO THE AMOUNT OF HFL. 25,= TO MR. W.L. VAN PELT, JAC. JORDAENSSTRAAT 15, 2923 CK KRIMPEN A/D IJSSEL, THE NETHERLANDS.
DISKETTES ARE FORMATTED FOR APPLE II DOS WITH THE FOLLOWING FILES:
HEXDUMP.BAS
HEXDUMP.OBJ (ALSO STAND ALONE USEABLE)
HEXDUMP ML.S (BIG MAG ASSEMBLER FORMAT)
AFTER 'BRUN HEXDUMP.OBJ' YOU ARE ABLE TO CALL THE ROUTINE IN MONITOR BY MEANS OF THE USER-VECTOR (CTRL-Y). USE FOLLOWING SYNTAX: xxxx.yyyy(CTRL-Y)H.
NO PAGE-GROUPING IS PROVIDED NOW! USE (CTRL-S) TO TEMPORARELY STOP TE OUTPUT. THE PROGRAM CHECKS WHETHER YOU ARE IN 80 COLUMN MODE (16 BYTES/LINE) OR IN 40 COLUMN MODE (8 BYTES/LINE).

SAMSON-65 OCTOPUS EC 65.
HOW TO GET MORE MEMORY-SPACE.

A Little modification of the CPU-board gives 1/2 k. more ram ($E200-$E3FF). As shown on fig. 1. is an extra or-gate between A9/A10 and N60 enough. A new IC16/N60 as an "piggy back" construction will do the job.

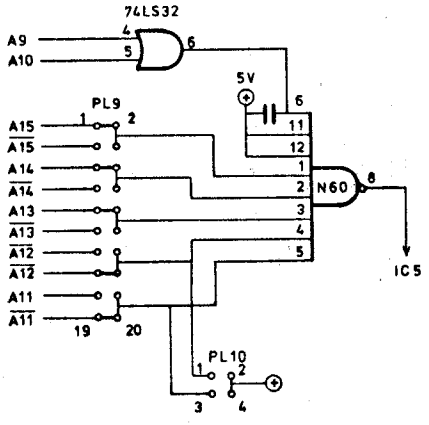
If you use BASICODE2 you have to do a few changes on your cassette interface-board.: Connect IC1 pin 9 to 0 volt and IC1 pin 12 to + 5 volt. This changes gives addr. $E18x instead of $E28x. Remember to change the addresses in the objectkode as well. BEWARE! of the wrong identifications on the Elektor diagram, IC 1 and IC 2 are swapped! Fig. 3.

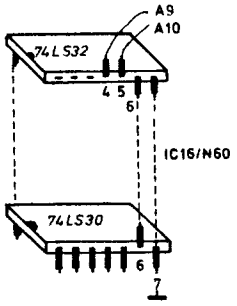
When i wish to equip my EC 65 whith an 6532 i improwed the address-decodning on the floppy controller-board, so it's only use $E000-$E03F, again using an "piggy-back" construction as a new IC 2. Fig. 2.

This leaves room for an 6532 whith ram area $E040-$E0BF and I/O and timer on $E0C0-$E0FF. The 6532 and it's decoding ic's are mounted on a wire wrap-board as shown on fig. 4.

PETER LINDSTROEM
SOLHAVEN 8
DK2990 NIVAA,
DENMARK.

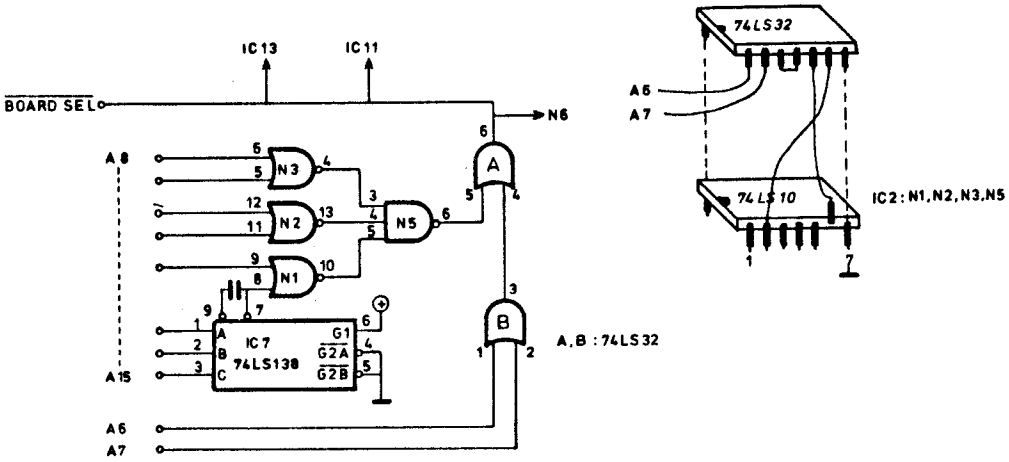
PART OF CPU





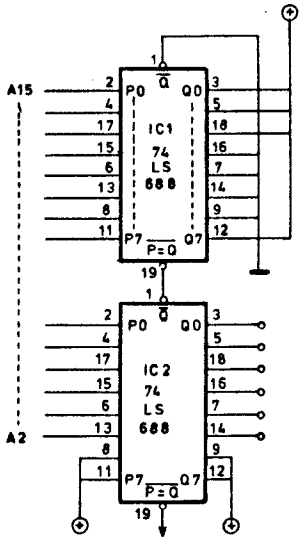
HERMAN ZONDAG

PART OF FDC



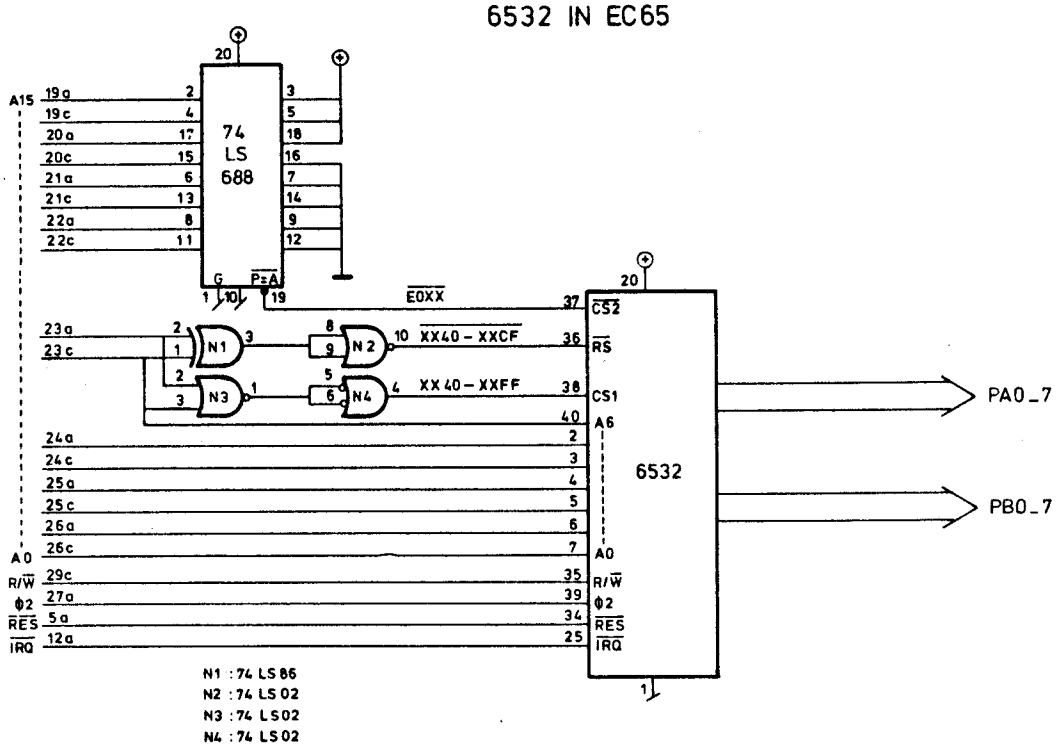
HERMAN ZONDAG

PART OF BASICODE INTERFACE



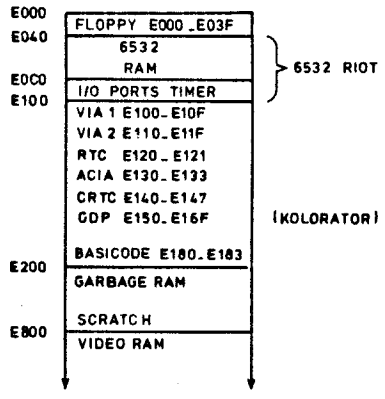
HERMAN ZONDAG

6532 IN EC65



N1 : 74 LS 86
N2 : 74 LS 02
N3 : 74 LS 02
N4 : 74 LS 02

HERMAN ZONDAG



THE MODIFIED PART OF THE MEMORY MAP

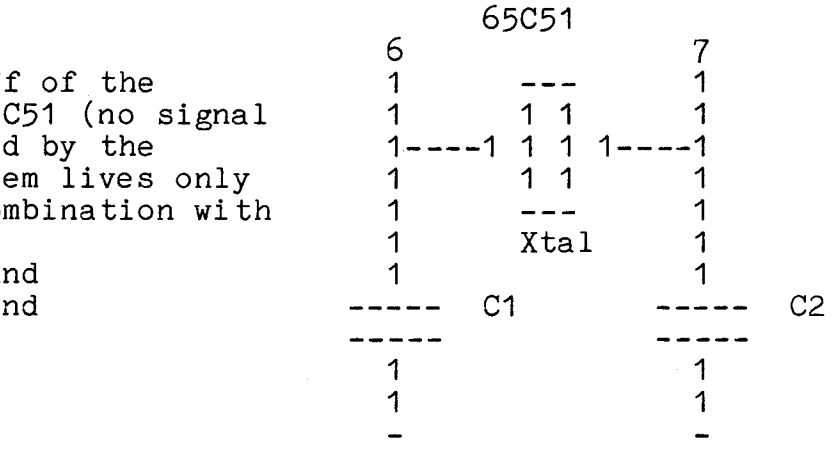
```
10    REM ***TOWERS OF HANOI***          ***TORENS VAN HANOI***
20    REM ********************************************************
30    REM THIS IS AN EXTENSION OF      DIT IS EEN UITBREIDING VAN
32    REM THE PROGRAMME "TOWERS        HET PROGRAMMA "TORENS VAN
34    REM OF HANOI", PUBLISHED         HANOI", GEPUBLICEERD IN DE
36    REM IN DE 6502 KENNER NO.38.     6502 KENNER NR.38.
38    REM THIS PROGRAMME SHOWS         DIT PROGRAMMA LAAT DE VER-
40    REM THE MOVES OF THE DISKS       PLAATSINGEN VAN DE SCHIJ-
42    REM AND COUNTS THE NUMBER        VEN ZIEN EN TELT HET AAN-
44    REM OF THE MOVED DISKS. ONE      TAL VERPLAATSTE SCHIJVEN.
46    REM CAN COUNT THE MOVES ONE-     MEN KAN DE VERPLAATSINGEN
48    REM SELF OR LET THE COMPU-       ZELF BEREKENEN OF HET DOOR
50    REM TER DO IT.                   DE COMPUTER LATEN DOEN.
52    REM THE MAXIMUM NUMBER OF        HET MAXIMAAL OP TE GEVEN
54    REM DISKS IS 12.                 SCHIJVEN IS 12.
100   REM ********************************************************
110   REM              W.E. BOER
120   REM              K. v. ENCKEVOIRTSTR. 14
130   REM              5645 EM   EINDHOVEN
140   REM              040 - 122217
150   REM              NEDERLAND
160   REM********************************************************
1000 PP(1)=14:PP(2)=40:PP(3)=66      :REM POSITIE PAAL
1010 HG(1)=0:HG(2)=0:HG(3)=0         :REM HOOGTE TOREN
1020 BO=20:TP=6:A=1:TL=0:HOME        :REM BODEM,TOP,VAR,TELLER
1030 GOSUB 3100                      :REM ZELF SPELEN
1040 GOSUB 6100                      :REM INVOER
1050 IF N=0 THEN 1270                :REM END
1060 DIM I(N),J(N),B(N),P1(N),P2(N),P3(N)
1070 HG(1)=HG(1)+N: HOME
1100 GOSUB 5000                      :REM TEKEN PALEN OP SCHERM
1110 GOSUB 5100                      :REM TEKEN BODEM OP SCHERM
1120 GOSUB 5200                      :REM TEKEN TOREN OP SCHERM
1125 GOSUB 3000                      :REM GEEF PALEN NUMMERS
1126 IF Y=1 THEN GOTO 2000           :REM ZELF SPELEN
1130 I(N)=1:J(N)=3:K=N
1140 IF K=1 THEN GOTO 1190
1150 I1=I(K):J1=J(K):K=K-1
1160 IF B(K+1)=1 THEN 1180
1170 I(K)=I1:J(K)=6-I1-J1:GOTO 1140
1180 I(K)=6-I1-J1:J(K)=J1:GOTO 1140
1190 IF B(K)=0 THEN GOTO 1230
1200 B(K)=0:K=K+1
1210 IF K<=N THEN GOTO 1190
1220 END
1230 VE=22:HO=28:CURSOR TO VE,HO
1240 PRINT"SCHIJF:";K;"VAN:";I(K);"NAAR:";J(K)
1250 GOSUB 5700                      :REM VERPLAATS SCHIJF
1255 TL=TL+1:VE=22:HO=1:CURSOR TO VE,HO:PRINT TL
1260 B(K)=1:GOTO 1140
1270 END
2000 FOR X=1 TO N                    :REM ZELF SPELEN
2010 P1(X)=X
2020 NEXT X:NN=N+1
2030 VE=22:HO=20:CURSOR TO VE,HO     :REM -VAN PAAL-
2040 INPUT;"SCHIJF VAN PAAL:";PA$
2050 IF PA$<CHR$(49) OR PA$>CHR$(51) THEN GOTO 2030
2060 PA=VAL(PA$):IF PA<1 OR PA>3 THEN GOTO 2030
2070 HO=40:CURSOR TO VE,HO           :REM -NAAR PAAL-
2080 INPUT;"NAAR PAAL:";PB$
2090 IF PB$<CHR$(49) OR PB$>CHR$(51) THEN GOTO 2070
2095 PB=VAL(PB$):IF PB<1 OR PB>3 THEN GOTO 2070
2100 IF PA=PB THEN GOTO 2120
2110 IF HG(PA)>0 THEN 2130
2120 GOSUB 3200:GOTO 2030            :REM FOUTMELDING
2130 HV=HG(PA):HN=HG(PB)
2140 AV=NN-HV:AN=NN-HN
2150 IF PA=1 THEN SN=P1(AV)          :REM BEPAAL SCHIJFNUMMER-VAN-
2160 IF PA=2 THEN SN=P2(AV)
2170 IF PA=3 THEN SN=P3(AV)
2180 IF HN=0 THEN GOTO 2230          :REM GEEN SCHIJF OP PAAL-NAAR-
2190 IF PB=1 THEN SR=P1(AN)          :REM BEPAAL SCHIJFNUMMER-NAAR-
2200 IF PB=2 THEN SR=P2(AN)
2210 IF PB=3 THEN SR=P3(AN)
2220 IF SR<SN THEN GOTO 2120         :REM FOUTMELDING
2230 K=SN:I(K)=PA:J(K)=PB
2240 GOSUB 5700                      :REM VERPLAATS SCHIJF
2250 HN=HG(J(K)):AN=NN-HN
2260 IF PB=1 THEN P1(AN)=SN          :REM SCHIJFNUMMER NAAR NIEUWE PAAL
2270 IF PB=2 THEN P2(AN)=SN
2280 IF PB=3 THEN P3(AN)=SN
2290 TL=TL+1:VE=22:HO=1:CURSOR TO VE,HO:PRINT TL :REM TELLER
2300 GOSUB 3300                      :REM TOREN VERPLAATST ?
2310 IF Y=1 THEN GOTO 2030           :REM VOLGENDE SCHIJF
2320 INPUT;"NOG EEN KEER? J/N:";A$
2330 IF A$="J" OR A$="j" THEN RUN
2340 END
3000 VE=BO:NR=1:INVERSE              :REM GEEF PALEN NUMMERS
3010 FOR HO=13 TO 65 STEP 26
3020 CURSOR TO VE,HO
3030 PRINT NR:NR=NR+1
3040 NEXT HO:NORMAL
3050 RETURN
3100 HOME:Y=0                        :REM ZELF SPELEN ?
3110 INPUT;"WILT U ZELF SPELEN ? J/N:";A$
3120 IF A$="j" OR A$="J" THEN Y=1:RETURN
3130 IF A$="n" OR A$="N" THEN RETURN
3140 GOTO 3100                       :REM ONJUISTE INGAVE
3200 V=23:HO=20:CURSOR TO VE,HO      :REM FOUTMELDING
3210 PRINT"       DEZE VERPLAATSING KAN NIET"
3220 FOR X=1 TO 1500
3230 NEXT X
3240 HO=1:CURSOR TO VE,HO
3250 FOR X=1 TO 80
3260 PRINT CHR$(32);
3270 NEXT X
3280 RETURN
3300 Y=1                             :REM TOEN VERPLAATST ?
3310 FOR X=1 TO N
3320 IF P3(Y)=X THEN Y=Y+1:NEXT X
3330 IF Y=X+1 THEN Y=0
3340 RETURN
5000 PL$=CHR$(160)                   :REM TEKEN PALEN OP SCHERM
5010 FOR VE=8 TO 19
5020 FOR HO=14 TO 66 STEP 26
5030 CURSOR TO VE,HO
5040 PRINT PL$
5050 NEXT HO
5060 NEXT VE
5070 RETURN
5100 VE=BO                           :REM TEKEN BODEM OP SCHERM
5110 BM$=CHR$(160)
5120 FOR HO=1 TO 79
5130 CURSOR TO VE,HO
5140 PRINT BM$
5150 NEXT HO
5160 RETURN
5200 K=N:VE=BO-A                     :REM TEKEN TOREN OP SCHERM
5210 GOSUB 5400                      :REM MAAK SCHIJF
5220 HO=PP(1)-K
5230 CURSOR TO VE,HO
5240 PRINT SF$
5250 HO=PP(1)+A
5260 CURSOR TO VE,HO
5270 PRINT SF$
5280 K=K-A:VE=VE-A
5290 IF K>0 THEN GOTO 5210           :REM VOLGENDE SCHIJF
5400 SF$=""                          :REM MAAK SCHIJF
5410 FOR X=A TO K
5420 SF$=SF$+CHR$(127)
5430 NEXT X
5440 RETURN
5500 SG$=""                          :REM MAAK LEGE SCHIJF
5510 FOR X=A TO K
5520 SG$=SG$+" "
5530 NEXT X
5540 RETURN
5550 SV$=" "+SF$+" "+SF$+" "         :REM MAAK VERPLAATS SCHIJF
5560 RETURN
5600 HO=PQ-K                         :REM PRINT SCHIJF
5610 CURSOR TO VE,HO
5620 PRINT SS$
5630 HO=PQ+A
5640 CURSOR TO VE,HO
5650 PRINT SS$
5660 RETURN
5700 Z=1    :REM SCHIJF NAAR RECHTS**VERPLAATS SCHIJF
5710 IF I(K)>J(K) THEN Z=-1          :REM SCHIJF NAAR LINKS
5720 GOSUB 5400                      :REM MAAK SCHIJF SF$
5730 GOSUB 5500                      :REM MAAK LEGE SCHIJF SG$
5740 GOSUB 5550                      :REM MAAK VERPLAATS SCHIJF SV$
5750 PV=PP(I(K))                     :REM POSITIE PAAL-VAN-
5760 PN=PP(J(K))                     :REM POSITIE PAAL-NAAR-
5770 HV=HG(I(K))                     :REM HOOGTE TOREN-VAN-
5780 HN=HG(J(K))                     :REM HOOGTE TOREN-NAAR-
5790 PQ=PV
5800 VE=BO-HV                        :REM VERPLAATS SCHIJF OMHOOG
5810 SS$=SG$
5820 GOSUB 5600                      :REM PRINT SCHIJF
5830 SS$=SF$
5840 VE=VE-A
5850 GOSUB 5600                      :REM PRINT SCHIJF
5860 IF VE>TP THEN 5810              :REM NIEUWE SCHIJF
5870 HG(I(K))=HV-A
5875 VA=PV-(K+A):NA=PN-(K+A)
5880 FOR X=VA TO NA STEP Z           :REM VERPLAATS SCHIJF-
5890 HO=X                            :REM HORIZONTAAL
5900 CURSOR TO VE,HO
5910 PRINT SV$
5920 NEXT X
5930 PQ=PN                           :REM VERPLAATS SCHIJF OMLAAG
5940 SS$=SG$
5950 GOSUB 5600                      :REM PRINT SCHIJF
5960 SS$=SF$
5970 VE=VE+A
5980 GOSUB 5600                      :REM PRINT SCHIJF
5990 IF((BO-A)-HN)>VE THEN 5940      :REM NIEUWE SCHIJF
6000 HG(J(K))=HN+A
6010 RETURN
6100 HOME                            :REM INVOER AANTAL SCHIJVEN
6110 FM$="FOUTIEVE INVOER!!!"
6120 N$="":N=0
6130 PRINT"AANTAL SCHIJVEN ? (max 12):  ";
6140 GET IN$
6150 IF IN$="" THEN GOTO 6140
6160 IF IN$=CHR$(13) THEN GOTO 6230
6170 IF IN$<CHR$(48)OR IN$>CHR$(57) THEN PRINT FM$:GOTO 6130
6180 PRINT IN$
6190 N$=N$+IN$
6200 N=VAL(N$)
6210 IF N<1 OR N>12 THEN PRINT FM$:GOTO 6120
6220 GOTO 6140                       :REM VOLGENDE INVOER
6230 RETURN
```

****
* **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

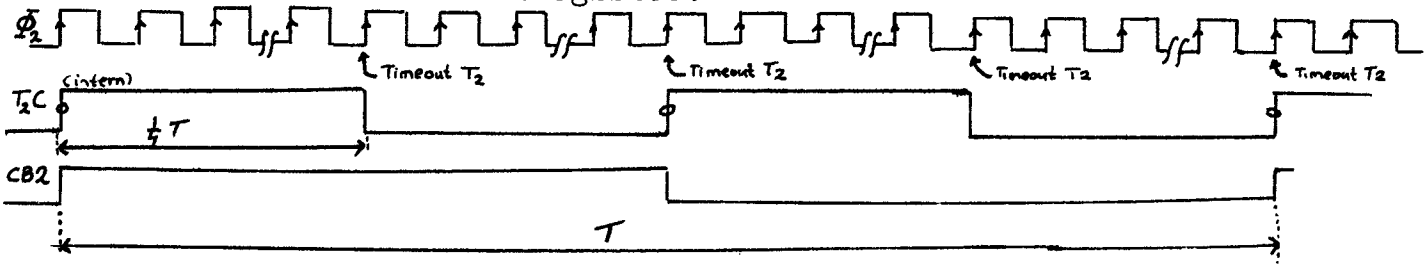


SPLIT SPEED WITH ACIA

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 $0F 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
  000B  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 0F              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==========

pin 5 6551 → pin 19 6522 (CB2)
or pin 17 6522 (PB7) for timer 1 free run.

Centronics input for DOS65 or junior computer.

If testing a centronics output or getting data from one computer to another, the following program could come in handy. It simulates a centronics input and it is possible to write bytes directly into memory. To get the input on the screen it is only necessary to change the 'jsr put' in 'jsr $c023' for the DOS65 computer or 'jsr $1334' for the junior computer. The program uses 6522 port b.

```
                ; file          centrin.mac
                ;               DOS65 system
                ; author        E.R.Elderenbosch
                ;
          0200          org     $0200
                ;
          0010  count   equ     $0010           ; 2 bytes
                ;
          E111  vbpad   equ     $e111
          E113  vbpadd  equ     $e113
          E11C  vbpcr   equ     $e11c
          E11D  vbifr   equ     $e11d
                ;
                ; for junior computer, use:
                ;
                ; vbpad  equ $1801
                ; vbpadd equ $1803
                ; vbpcr  equ $180c
                ; vbifr  equ $180d
                ;
0200 A0 00      init    ldy     #$00            ; setup variables
0202 A2 00              ldx     #$00
0204 A9 0A              lda     #$0a            ; set 6522 mode
0206 8D 1CE1            sta     vbpcr
0209 A9 00              lda     #$00            ; setup lines as input
020B 8D 13E1            sta     vbpadd
020E 85 10              sta     count           ; begin storing at $0300
0210 A9 03              lda     #$03
0212 85 11              sta     count+1
                ;
0214 20 1D02    loop    jsr     get             ; main program
0217 20 2802            jsr     put
021A 4C 1402            jmp     loop            ; end of program (endless loop)
                ;
021D AD 1DE1    get     lda     vbifr           ; centronics input routine
0220 29 02              and     #$02            ; is there a character?
0222 F0 F9              beq     get             ; if not, try again
0224 AD 11E1            lda     vbpad           ; if there is, exit
0227 60                 rts
                ;
0228 91 10      put     sta     [count],y       ; store character in memory
022A E6 10              inc     count           ; increment counter
022C D0 02              bne     ret
022E E6 11              inc     count+1
0230 60         ret     rts
                ;
          0200          end     init
                                label table

count     0010  get       021D  init      0200  loop      0214  put       0228
ret       0230  vbifr     E11D  vbpad     E111  vbpadd    E113  vbpcr     E11C

Errors detected: 0
```

COMMENT 13E 27 nov 1986 FOR APPLE

Author : Frans Verberkt, Hillekensacker 12-10,
6546 KG Nijmegen, The Netherlands.
Transl.: Nico Verberkt, The Netherlands.

In DE 6502 KENNER 46, page 10 I read the program COMMENT which also can be adapted to ASSM/TED of Moser in the APPLE. However with some alterations: ASSM/TED does not code the end of the line with CR ($0D), but recognizes it if the most significant bit of the character is high. So the test on this is CMP #$80...BCS.
It also would be nice to build in a function by which means the line would not be changed, this is avoided if directly after the semicolon (;) the minus sign (-) is used.
Furthermore directly after raising CURPOI is controled with CEME by which means this certainly cannot get beyond the text-area if nevertheless a character would be changed by wrong operating; e.g. when you walk about in the monitor and alter something on the textstring (never do this!!!).
The niceness of this program is that when you have typed it with this text completely, after assembling and running, you can read the same text in lowercase, which easier to read.

```
                ##################################
                # I AM CURIOUS TO KNOW HOW THIS  #
                # PROGRAM AS A KIND OF COMMAND   #
                # CAN BE BUILD IN, IN MOSER's    #
                # ASSEMBLER. WHO HELPS!          #
                ##################################
                          .OS
                0570  ;-------------------------------
                0590  ;#### PAGE ZERO ####
                0610 CURPOI     .DE $18           ;TEMPORARY MEMORY
                0640  ;-------------------------------
                0660  ;#### POINTERS ASSM/TED ####
                0680 LOME       .DE $100          ;BEGIN MEMORY
                0690 CEME       .DE $D3           ;END OF TEXT POINTER
                0720  ;-------------------------------
                0740  ;#### COMMENT ####
                0760            .BA $0900
0900- A5 18     0780 START      LDA *CURPOI       ;SAVE TEMPORARY MEMORY
0902- 48        0790            PHA
0903- A5 19     0800            LDA *CURPOI+1
0905- 48        0810            PHA
0906- AD 00 01  0830 COMMENT    LDA LOME          ;BEGIN ===" POINTER
0909- 85 18     0840            STA *CURPOI
090B- AD 01 01  0850            LDA LOME+1
090E- 85 19     0860            STA *CURPOI+1
0910- A0 00     0880            LDY #$00
0912- 20 52 09  0890 NEXT       JSR INCPOINT      ;INC+CMP CURPOI
0915- B0 34     0900            BCS COMEND        ;CURPOI = END OF FILE ?
0917- B1 18     0920 XIT        LDA (CURPOI),Y
0919- C9 3B     0930            CMP #';           ;FOUND SEMICOLON
091B- D0 F5     0940            BNE NEXT
091D- 20 52 09  0950            JSR INCPOINT      ;INC+CMP POINTER
0920- B0 29     0960            BCS COMEND        ;CURPOI = END OF FILE ?
0922- B1 18     0970            LDA (CURPOI),Y
0924- C9 2D     0980            CMP #'-           ;IF MINUS SIGN THEN
0926- F0 EA     0990            BEQ NEXT          ;NOTHING TO CHANGE
0928- 4C 42 09  1000            JMP SAME
092B- 20 52 09  1020 LOWER      JSR INCPOINT      ;FIRST CHAR = UPPERCASE
092E- B0 1B     1030            BCS COMEND        ;CURPOI = END OF FILE ?
0930- B1 18     1040            LDA (CURPOI),Y
0932- 29 7F     1050            AND #$7F          ;BIT 7=0 (POSITIVE ASCII)
0934- C9 41     1060            CMP #'A           ;TEST ALPHA UPPERCASE
0936- 90 0A     1070            BCC SAME
0938- C9 5B     1080            CMP #$5B          ;ASCII(Z)+1
093A- B0 06     1090            BCS SAME
093C- B1 18     1100            LDA (CURPOI),Y
093E- 09 20     1110            ORA #$20          ;BIT 5=1
0940- 91 18     1120            STA (CURPOI),Y    ;SET LOWER CASE
0942- B1 18     1140 SAME       LDA (CURPOI),Y
0944- C9 80     1150            CMP #$80          ;EOL BIT 7=1
0946- B0 CA     1160            BCS NEXT
0948- 4C 2B 09  1170            JMP LOWER
094B- 68        1200 COMEND     PLA               ;RESET TEMPORARY MEMORY
094C- 85 19     1210            STA *CURPOI+1
094E- 68        1220            PLA
094F- 85 18     1230            STA *CURPOI
0951- 60        1250            RTS               ;BACK TO CALLER
                1270 ;-------------------------------
                1290 ;#### INCPOINT + COMPARE ####
                1300
0952- E6 18     1310 INCPOINT   INC *CURPOI
0954- D0 02     1320            BNE COMPAR
0956- E6 19     1330            INC *CURPOI+1
0958- A5 19     1340 COMPAR     LDA *CURPOI+1     ;IF CURPOI "= CEME
095A- C5 D4     1350            CMP *CEME+1       ;THEN CARRY=SET
095C- 90 06     1360            BCC INCIT
095E- D0 04     1370            BNE INCIT
0960- A5 18     1380            LDA *CURPOI
0962- C5 D3     1390            CMP *CEME
0964- 60        1400 INCIT      RTS
                          .EN
```

HOW TO MODIFY THE ELEKTOR 64K MEMORY CARD FOR USE WITH DOS65

Andrew Gregory, England.

To make this card work with DOS65 V2.0 a few changes to the addressing are required. Proceed as follows:

1) Build the card out of TTL LS or TTL HC(T)MOS. I have built mine from TTL LS but do not envisage many problems with HCMOS versions provided it is remembered that only HCT devices can be driven from TTL LS. Seven 6264 rams are needed, the IC15 socket remains empty.

2) Reduce R27 (1K) to 390 ohms. Otherwise it will not function reliably with a 65C02 processor. On my card IC7 was a 7412, but I am not sure if this is crucial. See the note about HCMOS cpu cards elsewhere in this issue.

3) Set all the dip switches off. The links are made as follows:

E - L	D - K	C - J
R	Q	P no connection
O	N no connection	M -----

4) Lift pin 8 of IC6 out of its socket then make the following connections when viewing the card with the writing the correct way up:

Connect a 1K resistor from IC6 pin 13 to +5 volts.
Connect IC4 pin 3 to P left.
Connect IC6 socket pin 8 to N centre.
Connect IC6 pin 8 to N right.
Connect IC4 pin 12 to H.
Connect IC4 pin 14 ti IC6 pin 9.

The card now occupies $0000 to $DFFF and will operate at 1 of 2 MHz. The effect of these changes is to connect A13, A14 and A15 to the inputs op N9 and place an inverter (N2) in its output.

BRIEF AAN DE REDAKTIE

Wally E. Boer, Nederland

Als men bij DOS65-Basic de GET-instruktie gebruikt, dan komt de cursor niet op het scherm. Bij INPUT komt de cursor wel terug. Om bij GET toch de cursor op het scherm te krijgen moeten twee registers van de CRTC op de videokaart gevuld worden met bepaalde data. Dit kan in Basic door voor de regel met GET twee poke's te geven, t.w. POKE 57664,10 : POKE 57665,0
Men kan er ook een machinetaalprogramma van maken en ergens op een veilige plaats in het geheugen zetten, en als het nodig is oproepen met CALL. Dit wordt dan:

```
A9 0A
8D 40 E1     ; Adress register
A9 00
8D 41 E1     ; Register file
60
```

Zelfbouwer Elektuur's EC65K zoekt kontakt met Belgische zelfbouwers.
Erik Olaerts, Zavelputstraat 13, B-3020 Herent, 016/237378.

TE KOOP:
COMMODORE systeem, bestaande uit:
Basiseenheid (scherm, kast, toetsenbord) Commodore 8032-S 8 bits/32KByte, 80 kol, 25 regels, Basic 4.0 op Rom. 3 poorten: IEEE, User port, Cass. port.
Dual disk drive 8050, 2x512KByte, 5 1/4" en 4KByte intern geh. Matrix printer AS 8024, 132 kol, pinf. IEEE Ser.Interface. Seriële kabel. Doos printer papier.
Prijs n.o.t.k. (richtpr. ca. f.1.800,==)
Alleen als gehel te koop.
Victor Kroon, Zeisstr.54, 3075 NZ R'dam.
Telf.: 010-4842778 (na 18.00).

OP COMPUTERVAKANTIE?
COMPUTER WORLD organiseert in 1987 een drietal computervakanties in het Zwarte Woud. Ook geschikt voor meisjes. Vraag brochures aan bij COMPUTER WORLD, Hurstweg 62B, D-7800 Freiburg (0761/44775).
Daarbij naam en adres Redaktie DE 6502 KENNER opgeven is van belang!

== HOE WORDT DE VIDEO CONTROLLER 6845 GEPROGRAMMEERD ? ==

DOOR : Tony Lehaen, België.

Dit artikel maakt het U misschien duidelijker hoe de CRTC 6845 op de VDU kaart geprogrammeerd kan worden, door het berekenen van de registerinhouden. De 6845 beschikt over 17 registers. Alvorens deze 17 registers te berekenen dienen eerst enkele begrippen nader toegelicht te worden en enkele gegevens bepaald.
Bij een gewoon TV toestel met bewegende beelden in ons Europees 625 lijnenstelsel worden twee beeldrasters (met interliniëring) van elk 312,5 lijnen per beeld beschreven. De rasterfrequentie is 25 Hz of 25 beelden per seconde, dit om een flikkervrij beeld te verkrijgen. De horizontale frequentie is dan 625 x 25 = 15625 Hz.
Iets anders ligt het nu met de stilstaande beelden van onze video, waar het raster van 312,5 lijnen (zonder interliniëring) 50 maal per seconde geschreven wordt. Er moet opgemerkt worden dat het hier gaat over een Video monitor of TV toestel met video-ingang (dit in verband met de bandbreedte).
Voor de Video kunnen nu de volgende gegevens bepaald worden:

(A) De rasterfrequentie = 50 Hz = 50 beelden per seconde
(B) De horizontale frequentie = 312,5 x 50 = 15625 Hz
(C) De horizontale synchronisatie pulsbreedte = 4 uSec.
(D) De rasterscantijd = $\frac{1}{(A)} = \frac{1}{50}$ = 20 mSec.

(E) De videolijntijd = $\frac{1}{(B)} = \frac{1}{15625}$ = 64 uSec.

(F) De oscillator dotfrequentie (of klokfrequentie) = FX.
Bij de VDU kaart gaat men ervan uit dat elk karakter 8 dots breed is en dat er in één lijntijd van 64 uSec. (horizontale synchronisatiepuls van 4 uSec. inbegrepen) 128 karakters op het scherm geschreven worden.
Dit geeft ons als klokfrequentie: $Fx = \frac{128 \times 8}{64} = 16$ MHz
Op de VDU kaart moet dan een kristal van 16 MHz gebruikt worden.

(G) De CRTC karakterfrequentie = $\frac{(F)}{8} = \frac{16}{8}$ = 2 MHz.

(H) De CRTC karaktertijd = $\frac{1}{(G)} = \frac{1}{2}$ = 0,5 uSec.

(I) Aantal videolijnen per karakter = 8 + 1 lege lijn = 9.
(J) Totale vertikale karakterlijntijd = (E)x(I) = 64 x 9 = 576 uSec.

Bepalen van de registerinhouden

R0 : Horizontaal totaal = totaal van de zichtbare + niet-zichtbare karakters per regel - een.
$R0 = \frac{(G)}{(B)} - 1 = \frac{2000000}{15625} = 128 - 1 = 127$

R1 : Horizontaal display = aantal zichtbare karakters per regel.
R1 = 80 (elk formaat kan hier vrij gekozen worden)

R2 : Plaats van de horizontale synchronisatiepuls in aantal karakters op de horizontale lijn:
$R2 = \frac{R0+R1-R3}{2} = \frac{127 + 80 - 8}{2} = 99,5$ afgerond = 100.

R3 : De vertikale synchronisatie pulsbreedte ligt voor de 6845 vast op 16 videolijntijden.
De horizontale synchronisatie pulsbreedte in aantal karakters is:
R3 = (C)x(G) = 4 x 2 = 8

R4 : Vertikaal totaal = totaal van zichtbare en niet-zichtbare karakterregels - een. Dit mag niet groter zijn dan de rasterscantijd (D) = 20 mSec.
De totale karakterlijntijd (J) = 576 uSec.
$R4 = \frac{(D)}{(J)} - 1 = \frac{20000}{576} - 1 = 33,72$ afgerond = 33.

R5 : Vertikale fijnafstemming = aantal videolijnen van 64 uSec toe te voegen aan R4, omdat door de afronding van R4 de rasterscantijd geen 20 mSec meer is.
De fout is 20000 - (34 x 576) = 20000 - 19584 = 416 uSec.
$R5 = \frac{416}{64} = 6,5$ afgerond = 6.

R6 : Vertikaal display = aantal zichtbare karakterregels.
R6 = 24 (elk formaat kan hier vrij gekozen worden).

R7 : Plaats van de vertikale synchronisatiepuls in aantal karakterregels. Dit bepaalt de hoogte van de bovenrand en onderrand.
$R7 = \frac{R4+R6}{2} = \frac{33 + 24}{2} = 28,5$ afgerond 28

R8 : Interliniëringsmode (zet de rasterscanmode), de 6845 heeft 3 mogelijke modes.
Mode 0 : niet geïnterlinieerd (normale synchronisatie)
Mode 1 : geïnterlinieerde synchronisatie
Mode 3 : geïnterlinieerde synchronisatie met dubbel aantal karakterregels.
Men kiest voor Mode 0.
R8 = 0

R9 : Maximum videolijnen per karakterregel
R9 = (I) - 1 = 9 - 1 = 8

R10: Cursor startlijn = onderste videolijn waar de cursor begint. De bits van 0 tot 4 bepalen de beginlijn van de cursor. De bits 5 en 6 bepalen de cursor displaymode.

bit 6	bit 5	displaymode
0	0	niet knipperende cursor
0	1	geen cursor zichtbaar
1	0	snel knipperende cursor (1/16 van rasterscantijd: (D) 20 / 16 = 20 / 16 = 1,25 mSec.
1	1	traag knipperende cursor (1/32 van rasterscantijd: (D) 20 / 32 = 20 / 32 = 0,625 mSec.

Maakt men bijvoorbeeld volgende keuze:

b6	b5	b4	b3	b2	b1	b0
1	1	0	0	0	0	0

= Hex 60 of Dec 96

R10 = 96, dit betekent traag knipperend en startlijn op videolijn 0.

R11: Cursor eindlijn = hoogte van de cursor uitgedrukt in aantal videolijnen.
R11 = 7 formaat is vrij te kiezen tussen 1 en 8
1: de cursor is 1 videolijn hoog
8: de cursor is 8 videolijnen hoog
(= karakterhoogte)

R12-R13: Startadres van de controller 6845 op het scherm in het videogeheugen (voor de VDU kaart is dit $D000).
R12 en R13 vormen een 14-bits adres waarvan slechts de bits b0 tot b10 bij de VDU kaart gebruikt worden omdat de videoram 2K groot is ($D000 tot $D7FF).
R12 = R13 = 0, dit betekent dat het startadres links boven in het scherm staat.

R14-R15: Startadres van de cursor op het scherm in het videogeheugen. R14 en R15 vormen een 14-bits adres.
R14 = R15 = 0, dit betekent dat de beginpositie van de cursor links boven in het scherm is.

R16-R17: Lichtpen. Kan gebruikt worden als men over de nodige software beschikt.
R16 = 80 (karakters per regel) ¾ is het zichtbare
R17 = 24 (karakterregels) ¾ deel o.h. scherm.

Samenvatting van de CRTC timing table.

Reg's	Dec	Hex	
R0	127	$7F	Horizontaal totaal -1=128-1=127 karakters
R1	80	$50	Hor. aantal karakters/regel= 80 karakters
R2	100	$64	Hor. synchronisatie positie=100 karakters
R3	8	$08	Vert/Hor. synchron. pulsbreedte = 16/8
R4	33	$21	Vertikaal totaal = 34-1=33 karakterregels
R5	6	$06	Vert. totaal fijnafstemm.=6x64uSec=416uSec
R6	24	$18	Vert. aantal karakterregels = 24
R7	28	$1C	Vert. synchron. pos. = 28 karakterregels
R8	0	$00	Interliniëringsmode
R9	8	$08	Aantal scanlijnen per regel = 9 - 1 = 8
R10	96	$60	Cursor start op videolijn 0 en traag knip.
R11	7	$07	Cursor eind, bepaalt de hoogte v.d. cursor
R12	0	$00	¾Startadres controller linksboven scherm
R13	0	$00	¾
R14	0	$00	¾Startadres cursor linksboven i.h. scherm
R15	0	$00	¾
R16	80	$50	¾Zichtbaar veld voor de lichtpen
R17	24	$18	¾

* EEN PAAR TIPS VOOR COMMODORE BASIC *

Door : Nico de Vries, Nederland

INPUT zonder vraagteken.

Het is door middel van een POKE voorafgaand aan een INPUT-statement het door de INPUT gegenereerde vraagteken te onderdrukken. De POKE is voor iedere ROMset anders:

BASIC 1.0 (OR)	POKE3,1
BASIC 2.0 (NR)	POKE14,1
BASIC 4.0	POKE16,1
VIC 20	POKE19,1
C 64	POKE19,1

Na de input is het aan te bevelen de originele waarde (nul) terug te POKEn. Programma voorbeeld (oude ROMs):

```
10 POKE3,1
20 INPUT"TYP UW NAAM IN";A$
30 POKE3,0
```

Reset van CBM diskdrive.

U kunt met de volgende opdrachten de CBM diskdrive resetten. Dit is hetzelfde als een koude start (power up).

```
BASIC:
10 OPEN15,8,15
20 PRINT#15,"U:"
30 CLOSE15

DOS SUPPORT/UNIVERSAL WEDGE:
"U: (R)
of: @U: (R)

DISK-O-PRO/COMMAND-O
SEND"U:" (R)
```

Selectieve directories.

Het directory-commando van DOS SUPPORT of UNIVERSAL WEDGE heeft meer mogelijkheden dan de meeste gebruikers weten. Hier zijn ze allemaal (voorbeelden met behulp van DOS SUPPORT):

1. Complete directory:

```
"$0  Alleen drive 0.
"$1  Alleen drive 1.
"$   Beide drives
```

2. Bepaalde filenaam opvragen:

```
"$0:filenaam  Alleen drive 0.
"$1:filenaam  Alleen drive 1.
"$,filenaam   Beide drives.
```

Hierbij kunt U gebruik maken van ? en * om bepaalde namen uit te selecteren. Hierbij is een ? een willekeurig teken, en een * geeft aan dat de volgende tekens er niet toe doen. Voorbeelden:

```
"$0:????  geeft alle filenamen van 4 letters op drive 0.
"$1:G*    geeft alle filenamen die met een G beginnen op
          drive 1.
```

Zie ook de manual van de diskdrive, voor meer voorbeelden.

3. Bepaalde filetypen opvragen:

U kunt ook selecteren op filetype (dit staat in geen enkel manual !!!). Het gaat zo:

```
"$0:*=S   geeft alle sequentiële files op drive 0.
"$1:*=R   geeft alle REL-files op drive 1.
"$,*=U    geeft alle USR-files op beide drives.
"$1:*=P   geeft alle PRG-files op drive 1.
```

Tenslotte is het mogelijk om de mogelijkheden 2. en 3. te combineren:

```
"$0:G?T*=S  geeft alle SEQ-files op drive 0 waarvan de
            naam begint met een G en waarvan de tweede
            letter van de naam eenT is.
"$1:????=R  geeft alle REL-files op drive 1 waarvan de
            naam uit 4 tekens bestaat.
"$,PBE*=P   geeft alle PRG-files waarvan de naam begint
            met PBE op beide drives.
```

Met deze wetenschap is het uitermate vreemd dat in BASIC 4.0 DIRECTORY of CATALOG niet gevolgd mogen worden door een string, maar alleen door een drive nummer. Hierdoor zijn alle kunstjes onmogelijk (in DISK-O-PRO mag dit overigens wel, zodat het laatste voorbeeld in DISK-O-PRO zo moet worden ingetypt: DIRECTORY "PBE*=P" (R)).

APPLE NIEWS

Omzet van Apple Computer Inc. toegenomen met 24% in eerste kwartaal van fiscaal jaar 1987.

Cupertino/Zeist, 30 januari 1987.
Het eerste kwartaal van het fiscale jaar 1987 is voor Apple Computer Inc. afgesloten met een winst van 58,5 miljoen US $ ofwel 0,91 US $ per aandeel. In de vergelijkbare periode van het jaar daarvoor werd een winst behaald van US $ 56,9 miljoen, ofwel 0,91 US $ per aandeel. Het fiscaal jaar loopt van oktober tot en met september. De omzet in het afgelopen kwartaal bedroeg 662,3 miljoen US $ hetwelk een toename van 24% betekent vergeleken met dezelfde periode van het vorig jaar, toen de omzet 533,9 miljoen US $ bedroeg. De bruto winst uitgedrukt in een percentage van de omzet bedroeg in het eerste kwartaal 51,8 procent t.o.v. 50,7 procent in het eerste kwartaal van het fiscale jaar 1986. De nieuwe APPLE][GS is goed ontvangen.

EERSTE EUROPESE APPLE MACINTOSH TENTOONSTELLING: MACWORLD EXPO IN AHOY TE ROTTERDAM

Op 22, 23 en 24 april gaat de eerste MacWorld Expo van start in de Ahoy Hallen te Rotterdam. Aanleiding tot de organisatie van MacWorld Expo is de snelle penetratie van de Apple Macintosh in het bedrijfsleven als produktiviteits hulpmiddel op vrijwel elk denkbaar gebied. Tijdens MacWorld Expo kunnen contacten worden gelegd en de nieuwste ervaringen worden uitgewisseld, tussen leveranciers van hard- en software, distributeurs, Macintosh gebruikers, Apple dealers, en uitgevers. Naar verwachting zullen uit zowel Amerika als uit belangrijke Europese landen exposanten ruim 4.000 vierkante meter expositieruimte in de Ahoy Hallen bezetten. Deelnemers als Microsoft, Apple Computer, Blyth Software, Agfa Gevaert, Nantucket, Adobe, Association of Swiss Macintosh Developers, LetraSet, Hewlett-Packard en Symbiotic hebben hun deelname toegezegd. Naast interessante en nieuwe produkten is een lezingen/congresprogramma aan de expositie verbonden met sprekers uit binnen- en buitenland.

GEVRAAGD:
Zelfbouwer Elektuur's EC65K zoekt kontakt met Belgische zelfbouwers. Erik Olaerts, Zavelputstraat 13, 3020 Herent. Tel.: 016/23 73 78.

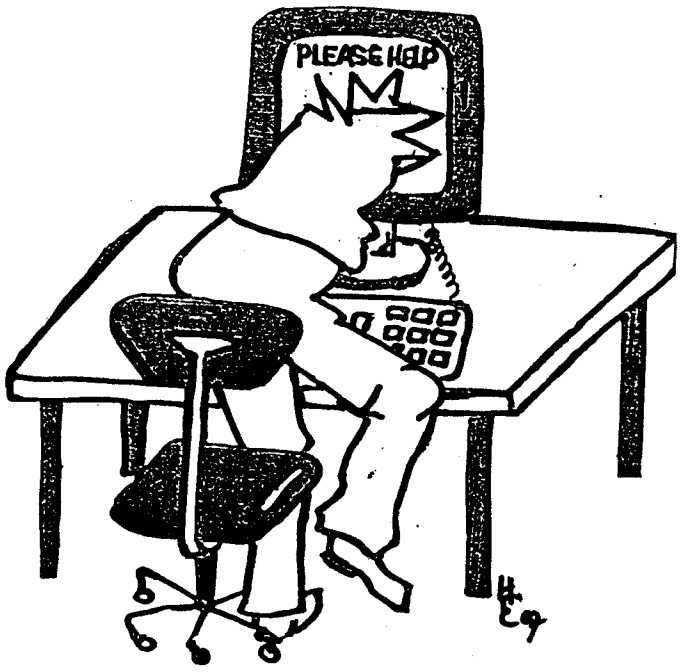
LETTER TO THE EDITOR by Andrew Gregory, England.

A note about HCMOS cpu cards: If you build your cpu card from HCMOS then leave IC 9 as 74LS01 and IC 20 as 74LS06. IC 7 can be 74HC04 provided R1 and R2 are increased to 1K5 but I found that it then worked unreliably with the 65C02.

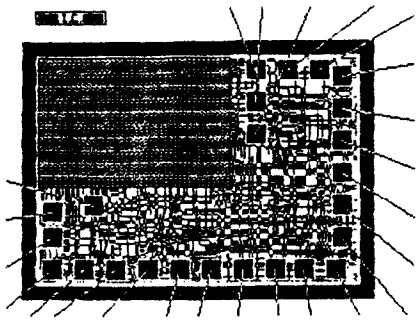
A note about VDU cards: Everyone encounters screen flicker with these cards. It can be cured by replacing IC 8 by a 74L30 when using an NMOS 6502. However the most effective solution is that suggested by Albert v.d. Beukel in DE 6502 KENNER issue 46 page 17: Reconnect IC 8 pin 1 to 02 (pin 27a of the 32 way connector). I built mine from HCMOS IC's with the exceptions of IC 20 and IC 21 which I could not obtain.

PLEASE, SEND ALL YOUR PROGRAMMES TO THE EDITORIAL OFFICE.

```
10 REM ************************************************************
20 REM *                                                          *
30 REM *                  S L I D I N G   G R I D                 *
40 REM *                  ***********************                 *
50 REM *                                                          *
60 REM *      AN AMUSING COMPUTER-GAME WRITTEN IN BASICODE-2       *
70 REM *                                                          *
80 REM *      FILL THE LINES 10 ---> 999 WITH THE BASICODE - 2    *
90 REM *             ROUTINES FOR YOUR OWN COMPUTER!              *
100 REM*                                                         *
110 REM************************************************************
120 REM
130 REM
1000 A=200:GOTO20:REM SLIDING GRID
1010 DIM A(42),RI(4),PS(256),B(42),C(42)
1100 DIM P1(256):REM WHERE THE CHARACT.
1150 REM SHOULD BE
1200 REM PS(I) CONTAINS THE POSITION
1250 REM OF CHR$(I) IN THE GRID
1300 RI(1)=-1:RI(2)=1:RI(3)=6:RI(4)=-6
1350 REM -----> SELECTION/INSTRUCTIONS <-----
1400 REM
1450 GOSUB10000
1500 VE=8:HO=3:GOSUB110
1550 PRINT"Do you want instructions (Y/N) ?";
1600 GOSUB210
1650 IF(IN$="Y")OR(IN$="y")THENGOSUB8750:GOTO1450
1700 IF (IN$<>"n") AND (IN$<>"N") THEN 1600
1750 GOSUB10000
1800 VE=3:HO=5:GOSUB110
1850 PRINT"You can select :"
1900 HO=5
1950 VE=06:GOSUB110:PRINT"1 ---> 3 * 3 figures grid"
2000 VE=08:GOSUB110:PRINT"2 ---> 3 * 3 letters grid"
2050 VE=10:GOSUB110:PRINT"3 ---> 4 * 4 letters grid"
2100 VE=12:GOSUB110:PRINT"4 ---> 5 * 5 letters grid"
2150 VE=18:HO=0:GOSUB110
2200 PRINT"Select 1 to 4 please ...";
2250 GOSUB210
2300 IF(VAL(IN$)<1)OR(VAL(IN$)>4)THEN2150
2350 M=VAL(IN$)+1
2400 M1=ASC("a"):IF M=2 THEN M1=ASC("1"):M=3
2450 M1=M1-1-M
2500 REM
2550 REM -----> DRAW THE GRID <-----
2600 REM
2650 GOSUB100
2700 FORK1=0TO(2*M)STEP2
2750 FORK2=18TO(4*M+18)
2800 VE=K1:HO=K2:GOSUB110
2850 PRINT"-";
2900 NEXTK2,K1
2950 FORK1=01TO(2*M-01)STEP2
3000 FORK2=18TO(4*M+18)STEP4
3050 VE=K1:HO=K2:GOSUB110
3100 PRINT"*";
3150 NEXTK2,K1
3200 REM
3250 REM -> INITIALISATION <-
3300 REM
3350 FORB=0TO42:A(B)=-1:NEXTB
3400 FORB=1TOM
3450 FORC=1TOM
3500 PS=6*B+C:CH=M1+C+B*M
3550 B(PS)=B:C(PS)=C
3600 A(PS)=CH:PS(CH)=PS:P1(CH)=PS
3650 NEXTC
3700 NEXTB:AN=M*M-1
3750 REM
3800 REM AN=NUMBER OF CHARS IN RIGHT
3850 REM POSITION
3900 REM
3950 B=7*M:A(B)=32:PS(32)=B
4000 REM
4050 REM -> DEGREE OF DIFFICULTY <-
4100 REM
4150 HO=0
4200 VE=2:GOSUB110:PRINT"1 - Easy";
4250 VE=3:GOSUB110:PRINT"2 - Normal";
4300 VE=4:GOSUB110:PRINT"3 - Difficult";
4350 VE=6:GOSUB110:PRINT"Select 1 to 3 ";
4400 GOSUB210
4450 IF(VAL(IN$)<1)OR(VAL(IN$)>3)THEN4350
```

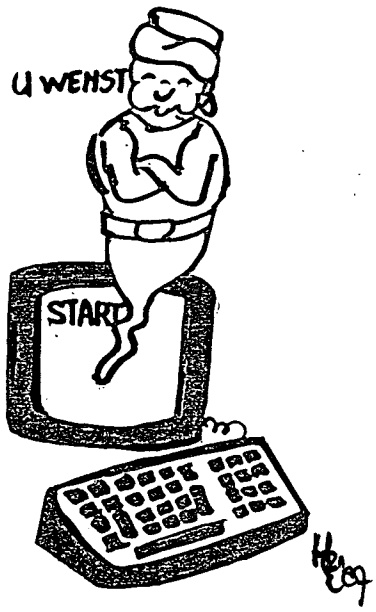


```
4500 F=VAL(IN$)
4550 GOSUB10400:REM DELETE PARTS OF TEXT
4600 REM
4650 REM -> SHUFFLE CHARACTERS  <-
4700 REM
4750 VE=8:GOSUB110:PRINT"One moment ...";
4800 IF F=1 THEN F=150
4850 IF F=2 THEN F=180
4900 IF F=3 THEN F=200
4950 C=0
5000 GOSUB260
5050 D=INT(RV*4+1)
5100 RI=RI(D)
5150 IF A(B+RI)<0THEN5000
5200 IF B    =P1(A(B)) THEN AN=AN-1
5250 IF B+RI=P1(A(B+RI)) THEN AN=AN-1
5300 A(B)=A(B+RI):PS(A(B))=B
5350 A(B+RI)=32  :PS(A(B+RI))=B+RI
5400 IF B    =P1(A(B)) THEN AN=AN+1
5450 IF B+RI=P1(A(B+RI)) THEN AN=AN+1
5500 B=B+RI
5550 C=C+1:IFC<F THEN 5000
5600 E=1:REM  COUNTER FOR TRIES
5650 GOT06450
5700 REM
5750 REM -> PRINT CHARACTERS <-
5800 REM
5850 FORB=1TOM:FORC=1TOM
5900 VE=(2*B)-1:HO=16+4*C:GOSUB110
5950 PRINTCHR$(A(6*B+C));
6000 NEXTC:NEXTB
6050 VE=8:HO=0:GOSUB110
6100 PRINT"                    ";
6150 GOSUB110
6200 PRINT"Try";E;"? ";
6250 RETURN
6300 REM
6350 REM -> INPUT SLIDING CHARACTER <-
6400 REM
6450 GOSUB5750
6500 GOSUB210:IFIN$="*"THEN8150
6550 REM
6600 REM -> CAN CHARACTER SLIDE ? <-
6650 REM
6700 D1=PS(ASC(IN$))
6750 D2=PS(32)
6800 DI=D1-D2
6850 IF ABS(DI) = 1 THEN 7000
6900 IF ABS(DI) = 6 THEN 7000
6950 GOTO 6500
7000 IF D1=P1(A(D1)) THEN AN=AN-1
7050 IF D2=P1(A(D2)) THEN AN=AN-1
7100 A(D2)=A(D1):A(D1)=32
7150 IF D2=P1(A(D2)) THEN AN=AN+1
7200 IF D1=P1(A(D1)) THEN AN=AN+1
7250 PS(A(D2))=D2:PS(A(D1))=D1
7300 VE=2*B(D1)-1:HO=16+4*C(D1)
7350 GOSUB110:PRINT" ";
7400 VE=2*B(D2)-1:HO=16+4*C(D2)
7450 GOSUB110:PRINTCHR$(A(D2));
7500 E=E+1:GOSUB6050:REM DISPL. NO OF TRIES
7550 IF AN<M*M-1THEN6500
7600 REM
7650 REM -> ALL IN GOOD ORDER <-
7700 REM
7750 VE=10:HO=0:GOSUB110:PRINT"WELL DONE ...";
7800 VE=VE+2:GOSUB110
7850 PRINT"Another game (Y/N) ?";
7900 GOSUB210
7950 IF(IN$="Y")OR(IN$="y")THENRUN
8000 REM
8050 REM -> END OF GAME <-
8100 REM
8150 GOSUB10000
8200 PRINT
8250 PRINT"A tele-coproduction by"
8300 PRINT:PRINT
8350 PRINT"Ludo Delpire,      Belgium":PRINT
8400 PRINT"Marc Lachaert,     Belgium":PRINT
8450 PRINT"Thomas Hofmeister, FR Germany"
8500 PRINT:PRINT"See you later!"
8550 END
8600 REM
8650 REM -----> INSTRUCTIONS <-----
```

IK SMURF GRAAG
OP COMPUTERS !

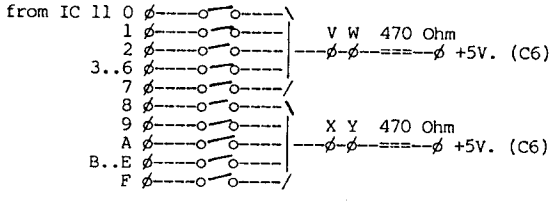
```
8700 REM
8750 GOSUB10000:VE=2:HO=10:GOSUB110
8800 PRINT"INSTRUCTIONS :"
8850 VE=VE+1:GOSUB110:PRINT"--------------"
8900 PRINT
8950 PRINT"In a moment, a  square  grid  with"
9000 PRINT"a  certain  amount of  letters  or"
9050 PRINT"numbers will appear. One square of"
9100 PRINT"the grid will be empty."
9150 PRINT
9200 PRINT"If you hit the key of a  character"
9250 PRINT"which is next to the empty square,"
9300 PRINT"the chosen  character will go into"
9350 PRINT"the empty  square. If you continue"
9400 PRINT"doing so, you  should  be  able to"
9450 PRINT"bring all characters into  numeric"
9500 PRINT"or alphabetic order.":PRINT
9550 PRINT"You can end the game at any moment"
9600 PRINT"by typing in an '*'":PRINT
9650 PRINT"If  possible: Disable  the  cursor"
9700 VE=23:HO=0:GOSUB110
9750 PRINT"Hit any key to continue ";:GOSUB210
9800 RETURN
9850 REM
9900 REM -----> HEAD LINE <-----
9950 REM
10000 GOSUB100
10050 VE=0:HO=2:GOSUB110
10100 PRINT"*** - S L I D I N G   G R I D - ***"
10150 PRINT"----------------------------------------"
10200 RETURN
10250 REM
10300 REM -> ERASE PARTS OF TEXT <-
10350 REM
10400 FORVE=2TO6
10450 IFVE=F+1THEN10550
10500 GOSUB110:PRINT"                    ";
10550 NEXTVE:RETURN
10600 REM
10650 REM
10700 REM     S L I D I N G   G R I D
10750 REM
10800 REM   NEW VERSION BASICODE 2
10850 REM
10900 REM WRITTEN IN JUNE 1985
10950 REM
11000 REM BY    THOMAS HOFMEISTER
11150 REM
11300 REM          BOCHUM B.R.D.
11350 REM
11400 REM AND   MARC LACHAERT
11450 REM
11500 REM                 BELGIUM
```



DRAM

Extension (64k) Dynamic Ram card (Elektuur April, 1982) with switches to select memory.

By: Ronald Hermens, The Netherlands.



With these switches you can select memory in steps of 4Kbyte. All you need is a total of 16 dipswitches (2*8 or the like), and 14 cm flatcable >= 18 wires.
BUILD: Put the dipswitches on a piece of print with holes (5*16) and copper lines. Remove the copper from the central holes. Cut lose the flatcable wires for 2 cm. every 4 wires, and all individual wires 7 mm. Strip and coat then with tin. Solder them on the print with dipswitches (switch 0 = black etc.), connect all other ends of the dipswitches to another, and to the wires left over from the flatcable. Put two 470 ohm resistors in holes V & Y on the DRAM-card, and connect the other ends to another and to + 5V. = C6 (isolated wire!). Connect V to W, X to Y and these to the flatcable wire #16 & #17. First connect wire #11 (brown) until #15 (green) to B..F (IC 11), and then #0..#10 to 0..A. Be very sure not to make any false connections (practice stripping of flatcable first!).

Maarten den Hertog, The Netherlands.

I use the 64K SRAM-card, built in accordance to Elektor's Computer Special no. 3. In the issued scheme the authors have forgotten to give the 8 pull-up resistors on the 'chip-select-lines'.

```
 10:TRACK 0 FOR NON BOOTABLE DISKS FOR EC65 OR DOS-JUNIOR
 40:                    By: Coen Boltjes, The Netherlands
 50NMIVEC=$E7CB
 60INIDSK=$F464          ;HEAD UP
 70AHOLD =$2363
 80:
 90*=$2200
100:
110BOOTUP  JSR STROUT     ;PRINT NEXT
120        JSR INIDSK
130        JMP (NMIVEC)
140:
150STROUT  LDX #$00
160STROUU  LDA TEXT,X
170        BEQ STROEX     :=>END
180        STA AHOLD
190        JSR $F000      ;PRINT
200        INX
210        JMP STROUU
220STROEX  RTS
230:
240TEXT    .BYTE $0A,$0D,$0A,$0A
250        .BYTE '********************'
260        .BYTE $0A,$0D
270        .BYTE 'Insert Systemdisk to'
280        .BYTE 'Boot up', $0A,$0D,$07
290        .BYTE $00
```

```
*********************************************************
*          HANDIGE SUBROUTINES VOOR DE 6502             *
*********************************************************
```

Door: Anton Mueller, Nederland.

Hierna vindt U een aantal handige subroutines voor elk willekeurig 6502 systeem. Zij zijn geheel systeemonafhankelijk.
De eerste twee, de PUSH en PULL routines, horen bij elkaar. De PUSH routine duwt de inhoud van alle 6502 registers op de stack en gaat daarna terug naar het aanroepende programma, met behoud van de oorspronkelijke registerinhouden. De PULL routine trekt de inhoud van alle 6502 registers, die door de PUSH routine op de stack waren geduwd, weer van de stack af en zet deze in de desbetreffende registers en gaat daarna terug naar het aanroepende programma.

Voorbeeld van het aanroepen van deze routines:

```
MAIN   JSR   SUBR          ; CALL SUBROUTINE
-
-
-
-
SUBR   JSR   PUSH          ; SAVE REGISTERS
-
-
-
-
       JSR   PULL          ; RESTORE REGISTERS
```

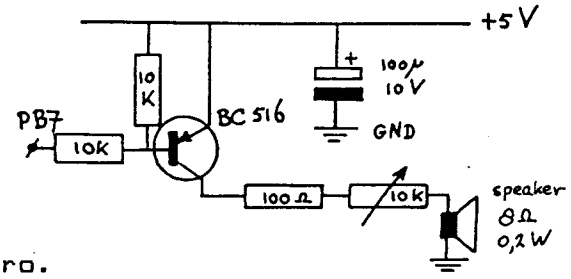
De daarna volgende subroutines zijn allen van het type van het omwisselen van twee operanden of twee registers, waarbij geen gebruik wordt gemaakt van hulplocaties. Het gebruik spreekt voorzichzelf.

```
*********************************************************
*    PUSH  - PUSH REGISTERS ON STACK                    *
*            SAVE REGISTERS P, A, X AND Y ON THE STACK  *
*********************************************************

STACK  *       $0100       ; TOP OF STACK
                           ; BEGIN
PUSH   PHP                 ;    PUSH (P)
       PHA                 ;    PUSH (A)
       TXA                 ;
       PHA                 ;    PUSH (X)
       TYA                 ;
       PHA                 ;    PUSH (Y)
       TSX                 ;    (X) := (S)
       LDAAX STACK +06     ;    PUSH (RETURN ADDRESS)
       PHA                 ;    (* COPY OF RETURN ADDRES ON
       LDAAX STACK +05     ;    ENTRY *)
       PHA                 ;
       LDAAX STACK +04     ;    PUSH (P)
       PHA                 ;    (* COPY OF (P) ON ENTRY *)
       LDAAX STACK +03     ;    PUSH (A)
       PHA                 ;    (* COPY OF (A) ON ENTRY *)
       LDYAX STACK +01     ;    (Y) := STACK(AX)+1
       LDAAX STACK +02     ;    (X) := STACK(AX)+2
       TAX                 ;
       PLA                 ;    PULL (A)
       PLP                 ;    PULL (P)
       RTS                 ; END

*********************************************************
*    PULL   - PULL REGISTERS FROM STACK                 *
*             RESTORE REGISTERS Y, X, A AND P FROM THE  *
*             STACK (REGISTERS MUST HAVE BEEN SAVED BY  *
*             'PUSH' ROUTINE).                          *
*********************************************************

                           ; BEGIN
PULL   TSX                 ;    (X) := (S)
                           ;    (* COPY RETURN ADDRESS *)
       LDAAX STACK +02     ;    STACK(AX)+8 := STACK(AX)+2
       STAAX STACK +08     ;
       LDAAX STACK +01     ;    STACK(AX)+7 := STACK(AX)+1
       STAAX STACK +07     ;
       PLA                 ;    (S) := (S) + 2
       PLA                 ;
       PLA                 ;    PULL (Y)
       TAY                 ;
       PLA                 ;    PULL (X)
       TAX                 ;
       PLA                 ;    PULL (A)
       PLP                 ;    PULL (P)
       RTS                 ; END
```

```
*********************************************************
*    EXCHGA - EXCHANGE SUBROUTINE                       *
*             EXCHANGES OPERAND A WITH OPERAND B        *
*             WITHOUT USAGE OF WORK AREA'S              *
*********************************************************

                           ; BEGIN
EXCHGA PHP                 ;    PUSH (P)
       PHA                 ;    PUSH (A)
       LDA   OPRNDB        ;    OPRNDA := OPRNDB EOR OPRNDA
       EOR   OPRNDA        ;
       STA   OPRNDA        ;
       EOR   OPRNDB        ;    OPRNDB := OPRNDA EOR OPRNDB
       STA   OPRNDB        ;
       EOR   OPRNDA        ;    OPRNDA := OPRNDB EOR OPRNDA
       STA   OPRNDA        ;
       PLA                 ;    PULL (A)
       PLP                 ;    PULL (P)
       RTS                 ; END

*********************************************************
*    EXCHGB - EXCHANGE SUBROUTINE                       *
*             EXCHANGES OPERAND A WITH OPERAND B        *
*             WITH USAGE OF THE STACK                   *
*********************************************************

                           ; BEGIN
EXCHGB PHP                 ;    PUSH (P)
       PHA                 ;    PUSH (A)
       LDA   OPRNDB        ;    PUSH OPRNDB
       PHA                 ;
       LDA   OPRNDA        ;    OPRNDB := OPRNDA
       STA   OPRNDB        ;
       PLA                 ;    PULL OPRNDA
       STA   OPRNDA        ;
       PLA                 ;    PULL (A)
       PLP                 ;    PULL (P)
       RTS                 ; END

*********************************************************
*    EXCHAY - EXCHANGE SUBROUTINE                       *
*             EXCHANGES ACCUMULATOR WITH REGISTER Y     *
*********************************************************

                           ; BEGIN
EXCHAY PHP                 ;    PUSH (P)
       PHA                 ;    PUSH (A)
       TYA                 ;    PUSH (Y)
       PHA                 ;
       TXA                 ;    PUSH (X)
       PHA                 ;
       TSX                 ;    (X) := (S)
       LDAAX STACK +03     ;    (Y) := STACK(AX)+3
       TAY                 ;    (* (Y) := (A) *)
       PLA                 ;    PULL (X)
       TAX                 ;
       PLA                 ;    (S) := (S) +1
       PLA                 ;    PULL (A)
       PLP                 ;    PULL (P)
       RTS                 ; END

*********************************************************
*    EXCHXY - EXCHANGE SUBROUTINE                       *
*             EXCHANGES REGISTER X WITH REGISTER Y      *
*********************************************************

                           ; BEGIN
EXCHXY PHP                 ;    PUSH (P)
       PHA                 ;    PUSH (A)
       TYA                 ;    PUSH (Y)
       PHA                 ;
       TXA                 ;    (Y) := (X)
       TAY                 ;
       PLA                 ;    PULL (X)
       TAX                 ;
       PLA                 ;    PULL (A)
       PLP                 ;    PULL (P)
       RTS                 ; END

*********************************************************
*    EXCHAX - EXCHANGE SUBROUTINE                       *
*             EXCHANGES ACCUMULATOR WITH REGISTER X     *
*********************************************************

                           ; BEGIN
EXCHAX PHP                 ;    PUSH (P)
       PHA                 ;    PUSH (A)
       TXA                 ;    PUSH (X)
       PHA                 ;
       TSX                 ;    (X) := (S)
       LDAAX STACK +02     ;    (X) := STACK(AX)+2
       TAX                 ;    (* (X) := (A) *)
       PLA                 ;    PULL (A)
                           ;    (* (A) := (X) *)
       PLP                 ;    (S) := (S) +2
       PLP                 ;
       PLP                 ;    PULL (P)
       RTS                 ; END
```

FORTH ON THE JUNIOR

For some time I have a fig-FORTH and a 79 STANDARD FORTH for the JUNIOR at my disposal. I felt the need to have some important I/O addresses accessible in FORTH without having to look them up in the books every time. And behold the friendliness of FORTH: all addresses can get familiar names from previous publications about the JUNIOR. A name like NMI is much easier to remember then the hex-addresses $1A7A and $1A7B. (I had to peep at the book for those two). Changing the NMI vector is, after loading screen 1 and 2, a piece of cake. Let's assume we want to point the NMI vector at address $2000, then typing the following FORTH statements: HEX 2000 NMI ! will do the job.
For those who are not familiar with FORTH: with the word HEX I tell FORTH that every number typed in is in hexadecimal notation. The exclamation mark ! is in FORTH a store operation (like POKE in BASIC). Note: FORTH takes care of the $00 being stored at address $1A7A, and the $20 at address $1A7B. To have a known delay at my disposal I made the word MS (milisecond). How to use this word can be found in screen 3. How it works: first the operand for MS (giving the delay in miliseconds) is checked because when it's equal to zero, we don't have to wait at all. If it's not zero then as many times as is neccessary the following operations are carried out: load timer CNTB (systemclock:8) with $7C (=124) and check if the timerflag (bit 7) is set (timerflag greater than $7F). In FORTH it is possible to handle numbers in decimal (after typing DECIMAL) or hexadecimal (after typing HEX). Because I have been working on an assembler, it would be handy to have binary in- and output as well. So I made the word BINARY. In this word one can see how easy it is in FORTH to choose a new basenumber: simply store the new base in the systemvariable BASE (basenumbers from 2 to 70 are possible) and "voila", FORTH is working in decimal, octal, binary or even quintal (numbers with base 5). For people having to deal with a lot of conversions like hexadecimal <-> decimal or decimal <-> binary, this is of course something they could only dream of. Screen 4 is just for fun, I made a small circuit (see figure) and connected it with PB7 of the 6522 from the JUNIOR. With the help of this little ciruit I can generate real "hifi" sound with the JUNIOR. The FORTH word TONE works as follows: first load the auxilary control register of the 6522 with $C0 (%1100 0000). This enables the free-running mode of timer 1 and connects its output with PB7. The argument of TONE is placed in the timerlatch. The timer will be loaded with this number every time it passes zero.



Last but not least the timer itself is loaded with the argument and off it goes! The word TOFF disables the tone by writing $00 in the auxilary control register thus stopping the timer. BELL shows how both words can be used, it generates a tone for about one second. To change the frequency alter the argument of TONE, here 300. To change the lenght alter the argument of MS, in the example 1000.

written by: Frans Bakx
Huissteden 1112
6605 HD Wijchen
tel: 08894 - 16389

```
SCR # 1
  0 ( VIA 6522 registers                          JUNIOR          )
  1 FORTH DEFINITIONS HEX
  2 1800 CONSTANT DRB          ( DATA REGISTER B                  )
  3 1801 CONSTANT DRA          ( DATA REGISTER A                  )
  4 1802 CONSTANT DDRB         ( DATA DIRECTION REG B             )
  5 1803 CONSTANT DDRA         ( DATA DIRECTION REG A             )
  6 1804 CONSTANT T1C          ( TIMER 1 LOW                      )
  7 1806 CONSTANT T1L          ( TIMER 1 LATCH LOW                )
  8 1808 CONSTANT T2C          ( TIMER 2 LOW                      )
  9 180A CONSTANT SR           ( SHIFT REGISTER                   )
 10 180B CONSTANT ACR          ( AUXILARY CONTROL REG             )
 11 180C CONSTANT PCR          ( PERIPHERAL CONTROL REG           )
 12 180D CONSTANT IFR          ( INTERRUPT FLAG REGISTER          )
 13 180E CONSTANT IER          ( INTERRUPT ENABLE REGISTER        )
 14 180F CONSTANT DRA2         ( DATA REG A NO HANDSHAKE          )
 15 -->
```

```
SCR # 2
  Ø ( PIA 6532 registers                                  JUNIOR          )
  1 1AD5 CONSTANT RDFLAG          ( FLAG REGISTER                         )
  2 1AF4 CONSTANT CNTA            ( CLK1T           DISABLE IRQ           )
  3 1AF5 CONSTANT CNTB            ( CLK8T           DISABLE IRQ           )
  4 1AF6 CONSTANT CNTC            ( CLK64T          DISABLE IRQ           )
  5 1AF7 CONSTANT CNTD            ( CLK1KT          DISABLE IRQ           )
  6 1AFC CONSTANT CNTE            ( CLK1T           ENABLE  IRQ           )
  7 1AFD CONSTANT CNTF            ( CLK8T           ENABLE  IRQ           )
  8 1AFE CONSTANT CNTG            ( CLK64T          ENABLE  IRQ           )
  9 1AFF CONSTANT CNTH            ( CLK1KT          ENABLE  IRQ           )
 1Ø
 11 ( interrupt vectors on page $1A                                       )
 12 1A7A CONSTANT NMI             ( NMI    VECTOR LOW                     )
 13 1A7C CONSTANT BRKT            ( BREAK  VECTOR LOW                     )
 14 1A7E CONSTANT IRQ             ( IRQ    VECTOR LOW                     )
 15 -->

SCR # 3
  Ø ( utilities                                                           )
  1
  2 ( MS         n ----  delay for aprroximately n milliseconds           )
  3 ( 79-STANDARD REFERENCE WORD SET                                      )
  4 : MS
  5     -DUP IF    Ø DO      7C CNTB !
  6                  BEGIN
  7                      RDFLAG C@ 7F >
  8                  UNTIL
  9                  LOOP
 1Ø           THEN ;
 11
 12
 13 : BINARY     ( ----  set I/O binary                                   )
 14         2 BASE ! ;
 15 -->

SCR # 4
  Ø ( sound with 6522 VIA and speaker connected with PB7                  )
  1 ( TONE       n ----  enable sound, n is frequency                     )
  2 : TONE
  3         CØ  ACR C!              ( free-running mode timer 1           )
  4         DUP T1L ! T1C ! ;       ( load timerlatch and timer           )
  5 ( TOFF        ----     disable sound                                  )
  6 : TOFF        Ø ACR ! ;          ( disable free-running mode          )
  7 DECIMAL
  8 ( BELL        ----     short beep                                     )
  9 ( 79-STANDARD REFERENCE WORD SET                                      )
 1Ø ( frequency can be changed by changing 3ØØ, length by                 )
 11 ( changing 1ØØØ                                                       )
 12 : BELL
 13       3ØØ TONE          1ØØØ MS ( aprox. 1 sec. delay then stop      )
 14           TOFF ;
 15 ;S     frank bakx     huissteden 1112   6605 HD Wijchen

SCR # 5
  Ø ( PEEP       n ----  beep (toggle) speaker                            )
  1 HEX
  2 1A82 CONSTANT DRB2                      ( DATA REGISTER B             )
  3 1A83 CONSTANT DDRB2                     ( DATA DIRECTION REG B        )
  4 : PEEP DDRB2 C@ 8Ø OR DDRB2 C!          ( enable PB7 output           )
  5        Ø DO DRB2 C@ 8Ø XOR DRB2 C!      ( toggle PB7 output           )
  6        LOOP                             ( n times                     )
  7        DDRB2 C@ 7F AND DDRB2 C!         ( disable PB7 output          )
  8 ;
  9
 1Ø
 11
 12
 13
 14
 15 ( frank bakx         huissteden 1112   6605 HD Wijchen                )
OK
```

****
** ADAPTATION MINI-MODEM BAUDRATE 1200/75 **
****

Author: A.v.d. Hombergh, The Netherlands

Eversince I have had a modem, I wanted to work according to the V21 protocol (transmission speed 300 Bd full duplex) and the V23 protocol. The latter works with a transmission speed on 1200/75 Bd, the so-called split baudrate.
Since I have built the Mini Modem from Elektor, the option V23 was not possible without further preface.
For this modem has no interspeeder i.e. the Modem provides for the translation of 1200 Bd to 75 Bd and the other way round from 75 Bd to 1200 Bd.
Yet in order to be able to work according tb both options I have introduced a hardware adaptation in the modem. The system whereby this modem jointly works is the EC65 from Elektor. I have also adapted the communication program in order to be able to work with both protocols.

Hardware adaptation Mini-Modem as described in the German Elektor Special EC-4.

1. Connect by means of wiring draw on IC 1 (AM7910)
   pin 10 with 28 = TD...BTD
   pin 13 with 14 = CTS..BCTS
   pin 25 with 27 = CD...BCD
   pin 26 with 15 = RD...BRD
2. Mount on a help print a divider Mos IC 4040 and use, if possible, a fourdecks switch with 4 settings/positions or else a separate switch with 3 or 4 settings/positions. The clock inlet of the divider (pin 10) is joined with the outlet of the oscilator of the modem (R24). This clock 2.4576 Mc is divided by the 4040 to a frequency which is 16 times higher than the receiving Data frequency, by 1200 Bd this is 19200 Hz, by 300 Bd 4800 Hz and by 75 Bd this frequency is 1200 Hz.
   The switch with 3 or 4 settings is joined with:
   pin 1 with Q9 from 4040 frequency outlet 4800 Hz
   pin 2 with Q9 from 4040 frequency outlet 4800 Hz
   pin 3 with Q7 from 4040 frequency outlet 19200 Hz
   pin 4 with Q11 from 4040 frequency outlet 1200 Hz
   The middle contact of the switch is connected with pin 17 of the D-25 connector V24 (RS 232) interface.

The Mini Modem is now able to be used with VIDITEL (VIDITEX) for example. Nevertheless it cannot be used without reservation with some modems such as FIDO whereby nodes are used.
The CCITT dictates that with data communication via the connected telephone-network, a frequency of 2100 Hz is send for the start of data communication. This is in the telephone systems for the disconnection of echo suppressors. This is employed in international communications. Therefore the AM7910 sends this tone as first frequency, except in the mode 300 Bd originate. The length of the tone is 3 seconds. There are FIDO modems reacting incorrectly on this 2100 Hz. The result of this is short circuiting of the connection. A so-called Multi Modem recognizes by the carrier with which sort of modem it is going to communicate.
If this mode, in answering, recognizes a carrier of 1080 Hz then the communication takes place according to the V21 protocol, this 300 Bd full duplex. By a recognition of 390 Hz, then the protocol is V23 split baudrate. The reception speed for the Multi Modem is then 75 Bd and the transmission speed 1200 Bd.
My experience is that some Multi Modems used by FIDO nodes react incorrectly at the frequency of 2100 Hz.
In order to get round the above described problem I have introduced the following adaptation in the Mini Modem.
Switch the AM7910 "ON LINE" before the telephone connection has been made. Therefor pin 1 of the AM7910 is made low but the line relay is not confirmed. For the buildup of the telephone connection lasts longer than 3 seconds.

Hardware adaptation Mini Modem by answer tone of 211 Hz.

1a Use for S1 a double switch.
   Scratch pin 13 from IC 6 loose from the connection to IC 1.
   Connect pin 13 with a resistor of 2K2 to + 5 Volts and connect pin 13 from IC 6 with S1(b).
   The other side of S1(b) is connected with IC 1 pin 1.
 b Scratch from IC 6 the pins 1 and 2 loose from earth.
   Connect pin 1 from IC 6 with middle contact from S2-b or with MC-1 from IC 1.
   Connect pin 2 from IC 6 with R 19 (47K) or with pin 6 from FF1 (IC 8).

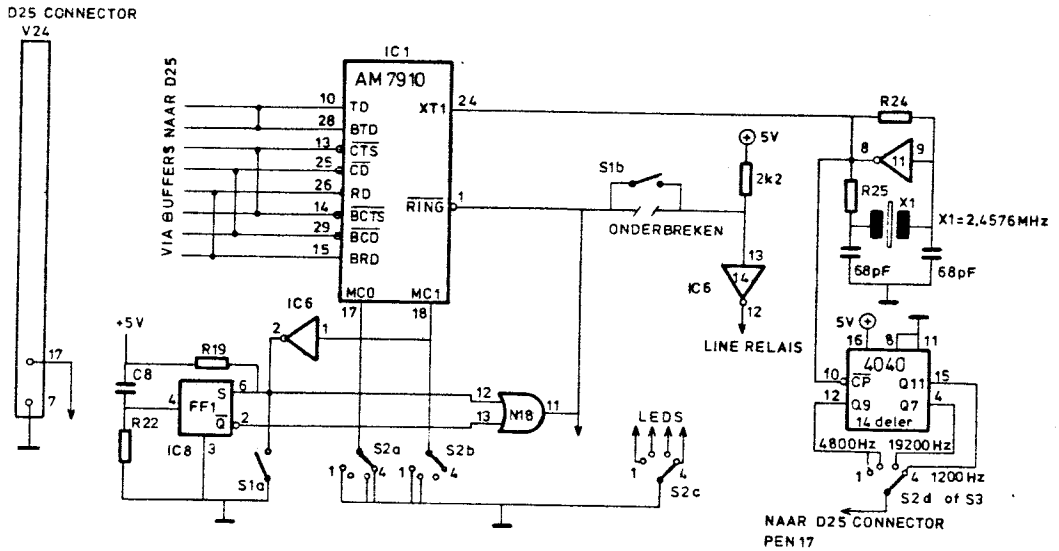
Hardware adaptation EC65 CPU board.

Connect pin 5 from the acia (6551) with pin 9 from PL 7, and bring pin 9 from PL 7 to the D-25 connector pin 17.

The above mentioned modem works together with the communication program that is available for the EC65(K).
I have also made some software adaptations in order to be able to work with the V21 as well as the V23 split-baudrate.

1. Boot your communication program in your system and choose option E from the main menu.
   Load track 13 to memory location $A274 with CA A274=13,1.
   Jump to the monitor and change the following data:

	Value			Value	
Addr:	Old	New	Addr:	Old	New
$A5FC	$33	$20	$A60D	$32	$31
$A5FD	$30	$37	$A60E	$34	$32
$A5FE	$30	$35	$A62A	$16	$02
$A602	$36	$33	$A633	$17	$06
$A607	$31	$20	$A63C	$18	$07
$A608	$32	$36	$A645	$1A	$08



HERMAN ZONDAG

Save track 13 with SA 13,1=A274/8

Next load track 05 to memory location $AA00 with CA AA00=05,1 and change the following data.

$AF70	$16	$06	$B124	$31	$20
$B118	$30	$20	$B125	$32	$36
$B119	$30	$37	$B12B	$32	$31
$B11A	$30	$35	$B12C	$34	$32
$B11E	$36	$33			

Save track 05 again with SA 05,1=AA00/8.

The software is also adapted. By booting your communication program the ACIA is now programmed on the externed receive clock. The transmission speed of the originate modem is displayed in the status bar in the right hand corner of the screen.
By, for example, VIDITEL (VIEUWTEX) or by FIDO Multi Modem this is 75 baud. The default values remain the same by formerly up-booting (300 Bd 8 bits disabled parity 1 stop bit).
By FIDO according to V23 you therefore only change the transmission-speed 75 Bd.
By VIDITEL (in Holland) the ACIA parameters are 75 Bd, 7 data bits even parity 1 stop bit.
By the menu option X (exchange ACIA mode) the transmission speed 2400 Bd is dropped because another modem has to be adapted anyway.
By the transmission speed 4800 Bd, 9600 Bd and 19200 Bd the ACIA remains programmed on internal clock.
In the V21 mode originate Switch S2 is set to position 1 300 Bd full duplex.
In the V21 mode originate Switch S2 is set to position 2 300 Bd full duplex.
In the V23 mode originate Switch S2 is set to position 3 75 Bd Trans. 1200 Bd Rec.
In the V23 mode originate Switch S2 is set to position 4 1200 Bd Trans. 75 Bd Rec.

The advantage of this modification is access to more databanks and the data transmission is quicker, for example by downloading, and this makes a difference in the telephone costs. Naturally, the modification of the Mini Modem can also put into use other modems which don't make use of an interspeeder, provided that the IC AM7910 or 7911 is employed.
For the complete wiring diagram of the Mini-Modem review the German Elektor Sonderheft 4.

****
** NEC PINWRITER P1 DUMP **
****

Frank Vergoossen, Holland.

Dit programma is voor de ACORN ATOM, om grafische mode 4 plaatjes te printen met de NEC Pinwriter P1. Om het printen te starten CTRL-SHIFT indrukken. Na een BREAK moet het programma opnieuw aangeroepen worden met LINK (beginadres). Dit programma kan alleen kleine plaatjes uitprinten, terwijl hier maar 8 van de 16 beschikbare printernaalden gebruikt worden, zodat het printen no wel sneller zou kunnen. Bit 7 van de printerconnector moet verbonden worden met bit 3 van de C-poort van de 8255, volgens de modificatie beschreven in Acorn Nieuws bundel 1982, blz. 17. Voor verbeteringen aan dit programma houd ik me aanbevolen.

```
 10REM NEC PINWRITER P1 DUMP
 20REM DOOR FRANK VERGOOSSEN
 30J=30:DIMLLJ:F.I=0TOJ:LLI=#FFFF:N.:@=0
 40P.$12"NEC PINWRITER P1 GRAPHICS DUMP"''
 50IN."GEEF STARTADRES "Z
 60P.$21:F.I=1TO2:P=Z
 70[
 80:LL0\INVERTEREN
 90LDA@0:STA#23F
100JSR#FD1A\PIEPTOON
110JSR#F7D1\PRINT TEKST
120]$P="INVERTEREN (J/N) ?":P=P+L.P:[
130NOP:JSR#FFE6\LEES TOETS
140\VERGELIJK MET "J"
150CMP@#4A:BEQLL2
160\NIET INVERTEREN
170DEC#23F
180:LL2JSR#FFED\CR/LF
190\INTERRUPT VECTOR
200LDA@LL3%256:STA#204
210LDA@LL3/256:STA#205
220\INTERRUPT AAN
230CLI
240LDA@#C0:STA#B80B:STA#B80E
250\TIMER 1
260LDA@#80:STA#B804:STA#B805
270JMP#C55B\TERUG NAAR BASIC
280:LL3\X EN Y OP STACK
290TXA:PHA:TYA:PHA
300\CTRL-SHIFT
310LDA#B001:CMP@#3F:BEQLL15
320JMPLL28\TERUG
330:LL15\MODE 4
340LDA@32:STA#56
350\PRINTER AAN
360LDA@2:JSR#FEFB
370\ESC."T08"
380LDA@#1B:JSRLL30
390LDA@#54:JSRLL30
400LDA@#30:JSRLL30
410LDA@#38:JSRLL30
420JSRLL29:JSRLL29\2*CR
430\ADRES #8000
440LDA@#80:STA#53
450LDY@0:STY#52
460:LL16\ESC."I0008"
470LDA@#1B:JSRLL30
480LDA@#49:JSRLL30
490LDA@#30:JSRLL30
500JSRLL30:JSRLL30
510LDA@#38:JSRLL30
520\X EN Y OP 0
530LDY@0:LDX@7
540:LL19\LEES 8 BYTES
550LDA(#52),Y:STA#5A,X
560\VOLGENDE BIT
570TYA:CLC:ADC@32:TAY
580DEX:BPLLL19
590LDY@8
600:LL20LDX@7
610:LL21\SCHUIF BIT
620ASL#5A,X
630\SCHUIF CARRY
640ROR#57
650\TEST 8 BITS
660DEX:BPLLL21
670\EVT. INVERTEREN
680LDA#57:EOR#23F:STA#57
690\PRINTERBIT 7="0"
700LDA@7:STA#B002
710\TEST BIT 7
720BIT#57:BPLLL22
730\PRINTERBIT 7="1"
740LDA@15:STA#B002
750:LL22\NAAR PRINTER
760LDA#57:JSRLL30
770LDA@7:STA#B002
780LDA@0:JSRLL30
790\TEST 8 BYTES
800DEY:BNELL20
810\VERHOOG ADRES
820INC#52:LDA#52
830\TEST REGELEINDE
840CMP#56:BNELL16
850LDX@7
860:LL23\VOLGENDE REGEL
870LDA#52:CLC:ADC@32
880STA#52:BCCLL24
890\VERHOOG MSB
900INC#53:LDA#53
910\TEST GEHEUGENGRENS
920CMP@#98:BEQLL26
930:LL24
940DEX:BNELL23
950JSRLL29\CR
960\ADRES VOLGENDE REGEL
970LDA#52:CLC:ADC@32:STA#56
980JMPLL16
990:LL26\PIEPTOON+CR
1000JSR#FD1A:JSRLL29
1010\PRINTER UIT
1020LDA@3:JSR#FEFB
1030:LL28\HERSTART TIMER
1040LDA@#80:STA#B804:STA#B805
1050\Y, X EN A VAN STACK
1060PLA:TAY:PLA:TAX:PLA
1070RTI\TERUG
1080:LL29\CR
1090LDA@13:JMPLL30
1100:LL30\NAAR PRINTER
1110PHA:JMP#FF08
1120]N.:P.$6"CODE VAN #"&Z" TOT #"&P'
1130LI.Z:E.
```

```
36 44 MLIST
SCR # 36
  0 ( MAANLANDER. 1 VAN 9. CASES.                  )
  1 DECIMAL ( CASES KOMT UIT FORTH OK)
  2 : RANGE ROT DUP ROT SWAP ) IF 0 ELSE 1 THEN
  3         ROT ROT SWAP ) IF 0 ELSE 1 THEN :
  4 : BEGIN-CASES CSP @ !CSP COMPILE DUP 0 4 ; IMMEDIATE
  5 : CASE 4 ?PAIRS COMPILE OVER COMPILE = COMPILE OBRANCH
  6       HERE 0 , COMPILE DROP 5 ; IMMEDIATE
  7 : RANGE-CASE 4 ?PAIRS COMPILE RANGE COMPILE = COMPILE
  8              OBRANCH HERE 0 , 5 ; IMMEDIATE
  9 : ELSE-CASE 4 ?PAIRS 0 , 5 ; IMMEDIATE
 10 : END-CASE 5 ?PAIRS COMPILE BRANCH HERE 0 , SWAP 2
 11          [COMPILE] ENDIF 4 ; IMMEDIATE
 12 : END-CASES 4 ?PAIRS BEGIN SP@ CSP @ = 0= WHILE 2
 13   [COMPILE] ENDIF REPEAT CSP ! COMPILE DROP ; IMMEDIATE
 14 : KLAAR ." SCHERM GECOMPILEERD" CR ; IMMEDIATE
 15 : WIS EMIT 2 WAIT : KLAAR  -->


SCR # 37
  0 ( MAANLANDER. 2 VAN 9 )
  1 VARIABLE VOORUIT  VARIABLE ACHTERUIT  VARIABLE REMMEN
  2 VARIABLE STIJGEN VARIABLE MOE VARIABLE HOOGTE
  3 VARIABLE SNELHEID VARIABLE BRAND VARIABLE ZWAAR
  4 VARIABLE HOR VARIABLE VERT VARIABLE GANG
  5 : BEGINWAARDEN 1 MOE ! 9 HOOGTE ! 0 SNELHEID !
  6                4 VERT ! 3 HOR ! ;
  7 : BEGIN1 99 BRAND ! 2 ZWAAR ! 3 GANG ! ;
  8 : BEGIN2 80 BRAND ! 3 ZWAAR ! 3 GANG ! ;
  9 : BEGIN3 70 BRAND ! 3 ZWAAR ! 6 GANG ! ;
 10 : DAAL 0 DO 10 EMIT LOOP ;
 11 : TABU 0 DO 9 EMIT LOOP ;
 12 : TUSSEN ROT DUP ROT ) SWAP ROT ( OR NOT ;
 13 : REGEL 10 EMIT 13 EMIT ;
 14 : CURS 28 WIS 13 DAAL 24 TABU ;
 15 : IE 73 EMIT :     KLAAR    -->


SCR # 38
  0 ( MAANLANDER. 3 VAN 9 )
  1 : STREEP 0 DO 45 EMIT LOOP ;
  2 : ISSEN 0 DO 61 EMIT LOOP ;
  3 : LANDER 28 WIS VERT @ DAAL HOR @ TABU ." -O-" ;
  4 : EXLAN 28 WIS VERT @ DAAL HOR @ TABU 3 SPACES ;
  5 : EIND CURS REGEL ." U BENT NEERGESTORT !!!! " ABORT ;
  6 : ?HOOG HOOGTE @ 0( IF EIND ELSE HOOGTE @ 9 ) IF
  7  ." U BENT TE HOOG !! DE MAANLANDER EXPLODEERT !! "
  8  ABORT THEN THEN ;
  9 : ?HOR HOR @ 0( IF ." U RAAKT ACHTER !! "  ELSE HOR @ 50
 10  ) IF ." U BENT TE VER WEG GEGAAN !! " ABORT THEN THEN ;
 11 : BOTS ." FATALE BOTSING MET HET MOEDERSCHIP ! " ABORT ;
 12 : PAR 28 WIS 2 DAAL 7 TABU HOOGTE ? 14 TABU
 13       SNELHEID ? 15 TABU BRAND ? ;
 14 : EXPAR 28 WIS 2 DAAL 6 TABU 2 SPACES 13 TABU 4 SPACES
 15         12 TABU 7 SPACES PAR ;  KLAAR  -->


SCR # 39
  0 ( MAANLANDER. 4 VAN 9 )
  1 : TEST5 SNELHEID @ DUP 0( 0= OR IF
  2      ." DE LANDING IS GESLAAGD ! " ELSE EIND THEN ;
  3 : TEST4 HOR @ MOE @ 1 + DUP 2 - SWAP TUSSEN
  4         IF BOTS ELSE HOR @ MOE @ 3 + DUP 2 +
  5         TUSSEN IF BOTS THEN THEN ;
  6 : TEST3 HOR @ 23 = IF TEST5 THEN ;
  7 : TEST2 VERT @ 13 = IF TEST3 THEN ;
  8 : BERG REGEL ." U BENT TEGEN DE BERG GEVLOGEN ! " ABORT ;
  9 : ?THUIS SNELHEID @ -2 = IF ." U BENT THUIS. PROFICIAT ! "
 10          ABORT ELSE TEST4 THEN ;
 11 : FUEL BRAND @ VOORUIT @ - ACHTERUIT @ - REMMEN @ -
 12       STIJGEN @ - DUP 0( IF CURS REGEL
 13       ." DE BRANDSTOF IS OP !!! " ABORT ELSE
 14       BRAND ! THEN ;
 15 KLAAR   -->
```

```
SCR # 40
  0 ( MAANLANDER. 5 VAN 9  )
  1 : REKEN REGEL ."                        MOMENTJE AUB" CURS
  2          EXLAN FUEL SNELHEID @ ZWAAR @ + STIJGEN @
  3          - REMMEN @ - SNELHEID ! HOR @ VOORUIT
  4          @ + HOR ! HOR @ ACHTERUIT @ - HOR !
  5          VERT @ SNELHEID @ + 1 + VERT !
  6          13 VERT @ - HOOGTE ! GANG @ MOE +! :
  7 : VOOR ." GEEFT U BRANDSTOF VOORUIT ? (0-9) "
  8        CURS KEY 48 - VOORUIT ! :
  9 : ACHTER ." GEEFT U BRANDSTOF ACHTERUIT ? (0-9) "
 10          CURS KEY 48 - ACHTERUIT ! :
 11 : REM ." GEEFT U BRANDSTOF OM AF TE REMMEN ? (0-9) "
 12       CURS KEY 48 - REMMEN ! :
 13 : STIJG ." GEEFT U BRANDSTOF OM OP TE STIJGEN ? (0-9) "
 14          CURS KEY 48 - STIJGEN ! :
 15  KLAAR   -->

SCR # 41
  0 ( MAANLANDER. 6 VAN 9 )
  1 : BEELD  CLS 2 WAIT IE 4 SPACES ." HOOGTE" 4 SPACES IE 2
  2  SPACES ." VALSNELHEID" 2 SPACES IE 2 SPACES
  3  ." BRANDSTOF-RESERVE" SPACE IE CR IE 14 STREEP IE 15 STREEP
  4  IE 20 STREEP IE CR IE 14 SPACES IE 15 SPACES IE 20 SPACES
  5  IE CR IE 14 ISSEN IE 15 ISSEN IE 20 ISSEN IE CR IE 51
  6  SPACES IE CR IE 51 SPACES IE CR IE 3 SPACES ." /\" 46
  7  SPACES IE CR IE 2 SPACES ." /" 2 SPACES ." \-\" 28
  8  SPACES ." /\" 13 SPACES IE CR IE SPACE ." /" 6 SPACES ." \"
  9  26 SPACES ." /  \" 12 SPACES IE CR IE ." /" 9 SPACES ." \"
 10  22 SPACES ." /" 6 SPACES ." \" 10 SPACES IE CR IE 12 SPACES
 11  ." \" 18 SPACES ." /" 10 SPACES ." \" 8 SPACES IE CR IE 14
 12  SPACES ." \---\" 10 SPACES ." /" 14 SPACES ." \" 6 SPACES
 13  IE CR IE 19 SPACES ." \" 7 SPACES ." /" 18 SPACES ." \"
 14  4 SPACES IE CR IE 20 STREEP ." \" 2 STREEP ." ^" 2 STREEP
 15  ." /" 20 STREEP ." \" 3 STREEP IE CR :  KLAAR  -->

SCR # 42
  0 ( MAANLANDER. 7 VAN 9 )
  1 : TEST1 VERT @ 3 - BEGIN-CASES 1 CASE MOE @ 2 + HOR @ =
  2  IF ?THUIS ELSE TEST4 THEN END-CASE
  3  3 CASE HOR @ 2 5 TUSSEN IF BERG THEN END-CASE
  4  4 CASE HOR @ 1 6 TUSSEN IF BERG ELSE HOR @ 35 38
  5  TUSSEN IF BERG THEN THEN END-CASE 5 CASE
  6  HOR @ 0 7 TUSSEN IF BERG ELSE HOR @ 34 39 TUSSEN
  7    IF BERG THEN THEN END-CASE 6 CASE HOR @ 0 11 TUSSEN IF
  8  BERG ELSE HOR @ 32 41 TUSSEN IF BERG THEN THEN END-CASE
  9  7 CASE HOR @ 0 13 TUSSEN IF BERG ELSE HOR @ 30 43 TUSSEN IF
 10  BERG THEN THEN END-CASE 8 CASE HOR @ 0 19 TUSSEN IF BERG
 11  ELSE HOR @ 28 45 TUSSEN IF BERG THEN THEN END-CASE 9 CASE
 12  HOR @ 0 20 TUSSEN IF BERG ELSE HOR @ 26 47 TUSSEN IF BERG
 13   THEN THEN END-CASE 10 CASE HOR @ 0 21 TUSSEN IF BERG
 14   ELSE HOR @ 25 48 TUSSEN IF BERG THEN THEN END-CASE
 15  END-CASES :    KLAAR   -->

SCR # 43
  0  ( MAANLANDER. 8 VAN 9 )
  1 : MOED 28 WIS 4 DAAL 9 EMIT MOE @ 0 DO 32 EMIT LOOP
  2          ." (   )" :
  3 : TEST MOE @ 50 ) IF 9 DAAL REGEL ." TE LAAT!!! " ABORT THEN :
  4 : UITVOER BEELD BEGIN MOED LANDER TEST EXPAR CURS REGEL ?HOOG
  5 ?HOR TEST2 TEST1 VOOR REGEL ACHTER REGEL REM REGEL STIJG
  6  REKEN AGAIN :
  7 : KIES 12 WIS 4 DAAL 15 SPACES ." KIES MOEILIJKHEIDS-GRAAD"
  8  CR CR CR 15 SPACES ." BEGINNER:        TOETS IN 1. " CR 15
  9  SPACES ." GEVORDERD:       TOETS IN 2." CR 15 SPACES
 10  ." ERVAREN PILOOT: TOETS IN 3." CR KEY DUP DUP 49 = IF
 11  BEGIN1 UITVOER ELSE 50 = IF BEGIN2 UITVOER ELSE 51 =
 12  IF BEGIN3 UITVOER THEN THEN THEN :
 13 : TEKST 12 WIS 20 SPACES ." MAANLANDER" CR 20 SPACES 10
 14  STREEP CR CR ." HET IS DE BEDOELING OM MET DE KLEINE" CR
 15  ." MAANLANDER - VANUIT HET MOEDERSCHIP - EEN" CR KLAAR -->
```

```
SCR # 44
  0 ( MAANLANDER. 9 VAN 9 )
  1 ." ZACHTE LANDING TE MAKEN OP HET KNIPPERENDE " CR
  2 ." ^ TEKENTJE. DAARNA MOET WEER WORDEN OPGESTEGEN" CR
  3 ." NAAR HET MOEDERSCHIP. DAT ZICH INMIDDELS VERPLAATST" CR
  4 ." HEEFT. DIT MOET GEBEUREN VOORDAT HET MOEDERSCHIP" CR
  5 ." BUITEN BEELD KOMT EN REKENING HOUDEND MET ZWAARTEKRACHT" CR
  6 ." EN DE BESCHIKBARE BRANDSTOF. LET OOK OP DE BERGEN !! " CR
  7  CR ." DURFT U DE REIS AAN ? " CR
  8    ." ZOJA. DRUK DAN OP DE TOETS 'J' " CR
  9   ." ZONEE. DRUK DAN OP DE TOETS 'N' "
 10    KEY 74 = IF KIES ELSE ABORT THEN :
 11 : MAANLANDER BEGINWAARDEN TEKST :
 12 KLAAR    :S
 13
 14
 15

OK
```

****
**** REACTIE OP PROBLEEM VAN R. BAARSLAG, EDITIE 48 ****
****

Door : Coen Boltjes, Nederland.

De decimale mode van de 6502 stelt de gebruiker in staat met decimale getallen te rekenen. 08 + 03 wordt dan 11, in plaats van 0B in de 'normale' hexadecimale mode.

Het probleem van R. Baarslag uit DE 6502 KENNER nummer 48, pagina 27, is terug te voeren tot het feit dat de meeste programmeurs bij het schrijven van routines uitgaan van de hexadecimale mode. Een voorbeeld: In de VDU software wordt de cursorpositie bepaald door een optelling van de CURRENT LINE POINTER, RAMBEG en COLUM. Hierbij wordt er door het programma vanuit gegaan dat deze routines in de hexadecimale mode worden doorlopen, en zijn er in de decimale mode problemen te verwachten. Hoe zijn deze op elegante wijze op te lossen? maak voor iedere externe subroutine een nieuwe subroutine bestaande uit:

```
PHP          ; Save processor status
CLD          ; Hexadecimal mode
JSR Routine  ; External routine
PLP          ; Restore processor status
RTS          ; Exit
```

Het voordeel van de Push en Pull operaties in deze routine is dat na het doorlopen van de routine de 'DECIMAL FLAG' de waarde heeft als voor het aanroepen van de routine. Hierdoor is de routine ook bruikbaar in de hexadecimale mode. Dit in tegenstelling tot de volgende routine,

```
CLD          ; Hexadecimal mode
JSR Routine  ; External routine
SED          ; Decimal mode
RTS          ; Exit
```

die altijd in de decimale mode wordt beëindigd.

Dit Pushen en Pullen van de processorstatus kan ook worden gebruikt in routines die niet mogen worden geinterrumpeerd, en in 'gewone' en in interruptroutines worden aangeroepen:

```
PSP          ; Save processor status
SEI          ; No interrupts
JSR Routine  ; External routine
PLP          ; Restore processor status
RTS          ; Exit
```

****
**** COMMODORE 64 I/O PORT VISIBLE ON MONITOR ****
****

By : Gerard van Roekel, The Netherlands
Transl.: Bart van Pelt

By using the following programme it's possible to watch the I/O. It shows the OUTPUT DATA REGISTER B, which can be affected by address 56577, and the DATA DIRECTION B, which is to be altered by address 56579.

```
100 FORI= 49152 TO 49264
110 READA:POKEI,A:NEXT
120 SYS 49152
130 DATA 120,169,013,141,020
140 DATA 003,169,192,141,021
150 DATA 003,088,096,169,058
160 DATA 141,031,004,141,071
170 DATA 004,141,111,004,169
180 DATA 031,133,250,169,004
190 DATA 133,251,160,008,185
200 DATA 104,192,145,250,136
210 DATA 208,248,160,008,169
220 DATA 071,133,250,169,004
230 DATA 173,003,221,074,144
240 DATA 005,162,129,076,063
250 DATA 192,162,133,072,138
260 DATA 145,250,104,136,208
270 DATA 238,160,008,169,111
280 DATA 133,250,169,004,133
290 DATA 251,173,001,221,074
300 DATA 144,005,162,177,076
310 DATA 094,192,162,176,072
320 DATA 138,145,250,104,136
330 DATA 208,238,076,049,234
340 DATA 055,054,053,052,051
350 DATA 050,049,048
360 END
```

```
                          START:

                          :76543210
                          :EEEEEEEE
                          :11111111

                          :76543210
POKE56579,111  :EAAEAAAA
POKE56577,77   :11011101

POKE56579,73   :76543210
POKE56577,117  :EAEEAEEA
                          :11110111

POKE56579,250  :76543210
POKE56577,33   :AAAAAEAE
                          :00100101

E = ENTRY
A = OUTPUT
```

By altering address 56577 one can decide the in- or output to be 0 or 1. By altering address 56579 a port is fixed to be in- or output.
Beside the programme there are some examples. It is to be taken into account , that the decimals behind the comma, are translated into hexadecimal notation by the computer. E.g. the figure 97 becomes 61 (hexadecimal) or 0110 0001. The I/O data stay at the upper right side of the monitor. Next programme shows all possible stages the I/O port can assume.

```
10 REM DETERMINE 1 OR 0 ON PORT
20 A=0
30 POKE56577,A
40 A=A+1
50 IFA=256THENEND
55 FORI=1 TO 100:NEXTI
60 GOTO30
```

```
10 DETERMINE IN- OR OUTPUT
20 B=0
30 POKE56579,B
40 B=B+1
50 IFB=256THENEND
55 FORI=1 TO 100:NEXTI
60 GOTO30
```

Hopefully this programme gives you some more insight on the I/O part of your computer.

Ernst Elderenbosch, Holland.

Een tip voor DOS65 Basicode gebruikers: De nieuwe Basic versie 2.10 staat niet toe dat er een REM statement op een regel staat zonder tekst. Deze regels komen nogal eens voor in de Basicode programma's en kunnen eenvoudig vervangen worden door regels met een dubbele punt (of helemaal weggelaten worden).

SEND YOUR SELF-DEVELOPED PROGRAMMES TO THE EDITORIAL OFFICE, JAC. JORDAENSSTR. 15, NL-2923 CK KRIMPEN/IJSSEL.

```
                *****   PRINTER ROUTINE   *****
                         STAR DP - 510

                DOOR :  ALFONS  VAN DE  MEUTTER
                        MECHELBAAN 49
                        B-3150 HEIST  O/D  BERG
                        BELGIE

Printer/outch. points here instead H# 1334

xxx1  48                 PHA                save Acc on stack
xxx2  2C 1C 0F           BIT ONLINE         is printer on ?
xxx5  10 18              BPL $xx1F          zero means OFF --) skip Printer
xxx7  2C 18 0F           BIT BUSY
xx0A  10 09              BPL $xx15          zero means Ready --) skip Wait
xx0C  20 3A 1A WAIT      JSR DELAY          300 mSec delay ) give time to
xx0F  20 3A 1A           JSR DELAY          600 mSec total ) empty buffer
xx12  18                 CLC
xx13  90 ED              BCC $xxx2          (On-line test)
xx15  68                 PLA                restore byte to print
xx16  48                 PHA                back on stack for entry or-Off
xx17  8D 20 0F           STA $0F20          on parallel-out port
                                            (must be in handshake mode by an
                                             extension of RESTTY or RESET)
xx1A  2C 1F 0F           BIT $0F1F          ACKNOWLEDGE
xx1D  30 FB              BMI $xx1A          (must be zero (PB-6))
          Cont. if hardcopy skipped.
xx1F  68                 PLA                restore Accu/adapt stackpointer
xx20  8E 60 0C           STX TEMP           buff X
xx23  4C 37 13           JMP CONT           at original video-output

You can patch $1334  4C xx x1     (8E 60 0C)

For independant printer routine  xx20  60      RTS

For JUNIOR :     Exchange 8E 60 0C into 8E 60 1A
                 and create a delay of 300 mSec. elsewhere
                 (1A3A = dedicated to other purposes)

CONNECTIONS:
===========
Used port = type 8154  addressed at 0F00-0F24 (0F80-0FFF=RAM)
0F01...0F07  =  read or reset (by write) bits port A
0F08...0F0F  =  idem for port B
0F10...0F17  =  read or set (by write) bit of port A
0F18...0F1F  =  idem for port B
0F20         =  in/out port A
0F21         =  in/out port B
0F22         =  Data Direction Reg. port A
0F23         =  idem port B
0F24         =  Mode Reg. (Write-only)
In Mode 03, port A = output. Writing a byte to it, pulls down bit 6 of port B.
Bit 7 port B must be pulled down (edge-detect) for ackn. and sets
bit 6 port B back high.
```

Bit 6 port B as wel as bit 7 port B may be tested for occurency of the ackn. by the printer.
DDR-A must be FF (all outputs)
DDR-B must be 40 (PB-6=output/rest=inputs)

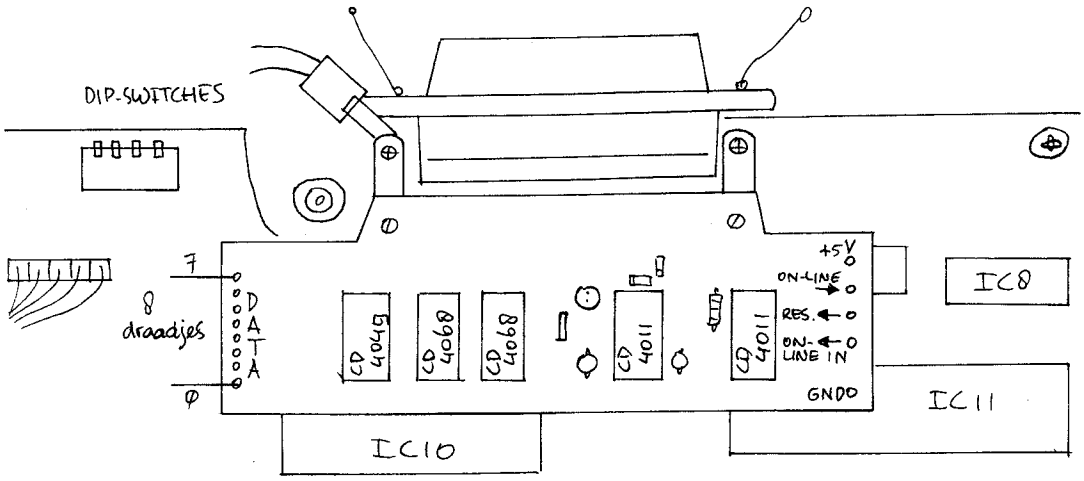
Bijkomende inlichtingen
==========

Teneinde ondubbelzinnige signalen te krijgen op de lijnen Busy, Error, On line mode, is het absoluut noodzakelijk om diode D 43 te overbruggen (of te verwijderen, en de doorgekraste baan te herstellen).
Je zal, zonder die ingreep, geen probleem hebben als de plug van de printer wordt uitgetrokken.
Echter, (zonder ingreep) als DAV laag gaat, zal die lijn via de pull-up weerstanden ook op de andere lijnen invloed hebben. Zo zal de "1" op de strobe doordringen tot op de "selected" lijn, de software geeft een byte, maar zal vruchteloos wachten op de acknowledge (printer staat immers af).
Kortom, de problemen ontstaan door HET ZWEVEN van de ingangen + uitgangen(!) als de printer niet onder spanning is.

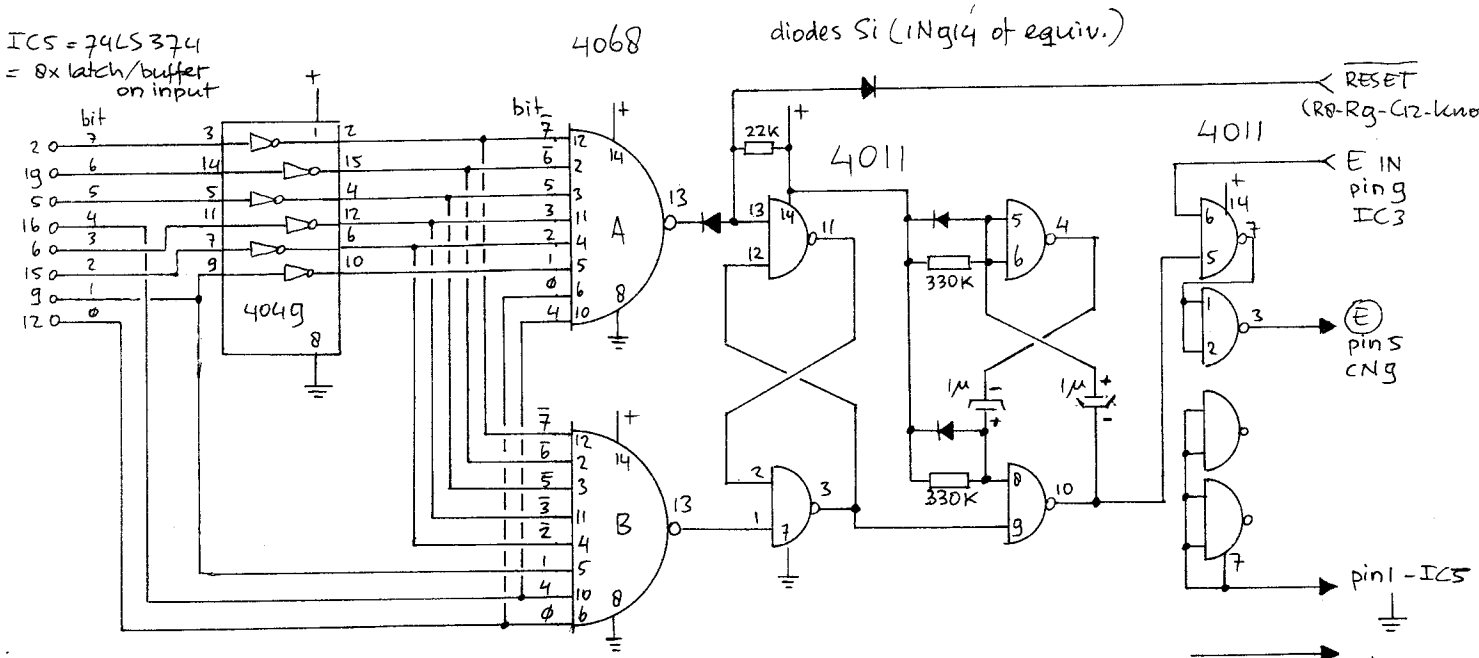
Waar vind je nu die diode ??
Er staan 2 weerstand-array's op de print, vlak achter de centronics-connector. De beide array's hebben een diode van pin 1 naar +5 Volt. Je mag ze beide kortleggen, of wegnemen + printspoor herstellen met een draadje.

Nog een kleine onhebbelijkheid is, dat het softwarematig uitschakelen door CHR$(19) of door DC-3 code, niet zichtbaar is. De handshake werkt perfekt, alleen, er wordt niets geprint. Ik heb dit opgelost door een huisprint toe te voegen, al moet ik zeggen dat dit een tekort is wat ik de producent aanwrijf. De ON-LINE (SELECTED) uitgang is immers de uitgang van een poort (IC-10 ofte D-7800 microprocessor), zodat het probleem via de software en timers van dit IC opgelost had moeten zijn (het lampje wordt trouwens uit een andere lijn van dit IC geschakeld)

Nu werkt het als volgt :
Bij aanschakelen van de printer gaat ON-LINE lampje aan. Bij druk op de toets ON-LINE gaat dit uit, en is ook het signaal SELECTED laag (origineel is selected altijd HOOG als de spanning opstaat !! Dit is principefout nr. 2). Door dat laag worden zal de software het deel Hardcopy van OUTCH patch overslaan (nogmaals drukken schakelt terug aan).
Indien een code 13 Hex of 19 Dec binnenkomt, dan gaat het lampje ON-LINE knipperen. Je weet dan dat alles wat je naar de printer stuurt, geneigeerd wordt, behalve code 11 Hex of 17 Dec waarmee je terug binnenhaalt. Het lampje gaat terug continue oplichten.
De schakeling nabouwen gaat probleemloos. Om ze in te bouwen heb je wel minstens het schema nodig. De diodes overbruggen kun je zonder schema.
Hierbij de schakeling om software ON/OFF te detecteren en te signaleren.
Tevens een schets over de diode-overbrugging.



PRINCIPE-schema – zie keerzijde



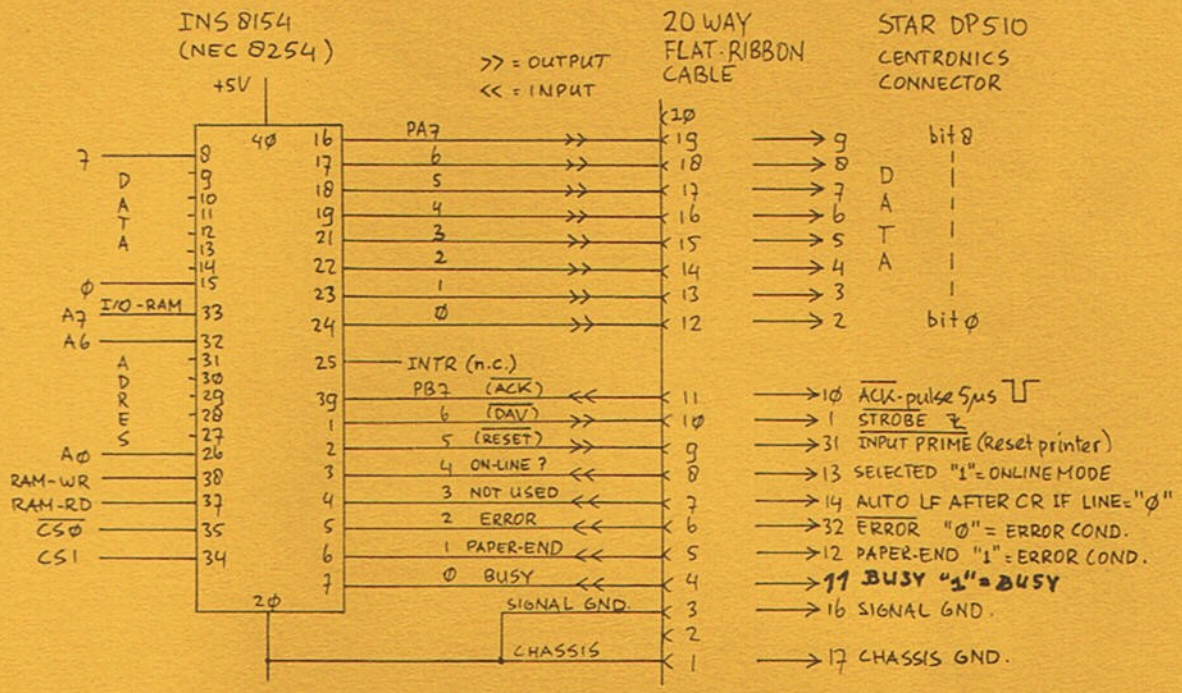
A ≡ $\overline{7.6.5}.4.\overline{3.2.1}.0$ of Hex 11 = start receiving (PRINTCHR$(17))

B ≡ $\overline{7.6.5}.4.\overline{3.2.1}.0$ of HEX 13 = stop receiving (PRINTCHR$(19))

=knipperen van LED "ON-LINE".

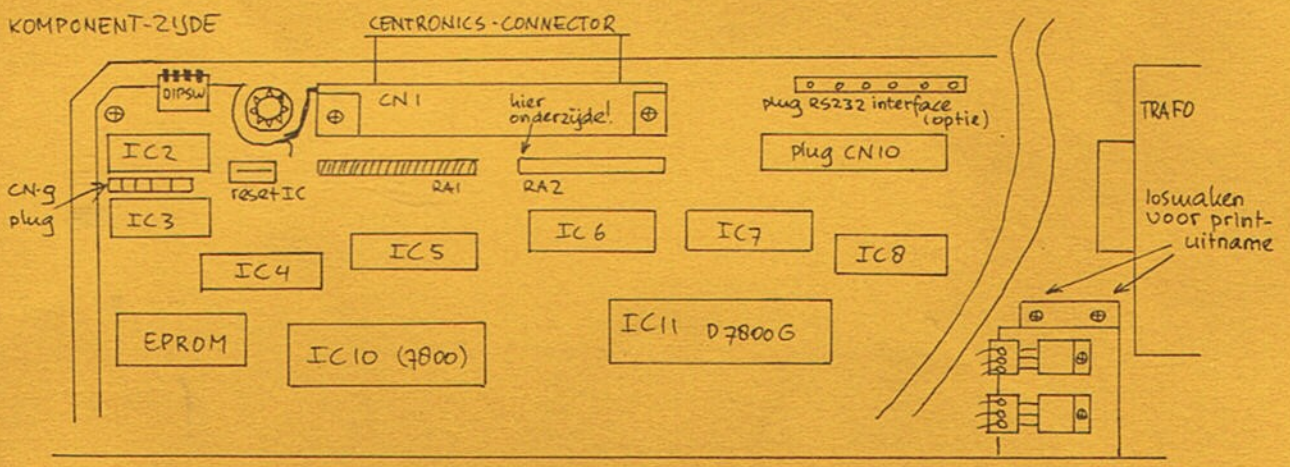
lijn (E) van IC3 pin9 naar plug CN9 onderbreken
naast plug CN9 pin5
aan komponentzijde alleen bereikbaar

Aan de tekentafel:
Fridus Jonkman



8154 Adressed at ØFØØ --- ØF24 for I/O
and ØF8Ø --- ØFFF for RAM
Mode word for MODE-3 (STROBED-OUT) = 6Ø
(Valid = 6Ø--7F)

STAR DP510



ONDER-PRINT