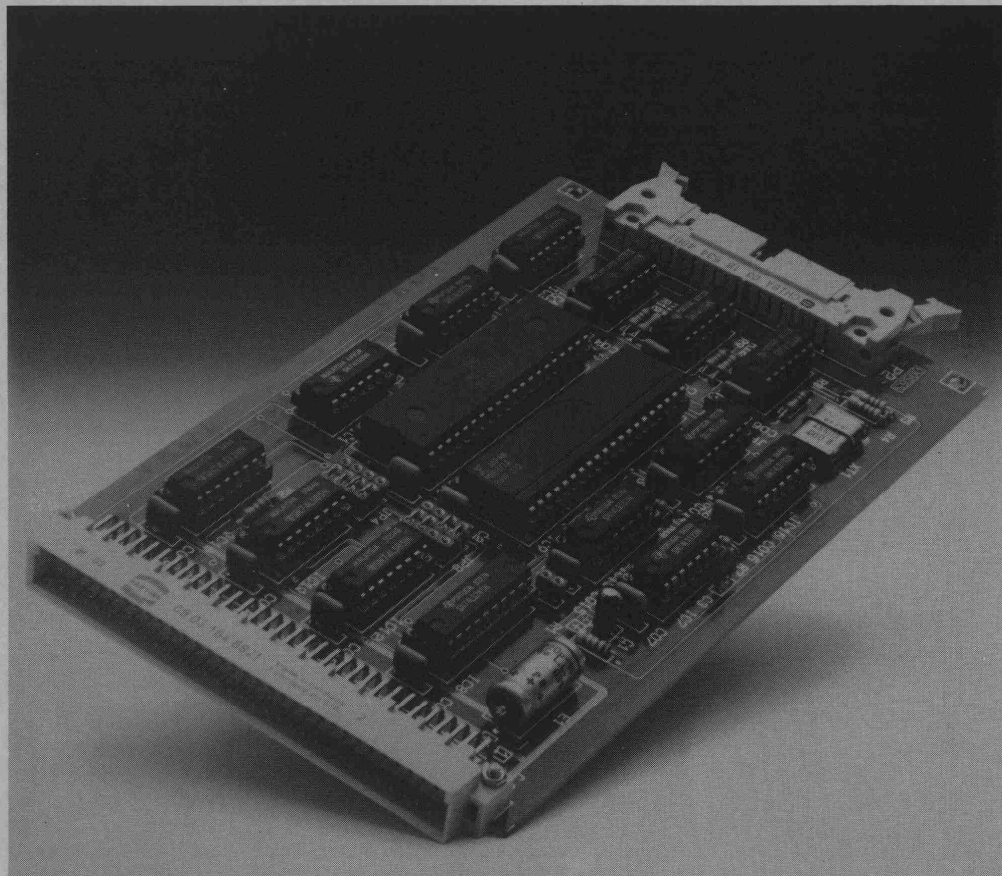




DE 6502 KENNER

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

De vereniging heeft leden in Nederland, België, Duitsland, Frankrijk, Engeland, Denemarken, Zweden, Spanje, Portugal, Oostenrijk, Finland, Israël, Amerika.

Het doel van de vereniging is: het bevorderen van de kennisuitwisseling tussen gebruikers van 6XXXX-computers, als COMMODORE-64, AMIGA, APPLE II/IIc/IIx/IIgs/III, MACINTOSH, ATARI 600/800XL/512/1024ST, CHE-1, PEARCOM, AIM-65, SYM, PET, BBC, VIC-20, BASIS 108, PROTON-computers, ITT-2020, OSI, ACC 8000, ACDORN ELECTRON, SYSTEM 65, PC-100, PALLAS, MINTA, FORMOSA, ORIC-1, STARLIGHT, CV-777, ESTATE III, SBC65/68, KIM, NCS, KEMPAC SYSTEM-4, Elektuur-computers (JUNIOR, en de OCTOPUS), LASER, dus ook 6800, 6809 en 68000-computers.

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

Verschijningsdata DE 6502 KENNER 1985

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

Bijeenkomsten van de club

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

Redactie-adres en informa- tie over paperware etc.

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

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

Voorzitter:
Rinus Vleesch Dubois
Fl. Nijtingalestraat 212
2037 AG Haarlem
Tel.: 023 - 330993

Sekretaris:
Bert Klein
Dierenweg 119
6706 CM Wageningen
Tel.: 08370 - 23646

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

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

Leden:

Adri Hanneke (05490 - 51151) Hardware/software/DOS65
Erwin Visschedijk (05490 - 71416) Hardware/software/DOS65
Bert van Oporok (01729 - 8636) 65802/65816/68000
Nico de Vries (010 - 4502239) Hard-/software/68000
Ervvoorzitter: Sijp de Vries
Ereleden: Mw. M. de Vries - Van der Winden

Lidmaatschap:
Anton Mueller
voor 1986: -fl. 45,-
voor 1987: -fl. 50,-
Buiten Europa:
voor 1986: -fl. 109,50 (incl.transfers)
voor 1987: -fl. 114,50 (incl.transfers)
Advertenties: Tarieven op aanvraag bij de redactie.

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

DE 6502 KENNER is een uitgave van de KIM Gebruikers Club Nederland. Het blad wordt verstrekt aan leden van de club. DE 6502 KENNER wordt van kopij voorzien door leden van de club, bij de opmaak van een publikatie bijgestaan door de redactie. De inzendingen van programma's dienen voorzien te zijn van commentaar in de listings en zo mogelijk door een inleiding voorafgegaan. Publikatie van een inzending betekent niet dat de redactie of het bestuur enige aansprakelijkheid aanvaardt voor de toepassing ervan. De inzendingen kunnen geschieden in assembly-source-listings, in Basic, in Basicode, Forth, Focal, Comal, Pascal, Fortran, Cobol, Logo Flan, etc. etc.

De leden schrijven ook artikelen over de door hen ontwikkelde hardware en/of aanpassingen daarop. Zij schrijven tevens artikelen van algemene aard of reageren op publikaties van andere inzenders.

DE 6502 KENNER IS EEN BLAD VAN EN DOOR DE LEDEN

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

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

DOS65 V2.01 is the new system of our club, build with Elektor's CPU, VDU, ROM-cards and our own professional Floppy-Disk-Controllercard for SS, DS, 40 or 80 tracks and a max. of 720 Kbytes capacity. For use with 6502 or 65022. For more information, write to E.J.M. Visschedijk
Dillenlaan 11
NL-7641 CX WIERDEN

The new DOS65 V2.01 is hardware compatible with Elektor's OCTOPUS/ED65 computer, except the controllercard.

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

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

Address all editorial, advertising and subscription inquiries DE 6502 KENNER: c/o Willem L. van Pelt Jacob Jordensstraat 15 2923 CK Krimpenerwaard The Netherlands

Editorial Staff:

Willem L. van Pelt
Gerard van Roekel
Frans Smeets
Coen Boltjes

Freelancers:

Rob Bamen
Fred Behringer, (Germany)
Fridus Jonkman
Gert Klein
Roger Langeveld
Marc Lachaert, (Belgium)
Fernando Lopes, (Portugal)
Frank Manshande
Gert van Opbroek
Leif Rasmussen, (Sweden)
Ruud Uphoff
Frans Verberkt
Herman Zondag

Translations:

Fred Behringer (Germany)
Willem van Asperen
Frank Bens
Albert v.d. Beukel
Rene Hettfleisch
Jaap de Hoop
Coen Kleipool (France)
Maarten van Lieshout

Nothing may be reprinted in part or in whole without permission from the publisher. Practising of the published programs and hardware etc. without responsibility of the publisher and for personal purpose only. DE 6502 KENNER appears in Febr, Apr, May, July, Sept, Oct, and Dec.

Copyright (C) 1986 KIM Gebruikers Club Nederland

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

The redaktion is not responsible for the contents of the articles, programs in the issues. The articles to be published have to be written by the sender.

CONTENTS OF DE 6502 KENNER NO. 47, DECEMBER 1986 VOL. 10/NO. 6

1. Uitnodiging Ledenvergadering/Landelijke Bijeenkomst 17 Januari 1987	2.
2. Van de redactie	3.
3. OCTOPUS	
Coen Boltjes	4.
... HARDWARE ADJUSTMENT FOR JUNIOR/OCTOPUS WITH VDU-CARD FOR USE AS A VIDEOTEX-TERMINAL (PRESTEL-STANDARD)	6.
... HOW TO ADJUST THE CHARACTER GENERATOR FOR VIDEOTEX	18.
... TIPS AND TRICKS FOR THE ED65K/JUNIOR	45.
... HARDCOPY ROUTINE FOR OCTOPUS	44.
... COLUMNS PRINT: for printing in two columns	28.
... "BELL" for ED65	33.
... SETTING TALLY-PRINTER	
4. COMMODORE	
Fred Behringer, Germany	9.
Gerard van Roekel	10.
... UPGRADE YOUR C-16 TO 512 KBYTE FROM 2364 TO 2764 EPROM	10.
... WARANTE PROBLEEMEN BIJ COMMODORE	
5. ACORN ELECTRON	
Sjaan Voortman	7.
6. ATARI	
Henk Speksnijder	11.
7. DOS65	
Peter Lasker	27.
Piet de Visser	29.
8. APPLE	
Frans Verberkt	13.
Hans Bosch	41.
9. ACORN ATOM	
Frank Vergoossen	14.
10. JUNIOR	
Gerrie van Woerkom	8.
... BREAK-KEY ON JUNIOR WITH SERIAL KEYBOARD	32.
... EXTENSION OF OS-650 V3.3 WITH DEL COMMAND	12.
Frank Bens	20.
Fernando Lopes, Portugal	38.
Ronald Hemaens	39.
Marcel Breukink	
11. FORTH	
A.G. Megens	42.
12. VARIOUS	
Andrew Gregory, England	16.
... A SIMPLE EPROM DISC FOR 6XXX COMPUTERS	38.
... SCREEN FLICKERING	19.
Erik v.d. Broek	12.
Marc Lachaert, Belgium	12.
Gert Klein	46.
Will Cuijpers	49.
Gert van Opbroek	10.
... NEW ADDRESS BRUTECH ELEKTRONICS	10.
... PHILIPS NEWSLINE	17.
... APPLE COMP. INC. VERDUBBELT WINST IN FISCAL JAAR 1986	38.
... STEVE CRACIA (America) TO JEFF CHATFIELD (Australia)	10.
... PRINTING ERRORS BASICODE INTERFACE ELEKTOR SPECIAL 2 PAGE 62	48.
... BLOCKDIAGRAM OF THE APPLE IIIGS	38.
... BRIEVEN AAN DE REDACTIE	13.
BOEKINFORMATIE: BOUW ZELF EEN EXPERTSYSTEEM	

Print your articles, programs etc. with a new ribbon and use 8 lines/inch by max. 73 lines/page.

Write your articles, programs etc. in English unless you need help to do it.

Send the names and addresses of 6xxx-users you know in your own country or you found in magazines to the editorial office. We will send information to these computerists and try to make them joining our club.

Send the names and addresses of the computer magazines in your country to the editorial office. We will send the magazines informations about our club to give attention to it and to get members in your own area.

PLEASE PAY YOUR 1987 SUBSCRIPTION (Hfl. 59,50)
HAVE A NICE YEAR OF HOBBYCOMPUTING IN 1987.

UITNODIGING LANDELIJKE BIJeenKOMST ASSENDELFT/KROMMENIE

Datum : zaterdag 17 januari 1987 ENTREE: Hfl. 10,=
Lokatie : nieuwe kantine FORBO-Krommenie
Industrieweg 12 te Assendelft. Tel: 075 - 291911

Auto : komende uit de richting Amsterdam

Coentunnel helemaal afrijden. Aan het eind rechtsaf (water aan linkerzijde). Dan 1e afslag rechts richting Uitgeest-Alkmaar. Doorrijden tot stoplichten. Linksaf spoorbaan over. Na 75 meter linksaf = Industrieweg. Links aanhoudende koert men op het FORBO-Krommenie terrein.

komende uit de richting Alkmaar

Snelweg Alkmaar-Haarlem. Afslag Uitgeest/Zaandam. Bij kruising linksaf. Bij 3e stoplichten rechtsaf spoorbaan over. Na 75 meter linksaf = Industrieweg. Links aanhoudende koert men op het FORBO-Krommenie terrein.

Trein: Station Krommenie-Assendelft. Rechtsaf tot over spoorbomen. Na 75 meter linksaf = Industrieweg. Links aanhoudende koert men op het FORBO-Krommenie terrein.

PROGRAMMA:

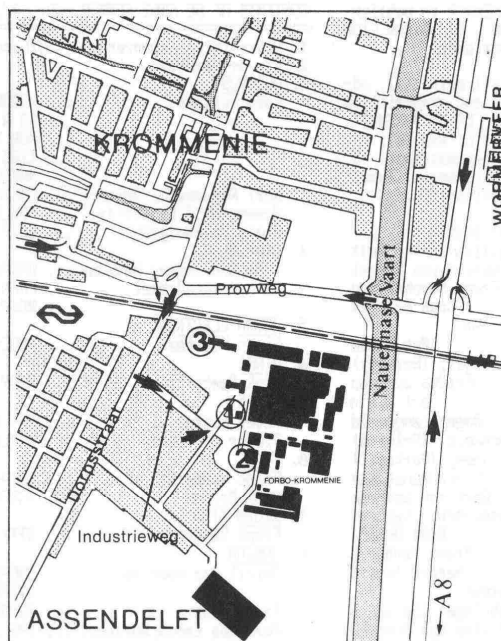
- 09.30 - Zaal open. Ontvangst met koffie.
- 10.00 - Opening door voorzitter en verwelkoming door gastheer Co Filmer. Diashow FORBO-Krommenie.
- 10.30 - Behandeling Jaarverslag 1986.
- 10.45 - Nico de Vries behandelt het onderwerp opvoeren van snelheid van 1 naar 2 Mhz.
- 11.30 - Koffiepauze
- 12.00 - Forum
Aansluitend lunchpauze
- 13.00 - MARKT. Ieder die iets aan te bieden heeft kondigt dit aan.
- 13.15 - INFORMEEL GEDEELTE.
In dit deel bestaat er volop gelegenheid kennis te maken met de meegebrachte systemen en de gebruikers ervan.
BRING OOK UW SYSTEEM MEE !!!
VERGEET DAARBIJ NIET EEN AANSLUITSDOER EN VERLENGKABELS.
- 17.00 - Sluiting.

Technische specificaties Apple IIGs.

- 16-bits microprocessor 65C816
- 256 Kilobyte werkgeheugen (RAM), uitbreidbaar tot 4 Megabyte
- 128 Kilobyte systeemgeheugen (ROM)
- ingebouwde RGB-poort voor 12 inch kleurenbeeldscherm
- ingebouwde poort voor 12 inch monochrome beeldscherm
- vijf grafische modi met maximale resolutie van 640x200 beeldpunten en kleurenpalet van 4096 kleuren
- 8 uitbreidings sleuven geschikt voor de standaard Apple II uitbreidingskit
- poort voor 3.5 en 5.25 inch schijf eenheden

- 2 RS422 seriële poorten voor modem, printer of AppleTalk-netwerk
- sound kit van 32 oscillatoren/16 gescheiden kanalen
- audio-aansluiting voor stereo-koptelefoon

- poort voor joystick en handregelaars
- toetsenbord met 89 toetsen en numeriek toetsenblok en cursortoetsen
- aansluitingsmogelijkheden voor de muis aan het toetsenbord
- muis



- | | | |
|---|--|---|
| ① Portier
Portier
Gateman
Portero | ② Ontvangstcentrum
Salle de réception
Empfangsraum
Reception building
Sala de recepcion | ③ Kantoor
Bureaux
Büros
Offices
Oficinas |
| ④ Centraal magazijn
Magasin central
Zentrallager
Central warehouse
Almacén central | | |

Treinverb. Amsterdam-Alkmaar (half uurdienst)
 Chemin de fer Amsterdam-Alkmaar (toutes les demi-heures)
 Eisenbahn Amsterdam-Alkmaar (jede halbe Stunde)
 Railway Amsterdam-Alkmaar (half hour service)
 Linea ferroviaria Amsterdam-Alkmaar (cada media hora)

FORBO-KROMMENIE BV
INDUSTRIEWEG 12
1566 JP ASSENDELFT - HOLLAND
075-28 06 00

Redaktioneel.

Het jaar 1986 is weer bijna voorbij. Nog wat feesten, nog wat drinken, en dan begroeten we 1987 met goede voornemens. Uw redakteur is daarin even loyaal als U. Hij neemt zich voor de edities van DE 6502 KENNER nog aantrekkelijker te maken. Het bestuur zal hem daarbij nog een handje helpen, door een daisywheelpriester aan te schaffen. Dat betekent nog niet dat alle pagina's daarmee bedrukt kunnen worden. Leden trachten mijn tijd te sparen door in samenspraak met mij zelf hun pagina's op te maken. Tijd, die hard nodig is voor vele andere nog te ontwikkelen of verder uit te bouwen ideeën. Doordat leden proberen hun eigen pagina's op te maken, doordat - vanwege de kans op fouten - niet alles overgetypt kan, zullen altijd pagina's aanwezig blijven die een ander leesbeeld vertonen, enigszins variërend in kwaliteit.

Uw redakteur neemt zich ook voor, net als in 1986, de informatiedichtheid in de edities nog verder tot onderwerp van zijn zorg te maken, zodat er voor hetzelfde lidmaatschapsgeld (in 1987: fl. 50,- voor Europese C.E.P.T.-landen) weer meer kennis wordt aangeboden. Om dat te verwezenlijken moet Uw redakteur meer dan nu al het geval is een beroep op U doen U in te zetten voor het verwerven van die kennis. Niet alleen, zoals nu al door een flink aantal leden gebeurt, mijn en Uw applaus verdienend, door programma's ter publikatie aan te bieden, maar door U op te geven als kontaktpersoon voor Uw gemeente, door Uw modem in te zetten voor het plaatsen van redaktionele berichten in databanken in Uw eigen telefoondistrikt, door namen en adressen door te geven van mensen met een 6xxxx-computer die U kunt vinden in Uw eigen omgeving, de lezersrubrieken van computertijdschriften, door mensen bij U uit te nodigen en Uw ervaringen met onze club door te geven en ons blad te laten zien.

Uw redakteur neemt zich voor, gezien zijn verwachtingen dat het moeilijker zal worden uit de Nederlandse dreven nieuwe leden aan te trekken, door te gaan met het aantrekken van nieuwe leden in andere landen. In dit verband moet het mij van het hart dat verzuchtingen over het Nederlandse karakter van onze club weliswaar begrijpelijk zijn gezien de geschiedenis van de club, maar dat ook van oudsher voor niemand een lidmaatschap kan worden geweigerd, dus ook niet voor in het buitenland wonenden, op grond van onze statuten, tenzij het gaat om niet natuurlijke personen.

Het is geen eenvoudige zaak de konsekwenties van deze openheid te aanvaarden, d.i. het gebruik van de Engelse taal in de edities van DE 6502 KENNER te handhaven. Soms kan dat zelfs heel omvangrijk zijn doordat de buitenlandse leden intensief gebruik maken van deze gelegenheid. De enquête, door het bestuur gehouden, wijst uit dat tegen het gebruik van de Engelse taal geen overwegende bezwaren zijn in te brengen. Het dient ook ons doel van elders kennis te verwerven. Ik ben mij ervan bewust dat sommigen moeite hebben met het Engels. Daar waar dat tot echte problemen voor hen leidt, laten zij zich met mij in verbinding stellen, dan zoeken we naar wegen om moeilijk leesbare artikelen leesbaar te maken. We zijn een club van mensen die elkaar willen helpen. De enkeling die op dit punt problemen heeft verdient van de leden hulp.

Als ik op die manier van dienst kan zijn wordt 1987 voor ons allemaal een fijn hobbyjaar.

Hardware adjustments for Junior/Octopus with VDU-card for use as a VIDEOTEX-Terminal. (PRESTEL-Standard)

by: Coen Boltjes tel: 015-136812
Nw. Plantage 9 vidibus: 400029830
2611 XH Delft
The Netherlands

Translated by: Elja van der Veer

Interactive VIDEOTEX is the international name for what is called VIDITEL by the Dutch PTT. In using the Junior/Octopus as a videotex terminal a number of hardware adjustments are necessary additional to a programme. In a next issue the software will be described. Below the hardware adjustments follow.

In using the computer as Videotex-terminal it is necessary that reverse- and mosaic-characters can be presented. This can be realised in at least three ways:

- 1) Construct the new VDU-card of Pieter Visser (see issue nr. 44) This is the most flexible, though the most expensive, solution.
- 2) Reprogram the charactergenerator. This is an inexpensive solution, but also quite a job and an EPROM-Programmer must be available
- 3) Carry out the adjustments as described below. Although this takes some time, it is cheap.

The graphic characters from the standard VDU-card are replaced by the reverse characters, which uses the codes \$80-\$FF. For this purpose the connection between pen 9 IC18 and pen 4 IC19 must be cut. The latter must be placed at zero, the first one must be connected to pen 12 of N33. The wire-bridge S/T must be removed, and will fall into disuse.

Now the adjustments for reverse characters are completed, and can be tested. (Reverse-codes in the VDU-RAM)

The mosaic characters are to be generated by the codes \$00-\$1F, because these codes are reserved for controlling characters in ASCII, so they are not used in the VDU-RAM. Each bit of the code that has been set corresponds with a fluorescent block (see fig 1). The block on the right below can be put on by using reverse-characters. If the mosaic code is offered to the character generator, the ROM is disabled, and the hardware of fig 2 is selected. This will then take care of transferring the building information.

Personally I have brared the ROM in a wire-wrap foot on to a small piece of print, on which the rest of the circuit was build as well. As the ROM-foot has long pins it can be placed completely in the foot of the VDU-card. (Be sure you cut pin20 from the VDU-card to the extension print) The numbers between the signalname and the pen number of an IC is the corresponding pen on the ROM.

note: It is possible that there are little lines visible between normal- and reverse characters. This is caused by the differend Propagation-Delay times of IC17 and IC18, which can result in a spike on the output of N34. Using an Edge-triggered Flip-Flop (74LS109, 74LS74, 74LS175) between the output of N31 and the input of N22 will synchronize the reverse- and building information. The DOT-frequentie is used as clock for the Flip-Flop.

For communication with a Videotex databank a modem is used (See Elektuur sept 1984). Communication between the (own) computer and the modem takes place by means of a serial RS232 interface.

However, as the sending and receiving speeds differ, it is not possible to use the interface on the print of the CPU-card. (Note: this does not hold for modems with a built-in re-speeder.) It is possible with the 6551 used on this print to send with a speed which differs from the receiving speed, but in that case we must offer a receive clock on the RxC line (5). Therefore you can use only part of the interface below. If you don't want to adjust the interface on the CPU-card, or when you use a Junior you have to build a new serial interface.

Therefore, an interface was constructed with the help of the 6850 which has separated RxC and TxC lines. By presented different frequencies it will be possible to work on different speeds.

The frequencies are deduced from IC1 which is transferred to the ACIA IC5 through the multiplexers IC2 and IC3. Establishing the multiplexers is done with the help of IC4. By writing 0-7 in the lower nibble and/or the higher nibble the frequencies can be selected.

By using the scalefactors 16 and 64 most of the current speeds can be moderately realised. (see table)

With the Junior the interface must be built twice because the keyboard must be read as well. This can not be done in the usual way, because as a result of the realtime character of the software the computer must be able to test whether a key was pressed or not.

Notice that if you don't want to change the speed of your keyboard you can directly connect the RxC line of the ACIA for the keyboard to one of the lines of IC1. For generating the /CS signals the article in Elektuur jan 1984 is referred to.

Also, one may use the circuits published earlier or get into contact with the author.

Take into account, after building the interface, that the ACIA's cannot be reset by a hardware reset. In the reset- or bootroutine this must be done by storing \$03 in the controlregisters.

Red.: The author also worked out the suggestion to solve the problem with software following this article.

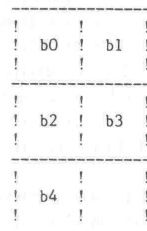


Fig. 1
Mosaic Defenition

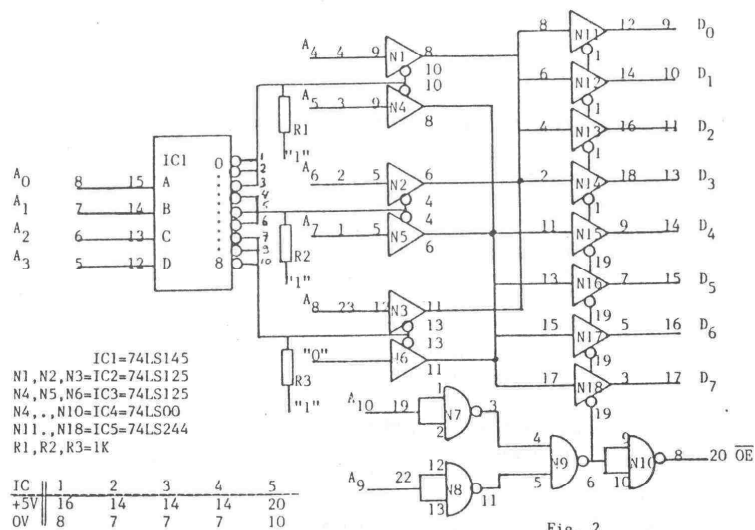
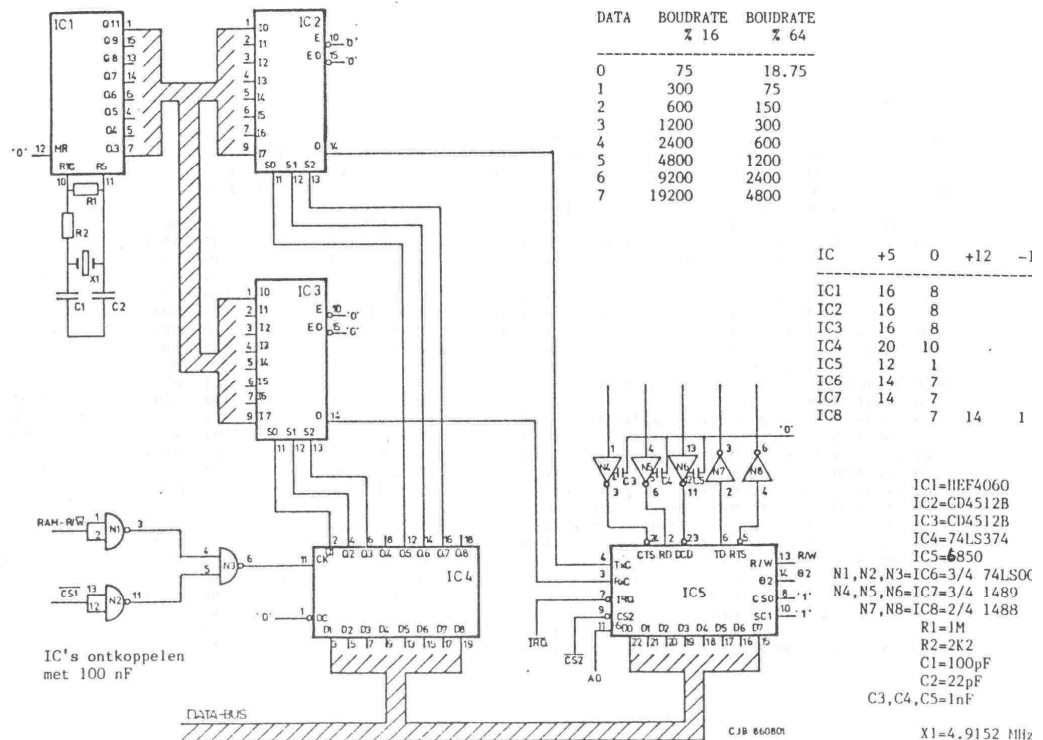


Fig. 2
Character-generator
Extensions

IC's ontkoppelen met 100 nF



How to adjust the character generator for VIDEOTEX

By: Coen Boltjes Telephone 015-136812
Vidibus 400029830

- 1) Load the first half of the old character generator in your system (Starting at \$8000)
- 2) Execute the MOVE program with an "ASL A" instruction instead of the "NOP" instruction. The characters are now shifted one dot to the left, and stored in \$9000-\$9800.
- 3) Move the characters one line down. (8000=8FFF, 9800) These shifts will cause better inverse characters.
- 4) Fill the locations \$8000-\$8200 with the contents of the hexdump, and save the first half of the new character generator on disk. (SA 13, 1=8000/8)
- 5) Execute the MOVE program with an "EOR #\$FF" instruction instead of the "NOP" instruction, and save the second half of the character generator on disk. (SA 14, 1=9000/8)
- 6) Program a new 4K EPROM with track 13 and 14.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
8000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
8010	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	
8020	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	
8030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
8040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	10 0040=
8050	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	SOURCE=\$40
8060	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	20 0042=
8070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	DEST = \$42
8080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	30 0044=
8090	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	COUNT = \$44
80A0	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	40 A000
80B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	*=\$A000
80C0	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	50 A000 A980
80D0	F0	F0	F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	MOVE
80E0	0F	0F	0F	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	LDA #\$80
80F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	STA SOURCE+1
8100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	70 A004 A990
8110	F0	F0	F0	00	00	00	00	00	00	00	00	00	00	00	00	00	LDA #\$90
8120	0F	0F	0F	00	00	00	00	00	00	00	00	00	00	00	00	00	STA DEST+1
8130	FF	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	00	;INIT TRANSFER POINTER
8140	00	00	00	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	90 A008 A9F0
8150	F0	F0	F0	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	LDA #\$F0
8160	0F	0F	0F	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	STA COUNT+1
8170	FF	FF	FF	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	100 A00A 8545
8180	00	00	00	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	LDY #\$00
8190	F0	F0	F0	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	110 A00C A000
81A0	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	LDY SOURCE
81B0	FF	FF	FF	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	120 A00E 8440
81C0	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	STY DEST
81D0	F0	F0	F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	130 A010 8442
81E0	0F	0F	0F	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	STY COUNT
81F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	140 A012 8444
:																	150 A014 B140
																	SAVSCT
																	LDA (SOURCE),Y
																	;GET CHARACTER
																	NOP
																	;PLACE HERE INSTRUCTION
																	STA (DEST),Y
																	;PALCE CHARACTER
																	170 A017 9142
																	INC SOURCE
																	180 A019 E640
																	BNE SAVSCU
																	190 A01B D002
																	INC SOURCE+1
																	200 A01D E641
																	SAVSCU
																	INC COUNT
																	210 A01F E644
																	BNE SAVSCV
																	220 A021 D004
																	INC COUNT+1
																	230 A023 E645
																	BEQ SAVSCW
																	;=>LAST CHARACTER
																	240 A025 F008
																	SAVSCV
																	INC DEST
																	250 A027 E642
																	BNE SAVSCT
																	260 A029 D0E9
																	INC DEST+1
																	270 A02B E643
																	BNE SAVSCT
																	;=>ALWAYS
																	280 A02D D0E5
																	SAVSCW
																	RTS
																	290 A02F 60

```

10REM *****
20REM *** PROCsearching ***
30REM *** (c) by S.Voortman ***
40REM *** Beatrixweg 28 ***
50REM *** 3253 BB OUDDORP ***
60REM *** The Netherlands ***
70REM *** Phone 01878-3113 ***
80REM *****
90REM *** For the Acorn Electron
100REM *** Program Length: 1911 bytes
110REM *** (p) 1986
120MODE4:VDU19,0,6;0;19,1,0;0::lnh=0:lnl=0
130PRINTTAB(3,3)"This program prints all DEFPROC's""and DEFFN's in BASIC-proc
rams.""Type in the page of the program which""DEFPROC's and DEFFN's you want
to know.""
140PRINT"The program's output is as follows:"
150PRINTSPC(4)"XXX DEF PROC/FNname""XXX stands for the line number of the""S
C4"DEF PROC/FN, and name for""SPC4"the name of the PROCedure."
160PRINT"At wich PAGE starts the program""(between %E00 and %6000)";
170INPUT page$:page=EVAL("&"+page$):IF page <%E00 OR page >%6000 THEN PRINT":
can't believe this is true!""You must have made a mistake!":GOTO 160
180IF ?page<>%0D THEN PRINT"Bad program at PAGE ":~page:END
190IF page?1=%FF AND ?page=%0D PROCend_of_program:END
200VDU12,2:PRINT"SPC3"PAGE &";~page
210lnh=page?1:lnl=page?2
220IF lnh=0 THEN linenummer=lnl ELSE linenummer=lnh*256+lnl
230linelenght=page?3
240IF page?4=%DD THEN def=TRUE ELSE def=FALSE
250IF def=TRUE AND page?5=%F2 THEN proc=TRUE ELSE proc=FALSE
260IF proc THEN 310
270IF def=TRUE AND page?5=%A4 THEN fn=TRUE ELSE fn=FALSE
280IF fn THEN 310
290IF def=TRUE AND proc=FALSE AND page?6=%F2 THEN page=page+1:proc=TRUE ELSE
roc=FALSE
300IF def=TRUE AND proc=FALSE AND page?6=%A4 THEN page=page+1:fn=TRUE ELSE f
FALSE
310IF def*proc=1 THEN PROCproc(linenummer)
320IF def*fn=1 THEN PROCfn(linenummer)
330page=page+linelenght:IF page?1=%FF THEN PROCend_of_program:VDU12:END
340GOTO 210
350
360DEFFPROCend_of_program:REM END
370PRINT"Program ends at line ";linenummer
380ENDPROC
390
400DEFFPROCproc(linenummer)
410proc$=(page+6):F%=0:L=LEN(proc$)
420REPEAT:F%=F%+1
430UNTIL MID$(proc$,F%,1)=":" OR F%=L+1
440proc$=LEFT$(proc$,F%-1)
450PRINT:linenummer;" DEFPROC";proc$
460ENDPROC
470
480DEFFPROCfn(linenummer)
490fn$=(page+6):F%=0:L=LEN(fn$)
500REPEAT:F%=F%+1
510UNTIL MID$(fn$,F%,1)="=" OR F%=L
520fn$=LEFT$(fn$,F%)
530PRINT:linenummer;" DEFFN";fn$
540ENDPROC

```



Description of PROCsearch:

Line: does:

```

120  MODE4: Set screen size (40*32 characters), VDU19...: Background
      colour magenta(6), Foreground colour black(0)
130  PRINTTAB(3,3)"text..." The ' means repeating the
      PRINT-statement, like PRINTTAB(3,3)"text...":PRINT"...."
180  IF ?page<>%OD ... Every BASIC-program starts with %OD, if it
      isn't present, it isn't a BASIC-program!
190  IF page?1=%FF... A '%FF' after '%OD' means the end of a
      BASIC-program
200  VDU12,2 This means: Clear screen(12) and printer on(2)
210  page?1 is linenumber MSB, page?2 is linenumber LSB
220  Calculate linenumber
230  page?3 is lenght of BASIC-line (inclusive '%OD' at the begin-
      ning and the '%OD' of the following line)
240  '%DD' is token for 'DEF'
250  '%F2' is token for 'PROC'
270  '%A4' is token for 'FN'
290-300 Search for a space between DEF and FN/PROC
310-340 Perform linenumber , PROC/FN name, set counters for next line
410-450 Read PROC name out of memory, check for ':' and PRINT
490-530 Same for FN's

```

A program line is stored (in memory) as follows:

```

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

1D00  0D 00 0A 0F F4 2D 2D 2D 54 45 53 54 2D 2D 2D 0D .....---TEST---.
1D10  00 14 05 F1 0D 00 1E 1C F1 22 47 65 74 61 6C 6C ..... "Getall
1D20  65 6E 20 65 6E 20 6B 77 61 64 72 61 74 65 6E 22 .. en en kwadraten"
1D30  0D 00 28 0E E3 20 5B 3D 31 20 B8 20 31 30 0D 00 .. (.. X=1 .. 10..
1D40  32 0C F1 3B 5B 2C 3B 5B 5E 32 0D 00 3C 05 ED 0D 2..;X,;X^2..<...
1D50  FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

 * BREAK-KEY ON JUNIOR WITH SERIAL KEYBOARD *

 By: Gerrit van Woerkom, The Netherlands

Referring to the article of A.C. Tijnas in issue 37 I hereby give my solution of a well functioning BREAK-key. The original situation has two disadvantages:

1. Every key on the keyboard acts as BREAK-key
2. A running BASIC program can only be interrupted during screen output

The following solution is made for the JUNIOR with OS-65D DOS, but can simply be adapted for the cassette JUNIOR. It is based on the assumption that a separate BREAK-key as described in Elektuur, May 1983, is present.

```

10          ;stop-key JUNIOR OS-65D V3.3 with
11          ;VDU-card
20          ;
30 FE81=    ; DELBIT=%FE81
40 FA80=    ; SPAD  =%FA80
50 2325=    ; KPDD  =%2325
60          ;
70 F01A     ; *=%F01A
80 F01A 4C9DF5 BRTST JMP BREAK
90          ;

```

```

100 F594
110 F594 2C80FA
120 F597 1004

130 F599 AD2523

140 F59C 60
150 F59D A20A
160 F59F 2081FE
170 F5A2 2C80FA

180 F5A5 30F2

190 F5A7 CA

200 F5A8 D0F5
210 F5AA 2C80FA

220 F5AD 10FB
230 F5AF 6C7CFA
240
250 0819
260 0819 2094F5

CHECK BIT SPAD ;key depressed?
      BPL BREAK ;if yes then
      ;test BREAK
      ;OS-65D needs
      ;this
RETURN LDA KPDD
      RTS
BREAK LDX #10 ;bitcounter=10
NEXT JSR DELBIT ;wait 1 bittime
      BIT SPAD ;serial output
      ;still high?
      BMI RETURN ;if no, then
      ;return
      DEX ;decrement bit-
      ;counter
      BNE NEXT ;10 bittimes?
      BIT SPAD ;wait until
      ;BREAK-key
      ;released
      BPL WAIT
      JMP (%FATC)

      *=%0819
CHKSTP JSR CHECK ;check BREAK-key

```

 ** UPGRADE YOUR C-16 TO 512 KByte **

By : Fred Behringer, München/Germany

Actually, it's but 484 K : whenever an address occurs within the first 4 K, the circuit switches to block no. 0. There are 8 blocks of 64 K each and the switching is done by POKE232,0 ... 7. 232 (i.e. \$EB) is the last (zero page) location of a block of addresses reserved for user software. Thus, by loading each 64 K block in turn with BASIC programs and putting the end-of-program vector \$2D/\$2E sufficiently high, the upgraded machine offers an immediately usable variety of 484 K (!) of BASIC programs which can be called upon by one of the pokes mentioned above.

I don't know yet how reliable my circuit is. The only thing I know is the fact that it works. And I'm happy about it since I had no oscilloscope at my disposal, and I needed some experimentation to find out which type of ICs worked best. This is why I used the ALS type for some of them, LS for others, and HCT for the 137.

And this is how the upgrade works:

- (1) The whole thing is based upon my 64 K upgrade which I reported on recently (replacement of the two 4416s by two 4464s).
- (2) Whenever the current address is less than \$1000, the output of one of the ALS260s goes high and switches the multiplexer LS257 such that the CAS signal is then applied to block no. 0.
- (3) If the current address is \$1000 or more, the CAS signal triggers the GE enable input of IC HCT137. The

(somewhat delayed) CAS pulse then triggers that DRAM block which is determined by the particular output of the HCT137 which has been decoded (see (4)). In order to wait for the current address to be stable before switching the LS257, the output of the ALS260 mentioned is first applied to the data input of an LS175 (D-type flip-flop). Only when the RAS line goes low, this information is transferred to the select input of the LS257 via the Q output of the LS175 (where it is held until the next going low of RAS). Actually, what was required was a high-going pulse at the clock input of the LS175. So I needed an extra LS00 inversion.

- (4) Whenever \$EB is called upon by a write operation (such as POKE, driving R/W low), the output of IC ALS 133 (working together with the two ALS260s) goes low. As a result the GL input of the HCT137 is driven low. The outputs of the HCT137 are now floating, depending on the data inputs currently applied. The input data (0 ... 7) will be properly decoded as soon as R/W goes high (driving GL high).

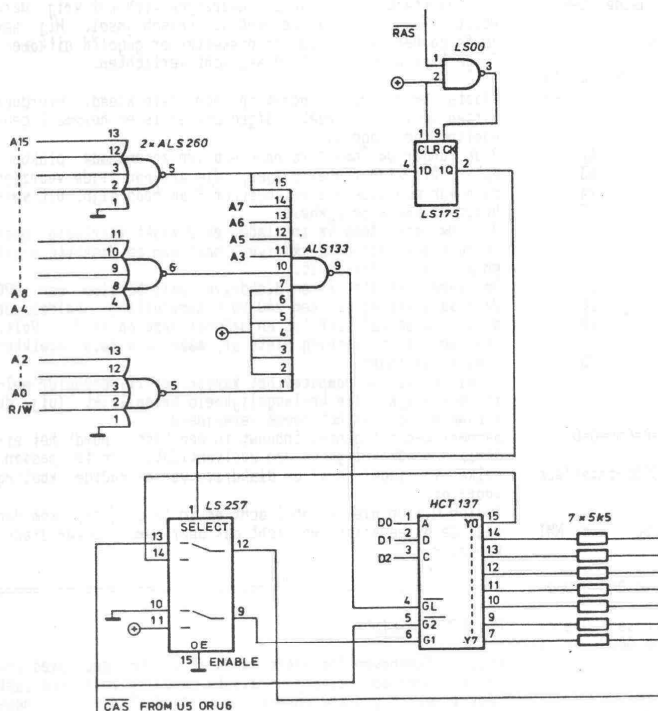
(R/W goes high prior to the data becoming invalid.)

In order to get hold of address lines A0 - A15, I glued one ALS260 on top of U7 (LS257), the ALS133 on top of the second ALS260.

(The dynamic RAMs' sockets would supply the multiplexed address lines only).

The rest of the circuit was mounted on a piece of veroboard and connected to the sockets where the two 4464s of my 64 K upgrade, reported on recently, were plugged in (U5 and U6).

The resistors at the outputs of the HCT137 were the results of some experimentation.



A series of tests has shown that the LS type ICs are too slow to guarantee no errors in bankswitching. After replacing LS175, LS257, LS00, and HCT137 by ALS175, ALS257, ALS00, and ALS137 there were no more errors to be observed. The resistors 5.6K should be removed.

TO CAS PINS OF DRAMS 4464
 EACH LINE TO 2x4464 (HIGH,LOW)

LAST LINE REPRESENTS BLOCK 0,
 WHICH IS SWITCHED IN WHENEVER
 ADDRESS IS BELOW \$1000

HERMAN ZONDAG

TRANSI : HAKON AUSTRO

DE 6502 KENNER

 * ATARI 600 XL AND MACHINE CODE PROGRAMMING *

 Henk Speksnijder, The Netherlands

The manual of the ATARI 600 XL contains almost useless information about machine code programming, but because I've been working on Elektor's JUNIOR computer, I was able to find out how to do. Try this:

```
10 FOR J=0 TO 9
20 READ D
30 POKE 1536+J,D
40 NEXT J
50 LET X=USR(1536)
60 ? X
100 DATA 169,0,133,212,169,1,133,213,104,96
```

When you run this program, there will be 256 printed on the screen. What happens is this. In line 100 is the machine code, line 10 upto 40 reads the machine code and stores it in RAM at address from 1536. Line 50 makes the machine code working. This machine code program is very small and only to show something:

LDAIM	0	169	0
STAZ	212	133	212
LDAIM	1	169	1
STAZ	213	133	213
PLA		104	
RTS		96	

The program in line 100 is in decimal and not hexadecimal because now line 20 and 30 are simple. This program stores a zero in address 212 and a one in 213, then it pulls one byte from stack and returns to BASIC. Then your BASIC makes this calculation:

$X = \text{PEEK}(212) + 256 * \text{PEEK}(213)$

That explains why 256 is printed in line 60.

The PLA instruction is always in this kind of machine code programs just like RTS. The RTS is clear because the program execution must return to BASIC. The PLA is there, because the ATARI BASIC puts one byte at the top of the stack, this byte contains the number of parameters. The use of parameters with the USR function is very interesting to do great things with machine code. Make line 50 like this: 50 LET X=USR(1536,555) and line 100 like this:

```
100 DATA 104,104,133,213,104,133,212,96
and line 10 FOR J=0 TO 7
```

The rest of the program remains the same. Now after RUN you will see printed on your screen 555. That is the same number as in line 50. This number can be any number from zero to 65535, but it is also possible to use variables. What happens in line 50 is: at first the BASIC puts three bytes on stack. The first has the value 1 because there is one parameter. The second and third bytes are the high order and low order of the parameter. The machine code in line 100 is now:

PLA	104		pull number of parameters
PLA	104		pull high order byte
STAZ	213	133 213	store this in 213
PLA	104		pull low order byte
STAZ	212	133 212	store this in 212
RTS	96		

Until now I always used in line 20 and 50 the number 1536. That is the beginaddress for me to work with machine code. The biggest program so far was 140 bytes. If you want to make bigger programs, then it is possible to do so, but the numbers in your program depend on the amount of RAM you have and on the size of the machine code program:

```
10 POKE 106,62:GRAPHICS 0
20 FOR J=0 TO 512
30 READ D
40 POKE 62*256+J,D
50 NEXT J
```

In line 10: If there is 16 K RAM then 64-p, where p is the number of pages (with 256 bytes) you need. In this example 64-2=62.

In line 20: The number of bytes of the program, in this example 512.

In line 40: The same number as used in line 10.

SOME QUESTIONS TO THE EXPERTS:

By: Henk Speksnijder, The Netherlands.

The ATARI BASIC programs usually start at 2060 or in this area. From 0 to 255 is page zero, and I only know the meaning of 35 page zero addresses. From 256 to 511 is the stack. From 512 to 2000 there are some interesting tables, but why is there so much filled with zero? From 1244 to 1400, 1447 to 1792 and 1820 to 2040 is only zero! Is that 700 bytes RAM wasted? Who knows about page zero? Send your letter to the editor. He takes note of it and will inform me.

AVAILABLE FOR YOUR OCTOPUS/EC65 COMPUTER :
 40 TR, SS, UNBOOTABLE DISKETTE ML No. 1

COPIER VERSION 2.3 + 4 BASIC PROGRAMS

Dr. Tietsch (Germany) published in DE 6502 KENNER no 43 and 44, April/June 1986, his Diskette Copier version 2.2, a modification for serial systems for double-sided disk-drives.

Marc Lachaert (Belgium) published in DE 6502 KENNER no 45 his error patches on Dr. Tietsch's Copier routine.

This diskette makes copying of your diskettes a lot easier.

EXTRA: Four BASIC games (Dutch language):
 GALGJE, KONG, SNAKEY, YATZIE

WARNING

To let the programs run properly the OHIO-DOS has to be changed. Boot first with the ML BASICCODE disk.

If you ordered the OS65-D patch diskettes earlier at the editorial office:
 Send empty diskette with label and R/W protect label to the editor's office. Send cheque of Hfl. 22,00 to W.L. van Pelt (Eurocheque = Hfl. 12,50).
 Price only for European (C.E.P.T.) countries.

viditel
 ptt telecomunicatie

Viditel - Den Haag

uw toegangsnummer a.u.b.

DIRECTORY DISK 5.a
for Elektor's OCTOPUS computers

By : Marc Lachaert, Belgium

TRACK	SEC	PAG	MEMORY	NAME	DESCRIPTION
00 -	1 -	8	2200,29FF	V3.3/1	OS-65D V3.3 part 1
01 -	1 -	8	2A00,31FF	V3.3/2	OS-65D V3.3 part 2
02 -	1 -	8	0200,09FF	BAS/1	BASIC part 1
03 -	1 -	8	0A00,11FF	BAS/2	BASIC part 2
04 -	1 -	8	1200,19FF	BAS/3	BASIC part 3
05 -	1 -	8	1A00,21FF	BAS/4	BASIC part 4
06 -	1 -	1	2200,22FF	B/5V/3	BASIC part 5
	2 -	1	3200,32FF		OS-65D part 3
	3 -	1	0000,00FF		OS-65D part 4
07 -	1 -	8	0200,09FF	ASM/1	OSI Assembler/editor/ext. monitor
					part 1
					part 2
08 -	1 -	8	0A00,11FF	ASM/2	part 3
09 -	1 -	8	1200,19FF	ASM/3	part 4
10 -	1 -	8	1A00,21FF	ASM/4	part 5
11 -	1 -	1	2200,22FF	ASM/D1	'directory' copy, contains only 'DIRECT' and 'TRKNUL'
	2 -	1	2E79,2F78		Empty 'Directory' copy
	3 -	1	2E79,2F78		} ML-progs in use for disk operat.
	4 -	1	5E00,5EFF		} util in BEXEC*
	5 -	1	5F00,5FFF		'Directory' copy with users file
	6 -	1	2E79,2F78		BASIC overlay
	7 -	1	25A0,269F		Directory part 1
12 -	1 -	1	2E79,2F78		Directory part 2
	2 -	1	2E79,2F78		BASIC overlay
	3 -	1	2C04,2D03		PUT/GET overlay
	4 -	1	2E79,2F78		OS-65D part 5
13 -	1	8	3274,3A73	V3.3/4	

DOS65 Basic version 2.10 available now.

For DOS65 users a fine Basic interpreter is available now. It has been developed from the well known Microsoft KB9 Basic, which has been used for many years by KIM and JUNIOR owners in our club. Those of you who are familiar with KB9 Basic will find many changes and enhancements in it. Among the most important are:

- Support of hexadecimal numbers.
- Cursor movement with CURSOR TO command.
- Support of normal and inverse video output.
- Support of all eight I/O devices with DEVICE command.
- Saving programs in disk files both in binary and ascii format.
- Loading programs from disk files both in binary and ascii format. (Now you can write your programs with the full screen editor)
- Executing Basic programs directly from DOS65 or DOS65 command files, without entering the Basic environment.
- Executing DOS commands and utilities within the Basic environment.
- Input/Output redirect.
- Sequential access data-files.
- Random access data-files.
- 'Intelligent' ONERR GOTO command with associated error variables for proper error trapping.

* DUBBELADRESSERING *

FRANK BENS

In het verleden heb ik de indruk gekregen dat men de dubbeladressering op de uitbreidingsprint van de JUNIOR niet kon waarderen en daardoor is er naar diverse oplossingen gezocht in relatie met de VIA.

Mijn vraag is nu waarom er op de disk-controllerkaart geen maatregelen getroffen voor dubbeladressering. Mijn inziens is er dubbeladressering van \$1900 t/m \$197F, en wel hierom:

A15 t/m A18 gaan via IC3,4 naar IC5,6 t.b.v. CS1,3. A7 gaat via IC4 naar IC7 t.b.v. CS2. A1,0 gaan ook via IC4 naar IC7 als AA1 en AA0.

De adreslijnen A6 t/m A2 blijven ongebruikt, Don't care, hetgeen dus dubbeladressering inhoudt. Een voorbeeld hiervan:

```

A15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 1 1 0 0 1 0 X X X X X 0/1 0/1
|               |               |
o.a. via IC5,6      Don't care  AA1 AA2
t.b.v. CS1,3        CS2

```

Hieruit blijkt dat:

A15-A12 = 1 HEX
A11-A 8 = 9 HEX
A 7-A 4 = 0000 tot 0111 => 0-7 HEX
A 3-A 0 = 0000 tot 1111 => 0-F HEX

Achter elkaar opgeschreven wordt dit:
\$1900 - \$197F

Diegenen die dit ook opgevallen is worden verzocht mij daarvan via de redactie in kennis te stellen. Mogelijk heeft men tevens ook een oplossing ervoor, dan hoor ik dat graag.

- RESUME command to resume program execution after error handling.
- Advanced control statements for structured programming:
* IF ... THEN ... ELSE
* WHILE ... ENDWHILE
* REPEAT ... UNTIL
- Executing Basic commands from string variables with EXECUTE command.

A comprehensive manual is available with many examples on each subject. Since KB9 Basic is a copyright of Microsoft, we can only supply the object code to those who have bought a legal Microsoft Basic. (send copy of invoice) The manual is available to all members of the '6502 KENNERS'. Please contact Erwin Visschedijk, Dillelaan 11, Wierden.

For those of you who are fed up with writing obscure PRINT statements for Elektor's graphics card there is good news too. A graphical Basic is already in preparation. Draw your pictures with easy to use Basic commands like PLOT, MOVE, COLOR etc. The graphical Basic will also support many commercial available plotters.

 ** DATA-INPUT (EXEC) **

Systeem: APPLE met Disk
 Auteur : Frans Verberkt
 Hillekensacker 12-10
 6546 KB NIJMEGEN
 TEL.: 080 - 779555

Dit programma maakt DATA-regels aan middels een EXEC-file. Na het intypen en starten van dit programma zal het mogelijk zijn vanaf regel 25000 DATA-regels toe te voegen. Het geeft U tevens enig inzicht in het werken met EXEC-files. Het programma is traag, maar goed. Na het intypen van elke regel zult U even moeten wachten, omdat deze regel toegevoegd wordt. Het is natuurlijk mogelijk meer regels in te typen voor de disk geactiveerd wordt, maar heeft als nadeel dat bij tussentijdse RESET de ingetoetste informatie verloren raakt. Het voordeel van dit programma is, dat bij een RESET het programma weer gestart kan worden met RUN zonder dat er iets aan mankeert. U moet wel het programma SAVEN als er na een nachtje slapen Uw eigen gegevens gelezen of aangevuld moeten worden.

```
1000 REM DATA-INPUT (EXEC) 19sep86
1010 GOSUB 2010: REM INIT
1020 GOSUB 3010: REM PRINT DATA-REGELS
1030 GOSUB 4010: REM REGEL INVOEREN
1040 GOSUB 5010: REM MAAK TEXT-FILE
1050 PRINT D$;"EXEC";FI$
1060 END
2000 REM INIT
2010 DI$ = CHR$(4): REM DOS commando
2020 FI$ = "DATA": REM naam van EXEC-file
2030 QO$ = CHR$(34): REM ASCII voor QUOTE
2040 DR = 25000: REM begin DATA-regels
2050 RETURN
3000 REM PRINT DATA-REGELS
3010 HOME
3020 RESTORE
3030 DR = DR + 10
3040 READ DA$
3050 IF DA$ = "DATAEND" THEN 3080
3060 PRINT " ";DA$
3070 GOTO 3030
3080 RETURN
4000 REM REGEL INVOEREN
4010 PRINT "Type volgende regel:"
4020 INPUT IN$
4030 RETURN
5000 REM MAAK TEXT-FILE
5010 PRINT
5020 PRINT DI$;"OPEN";FI$
5030 PRINT DI$;"DELETE";FI$
5040 PRINT DI$;"OPEN";FI$
5050 PRINT DI$;"WRITE";FI$
5060 GOSUB 5510: REM MAAK EXECFILE
5070 PRINT DI$;"CLOSE";FI$
5080 RETURN
5500 REM MAAK EXECFILE
5510 PRINT DR;"DATA";QO$;IN$;QO$
5520 PRINT "RUN"
5530 RETURN
25000 REM DATA-REGELS
29000 DATA "DATAEND"
```

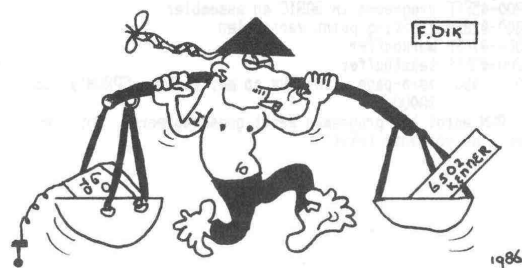
BOEKINFORMATIE

Bouw zelf een expertsysteem in BASIC; met programma-listings voor APPLE II, Commodore 64/128, GW BASIC, IBM BASIC MSX BASIC en ZX Spectrum.
 Chris Naylor, Academic Service 1986, 256 p., 4fl.45,-.
 ISBN 90 6233 167 X

Je staat voor de etalage van een boekhandel. Het oog valt op of wordt getrokken door een boektitel die je in je onderbewustzijn al kende: Bouw zelf een expertsysteem in Basic. Laatst op de bijeenkomst werd zo'n expertsysteem, inderdaad met enige trots, getoond in FORTH op de OCTOPUS. Je had er al veel over horen praten. Wie straks geen expertsysteem bezit, blijft achter. Is het 'het einde' in artificial intelligence? Zoiets wil je natuurlijk ook, antwoord krijgen op alle vragen die je stelt. Dus, kopen dat boek.

Thuisgekomen lees je de omslag van het boek. "Een expertsysteem is een computerprogramma dat een bepaalde hoeveelheid menselijke 'kennis' kan opnemen. De gebruiker kan dit programma, de 'expert', vervolgens consulteren om van die kennis gebruik te maken. Huisartsen kunnen het gebruiken voor het helpen stellen van een diagnose; chemici kunnen het gebruiken bij het afleiden van chemische structuren en ingenieurs gebruiken zo'n systeem bij het zoeken naar olie en andere fossiele brandstoffen. Tot nu toe draaien dergelijke programma's hoofdzakelijk op de grotere computers, waarbij van heel bijzondere programmeertalen gebruik gemaakt wordt. Dit boek laat op een eenvoudige manier zien wat zo'n expertsysteem is en hoe je ermee omgaat. De programma's die hierbij gebruikt worden zijn gewone BASIC-programma's, die in elke microcomputer direct ingetikt kunnen worden. De lezer is in staat zich hierdoor een beeld van kunstmatige intelligentie te vormen en kan zelf ervaren hoe een expertsysteem werkt. ...".

Chris Naylor heeft kans gezien een brug te slaan over de kloof tussen de gevoelsmatige afstand tot het begrip 'expertsysteem' - dat een verwijzing in zich draagt naar ingewikkelde dingen die alleen door deskundigen kunnen worden begrepen - en de werkelijkheid van de betrekkelijke eenvoud van de gemiddelde gebruiker van zakelijke en hobbycomputers. Op een wel heel speciaal humoristische wijze, waarbij me soms de tranen in de ogen schoten, ziet hij kans je steeds meer te boeien en voor te bereiden op zijn heel serieuze verhandeling over waarschijnlijkheden en kansen en het daarbij stap voor stap bouwen van het eigen systeem, rekening houdend met de noodzaak dat de lezer zelf variabelen zal moeten vinden en in een matrix vastleggen. Het boek beoogt niet dat de gegeven programma's uitsluitend worden ingetikt, daar is het te duur voor, het is bedoeld om de bouw te laten begrijpen hoe het systeem in elkaar zit om vervolgens te laten ontdekken dat het feitelijk niet is gebaseerd op ingewikkelheden, maar op nuchter en logisch redeneren. Chris Naylor ontkomt er echter niet aan wiskundige formules en begrippen te hanteren om het thema van zijn boek overeind te houden. Hier wordt duidelijk dat geboeid worden door de wijze van schrijven niet voldoende is. Een redelijke mate van inzicht in wiskunde is eveneens vereist. Het mooie werk van Chris Naylor lijkt me dan ook eerder geschikt voor gevorderden in Basic, niet voor beginners.



 ** RTTY TX/RX VOOR DE ACORN ATOM VAN PA3ELB **

Door: F.ank Vergoossen (PA3ELB), Nederland
 Tel.: 04754 - 1972

Inleiding

Er zijn reeds enkele programma's verschenen om RTTY signalen te ontvangen en/of te verzenden, die overigens bij vele zend- en luisteramateurs met succes in gebruik zijn. Nadat ik aan het experimenteren was geslagen met de VIA besloot ik om zelf een (eenvoudig) RTTY-programmaatje te maken. Omdat ik echter voortdurend wijzigingen en verbeteringen aanbracht, is dit korte programma al snel uitgegroeid tot een groter programma, waar ik mogelijkheden heb ingebouwd naar mijn eigen voorkeur, die ik helaas miste bij sommigen andere programma's. Na herhaaldelijke verzoeken van diverse personen heb ik dan ook besloten dit programma aan iedereen beschikbaar te stellen. Graag zou ik zien dat dit programma anderen aanzet om zelf aan de slag te gaan, want ook dit programma bestaat maar uit tamelijk eenvoudige recht-toe-recht-aan instructies, wat toch maar weer bewijst dat leuke programma's niet echt moeilijk moeten zijn. Dan volgt nu de handleiding voor het gebruik van het programma.

Hardware

Voor het ontvangen van RTTY signalen is een telexdecoder benodigd om de mark- en spacefrequenties om te zetten in een "1" of "0", zodat deze signalen eenvoudig door de computer te decoderen zijn. Dergelijke schakelingen zijn te vinden in o.a.:

Acorn Nieuws samenvatting 1982 blz. 96

Elektuur juli/augustus 1982 blz. 8-14

Elektuur juni 1983 blz. 6-58

Deze interface wordt aangesloten op poort c, bit 3 van de 8255 (mark="1", space="0"). Deze aansluiting is de meest rechtse als men van achter de computer naar de DIN-plug kijkt (pen 7). Ik heb voor deze aansluiting gekozen om niet een grote 64-polige connector te hoeven gebruiken voor de aansluiting van de decoder. Desgewenst kunt u het programma wijzigen voor gebruik met de VIA-poort.

Voor het zenden van RTTY wordt een AFSK-sigitaal opgewekt op PB7 van de VIA. Als u nu een weerstand van 4,7 kOhm tussen PB7 en de cassette-reorder-uitgang opneemt, kunt u voor zenden en ontvangen gebruik maken van slechts 1 zeven-polige DIN-plug, en u kunt de signalen ook nog op de recorder opnemen voor experimenten. Als u de MJCDS kan gebruiken heeft u deze weerstand reeds gemonteerd. Na verzwakking (bij mij een spanningsdeler bestaande uit een weerstand van 10 kOhm en een van 100 Ohm) en eventueel filtering kan men het AFSK-sigitaal op de ingang van de transceiver aanbieden.

Om de zendontvanger automatisch op zenden te schakelen als u gaat zenden is de mogelijkheid ingebouwd om de push-to-talk ingang van uw set aan te sluiten op PB6 met tussen-schakeling van een transistor (TX="1", RX="0").

Software

Geheugengebruik:

#2900-#5FFF programma in BASIC en assembler

#2800-#28FF floating-point variabelen

#6000-#7FFF werkbuffer

#8200-#9FFF tekstbuffer

#80 -#90 zero-page (let dus op met sommige EPROM's op #A000)

Na RUN wordt het programma eerst geassembleerd, intussen ziet u de volgende tekst:

BEDIENING:

t = TX (=zenden)
 r = RX (=ontvangen)
 c = CW ID. (=morse-identificatie)
 b = BUFFER (op voorhand intypen)
 a = MENU (hier komt u de eerste keer vanzelf in)

Nadat het programma klaar is met assembleren, komt u vanzelf in het menu terecht. Hier kunt u de diverse wijzigingen aanbrengen, en wel door op de gewenste cijfertoets te drukken. De nieuwe (standaard-)waarde komt dan vanzelf op de plaats van de oude waarde. Om ervoor te zorgen dat alle output naar het scherm snel genoeg verloopt is een VDU-routine in het programma opgenomen. Hieronder volgen de waarden waarop u alles kunt instellen, waarbij de waarden die bij het de eerste keer binnenkomen van het menu zijn ingesteld hier als eerste staan vermeld:

- 1 BAUDRATE 45.45/50/57/75/100/110 Baud
- 2 SHIFT 170/300/425/850/1000/85 Hertz
- 3 LAAGSTE FREQUENTIE 1275/2125 Hertz
- 4 POLARITEIT MLSH/MHSL
 (mark laag, space hoog/mark hoog, space laag)
- 5 BLANK TEKEN UIT/AAN
- 6 AUTO CR/LF UIT/AAN
- 7 UNSHIFT ON SPACE UIT/AAN
- 8 STOPBIT 1.5/1 bit
- 9 LEESTEKENS INT/WU/MIL
 (internationaal/Western Union/military)

t = NAAR TX (zenden)
 r = NAAR RX (ontvangen)
 b = WIJZIG BUFFER (tekstbuffer van 7,5 kB)

Als u het programma alleen voor ontvangen gebruikt is voor u alleen de baudrate, polariteit, unshift on space en leestekens van belang. Tijdens ontvangen wordt het bell-karakter afgedrukt als b, een niet bestaand karakter uit de set leestekens als e. Wilt u de tekst tevens op de printer zetten (ook mogelijk tijdens zenden) dan hoeft u slechts de printer aan te zetten en verder niets. Het programma controleert namelijk zelf of de printer wel of niet aanstaat, en zet zo ja de tekst op papier.

Tijdens zenden is er een werkbuffer die er voor zorgt dat u gerust zo'n 8 kB tekst sneller typen mag dan de computer uitzendt, deze buffer kunt u overigens ook al beginnen vol te typen tijdens ontvangen. Als u dan overschakelt op zenden wordt de tekst hierin direct uitgezonden. Verder is er een tekstbuffer voor het op voorhand intypen van berichten of standaardteksten. Deze buffer kunt u vanuit het menu kiezen om er tekst in op te slaan en is max. 7,5 kB lang. Als u dan tijdens het zenden b intypt wordt de tekstbuffer gecopieerd in de werkbuffer en uitgezonden. Verder is er een call-gever aanwezig waarin u de roepnaam van uw amateurstation kunt plaatsen. Om deze te activeren typt u c, de roepnaam wordt dan in morse uitgezonden en wel met de mark- en spacetoon die ook voor RTTY gebruikt wordt. Verder is er nog een handigheidje ingebouwd om fouten tijdens het typen te kunnen verbeteren, want de Baudotcode kent geen code om 1 teken terug te gaan. Als u namelijk sneller typt dan de computer zendt kunt u nog voor het uitzenden van de ingetypte tekst de fout verbeteren met DELETE. Niet voorkomende tekens worden tijdens uitzenden genegeerd. Als u de auto CR/LF aanzet wordt na 64 tekens automatisch carriage return en linefeed gegeven, voor het geval de ontvangstzijde gebruik maakt van een telex-machine (die kunnen na het einde van de regel niet verder).

Indicators

Tijdens ontvangen wordt rechtsonder een r gezet (van RX), tijdens zenden een t (van TX). Links hiervan wisselt een wit streepje in het ritme van de mark- en spacetoon van plaats (omhoog=mark, omlaag=space), dit werkt tijdens zenden en ontvangen.

Wijzigingen

U kunt in de listing de volgende dingen naar eigen voorkeur aanpassen:

- *K = string waarin de roepnaam van uw station voor de morseidentificatie, max. 255 karakters, dus meer dan genoeg.
- V = snelheid van de morsetekens, 10-30 woorden per minuut, hier 20 want dit is nog net niet te snel volgens de machtigingsvoorwaarden van de PTT.
- C = clockfrequentie van uw ATOM, normaal 1 MHz, ik heb echter het programma nog niet getest op andere clockfrequenties.

Ook de beginwaarden van het menu kunt u wijzigen waarbij de volgende variabelen van belang zijn:

- %B = baudrate in Baud
- S = frequentieshift in Hertz
- F = frequentie van de laagste toon in Hertz
- I = polariteit 0=MLSH, 8=MHSL
- G = blank teken 1=uit, 0=aan
- D = auto CR/LF 1=uit, 0=aan
- A = unshift on space 3=uit, 0=aan
- E = stopbit 1=1.5 bit, 0=1 bit
- N = leestekens 1=INT, 2=NU, 3=MIL

Wanneer u de interface aan wilt sluiten op PBO van de VIA moet u de volgende wijzigingen aanbrengen:

- wijzigen : regel 40 REA TTL IN PBO VAN 6522
- regel 1000 de 8 in 1 veranderen
- regel 1040 de 8 in 1 veranderen
- regel 2580 LDA#B800;AND#1
- regel 2750 LDA#B800;AND#1

- weglaten : regel 2410, 2420, 3430, 3440
- toevoegen: regel 2175 LDA#BA;STA#B003

Verdere gegevens over RTTY (die ik ook heb gebruikt voor dit programma) kunt u vinden in:

Elektuur juni 1983 blz. 6-37

Vademecum VERON blz. 279

(Redaktie: Nog enkele referenties :

- RTTY met de JUNIOR; Telexberichten via korte golf op uw scherm. DE 6502 KENNER 6(1982)23(okt)25-32
- RTTY ontvangst. HCCN 6(1983)2(apr)21-23
- RTTY-decoder. Elektuur 23(1983)6(jun)58.)

Laat het eens weten voor het geval u interessante wijzigingen aanbrengt, misschien hebben meer mensen er iets aan.

Veel succes gewenst.

N.B. Het volgende is alleen van belang voor de echte "freaks" die wijzigingen willen aanbrengen.

Zero-page adressen

- 80 RTTY: Baudotcode bij TX en RX.
- morse: Morsecode.
- 81 Asciiwaarde af te drukken karakter.
- 82 RTTY: 0=letters, 1=figures (voor TX eerst 3=nog niks).
- 83 RTTY: 2e Baudotcode na (un)shift of CR/LF.
- 84 Key-scan routine asciiwaarde vorige ingedrukte toets.
- 85 Key-scan routine 0=geen toets ingedrukt, 1=wel toets ingedrukt.

- 86 bewaaradres voor Y, index van computer in buffer.
- 87 bewaaradres voor Y, index van keyboard in buffer.
- 88 RTTY: aantal tekens na laatste CR/LF.
- 89 morse: bewaaradres voor A, lengte morsetoon 1 of 3.
- 8A RTTY: teller bij RX
- morse: bewaaradres voor Y, index karakterno. in call.
- bewaaradres voor Y bij de buffer.
- 8B LSB tekstbuffer altijd 0.
- 8C MSB tekstbuffer 82-9F.
- 8D LSB computer in werkbuffer altijd 0.
- 8E MSB computer in werkbuffer 60-7F.
- 8F LSB keyboard in werkbuffer altijd 0.
- 90 MSB keyboard in werkbuffer 60-7F.

Variabelen

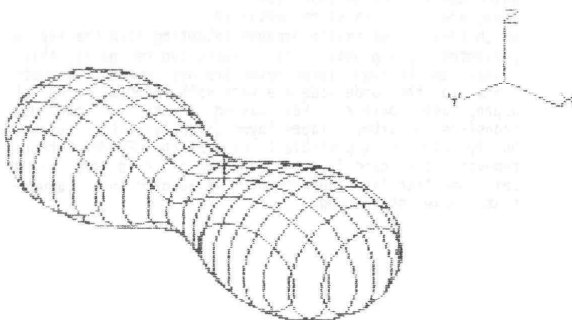
- A unshift on space 0=on 3=off
- %B Baudrate in Baud.
- B tellerstand lengte in bit, afh. van %B.
- C clockfrequentie, normaal 1 MHz.
- D auto CR/LF na 64 tekens 0=on 1=off.
- E lengte stopbit 0=1 bit 1=1.5 bit.
- F frequentie laagste toon in Hertz.
- G blankteken 0=on 1=off.
- H tellerstand markfrequentie.
- L tellerstand spacefrequentie.
- I polariteit 0=mlsh 8=mhsl (bij laatste wordt H met L verwisseld).
- J tellerstand morse-snelheid, lengte 1/2 punt.
- K stringadres call voor cw-identificatie.
- N leestekens 1=int 2=wu 3=mil.
- P assembleradres tijdens assembleren.
- Q beginadres assembler.
- R 0-25 baudottabel 26-83 morsetabel.
- S frequentieshift tussen mark en space in Hertz.
- T stringadres figures int, wu of mil.
- V morse-snelheid 0-30 wpm, normaal 20.
- W dummy voor inlezen tekstbuffer 8200-9FFF.
- Y dummy passno. assembler en H-L verwisseling.
- Z dummy ascii-toetsleesroutine menu.

Arrays

- II initialisatie via, snelle vdu, printer routine, clear flags.
- TT zendprogramma.
- RR ontvangsprogramma.
- ZZ morseidentificatie en buffer routine.

DE ASSEMBLY SOURCE-LISTING BEHOORENDE TOT DIT ARTIKEL WORDT IN EEN VOLGENDE EDITIE GEPUBLICEERD.

=====



```
*****
**      A SIMPLE EPROM DISC FOR 6XXX COMPUTERS      **
*****
```

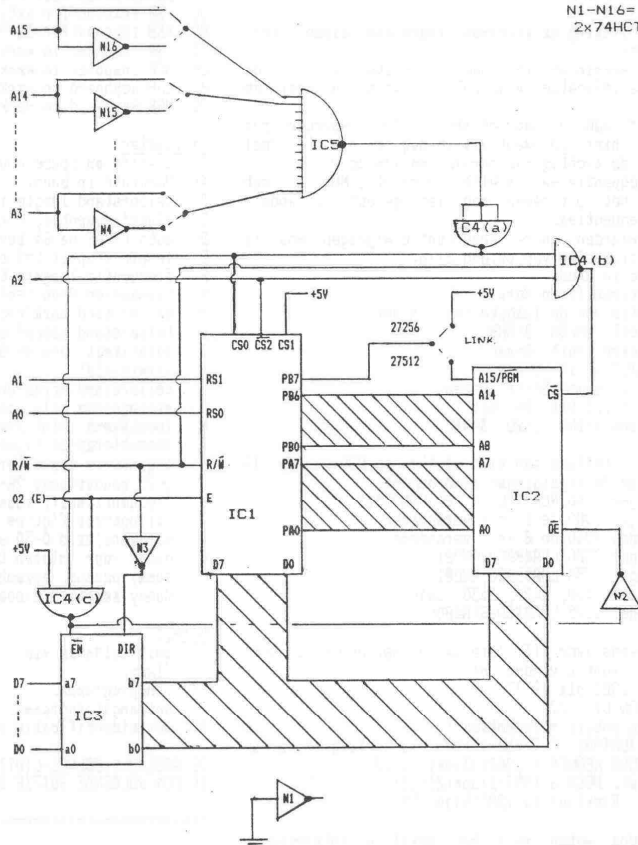
Author: Andrew Gregory, England.

The advent of 32K and 64K EPROMs has made it easy to construct an EPROM 'disc' of useful capacity. It is ideal for storing languages, operating systems and assemblers especially in cassette based systems. I built mine so that I could quickly load Micro-ADE and Epron programmer software into my expanded JUNIOR. The design has been kept as simple as possible but could be further developed, perhaps into a cartridge system.

HARDWARE DESCRIPTION

As can be seen from the circuit diagram the address lines of the 'disc' EPROM are set up by a 6821 PIA. This occupies four out of the eight memory locations occupied by the card. The EPROM can be read at the others. If the system address lines A0 and perhaps A1 are connected to the EPROM directly (preferably through buffers) then consecutive addresses can be read, but I have not done this. The address decoder is similar to that of Elektor's VDU card. The 74HCT540/74LS540 inverting buffers are functionally identical to 74HCT240/74LS240 but have a pin-out more convenient for home-made PCBs. To make a eurocard sized PCB for the Elektorbus without photographic techniques is not too difficult provided it is double-sided. The tracks to the 'a' row of the DIN14162 connector are layed and soldered on the component side allowing them to be taken through the 'c' row which has no pads on this side. There are varieties of connector in which this is impossible because soldering from the top is prevented by a plastic foot. Tracks can be quite thick (1mm) and through-plated holes are not needed. Connections to the underside are made with special pins at appropriate points. For laying tracks you can use transfers or artwork tapes layed directly on the copper. On my card it is possible to replace the EPROM without removing the card from the busboard as it is held in a zero insertion force socket which is mounted on an aluminium bracket at the end.

CIRCUIT OF THE EPROM DISC



IC1=6821
IC2=27256/27512
IC3=74HCT245/74LS245
IC4=74LS10
IC5=74LS133
N1-N16=
2x74HCT540/74LS540

SOFTWARE DESCRIPTION

The source listing is of a "loader" which copies files to RAM. I have blown this into the JUNIOR PMW ROM at address \$1750. The first page of the EPROM is a directory which holds up to 28 entries. Each entry consists of a hex file-number, EPROM address, load address, length and an execute address, nine bytes in total. Unused entries are left unprogrammed as \$FF is an illegal file number. The loader has been divided into subroutines to allow calling from within

programs. I have stored Micro-ADE under the filename 'A0'.
It is loaded from the PM monitor as follows:

```
1750(space) 20 R      Run loader
?A0(CR)       Filename
READY
JUNIOR
R              Run execute address
Date(DDMMYY)    Micro-ADE cold start
```

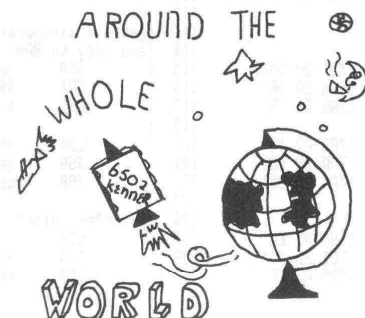
It loads almost instantly.

```

2  : EPROM DISC SOFTWARE
3  :
4  : This version intended for expanded JUNIOR.
5  : It may be blown into top of PMV or PME rom.
6  :
7  : DIRECTORY FORMAT:-
8  :
9  :   byte 0 file number
10 :   byte 1 eeprom address low
11 :   byte 2 " address high
12 :   byte 3 load address low
13 :   byte 4 " address high
14 :   byte 5 length low
15 :   byte 6 " high
16 :   byte 7 execute address low
17 :   byte 8 " high
18 :
19 : Up to 28 files are allowed. The directory fills the
20 : first page of the "disc" eeprom. (27256/27512)
21 :
22 : MONITOR ROUTINES
23 :
24 : 105F labjun equ $105F      warn start monitor
25 : 1334 prcha equ $1334      print ascii character
26 : 12AE reccha equ $12AE     keyboard input
27 : 13A2 ipb equ $13A2        key input a byte
28 : 11E8 crlf equ $11E8       carriage ret line feed
29 : 11D6 messy equ $11D6     print a message
30 :
31 : CARD ADDRESSES
32 :
33 : Note - page 19 on an expanded JUNIOR can be freed by
34 : reconnecting pin 23 of IC1 (6522) to A88. In my
35 : system the 6845 on the VDU card is addressed at
36 : $1900 and the RTC card at $1920.
37 :
38 : 8 addresses are required.
39 :
40 : 1908 card equ $1908       convenient i/o space
41 :
42 : 1908 dra equ card+00      6821 registers
43 : 1909 cra equ card+01
44 : 190A drb equ card+02
45 : 190B crb equ card+03
46 : 190C eprom equ card+04    address of eeprom
47 :
48 : ZERO PAGE ADDRESSES
49 :
50 : 00EA 50 org $EA           points to "disc" byte
51 : 00EB 51 epadl res 1
52 : 00EC 52 epadh res 1
53 : 00ED 53 memadl res 1      points to RAM byte
54 : 00EE 54 memadh res 1
55 : 00EF 55 lenl res 1        length of file
56 : 00FA 56 lenh res 1
57 : 00FA 57 org $FA
```

APPLE COMPUTER INC. VERDUBBELT WINST
IN FISCAL JAAR 1986.

Cupertino/Leist. 15 oktober 1986.
De netto winst en de winst per aan-
deel van Apple Computer Inc. zijn in
het fiscaal jaar 1986, dat loopt van
oktober t/m september, meer dan ver-
dubbeld. In het afgelopen jaar steeg
de winst met 151 procent tot 154
miljoen US dollar, vergeleken met
61.2 miljoen dollar in het jaar er-
voor. De winst per aandeel steeg met
141 procent tot 2.39 US dollar verge-
leken met 0.99 US dollar per aandeel
in het fiscale jaar 1985. De omzet in
het afgelopen jaar bedroeg 1.902
miljard US dollar. In 1985 was dit
1.918 miljard US dollar. De omzet in
het vierde fiscale kwartaal bedroeg
510.8 miljoen US dollar, een stijging
van 25 procent t.o.v. het jaar ervoor
toen de omzet 409.7 miljoen dollar
bedroeg. De winst in het vierde kwar-
taal steeg met 47 procent tot 32.9
miljoen US dollar ofwel 0.51 US dol-
lar per aandeel vergeleken met 22.4
miljoen ofwel 0.36 US dollar per
aandeel in dezelfde periode van het
jaar ervoor. De bruto marge uitge-
drukt in percentages van de omzet
bedroeg in het vierde kwartaal 53.3
hetgeen over het gehele jaar genomen
neerkomt op 53.1 procent. De overeen-
komstige percentages van het jaar
ervoor bedroegen respectievelijk 45.9
en 41.7. In Nederland is Apple Compu-
ter gegroeid met een percentage van
maar liefst 42 procent in vergelij-
king met het fiscale jaar 1985.



F. Dik
86

```

00FA      58 pointl res 1
00FB      59 pointh res 1
60      60
61      61
62      62 *****MAIN PROGRAM*****
63      63
1750      64      org $1750
1750 20 E811 65      JSR crlf          CR + LF
1753 A9 3F 66      LDA #'?
1755 20 3413 67      JSR prcha        print "?"
1758 20 A213 68      JSR ipb         filename in A reg
175B 20 6A17 69      JSR load         load the file
175E A0 4C 70      LDY #$4C        "READY" message
1760 90 02 71      BCC 1.f         branch if loaded OK
1762 A0 46 72      LDY #$46        "WHAT" message
1764 20 D611 73      JSR messy       print message
1767 4C 5F10 74      JMP labjun      warm start monitor
75      75
76      76 *****SUBROUTINES*****
77      77
78      78 ;Separate subroutines here to allow calling by programs
79      79
176A C9 FF 80      load CMP      #$FF      illegal entry?
176C F0 31 81      BEQ      err
176E 48 82      PHA          save entry
83      83
84      84 ;initialise PIA
176F A9 04 85      LDA      #$04
1771 A0 FF 86      LDY      #$FF
1773 A2 00 87      LDX      #$00
1775 8E 0919 88      STX      cra        access direction reg A
1778 8E 0B19 89      STX      crb        access direction reg B
177B 8C 0A19 90      STY      dra        A is outputs
177E 8C 0A19 91      STY      drb        B is outputs
1781 8D 0919 92      STA      cra        access data reg A
1784 8D 0B19 93      STA      crb        access data reg B
94      94
1787 8E 0B19 95      STX      dra        "disc" address = $0000
178A 8E 0A19 96      STX      drb
97      97
178D 68 98      PLA          retrieve entry
178E A8 99      TAY
100      100
101      101 ;Search for correct directory entry
102      102 2 CPY      eprom      is this it?
1792 F0 0D 103      BEQ      4.f         branch if yes
1794 A9 09 104      LDA      #$09        set up next address
1796 18 105      CLC
1797 6D 0B19 106      ADC      dra
179A 8D 0B19 107      STA      dra
179D 90 F0 108      BCC      2.b         branch if more entries
109      109
179F 38 110      err SEC          set carry if not found
17A0 60 111      RTS
112      112
113      113 ;get six parameter bytes from directory
114      114 ;and copy to RAM.
17A1 20 D817 115      JSR      get-param
17A4 E0 06 116      CPX      #$06        more?
17A6 D0 F9 117      BNE      4.b         branch if yes
118      118
17A8 A2 10 120      LDX      #$10
17AA 20 D817 121      JSR      get-param
17AD 20 D817 122      JSR      get-param
123      123
124      124 ;transfer "disc" start address to PIA
17B0 A5 EA 125      LDA      epadl       low byte
17B2 8D 0B19 126      STA      dra
17B5 A5 EB 127      LDA      epadh       high byte

```

Tips and tricks for the EC65K/Junior.

By: Coen Boltjes
Nw. Plantage 9
2611 XH Delft.

1) It's possible that while the computer don't acces the VDU-Ram, some video-noise is visible on the screen. This is caused by the fact that between valid address-lines the VDU-Ram address is offered to N37 of the VDU-card. This results in resetting the flip-flops FF1-FF4, which causes the video-noise. Connecting pen 1 of N37 with 02, instead of logic one, will solve this problem.

2) There is a lot of misunderstanding about the track to track access time. If you want to speed up this time, don't change location \$26A5. This must be \$D4. Location \$26A3 must contain the time in milliseconds. Standard is this \$28, but in a lot of configurations lower values will work.

3) If you want to run your computer on 2 MC, a parameter in a 10 milisecond loop must be changed. \$267B contains this parameter, which is \$31 or \$62 for 1 MC and 2 MC respectively. To avoid manual adjusting of this parameter if you use another speed than usual a little routine is developed which can be referred in the bootroutine.

4) If you want to use a boot-routine which is too large to fit in \$2200-\$2300 you can temporary use the space \$2E79-\$3179. Because this area is used for the directory and for the page 0 and 1 swap, the routines in this area will be available until a DIR command or a swap is executed.

5) If you use the CPU-card with a 65C02, you don't need special software to program EPROM's with the Elektuur programmer (jan 1982). The RDY line will hold the processor during the program pulse, so you can use the MOVE routine of the monitor.

=====

YOUR 1987 SUBSCRIPTION :
(for European countries only)
SEND CHEQUE OF Hfl. 59,50 TO
W.L. VAN PELT AT THE EDITORIAL
OFFICE (eurocheque = Hfl. 50,=)

```

17B7 8D 0A19      128      STA      drb
17B8 A0 00        129      ;
17BC AD 0C19      130      ;copy file from "disc" to ram
17BF 91 EC        131      LDY      #$00
17C1 EE 0819      132      LDA      eprom      get byte
17C4 D0 03        133      STA      [memad],Y    store in RAM
17C6 EE 0A19      134      ;
17C9 C8           135      INC      dra        next "disc" address
17CA D0 02        136      BNE      1.f
17CC E6 ED        137      INC      drb
17CE C6 EF        138      1      INY          increment ram pointer
17D0 D0 EA        139      BNE      1.f
17D2 C6 EF        140      INC      memadh
17D4 D0 E6        141      1      DEC      lenl    decrement length
17D6 18           142      BNE      3.b
17D7 60           143      DEC      lenh
17D8 EE 0819      144      BNE      3.b        branch if more
17DB AD 0C19      145      ;
17DE 95 EA        146      CLC          clear carry as OK
17E0 E8           147      RTS
17E1 60           148      ;
17E2 60           149      ;SUBROUTINE
17E3 60           150      ;
17E4 60           151      get-param INC dra    get next entry
17E5 60           152      LDA      eprom
17E6 60           153      STA      epadl,X
17E7 60           154      INX
17E8 60           155      RTS
17E9 60           156      ;
17EA 60           157      ;END

```

PAPERWARE-SERVICE ENGLISH

All prices in the PPWS only for European countries!

KINTAP is a Tape Read/Write utility program written for Elektor's OCTOPUS/EC-65 in combination with Elektor's BASIC-CODE/KIM tape interface card 65028 (Computer Special 2). by Marc Lachaert, Belgium. The tapes written by KINTAP are fully compatible with those written on any KIM or JUNIOR. Through this tapes, the OCTOPUS / EC-65 can communicate either with a KIM, a basic JUNIOR, an extended JUNIOR or a DOS-JUNIOR. This document together with the TAPUTL subroutines in assembly-source listings. Send cheque of Hfl. 31,50 to W.L. van Pelt (eurocheque 22,=) at the editorial office.

+++++ Want COMAL on your DOS65 ? +
+ Ask the editorial office +
+++++

** MON65 OP JUNIOR **

Door: Erik v.d. Broek, Nederland.

Waarom een Octopus gebouwd, als je al een JUNIOR hebt? Om MON65 te implementeren? Dan bouw je JUNIOR toch om! En al mijn software dan? Dan bouw je toch een uitschakelbare, omschakelbare overschakeling! Kortom, een JUNIOR in native-mode met een Octopus-MON65-simulationmode. De schakeling bevat een vier-standen-schakelaar met de volgende functies:

- 1) JUNIOR met afgekoppelde bus, voor het foutzoeken in uitbreidingen.
- 2) JUNIOR met bus. Reset vektor en standaard monitor worden opgezocht, echter in die oude 2708, waarvoor de bovenste 8 K (op bus zit daar MON) op de basis- en interface kaart wordt geadresseerd. Cassette en video-routines blijven bruikbaar.

3) MON65-halfsimulatie :

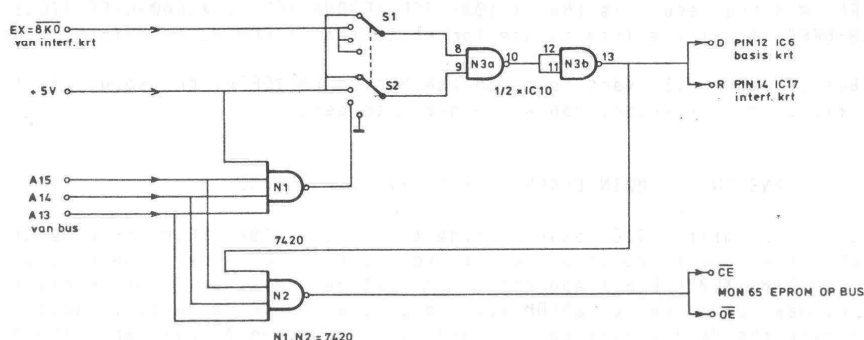
- Een nieuwe VIA voor keyboard op C110 - C1FF
- 2 K RAM op C800 - CFFF
- VDU RAM op D000 - D7FF
- MON65 Eprom uiteraard op E000 - FFFF
- Toeters en bellen naar eigen inzicht te bevestigen.

Verschil met de Octopus MON65:

- De ACIA, nog een VIA (de 'centronics') en de pieper. Dit is door eenieder zelf in te vullen, dunkt me.
- De onderste 8 K blijft op standaard geadresseerd. Software die op 0800 - 1FFF RAM verwacht, gaat dus de mist in, maar JUNIOR programma's blijven bruikbaar.
- 4) Idem 3), doch ook onderste 8 K wordt op de bus geadresseerd. Hier hebben we de meest realistische OCTMON65 simulatie.

De schakeling : 2 nieuwe poortjes, een 6522 en een 6116 + adres-dekodering. En niet te vergeten! de schakelaar.

- De RST draadbrug en D - EX verbinding verwijderen.
- Adres-dekoderingen voor de EPROM, de VIA en de 6116 spreken voor zich.



HERMAN ZONDAG

THE JUNIOR COMPUTER REVISITED

I - THE COMPLETE ADDRESS DECODING

Did you built the Junior Computer before the SAMSON/OCTOPUS project appeared? Are you reluctant to leave great part of the Junior, to go to the architecturally new SAMSON system? Are you one of those who loves to pick up your soldering iron, some ICs, resistors and other hardware stuff, and to ASSEMBLE, and to test, and to debug, as much as you program in software?

Then this is for you.

We can improve our old J.C. in many ways, up to an extent where it turns far more perfect (in terms of memory) than the latter *elektor computing* project.

One of the first steps is to resolve one of the greatest drawbacks of the J.C.: the memory organization. The Junior was born with 1024 bytes of RAM in the lowest address space, and other 1024 bytes of EPROM in the \$1C00-1FFF range; more, the basic input/output device, the 6532 PIA-RAM-TIMER, was laid in the address range of \$1A00-1BFF, occupying that way 512 places, where 128+32 were more than enough. Then the interface board came, with RAM growing from \$0400 to 07FF, EPROMs at \$0800-17FF, and a new I/O device at \$1800-19FF, this being another loss: 512 bytes where only 16 needed.

The bus board opened the frontiers to an endless growing, with marks as the dynamic RAM card, the VDU board, and the floppy controller card. And it was so, that in Nov.82 we were told how to change the memory scheme, turning its organization upside-down, in a way that we got RAM from \$0000 to BFFF, but with the 8K upper space disorganized as before. To complicate matters further, the PROM 82S23, IC 17 on the interface card, was surveying what could be modified or not, in the way that it forbides writing on certain locations or reading from other sources. And, of course, adapting the disk software from Ohio Scientific, brought further losses: the disk "controllers", located on \$C000-C0FF could be satisfied with only 6 locations! In the same way, the VDU board lost 6 addresses on the CRTC decoding, leaving aside other 2K in the whole project.

Soon I realized that, and found a way to gain the undecoded addresses in the \$C000 → DFFF (disk + video) space. Although this now went beyond the scope of this article, may be some users will benefit from the idea, so it follows. I've used for some time the 8K RAM + EPROM card (2) with some utility programs by means of the addition of only one IC. A 4-input and gate (fig.1) enables the card in the locations unused by the VDU or the FLOPPY cards; two lines must run to this gate: one (\$C000→C0FF) is already wired in the DISK board from pin 6 of IC2 (74LS10) to pin 6c of the bus connector (ref.4, fig.9); the other (\$D800→D807), in the VDU board, can be wired by an insulated jump from pin 9 of IC9 (74S133) to pin 6a of the bus (ref.6, fig.4). The opposite is done in the EPROM card from these bus-connector pins to the inputs of the new gate as illustrated here. Every time one of these signals is low, the card cannot be read; this memory board shall work only with EPROM (3x2716) and addressed in the base address of \$C000-DFFF. And the result is that \$C100-C7FF (EPROM IC25), \$C800-CFFF (IC26) and \$D808-DFFF (IC28) are free to use for utilities in EPROM, totalizing a good 6K space.

But if you really want to clean the RAM up to \$DFFF, the solutions follow. See fig.2 for the memory map we're going to get.

A - MODIFICATIONS ON THE MAIN BOARD (ref.1, fig.2); fig.3.

In the primitive J.C. board reside the CPU, a 2708 EPROM, 1K byte of RAM and the 6532 PIA. Due to construction implications, I left there the 6532 and the CPU; the RAM (\$E000-E3FF) and the EPROM will be addressed in other place.

The new (2732) system EPROM will be on the interface card. I choose so, because during the development period, and even now, when I reassemble the MONI-

TOR, erasing and burning another EPROM, I do not have to detach the interface from the main board; it's just a matter of turning them upside-down to change the EPROMs.

The RAM chips (IC4 and 5) and R16 must jump out of their sockets, and the 6532 IC3 re-addressed; this is done with the addition of a 74LS85 (plus a 74S30 if possible) in a little PCB.

The PIA has 128 bytes of RAM, two I/O ports, timers, flags and edge detectors. Elektor scattered all the PIA registers by \$1A80 → 1AFF, but we can learn from a manufacturer's sheet how to put them in 32 bytes (7). It's enough to decode the \$XX80-XX9F space after the 128 bytes of RAM (XX00-XX7F). A 4-bit comparator, the 74LS85 will be preset to \$A in one of its sets of inputs, while the other set will read the four address lines A4 through A7. Then, the three outputs will flag the addresses \$XX0X → XX9X (A<B, pin 5), the 16 addresses \$XXAX (A=B, pin 6) and the remaining \$XXBX → XXFX addresses (A>B, pin 7).

Using an (available) signal of \$EFXX, the PIA is laid down on the addresses \$EF00-EF9F, as shown in the drawing. The \$EFXX line is generated with an 8-input nand gate 74S30, which I've located on the disk board, for convenience; but I agree that this gate would better be joint with the 74LS85 on the main board, so that the system could work without the disk card, for debugging purposes. The new addresses of the PIA will be:

\$EF00 → EF7F : RAM	
\$EF80 - PAD (FA80)	
\$EF81 - PADD (FA81)	BETWEEN BRACKETS: the old addresses
\$EF82 - PBD (FA82)	
\$EF83 - PBDD (FA83)	\$EF94 - CNTA (FAF4)
\$EF84 - RDTDIS (FA84)	\$EF95 - CNTB (FAF5)
\$EF85 - RDFLAG (FA85)	\$EF96 - CNTC (FAF6)
\$EF8C - RDTEN (FADC)	\$EF97 - CNTD (FAF7)
\$EF84 - EDETA (FAE4)	\$EF9C - CNTE (FAFC)
\$EF85 - EDETB (FAE5)	\$EF9D - CNTF (FAFD)
\$EF86 - EDETC (FAE6)	\$EF9E - CNTG (FAFE)
\$EF87 - EDETD (FAE7)	\$EF9F - CNTH (FAFF)

This is the first part of the input/output page (\$EF00-EFFF) which will contain, besides the PIA, the CRTC, the disk PIA and ACIA, the VIA (Centronics), the real-time-clock,... and other user-defined addresses.

The PIA re-addressing can not be done independently of the other modifications because the new \$EF00-EF9F space is enabled in the interface/bus by the PROM + bus buffers, and if they are enabled, something will be buffered from them to the main board, surely different from what should be read from the PIA.

B - MODIFICATIONS ON THE INTERFACE CARD (ref.3, fig.1); fig.4

This will be one of the more heavily modified boards.

The bus-buffers must "close" the traffic in one only address range, IC5 will turn into a 4K EPROM, IC4 in a 2K CMOS RAM without one page, the VIA re-addressed, and a new IC added.

Following fig.4, let us begin with IC1, 6522 VIA: previously located at \$F800-F9FF (\$1800-19FF) it shall fall into the tiny \$EFA0-EFAF 16 addresses that it deserves. Line \$EFXX, already mentioned, will substitute K6 on the CS2. And A=B, from the 'LS85 (main board) will be linked to CS1 instead of the old VIA line (the output of gate N35).

Second, the bus buffers problem:

The traffic of information between the main board and the interface card (the bus board is connected to the interface, after the bus buffers), is controlled by the PROM IC17. This PROM, 82523, was programmed in a way that the bus buffers 74LS243 IC11 and 12, are disabled in 2 ranges, \$E000-E3FF (or \$0000-03FF)

and \$FA00-FFFF (\$1A00-1FFF) corresponding respectively to the 1K RAM, and to the PIA + EPROM, all in the main board, so that no read no write is permitted on or from the interface card and bus.

With the proposed location of the PIA (only) in the main board, it will be enough to limit the limitations to the range \$EF00-EF9F, of the sole device addressed above the interface card's buffers. All the remaining space can be addressed as required. So, the PROM is extracted and "substituted" by a 'LS157 selector. Its Y output is made low in the PIA address space, so that the buffers permit writing to the interface and the bus (no harm if no other device is located there in the same addresses) but no byte is read from there.

A (new, left unused from the memory reorganization for the floppy-disk interface) (ref.5, fig.1) 3-input nor gate (N33 or 34), will add signals \$EFXX, A=B (or XXAX) and A>B (or XXBX $\rightarrow$ XXFX). What means that the SEL input of the 'LS 157 will only be logic high when neither of the 3 nor-inputs is high: what means finally, the space \$EF00-EF9F. The consequence of this SEL input being high, is that the B input is "connected" to the Y output, and so the bus buffers' controller lines are fed with a low level, what determines that only write is permitted to the interface and bus, and no read is possible.

Third, IC4, RAM 6116 (or 51xx) was already addressed at \$E800-EFFF (or \$0800-0FFF, EPROM, originally), but now it must be disabled in the new I/O address page or \$EF00-EFFF. It's just enough to utilize one of the unused 3-input nand gates (again after the disk-interface, ref.5 fig.1 and 3). This gate N46, will replace functionally N41 on the $\overline{CS}$ and the $\overline{OE}$ RAM pins. Its inputs will be the $\overline{I2}$ as before, the inverted K2 and K3 as before, plus the \$EFXX line. The effect is the subtraction of \$EF00-EFFF from this RAM. If you want, you can unsolder R2, R4.

Fourth, the new system-monitor EPROM, on IC5, (previously 2716). The $\overline{OE}$ doesn't suffer modifications neither the $\overline{CE}$ pin; but pin 21 is now line A11. Of course, gate N43 must add the four K lines (K4 through K7) which must be linked together on points D-E-F, while R14 and R15 must withdraw from their places on the main board.

C - DISK BOARD - fig.5; (ref.4, fig.9)

Some ICs must be added to this card if the complete address decoding is desired. Six addresses are needed. A 13-input nand gate ('S133) will decode up to 8 addresses, and the so obtained \$EFC0-EFC7 line is directed to CS2 of either the 6850 ACIA and 6821 PIA. Then the ACIA's CS1 is linked with A2 and CS0 with A1, so it lies at \$EFC4-EFC5, where A2 is high and A1 is low. The CS1 of the 6821 links to A2, CS0 with $\oplus$ and RS1 and RS0 with A1 and A0 as before; this chip will be active on \$EFC0,1,2,3. A drawback of this arrangement is that the data buffers of the board (IC13, 'LS245) must be active on the 2+4 addresses. I ought to do that with two new gates, a 3-input-and, plus a 2-input-or, as drew in figure 5. Summing it all, it is a '133, a '240 (six inverters), a '11, a '32 and a '30 if this last gate was not put on the main board.

This card was already "backed" with the logic necessary to the control of the drive motor (10) so in the end, it will be a heavy one!

D - VDU BOARD - fig.6; (ref.6, fig.4)

This is the slightest modified one, with one more gate, necessary to reduce the decoding from 8 to 2 bytes, the only two registers necessary for the communication with the 6845 CRTC. The 3-input and gate (74S11) adds two more address lines to the 13 already present to the 'S133 IC9. The remaining address lines present to N38 and to N37 must follow the chosen addresses for the 6845 CRTC (\$EFC6,7) and to the circuitry conducting to the 6116 (or 51xx) RAM IC. The video RAM is now at \$E000-E7FF; the first K of this 2K space corresponds to the

RAM in the main board, and the (previously) closed bus-buffers. It implies that those RAM chips in the main board and the PROM on the interface card, be already out. The second 1K was the RAM in the interface card, which must be out too.

The location of the video memory may be arguable; I decided so, because, at the end, I have RAM continuously from \$0000 to \$EF80, which passes through the VIDEO RAM, and includes the PIA RAM, some 61312 RAM addresses which can be useful to some applications.

If you want to verify each step, a possible time-table to do all these modifications, could be:

- 1- install 'LS85 + 'LS30 on the main board, to get signals $\overline{EFXX}$, $A < B$, $A = B$ & $A > B$
- 2- "subtract" I/O page from RAM IC4 on the interface board: wire new N46; (re-program the VIDEO routines on EPROM, relocating the video vectors from $\overline{EFXX}$ to other place). Take off R2 (K3).
- 3- re-address all your "user" I/O addresses, as the real-time clock, A/D boards, peripheral control..., to the new I/O page unused addresses; if not enough space, subtract one more page (previous point 2-).
- 4- re-address the VIA 6522 in the interface board; (reprogram the Centronics routines on EPROM: $\overline{F800-F9FF} \rightarrow \overline{EFA0-EFAF}$).
- 5- re-address the ACIA 6850 and PIA 6821 on the disk board; (reprogram the boot-up routines on EPROM, and OS65DV3.3 addresses on diskettes: $\overline{C0XX} \rightarrow \overline{EFC0-EFC5}$, tracks 00, 01, 06-4, 17 & 18 (COPIER), 39-1).
- 6- re-address the 6845 CRTC on the VDU board; (reprogram the VIDEO routines on EPROM, addresses on diskettes as in BEXEC*: $\overline{D800-D807} \rightarrow \overline{EFC6,7}$).
- 7- program, beforehand, a 2732 EPROM with all your system/monitor routines; you will have: $\overline{F000-F7FF}$ as before, $\overline{F800-FBFF}$ afresh (old 6522 VIA + 6532 PIA), and $\overline{FC00-FFFF}$, the old system monitor space at the main board, with all the new addresses.
- 8- a) on the interface board: take off the PROM and associated N40-IC15 ('LS30), unsolder the pull-up resistors R32, 34 & 35
take off the RAM chips IC2 and IC3 plus R33 (K1)
install the 'LS 157
and re-address the new EPROM 2732: wire A11 to it
b) on the main board: take off the EPROM and R14 (K7)
take off the RAM chips IC4 and IC5, plus R16 (K0)
and re-address the 6532 PIA; unsolder R15 (K6)
c) on the VDU board: re-address the CMOS video-refresh-RAM.

CONCLUSION

If you feel skilfull to undertake such an enterprise, don't give up. It is worthy! And be patient... it took me months.

If you need, I'll try to answer your questions.

References:

1. junior computer - elektor MAY.80
2. 8K RAM + 4, 8 or 16K EPROM on a single card - elektor SEP.80
3. the fully fledged junior computer - elektor MAY.81
4. floppy-disk interface for the Junior (I) - elektor NOV.82
5. floppy-disk interface for the Junior (II) - elektor DEC.82
6. VDU card - elektor SEP.83
7. R6532 - RAM-I/O-TIMER (RIOT) - Rockwell data sheet D42-Rev.6 - OCT.83
8. address decoding - elektor JAN.84
9. memory timing - elektor FEB.84
10. controlling the floppy-disk drive motor - elektor APR.84

Next time: ANOTHER VIEW OF THE DOS JUNIOR (SOFTWARE TIPS FOR THE DOS JUNIOR)

(F.LOPES, Colégio dos Orfãos, Lg, Baltazar Guedes, 4300 PORTO PORTUGAL)

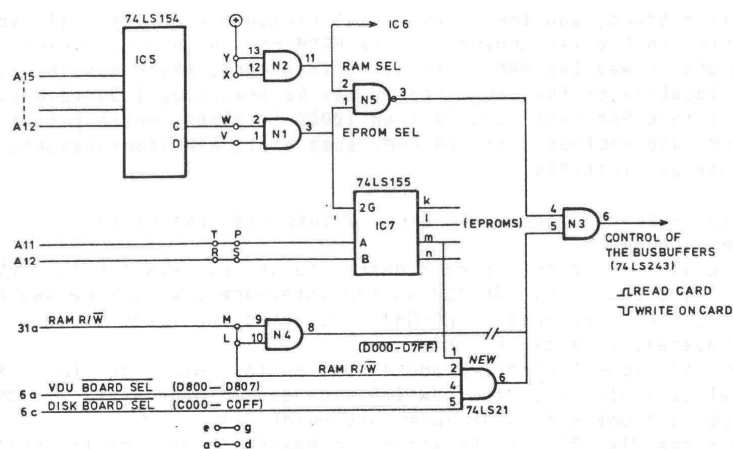


figure.1 - modifications on the RAM EPROM card: how to take advantage of the un-decoded memory in the FLOPPY + VDU cards, to use 3 EPROMs in the area C100-C7FF, C800-CFFF and D800-DFFF.

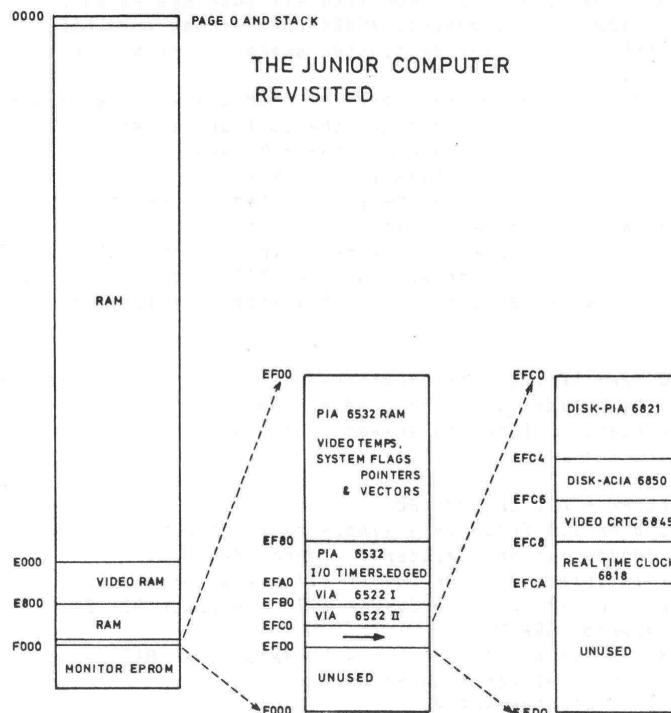


figure.2 - a new memory map for the Junior: a maximum clear-up of the user RAM, and a condensed I/O page.

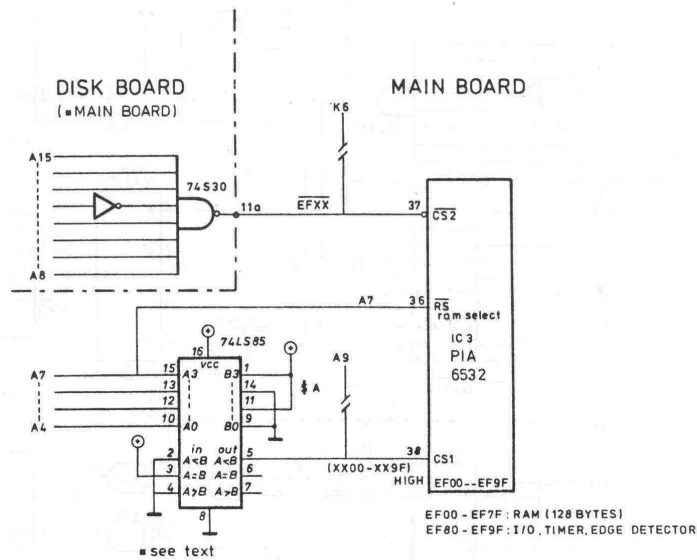


figure.3 - a 'LS85 plus a 'S30 re-address the PIA, from a 512 wide to a 32 tiny address.

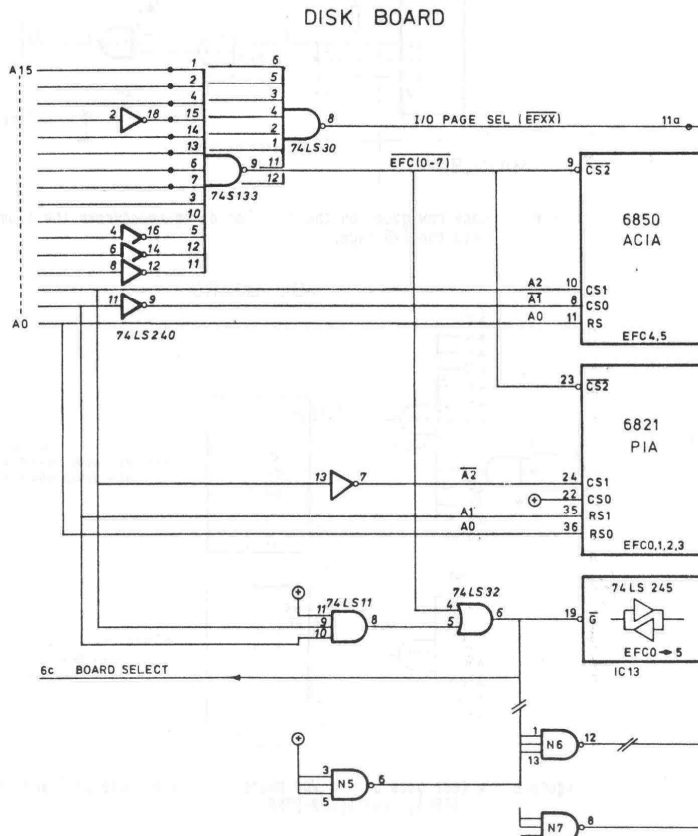


figure.4 - on the interface board, the PROM retires, and the bus-buffers are on-ly disabled on the PIA addresses.

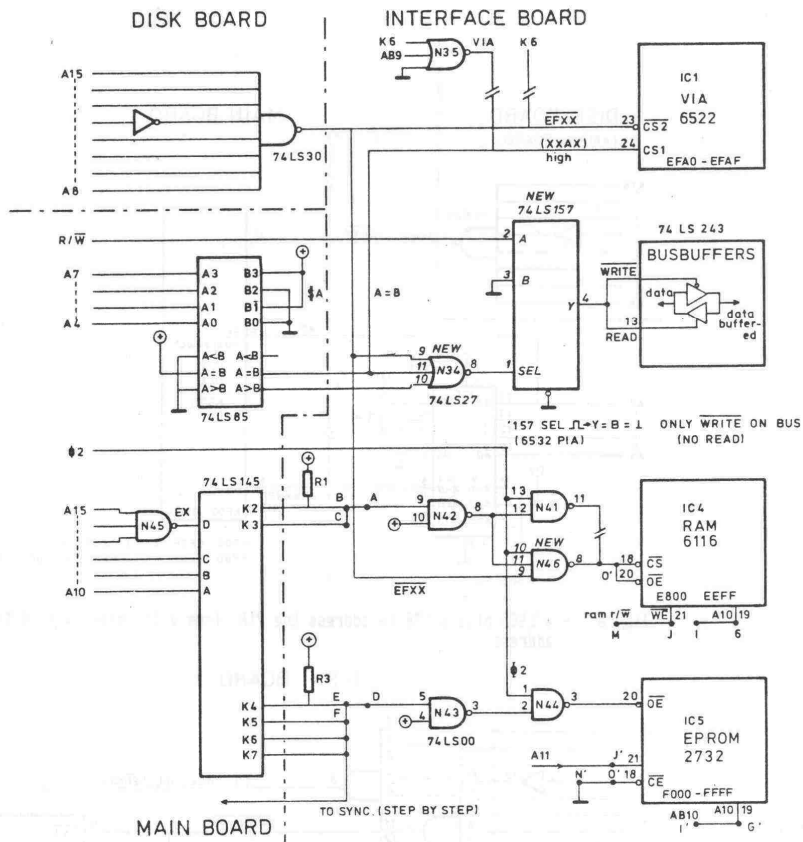


figure.5 - some new gates on the disk board, to re-address the floppy addresses into the I/O page.

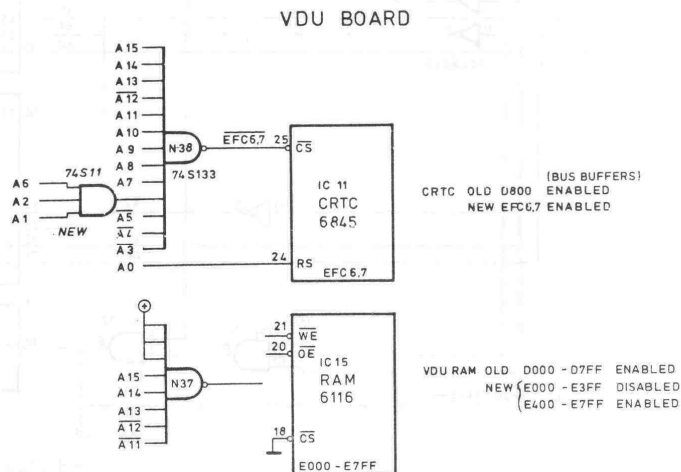


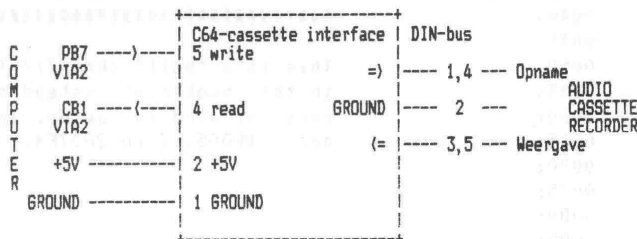
figure.6 - a last gate on the VDU board, to address the CRTC and the Video RAM at \$EFC6,7 and \$E000-E7FF.

HET LEZEN VAN BASICODE-2 VOOR DOS-65

1 INLEIDING

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

Het interface heb ik als volgt met de computer verbonden:



2 GEBRUIK

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

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

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

Met behulp van de EDITOR kan de ontvangen ASCII-file aangevuld worden met de standaard subroutines (BCSTART). Deze nieuwe file kan door BASIC ingelezen worden mid-dels het input redirect. Daarna kan de BASIC-file gesaved worden en kan het pro-gramma gerund worden.

3 CASSETTE-INTERFACE

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

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

4 ONTVANG-PROGRAMMA

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

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

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

5 STANDAARD ROUTINES

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

Voorbeeld: BASICODE (ASCII commando file)

```
LO 0:BCSUBR
BASIC
```

Hierna kan het BASICODE programma geladen en gerund worden.

6 LITERATUUR

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

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

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

```

0005: F427          ORG    $F427
0010:
0015:                #####
0020:                ##      "B E L L"    for   E C 6 5      ##
0025:                ##      by Lindstrøm & Rasmussen      ##
0030:                ##      Parkvej 1, DK-4534 Hørve      ##
0035:                ##      juli/86      ##
0040:                #####
0045:
0050:                This is a "bell"/chr$(7)/ for ec65. The routine is placed
0055:                in the 'boot-rom' instead of the different screen-paramet-
0060:                ers not used in samson. To implement the routine, change
0065:                adrs. $F005..7 to 2027F4.
0070:
0075:
0080:
0085:
0090:
0095:
0100:                +5v
0105:                |
0110:                |
0115:                |
0120:                |
0125:                |
0130:                |
0135:                |
0140:                |
0145:                |
0150:                |
0155:                |
0160:                |
0165:                |
0170:                |
0175:                |
0180:                |
0185:                |
0190:                |
0195:                |
0200:                |
0205:                |
0210:                |
0215:                |
0220:                |
0225:                |
0230:                |
0235:                |
0240:                |
0245:                |
0250:                |
0255:                |
0260:                |
0265:                |
0270:                |
0275:                |
0280:                |
0285:                |
0290:                |
0295:                |
0300:                |
0305:                |

```

via2 / PB7

2k2

0

PB7 of via 2 is used for output of the tone. Use a high-ohm speaker or a mini-buzzer and add a transistor for louder sound (but not too loud! it will get on your nerves)

```

0135:                # via 2  addresses #
0140: F427          VBPBD *    $E110
0145: F427          VBTACH *    VBPBD +05 timer 1, counter-high
0150: F427          VBTALL *    VBPBD +06 timer 1, latch low
0155: F427          VBTALH *    VBPBD +07 timer 1, latch high
0160: F427          VBTBCL *    VBPBD +08 timer 2, latch low, counter low
0165: F427          VBTBCH *    VBPBD +09 timer 2, counter high
0170: F427          VBACR *    VBPBD +0B auxillary control register
0175: F427          VBIFR *    VBPBD +0D interrupt flag register
0180: F427          AHOLD *    $2363
0185:
0190: F427 AD 63 23    LDA    AHOLD    get character
0195: F42A C9 07      CMPIM $07      is it CHR$(07)?
0200: F42C F0 01      BEQ    BELL      if yes, goto bell
0205: F42E 60         RTS            else return
0210: F42F 4B         PHA            save A
0215: F430 A9 C0      LDAIM $C0      set bit 7,6 "1"->T1:en.PB7 out/fre run
0220: F432 8D 1B E1  STA    VBACR    bit 5 to "0"---> T2: one shot
0225: F435 A9 00      LDAIM $00      timer 1 generates the tone
0230: F437 8D 16 E1  STA    VBTALL
0235: F43A A9 02      LDAIM $02      set tone-frequency to approx. 1000 Hz
0240: F43C 8D 17 E1  STA    VBTALH
0245: F43F 8D 15 E1  STA    VBTACH    start the tone
0250: F442 A9 00      LDAIM $00      timer 2 controls duration of the tone
0255: F444 8D 18 E1  STA    VBTBCL
0260: F447 A9 C3      LDAIM $C3      gives a duration of approx. 50 ms.
0265: F449 8D 19 E1  STA    VBTBCH
0270: F44C A9 20      LDAIM $20      timer 2 set bit 5 of IFR
0275: F44E 2C 1D E1  TIMOUT BIT    VBIFR    is time out ?
0280: F451 F0 FB      BEQ    TIMOUT    if not wait,
0285: F453 AD 18 E1  LDA    VBTBCL    time out, clear timer 2 interrupt flag
0290: F456 A9 00      LDAIM $00
0295: F458 8D 1B E1  STA    VBACR    and stop tone.
0300: F45B 68         PLA            restore A
0305: F45C 60         RTS            and return

```

```

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

00D2          org      $00d2          8 zero page addresses used

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

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

2000          org      $2000

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

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

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

202A A0 49          headr  ldy     #$49          write header

```

assette routines

20-Sep-86 13:23

Page 1

```

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

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

```

```

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

```

```

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

```

```

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

```

```

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

```

sette routines

20-Sep-86 13:23

Page 2

```

P1 A0 3A      ldy    #$3a
P3 20 BF20    cassv1 jsr    rdbyte    read byte
P6 D2 D4      cmp    [begad]    check it
P8 F0 06      beq     1.f        ok ?
PA E6 D8      inc     ercnt      then it's an error
PC D0 02      bne     1.f
PE E6 D9      inc     ercnt+1
P0 A0 35      1      ldy    #$35
P2 20 5420    jsr     next        next address
P5 B0 EC      bcs     cassv1      not yet ready ?
P7 28         plp
P8 60         rts

P9 A2 00      findst ldx    #$00    find start of transmission
PB A0 2B      1      ldy    #$2b
PD 20 D020    jsr     rdbit        read half bit
P0 90 F7      bcc     findst      if it's not a 'long 1', then restart
P2 CA         dex
P3 D0 F6      bne     1.b        check for another 'long 1' (256 needed)
P5 A0 22      2      ldy    #$22    search 'short 0'
P7 20 D020    jsr     rdbit
P9 B0 F9      bcs     2.b
PB 4C D020    jmp     rdbit        skip other half of 'short 0'

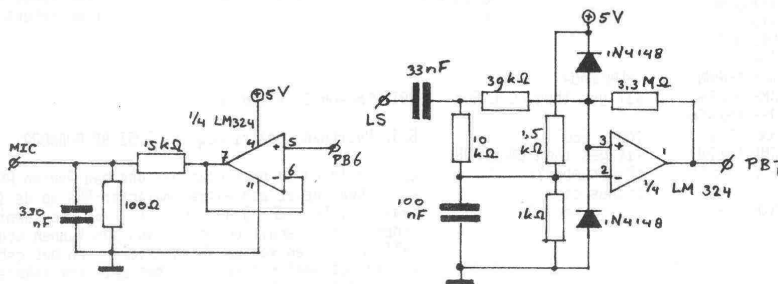
PF A2 08      rdbyte  ldx    #$08    read byte = 8 bits
P1 48         1
P2 20 CD20    jsr     rd2bit        read bit
P5 68         pla
P6 2A         rora        rotate new bit
P7 A0 3A      ldy    #$3a
P9 CA         dex
PB D0 F5      bne     1.b
PC 60         rts

PD 20 D020    rd2bit  jsr     rdbit    read full bit
P0 88         rdbit   dey          read half bit
P1 AD 80F7    lda     orb          wait for transition
P4 45 D3      eor     lastin
P6 10 F8      bpl     rdbit
P8 45 D3      eor     lastin
PA 85 D3      sta     lastin
PC C0 80      cpy     #$80
DE 60         rts

end

```

rors detected: 0



EXTENSION OF OS-650 V3.3 WITH DEL COMMAND

This extension applies to the JUNIOR with OS-650 V3.3 DOS.

While developing BASIC programs it is often necessary to remove a number of lines at the same time. For this purpose many BASIC interpreters have a DELETE command with which it is possible to remove several lines at the same time. This extension adds a DEL command to the BASIC interpreter from address \$3591 where some cursor-routine resides that is not effective on the JUNIOR. The DEL command replaces the LET statement that is not really used mostly (one could make a small program that enables either DEL or LET). In the addressstable and the keywordtable the following changes have to be made:

address	old	new
\$020E	\$A5	\$90
\$020F	\$09	\$35
\$029E	\$4C	\$44
\$029F	\$45	\$45
\$02A0	\$D4	\$CC

Directions for use:

DEL10 :deletes line 10
 DEL-10 :deletes up to line 10 inclusive
 DEL10- :deletes from line 10
 DEL10-20 :deletes from line 10 up to 20 inclusive
 DEL- :deletes the entire program

Gerrit van Woerkom
 Appelaar 55
 4907 KD Oosterhout
 The Netherlands

```

10      ;BASIC DEL CMD
20 0019=      INTEG =#19
30 006F=      INDEX1=#6F
40 0071=      INDEX2=#71
50 007A=      STVAR =#7A
60 00AC=      LOWTR =#AC
70      ;
80 00C0=      CHRGET=#00C0
90 00C6=      PREVCH=#00C6
100 051D=      RECOMP=#051D
110 0633=      RECHER=#0633
120 096C=      COLL =#096C
130 0E1E=      SNERR =#0E1E
140 10DD=      ILLEG =#10DD
150      ;
160 3590      *=#3590
170 3590 60      RTS
180 3591 A487      DEL      LDY #87
190 3593 C8      INY
200 3594 F003      BEQ NORUN      ;RUN-mode?
210 3596 4CD010      JMP ILLEG      ;if yes then FC ERROR
220 3599 20C600      NORUN      JSR PREVCH
230 359C D003      BNE DEL1      ;only DEL?
240 359E 4C1E0E      ERROR      JMP SNERR      ;if yes then SN ERROR
250 35A1 9004      DEL1      BCC LNNMBR      ;linenumber?
260 35A3 C9A4      CMP #A4      ;minus sign?
270 35A5 D0F7      BNE ERROR      ;if no then SN ERROR

```

```

280 35A7 206C09      LNNMBR      JSR COLL      ;fetch 1st linenumber
290 35AA 203306      JSR RECHER      ;search for 1st line
300 35AD 20C600      JSR PREVCH
310 35B0 F00C      BEQ DEL2      ;2nd linenumber?
320 35B2 C9A4      CMP #A4      ;minus sign?
330 35B4 D0E8      BNE ERROR      ;if no then SN ERROR
340 35B6 20C000      JSR CHRGET
350 35B9 206C09      JSR COLL      ;fetch 2nd linenumber
360 35BC D0E0      BNE ERROR      ;more, then SN ERROR
370 35BE A519      DEL2      LDA INTEG
380 35C0 051A      ORA INTEG+1
390 35C2 D006      BNE NONUL
400 35C4 A9FF      LDA #FF
410 35C6 B519      STA INTEG
420 35C8 B51A      STA INTEG+1
430 35CA A5AC      NONUL      LDA LOWTR
440 35CC A6AD      LDX LOWTR+1
450 35CE B571      STA INDEX2
460 35D0 B672      STX INDEX2+1
470 35D2 203306      JSR RECHER
480 35D5 9012      BCC DEL3
490 35D7 A001      LDY #01
500 35D9 B1AC      LDA (LOWTR),Y
510 35DB B8      DEY
520 35DC AA      TAX
530 35DD D004      BNE DEL4
540 35DF B1AC      LDA (LOWTR),Y
550 35E1 F006      BEQ DEL3
560 35E3 B1AC      DEL4      LDA (LOWTR),Y
570 35E5 B5AC      STA LOWTR
580 35E7 B6AD      STX LOWTR+1
590 35E9 A571      DEL3      LDA INDEX2
600 35EB 38      SEC
610 35EC E5AC      SBC LOWTR
620 35EE AA      TAX
630 35EF A572      LDA INDEX2+1
640 35F1 E5AD      SBC LOWTR+1
650 35F3 A8      TAY
660 35F4 B01E      BCS READY      ;remove to end program
670 35F6 BA      TAX
680 35F7 18      CLC
690 35F8 B57A      ADC STVAR
700 35FA B57A      STA STVAR
710 35FC 98      TYA
720 35FD B57B      ADC STVAR+1
730 35FF B57B      STA STVAR+1
740 3601 A000      LDY #00
750 3603 B1AC      NEXT      LDA (LOWTR),Y ;shift BASIC text
760 3605 9171      STA (INDEX2),Y ;upward in workspace
770 3607 C8      INY
780 3608 D0F9      BNE NEXT
790 360A E6AD      INC LOWTR+1
800 360C E672      INC INDEX2+1
810 360E A57B      LDA STVAR+1
820 3610 C572      CMP INDEX2+1
830 3612 D0EF      BCS NEXT      ;are we ready?
840 3614 4C1D05      READY      JMP RECOMP      ;rearrange BASIC lines
850                      ;and return to BASIC

```

BRIEVEN AAN DE REDAKTIE

S.J. Voortman, Beatrixweg 28, 3253 BB OUDDORP

Zijn er leden in onze club die ook een Proton PC-2 hebben, en weten hoe ze een extra character-RAM op de (Video-processor IC's) EF 9340 / EF 9341 (op de semi-grafische video-kaart) vanuit een programma aan kunnen sturen? (het zelf definieren van de 96 karakters, en het gebruik hiervan op het beeldscherm). Ik heb zelf wel enkele kopietjes van de data-sheet, maar weet er niet goed raad mee.

SETTING TALLY-PRINTER

PROTON 650X ASSEMBLER V4.4 PAGE: 0001

```

0001 0000      .TIT 'SETTING TALLY-PRINTER'
0002 0000      ;
0003 0000      ; *****
0004 0000      ;
0005 0000      ; THIS ROUTINE IS DEVELOPED TO CHANGE SEVERAL FUNCTION
0006 0000      ; SETTINGS OF A TALLY 1612 IMPACT PRINTER VIA THE SERIAL
0007 0000      ; INTERFACE, USING THE ACIA OF ELEKTOR'S CPU CARD.
0008 0000      ; THE ROUTINE IS RUNNING ON AN ELEKTOR'S CPU/VDU SYSTEM
0009 0000      ; USING THE PROTON DOS.
0010 0000      ; AN EXPLANATION OF THE VARIOUS MONITOR ROUTINES IS GIVEN
0011 0000      ; AT THE END OF THE SOURCE LISTING.
0012 0000      ; THE ROUTINE IS STARTED BY A FUNCTION KEY WICH VECTOR
0013 0000      ; IS LOADED AT THE ADDRESS $0521.
0014 0000      ;
0015 0000      ; *****
0016 0000      ;
0017 0000      ; AUTHOR : F.J.M. SMEEHUIJZEN
0018 0000      ; LIPPEDAL 19
0019 0000      ; 2904 CL CAPELLE AAN DEN IJSSEL
0020 0000      ; TEL: 010-4512507
0021 0000      ;
0022 0000      ; *=$0000
0023 0000      ;
0024 0000      ; HULP      *$+2
0025 0002      ; PARAM    *$+3
0026 0005      ;
0027 0005      ; *=$0506
0028 0506      ;
0029 0506 0011      .WOR TALLY
0030 0508      ;
0031 0508      ; *=$0521
0032 0521      ;
0033 0521      ; OUTFLG  *$+1      ; OUTPUT DEVICE FLAG
0034 0522      ;
0035 0522      ; *=$1100
0036 1100      ;
0037 1100 A9AD TALLY LDA # (TEXT0      ; DISPLAY
0038 1102 A213 LDX #) TEXT0      ; SCREEN HEADING
0039 1104 20BF13 JSR OUTPUT
0040 1107 A970 TAL1 LDA # (TEXT1      ; DISPLAY
0041 1109 A214 LDX #) TEXT1      ; '6 LINES PER INCH'
0042 110B 20BF13 JSR OUTPUT
0043 110E 203A16 JSR REDOUT      ; GET KEYBOARD
0044 1111 C959 CMP #'Y'      ; IF YES, CHANGE
0045 1113 F007 BEQ LPI6      ; INTO 6 LPI
0046 1115 C94E CMP #'N'      ; IF NOT, CHANGE
0047 1117 F010 BEQ TAL2      ; VALUE ON SCREEN
0048 1119 4C0711 JMP TAL1      ; WRONG CHOICE, ALWAYS BACK
0049 111C A9FD LPI6 LDA # (TEXT13      ; SEND 6 LPI
0050 111E A215 LDX #) TEXT13      ; ESCAPE STRING
0051 1120 20A313 JSR OUTP3      ; TO PRINTER
0052 1123 203D16 JSR OUTLOW      ; SET OUTPUT TO DISPLAY
0053 1126 4C4811 JMP TAL3      ; JUMP TO NEXT QUESTION
0054 1129 A991 TAL2 LDA # (TEXT2      ; DISPLAY
0055 112B A214 LDX #) TEXT2      ; ' 8 LINES PER INCH'
0056 112D 20BF13 JSR OUTPUT
0057 1130 203A16 JSR REDOUT      ; GET KEYBOARD
0058 1133 C959 CMP #'Y'      ; IF YES, CHANGE
0059 1135 F007 BEQ LPI8      ; INTO 8 LPI
0060 1137 C94E CMP #'N'      ; IF NOT, CHANGE
0061 1139 F0CC BEQ TAL1      ; VALUE ON SCREEN
0062 113B 4C2911 JMP TAL2      ; WRONG CHOICE, ALWAYS BACK
0063 113E A901 LPI8 LDA # (TEXT14      ; SEND 8 LPI
0064 1140 A216 LDX #) TEXT14      ; ESCAPE STRING
0065 1142 20A313 JSR OUTP3      ; TO PRINTER
0066 1145 203D16 JSR OUTLOW      ; SET OUTPUT TO DISPLAY
0067 1148 203716 TAL3 JSR CRLF
0068 114B 203716 JSR CRLF
0069 114E A9B2 TAL3A LDA # (TEXT3      ; DISPLAY
0070 1150 A214 LDX #) TEXT3      ; '10 CHAR/INCH'
0071 1152 20BF13 JSR OUTPUT
0072 1155 203A16 JSR REDOUT      ; GET KEYBOARD
0073 1158 C959 CMP #'Y'      ; IF YES, CHANGE
0074 115A F007 BEQ CPI10      ; TO 10 CPI
    
```

SETTING TALLY-PRINTER

PROTON 650X ASSEMBLER V4.4 PAGE: 0002

```

0075 115C C94E      CMP #'N'      ; IF NO, CHANGE
0076 115E F010      BEQ TAL4      ; VALUE ON SCREEN
0077 1160 4C2911    JMP TAL2
0078 1163 A905      LDA #TEXT15    ; CHANGE TO
0079 1165 A216      LDX #TEXT15    ; 10 CPI
0080 1167 20A313    JSR OUTP3
0081 116A 203D16    JSR OUTLOW    ; SET OUTPUT TO DISPLAY
0082 116D 4CB111    JMP TAL6
0083 1170 A9D3      LDA #TEXT4     ; DISPLAY
0084 1172 A214      LDX #TEXT4     ; '12 CHAR/INCH'
0085 1174 208F13    JSR OUTPUT
0086 1177 203A16    JSR REDOUT
0087 117A C959      CMP #'Y'      ; GET KEYBOARD
0088 117C F007      BEQ CPI12     ; IF YES, CHANGE
0089 117E C94E      CMP #'N'      ; INTO 12 CPI
0090 1180 F010      BEQ TAL5      ; IF NO, CHANGE
0091 1182 4C7011    JMP TAL4      ; VALUE ON SCREEN
0092 1185 A909      LDA #TEXT16    ; CHANGE TO
0093 1187 A216      LDX #TEXT16    ; 12 CPI
0094 1189 20A313    JSR OUTP3
0095 118C 203D16    JSR OUTLOW    ; SET OUTPUT TO DISPLAY
0096 118F 4CB111    JMP TAL6
0097 1192 A9F4      LDA #TEXT5     ; DISPLAY
0098 1194 A214      LDX #TEXT5     ; '16.5 CHAR/INCH'
0099 1196 208F13    JSR OUTPUT
0100 1199 203A16    JSR REDOUT
0101 119C C959      CMP #'Y'      ; GET KEYBOARD
0102 119E F007      BEQ CPI16     ; IF YES, CHANGE
0103 11A0 C94E      CMP #'N'      ; INTO 16.5 CPI
0104 11A2 F0AA      BEQ TAL3A     ; IF NO, CHANGE
0105 11A4 4C9211    JMP TAL5     ; VALUE ON SCREEN
0106 11A7 A90D      LDA #TEXT17    ; WRONG CHOICE, REPEAT
0107 11A9 A216      LDX #TEXT17    ; SEND
0108 11AB 20A313    JSR OUTP3     ; ESCAPE STRING
0109 11AE 203D16    JSR OUTLOW    ; TO PRINTER
0110 11B1 203716    JSR CRLF      ; SET OUTPUT TO DISPLAY
0111 11B4 203716    JSR CRLF
0112 11B7 A915      LDA #TEXT6     ; DISPLAY
0113 11B9 A215      LDX #TEXT6     ; 'CHARACTER WIDTH'
0114 11BB 208F13    JSR OUTPUT
0115 11BE 203A16    JSR REDOUT
0116 11C1 C94E      CMP #'N'      ; GET KEYBOARD
0117 11C3 F007      BEQ NORMAL    ; NORMAL WIDTH ?
0118 11C5 C944      CMP #'D'      ; IF YES, CHANGE INTO NORMAL
0119 11C7 F010      BEQ DOUBLE    ; DOUBLE WIDTH ?
0120 11C9 4CB711    JMP TAL6A     ; IF YES, CHANGE INTO DOUBLE
0121 11CC A911      LDA #TEXT18    ; WRONG CHOICE, REPEAT
0122 11CE A216      LDX #TEXT18    ; SEND NORMAL WIDTH
0123 11D0 20A313    JSR OUTP3     ; ESCAPE STRING
0124 11D3 203D16    JSR OUTLOW    ; TO PRINTER
0125 11D6 4CE311    JMP TAL7     ; SET OUTPUT TO DISPLAY
0126 11D9 A915      LDA #TEXT19    ; SEND DOUBLE WIDTH
0127 11DB A216      LDX #TEXT19    ; ESCAPE STRING
0128 11DD 20A313    JSR OUTP3     ; TO PRINTER
0129 11E0 203D16    JSR OUTLOW    ; SET OUTPUT TO DISPLAY
0130 11E3 203716    JSR CRLF
0131 11E6 203716    JSR CRLF
0132 11E9 A936      LDA #TEXT7     ; DISPLAY
0133 11EB A215      LDX #TEXT7     ; 'SET LEFT MARGIN'
0134 11ED 208F13    JSR OUTPUT
0135 11F0 203A16    JSR REDOUT
0136 11F3 C959      CMP #'Y'      ; GET KEYBOARD
0137 11F5 F007      BEQ SETLM     ; IF YES, ASK
0138 11F7 C94E      CMP #'N'      ; FOR MARGIN VALUE
0139 11F9 F037      BEQ TAL8     ; IF NO, JUMP
0140 11FB 4CE911    JMP TAL7A     ; TO NEXT QUESTION
0141 11FE A919      LDA #TEXT20    ; WRONG CHOICE, ALWAYS BACK
0142 1200 A216      LDX #TEXT20    ; SEND LEFT MARGIN
0143 1202 20A313    JSR OUTP3     ; ESCAPE STRING
0144 1205 203D16    JSR OUTLOW    ; TO PRINTER
0145 1208 A92D      LDA #TEXT25    ; SET OUTPUT TO DISPLAY
0146 120A A216      LDX #TEXT25    ; DISPLAY ' = '
0147 120C 208F13    JSR OUTPUT
0148 120F A200      LDX #*00
0149 1211 203A16    JSR REDOUT
0150 1214 9502      STA PARAM,X
0151 1216 E002      CPX #*02
; TYPE IN
; LEFT MARGIN
; VALUE
; MAX 3 DIGITS

```

```

0152 1218 F004      BEQ SETLM2
0153 121A EB        INX
0154 121B 4C1112    JMP SETLM1
0155 121E A941      LDA #'A'      ; SET OUTPUT
                                ; TO SERIAL PRINTER
0156 1220 8D2105    STA OUTFL6
0157 1223 A200      LDX #00      ; SET PRINTER
0158 1225 B502      LDA PARAM,X   ; TO LEFT MARGIN
0159 1227 203416    JSR OUTALL    ; USING INPUT
0160 122A EB        INX          ; VIA KEYBOARD
0161 122B E003      CPX #03
0162 122D D0F6      BNE SETLM3
0163 122F 203D16    JSR OUTLOW    ; SET OUTPUT TO DISPLAY
0164 1232 203716    JSR CRLF
0165 1235 203716    JSR CRLF
0166 1238 A957      LDA #TEXT8    ; DISPLAY
0167 123A A215      LDX #TEXT8    ; 'SET RIGHT MARGIN'
0168 123C 208F13    JSR OUTPUT
0169 123F 203A16    JSR REDOUT
0170 1242 C959      CMP #'Y'      ; IF YES, ASK
                                ; FOR MARGIN VALUE
0171 1244 F007      BEQ RIGHTM     ; IF NOT
0172 1246 C94E      CMP #'N'      ; SKIP SETTING
0173 1248 F037      BEQ TAL9       ; WRONG CHOICE, ALWAYS BACK
0174 124A 4C3812    JMP TAL8A
0175 124D A91D      LDA #TEXT21    ; SEND RIGHT MARGIN
0176 124F A216      LDX #TEXT21    ; ESCAPE STRING
0177 1251 20A313    JSR OUTP3     ; TO PRINTER
0178 1254 203D16    JSR OUTLOW    ; SET OUTPUT TO DISPLAY
0179 1257 A92D      LDA #TEXT25    ; DISPLAY ' = '
0180 1259 A216      LDX #TEXT25
0181 125B 208F13    JSR OUTPUT
0182 125E A200      LDX #00        ; TYPE IN
0183 1260 203A16    JSR REDOUT     ; RIGHT MARGIN
0184 1263 9502      STA PARAM,X    ; VALUE
0185 1265 E002      CPX #02        ; MAX 3 DIGITS
0186 1267 F004      BEQ SETRM2
0187 1269 EB        INX
0188 126A 4C6012    JMP SETRM1
0189 126D A941      LDA #'A'      ; SET OUTPUT
0190 126F 8D2105    STA OUTFL6    ; TO SERIAL PRINTER
0191 1272 A200      LDX #00      ; SET RIGHT
0192 1274 B502      LDA PARAM,X    ; MARGIN USING
0193 1276 203416    JSR OUTALL    ; INPUT VIA
0194 1279 EB        INX          ; KEYBOARD
0195 127A E003      CPX #03
0196 127C D0F6      BNE SETRM3
0197 127E 203D16    JSR OUTLOW    ; SET OUTPUT TO DISPLAY
0198 1281 203716    JSR CRLF
0199 1284 203716    JSR CRLF
0200 1287 A978      LDA #TEXT9    ; DISPLAY
0201 1289 A215      LDX #TEXT9    ; 'SET PAGE LENGTH'
0202 128B 208F13    JSR OUTPUT
0203 128E 203A16    JSR REDOUT
0204 1291 C959      CMP #'Y'      ; IF YES, ASK FOR
0205 1293 F007      BEQ LENGTH     ; PAGE LENGTH VALUE
0206 1295 C94E      CMP #'N'      ; IF NOT
0207 1297 F037      BEQ TAL10     ; SKIP SETTING
0208 1299 4C8712    JMP TAL9A     ; WRONG CHOICE, ALWAYS BACK
0209 129C A921      LDA #TEXT22    ; SEND PAGE LENGTH
0210 129E A216      LDX #TEXT22    ; ESCAPE STRING
0211 12A0 20A313    JSR OUTP3     ; TO PRINTER
0212 12A3 203D16    JSR OUTLOW    ; SET OUTPUT TO DISPLAY
0213 12A6 A92D      LDA #TEXT25    ; DISPLAY ' = '
0214 12A8 A216      LDX #TEXT25
0215 12AA 208F13    JSR OUTPUT
0216 12AD A200      LDX #00        ; TYPE IN
0217 12AF 203A16    JSR REDOUT     ; PAGE LENGTH
0218 12B2 9502      STA PARAM,X    ; VALUE
0219 12B4 E002      CPX #02        ; MAX 3 DIGITS
0220 12B6 F004      BEQ SETPL2
0221 12B8 EB        INX
0222 12B9 4CAF12    JMP SETPL1
0223 12BC A941      LDA #'A'      ; SET OUTPUT
0224 12BE 8D2105    STA OUTFL6    ; TO SERIAL PRINTER
0225 12C1 A200      LDX #00      ; SET PRINTER
0226 12C3 B502      LDA PARAM,X    ; TO PAGE LENGTH
0227 12C5 203416    JSR OUTALL    ; USING INPUT
0228 12C8 EB        INX          ; VIA KEYBOARD

```

SETTING TALLY-PRINTER

PROTON 650X ASSEMBLER V4.4 PAGE: 0004

```

0229 12C9 E003      CPX #03
0230 12CB D0F6      BNE SETPL3
0231 12CD 203D16     JSR OUTLOW      ; SET OUTPUT TO DISPLAY
0232 12D0 203716     JSR CRLF
0233 12D3 203716     JSR CRLF
0234 12D6 A999      TAL10A LDA #TEXT10      ; DISPLAY
0235 12D8 A215      TAL10A LDX #TEXT10      ; 'SET TOP MARGIN'
0236 12DA 208F13     JSR OUTPUT
0237 12DD 203A16     JSR REDOUT
0238 12E0 C959      CMP #'Y'      ; IF YES, ASK FOR
0239 12E2 F007      BEQ TOPM      ; TOP MARGIN VALUE
0240 12E4 C94E      CMP #'N'      ; IF NOT
0241 12E6 F037      BEQ TAL11     ; SKIP SETTING
0242 12E8 4CD612     JMP TAL10A    ; WRONG CHOICE
0243 12EB A925      TOPM LDA #TEXT23     ; SEND TOP MARGIN
0244 12ED A216      TOPM LDX #TEXT23     ; ESCAPE STRING
0245 12EF 20A313     JSR OUTP3      ; TO PRINTER
0246 12F2 203D16     JSR OUTLOW     ; SET OUTPUT TO DISPLAY
0247 12F5 A92D      LDA #TEXT25     ; DISPLAY ' = '
0248 12F7 A216      LDX #TEXT25
0249 12F9 208F13     JSR OUTPUT
0250 12FC A200      LDX #000        ; TYPE IN
0251 12FE 203A16     SETTM1 JSR REDOUT     ; TOP MARGIN
0252 1301 9502      STA PARAM,X      ; VALUE
0253 1303 E002      CPX #02        ; MAX 3 DIGITS
0254 1305 F004      BEQ SETTM2
0255 1307 E8        INX
0256 1308 4CFE12     JMP SETTM1
0257 130B A941      SETTM2 LDA #'A'      ; SET OUTPUT
0258 130D 8D2105     STA OUTFLG     ; TO SERIAL PRINTER
0259 1310 A200      LDX #000        ; SET TOP
0260 1312 B502      SETTM3 LDA PARAM,X      ; MARGIN USING
0261 1314 203A16     JSR OUTALL     ; INPUT VIA
0262 1317 E8        INX            ; KEYBOARD
0263 1318 E003      CPX #03
0264 131A D0F6      BNE SETTM3
0265 131C 203D16     JSR OUTLOW     ; SET OUTPUT TO DISPLAY
0266 131F 203716     JSR CRLF
0267 1322 203716     JSR CRLF
0268 1325 A9BA      TAL11A LDA #TEXT11     ; DISPLAY
0269 1327 A215      TAL11A LDX #TEXT11     ; 'SET BOTTOM MARGIN'
0270 1329 208F13     JSR OUTPUT
0271 132C 203A16     JSR REDOUT
0272 132F C959      CMP #'Y'      ; IF YES, ASK FOR
0273 1331 F007      BEQ BOTM      ; BOTTOM MARGIN VALUE
0274 1333 C94E      CMP #'N'      ; IF NOT
0275 1335 F037      BEQ TAL12     ; SKIP SETTING
0276 1337 4C2513     JMP TAL11A    ; WRONG CHOICE
0277 133A A929      BOTM LDA #TEXT24     ; SEND BOTTOM MARGIN
0278 133C A216      BOTM LDX #TEXT24     ; ESCAPE STRING
0279 133E 20A313     JSR OUTP3      ; TO PRINTER
0280 1341 203D16     JSR OUTLOW     ; SET OUTPUT TO DISPLAY
0281 1344 A92D      LDA #TEXT25     ; DISPLAY ' = '
0282 1346 A216      LDX #TEXT25
0283 1348 208F13     JSR OUTPUT
0284 134B A200      LDX #000        ; TYPE IN
0285 134D 203A16     SETBM1 JSR REDOUT     ; BOTTOM MARGIN
0286 1350 9502      STA PARAM,X      ; VALUE
0287 1352 E002      CPX #02        ; MAX 3 DIGITS
0288 1354 F004      BEQ SETBM2
0289 1356 E8        INX
0290 1357 4C4D13     JMP SETBM1
0291 135A A941      SETBM2 LDA #'A'      ; SET OUTPUT
0292 135C 8D2105     STA OUTFLG     ; TO SERIAL PRINTER
0293 135F A200      LDX #000        ; SET BOTTOM
0294 1361 B502      SETBM3 LDA PARAM,X      ; MARGIN USING
0295 1363 203A16     JSR OUTALL     ; INPUT VIA
0296 1366 E8        INX            ; KEYBOARD
0297 1367 E003      CPX #03
0298 1369 D0F6      BNE SETBM3
0299 136B 203D16     JSR OUTLOW     ; SET OUTPUT TO DISPLAY
0300 136E 203716     JSR CRLF
0301 1371 203716     JSR CRLF
0302 1374 A9DB      TAL12A LDA #TEXT12     ; DISPLAY
0303 1376 A215      TAL12A LDX #TEXT12     ; 'SETTING OK ? : '
0304 1378 208F13     JSR OUTPUT
0305 137B 203A16     JSR REDOUT

```

SETTING TALLY-PRINTER

PROTON 650X ASSEMBLER V4.4 PAGE: 0005

```

0306 137E C959      CMP #'Y'      ; IF YES,
0307 1380 F00A      BEQ END        ; SETTING COMPLETED
0308 1382 C94E      CMP #'N'      ; IF NO, START
0309 1384 F003      BEQ TAL12B     ; OVER AGAIN
0310 1386 4C7413     JMP TAL12B
0311 1389 4C0011     JMP TALLY
0312 138C 4C3116     END           ; RETURN TO MONITOR
0313 138F           ;
0314 138F           ; ROUTINE TO DISPLAY TEXT-STRINGS POINTED BY A & X
0315 138F           ;
0316 138F 8500      OUTPUT STA HULP
0317 1391 8601      STX HULP+1
0318 1393 A000      LDY #00
0319 1395 B100      OUTP1 LDA (HULP),Y
0320 1397 C903      CMP #03        ; END OF TEXT
0321 1399 F007      BEQ OUTP2
0322 139B 203416     JSR OUTALL     ; OUTPUT CHARACTER
0323 139E C8        INY
0324 139F 4C9513     JMP OUTP1
0325 13A2 60        OUTP2 RTS
0326 13A3           ;
0327 13A3           ; ROUTINE TO SEND ESCAPE STRINGS VIA ACIA TO PRINTER
0328 13A3           ;
0329 13A3 48        OUTP3 PHA
0330 13A4 A941      LDA #'A'        ; SET OUTFLG TO ACIA
0331 13A6 8D2105     STA OUTFLG
0332 13A9 68        PLA
0333 13AA 4C8F13     JMP OUTPUT
0334 13AD           ;
0335 13AD           ; TEXT-STRINGS FOR DISPLAY & PRINTER
0336 13AD           ;
0337 13AD 1A        TEXT0 .BYT $1A,' *****'
0338 13DA 2A2A      .BYT '*****'
0339 13EC 0A        .BYT $0A,$0D,' ***** PRINTER SETTINGS'
0340 1411 204D      .BYT ' MENU FOR TALLY 1612 *****', $0A,$0D
0341 142E 2020      .BYT ' *****'
0342 1454 2A2A      .BYT '*****', $0D,$0A,$0A,$03
0343 1470 0D        TEXT1 .BYT $0D,'- 6 LINES PER INCH :', $03
0344 1491 0D        TEXT2 .BYT $0D,'- 8 LINES PER INCH :', $03
0345 14B2 0D        TEXT3 .BYT $0D,'- 10 CHARS PER INCH :', $03
0346 14D3 0D        TEXT4 .BYT $0D,'- 12 CHARS PER INCH :', $03
0347 14F4 0D        TEXT5 .BYT $0D,'- 16,5 CHARS PER INCH :', $03
0348 1515 0D        TEXT6 .BYT $0D,'- CHAR WIDTH (NORM/DOUBLE) :', $03
0349 1536 0D        TEXT7 .BYT $0D,'- SET LEFT MARGIN :', $03
0350 1557 0D        TEXT8 .BYT $0D,'- SET RIGHT MARGIN :', $03
0351 1578 0D        TEXT9 .BYT $0D,'- SET PAGE LENGTH :', $03
0352 1599 0D        TEXT10 .BYT $0D,'- SET TOP MARGIN :', $03
0353 15BA 0D        TEXT11 .BYT $0D,'- SET BOTTOM MARGIN :', $03
0354 15DB 0A        TEXT12 .BYT $0A,$0A,$0D,' SETTING OK ? :', $03
0355 15FD 1B        TEXT13 .BYT $1B,$37,$30,$03; 6 LPI
0356 1601 1B        TEXT14 .BYT $1B,$37,$31,$03; 8 LPI
0357 1605 1B        TEXT15 .BYT $1B,$37,$33,$03; 10 CPI
0358 1609 1B        TEXT16 .BYT $1B,$37,$34,$03; 12 CPI
0359 160D 1B        TEXT17 .BYT $1B,$37,$35,$03; 16.5 CPI
0360 1611 1B        TEXT18 .BYT $1B,$37,$37,$03; NORMAL WIDTH
0361 1615 1B        TEXT19 .BYT $1B,$37,$38,$03; DOUBLE WIDTH
0362 1619 1B        TEXT20 .BYT $1B,$35,$31,$03; SET LEFT MARGIN
0363 161D 1B        TEXT21 .BYT $1B,$35,$32,$03; SET RIGHT MARGIN
0364 1621 1B        TEXT22 .BYT $1B,$35,$33,$03; SET PAGE LENGTH
0365 1625 1B        TEXT23 .BYT $1B,$35,$34,$03; SET TOP MARGIN
0366 1629 1B        TEXT24 .BYT $1B,$35,$35,$03; SET BOTTOM MARGIN
0367 162D 203D20    TEXT25 .BYT ' = ', $03
0368 1631           ;
0369 1631           ; MONITOR ROUTINES
0370 1631           ;
0371 1631 6C00E0     COMIN JMP ($E000) ; WARM START OF MONITOR
0372 1634 6C08E0     OUTALL JMP ($E008) ; OUTPUT TO ACTIVE OUTPUT DEVICE
0373 1637 6C0EE0     CRLF JMP ($E00E) ; CRLF TO ACTIVE OUTPUT DEVICE
0374 163A 6C1CE0     REDOUT JMP ($E01C) ; READ KEYBOARD AND ECHO CHARACTER
0375 163D 6C56E0     OUTLOW JMP ($E056) ; SET OUTPUT DEVICE TO DISPLAY
0376 1640           ;
0377 1640           ; .END

```

ERRORS: 0000

(0000)

SLAVE THE JUNIOR PROMOTING CY (W&J) 7JUL86

```

0010: 0200 SLAVE ORG $0200 BOOTSTRAP LOADER
0011:
0020: THIS PROGRAM LOADS FILES WITH ID=01 AND FURTHER,
0030: IF FILE IS LOADED THE ID NUMBER WILL BE PRINTED.
0040: FILES MUST BE IN NUMERICAL ORDER.
0050:
0060: USE THIS PROGRAM TO LOAD PATCHES (E.G. MICRO-ADE).
0070: PLACE THIS PROGRAM ON THE BEGINNING OF THE TAPE
0080: AND THE PATCHES DIRECTLY BEHIND.
0090:
0100: AUTHOR : RONALD HERMENS, THE NETHERLANDS
0110:
0120: * CONSTANTS *
0130: 0D 0D NUMBER * $0D THE # OF FILES TO BE READ
0140: * ROUTINES OF PM *
0150: 02 0B RDTAPE * $0B02 READ FILE FROM TAPE
0160: 6A 10 RESALL * $106A WARM START OF PM
0170: F3 11 PRSP * $11F3 PRINT A SPACE ON SCREEN
0180: F8 11 PRBUFS * $11F8 PRINT CURRENT ADDRESS & CONTENTS
0190: 8F 12 PRBYT * $128F CONTENTS ACCU (HEX) TO SCREEN
0200:
0210: * POINTERS USED BY PM *
0220: FA 00 POINTL * $00FA ADDRESS POINTER
0230: FB 00 PINTH * $00FB ADDRESS POINTER
0240: 79 1A ID * $1A79 ID OF FILE TO BE READ
0250: * I/O ADDRESSES *
0260: 83 1A PBDD * $1A83 DIRECTION REGISTER OF PORT B
0270: 82 1A PBD * $1A82 DATA REGISTER OF PORT B
0280: 0200 A9 01 BEGIN LDAIM $01 FIRST ID TO BE READ
0290: 0202 8D 79 1A STA ID IN ID COUNTER
0300: 0205 A2 0D LOOP LDXIM NUMBER CHECK IF ALL
0310: 0207 EC 79 1A CPX ID FILES ARE READ
0320: 020A 30 15 BMI END IF SO: BRANCH
0330:
0340: 020C 20 02 0B JSR RDTAPE READ A FILE
0350: 020F 20 2F 02 JSR SETPIA SET I/O FOR TERMINAL
0360: AND DISPLAY OFF
0370: 0212 AD 79 1A LDA ID PRINT JUST READ ID
0380: 0215 20 8F 12 JSR PRBYT
0390: 0218 20 F3 11 JSR PRSP SOME SPACE
0400: 021E 4C 05 02 INC ID INCREMENT ID COUNTER
0410: JMP LOOP
0420: 0221 A9 00 END LDAIM $00 SET PM ADDRESSPOINTER
0430: 0223 85 FA STA POINTL ON COLD START ADDRESS
0440: 0225 A9 20 LDAIM $20 USER PROGRAM
0450: 0227 85 FB STA PINTH ($2000)
0460: 0229 20 F8 11 JSR PRBUFS PRINT ADDRESSPOINTER
0470: 022C 4C 6A 10 JMP RESALL GOTO PM
0480:
0490: 022F A9 7F SETPIA LDAIM $7F SET OUTPUTS
0500: 0231 8D 83 1A STA PBDD FOR TERMINAL
0510: 0234 AD 82 1A LDA PBD SWITCH
0520: 0237 09 58 ORAIM $58 MOTORS
0530: 0239 8D 82 1A STA PBD OFF
0540: 023C 60 RTS

```

Te koop org. Apple II+, 48 + 16 kb RAM, num. keyb., 80 koll., 2 drives, serial port, monitor, Silentye printer, RS232 port, VIA (Centronics) par. ports, Veel software met zèer veel doc. (100+ disks: Assemblers, Forth, Lisp, Logo, Pascal, etc, etc). Veel boeken (assembly lang., technisch voor Apple) en tijdschr. (Micro, Nibble, Apple Orchard; 100+ nummers). T.e.a.b. M. de Vries, 020-711253 ('s avonds).

STEVE CIARCIA (America) to
JEFF CHATFIELD (Australia)

The KIM-1 was a single board microcomputer that was manufactured by MOS Technology in the late 1970's. It features a 2K monitor ROM supporting a teletype and cassette interface, and had 1K of RAM and a hex keypad on board. It had expansion capability to 64K with external cards, and was an extremely popular device. Many articles for expanding and modifying the KIM appeared in the 1977 and 1978 issues of BYTE and KILOBAUD (later Microcomputing) magazines. MOS Technology Inc., is now a division of Commodore Corp., and no longer makes or supports the KIM. The Synertek SYM-1, and the Rockwell AIM were similar type computers. A 9K BASIC was configured for the KIM by Microsoft Corp., and was distributed by Micro-Z Co., P.O.Box 2426, Rolling Hills, CA 90274 for \$100. This included a cassette tape and complete documentation. The Ohio Scientific OS-650 was an operating system for their 6502 based computers. Like many of the pioneers in microcomputers, Ohio Scientific is no longer in business. My research manager, Harv Weiner, mentioned that he had the KIM 9K BASIC and some other cassette programs that he would be willing to sell for "any reasonable offer". If you are interested, contact him at 67 Scott Drive, South Windsor, CT 06074.

Steve Ciarcia, Ciarcia's Circuit Cellar, P.O.Box 582, Glastonbury, Connecticut 06033.

DID YOU PAY YOUR 1987 SUBSCRIPTION ?

** SCREEN FLICKERING **

By: Andrew Gregory, England.

In issue No. 45 I found an article on how to modify the VDU-card to prevent screen flicker. Well, quite by accident I have found another solution which works on TTL LS or TTL HC versions at 1 or 2 MHz. All you have to do is replace the 74LS30 or 74HC30 (IC8) with a 74L30. This is probably not useful to many people as the device is obsolete. I discovered this when I was short of 74LS30's and opportunistically tried a 74L30 instead.

ASS L

```

0010 :*****
0020 :* T E L E X  PROGRAMMA MCB 1984 *
0030 :* Telex programma voor de JUNIOR computer *
0040 :*****
0050 :
0060 : DOOR: M.C. BREUKINK
0070 : SCHOUTENDREEF 66
0080 : 2542 LN DEN HAAG
0090 :
0100 : DIT PROGRAMMA IS GESCHIKT VOOR EEN JUNIOR MET
0110 : INTERFACE-KAART. HET ZET EEN ASCII-KARAKTER
0120 : IN DE ACCU OM NAAR BAUDOT EN PRINT DEZE OP
0130 : EEN TELEX. DE BAUDOTCODE BESTAAT UIT 1 START-
0140 : BIT, 5 DATA-BITS EN 1 STOP-BIT.
0150 : DE SNELHEID VAN DE MEESTE TELEXEN LIGT ROND
0160 : DE 50 BAUD. IN DE PRAKTIJK KOMT DIT NEER OP
0170 : ONGEVEER 6 KARAKTERS PER SECONDE.
0180 : BIJ HET OMZETTEN VAN ASCII NAAR BAUDOT WORDT
0190 : GEBRUIK GEMAAKT VAN EEN OPZOEKTABEL.
0200 : EEN CARRIAGE RETURN WORDT 2 MAAL AFGEDRUKT
0210 : AANGEZIEN SOMMIGE OUDERE TELEXEN NOGAL WAT
0220 : TIJD NODIG HEBBEN OM DE WAGEN VANAF HET EINDE
0230 : NAAR HET BEGIN VAN EEN REGEL TE BRENGEN.
0240 :
0250 :
0260 : DE SERIELE DATA KOMT VIA PBO NAAR BUITEN. TER-
0270 : OP PB1 EEN SCHAKELAAR WORDT AANGESLOTEN. WEL-
0280 : KE DIENST ALS AAN/UIT SCHAKELAAR VOOR DE TELEX
0290 : (PB1=0 TELEX IS AAN).
0300 : DE TELEX KAN NATUURLIJK NIET ZONDERMEER AAN-
0310 : GESLOTEN WORDEN OP DE VIA 6522. DIT MOET GE-
0320 : BEUREN VIA EEN (GALVANISCH GESCHEIDEN !) IN-
0330 : TERFACE. WELKE D.A. BESTAAT UIT EEN 20-60 mA
0340 : STROOMBRON. SCHEMA'S HIERVAN ZIJN TE VINDEN
0350 : IN DE BEKENDE ELECTRONICA BLADEN.
0360 :
0370 : VIA 6522 ADRESSEN
0380 :
0390 PBD .DE $1800 : ORB/IRB INPUT/OUTPUT REG."B"
0400 PAD .DE $1801 : ORA/IRA INPUT/OUTPUT REG."A"
0410 PBDD .DE $1802 : DDRB DATA DIRECTION REG."B"
0420 PADD .DE $1803 : DDRA DATA DIRECTION REG."A"
0430 T2C/L .DE $1808 : T2 LOW-ORDER LATCHES/COUNTER
0440 T2C/H .DE $1809 : T2 HIGH-ORDER COUNTER
0450 ACR .DE $180B : AUXILARY CONTROL REGISTER
0460 IFR .DE $180D : INTERRUPT FLAG REGISTER
0470 :
0480 : TEMPS IN 6532 PIA-RAM
0490 :
0500 X/TEMP .DE $1A56
0510 BAUDOT .DE $1A57
0520 CYFLET .DE $1A58
0530 X/SAVE .DE $1A60
0540 SACCU .DE $1A62
0550 :
0560 :
0570 : .BA $0D0C
0580 :
0590 :
0600 :
0610 :
0620 :
0630 :
0640 :
0650 :
0660 :
0670 :
0680 :
0690 :
0700 :

```

0D0C- 8E 60 1A 0590 TELEX STX X/SAVE : RED X-REGISTER
0D0F- 20 1D 0D 0600 JSR TPRINT : PRINT KARAKTER
0D12- AD 62 1A 0610 LDA SACCU :
0D15- C9 0D 0620 CMP #\$0D : CARRIAGE RETURN ?
0D17- D0 03 0630 BNE RETURN :
0D19- 20 1D 0D 0640 JSR TPRINT : JA. NOGMAALS PRINTEN
0D1C- 60 0650 RETURN RTS :
0660 :
0670 :
0680 TPRINT AND #\$7F :
0D1D- 29 7F 0680 STA SACCU : RED ACCU
0D1F- 8D 62 1A 0690 LDA #\$0FD : PBO=UITGANG PB1=INGANG
0D22- A9 FD 0700

0D24- 8D 02 18	0710	STA PBDD	
0D27- A9 01	0720	LDA #01	: PB0=1 (RUSTTOESTAND)
0D29- 8D 00 18	0730	STA PBD	
0D2C- A9 02	0740	LDA #2	
0D2E- 2C 00 18	0750	BIT PBD	: TELEX AAN ?
0D31- F0 01	0760	BEQ OK	
0D33- 60	0770	RTS	
0D34- AE 62 1A	0780 OK	LDX SACCU	: X=ASCII
0D37- E0 60	0790	CPX #60	: KLEINE LETTER ?
0D39- 90 04	0800	BCC BIG	
0D3B- 8A	0810 SMALL	TXA	
0D3C- 29 DF	0820	AND #0DF	: BIT 6=0
0D3E- AA	0830	TAX	
0D3F- 8E 56 1A	0840 BIG	STX X/TEMP	
0D42- E0 41	0850	CPX #41	: LETTER ?
0D44- 30 17	0860	BMI CIJFER	: ZONEE. TELEX OP CIJFERS
0D46- E0 5B	0870	CPX #5B	
0D48- 10 13	0880	BPL CIJFER	: ZONEE. TELEX OP CIJFERS
0D4A- AD 58 1A	0890	LDA CYFLET	
0D4D- C9 FE	0900	CMP #0FE	: TELEX OP LETTERS ?
0D4F- F0 08	0910	BEQ OUT	
0D51- A9 FE	0920	LDA #0FE	: CODE VOOR LETTERS
0D53- 8D 58 1A	0930	STA CYFLET	
0D56- 20 6C OD	0940 DOUT	JSR OUTPUT	: CODE --> TELEX
0D59- 20 69 OD	0950 OUT	JSR OUTCH	: ASCII-BAUDOT-TELEX
0D5C- 60	0960	RTS	
0D5D- AD 58 1A	0970 CIJFER	LDA CYFLET	
0D60- C9 FE	0980	CMP #0FE	: TELEX OP CIJFERS ?
0D62- F0 FE	0990	BEQ OUT	: JA. PRINT KARAKTER
0D64- A9 FE	1000	LDA #0FE	: NEE. ZET TELEX OP CIJFERS
0D66- 4C 56 OD	1010	JMP DOUT	
	1020 :		
0D69- BD 9E OD	1030 OUTCH	LDA TABEL, X	: ASCII-->TABEL-->BAUDOT
0D6C- 8D 57 1A	1040 OUTPUT	STA BAUDOT	
0D6F- A2 07	1050	LDX #7	: 1 START 5 DATA 1 STOP
0D71- AD 57 1A	1060 OUTBIT	LDA BAUDOT	
0D74- 8D 00 18	1070	STA PBD	: B0-->PB0
0D77- 20 84 OD	1080	JSR DELAY	: DELAY 20 MS (50 BAUD)
0D7A- 6E 57 1A	1090	ROR BAUDOT	: VOLGENDE BIT
0D7D- CA	1100	DEX	
0D7E- D0 F1	1110	BNE OUTBIT	
0D80- AE 56 1A	1120	LDX X/TEMP	: RESTORE X-REGISTER
0D83- 60	1130	RTS	
	1140 :		
0D84- A9 00	1150 DELAY	LDA #00	
0D86- 8D 0B 18	1160	STA ACR	: RESET ACR REGISTER
0D89- A9 20	1170	LDA #20	
0D8B- 8D 08 18	1180	STA T2C/L	: LOW ORDER
0D8E- A9 4E	1190	LDA #4E	
0D90- 8D 09 18	1200	STA T2C/H	: HIGH ORDER. START TIMER
0D93- A9 20	1210	LDA #20	
0D95- 2C OD 18	1220 TEST	BIT IFR	: TIME OUT ?
0D98- F0 FB	1230	BEQ TEST	
0D9A- AD 08 18	1240	LDA T2C/L	
0D9D- 60	1250	RTS	
	1260 :		
	1270 :		
	1280 :		
0D9E- FF FF FF	1290 TABEL	.BY \$FF \$FF \$FF \$FF \$FF \$FF \$FF \$D6	
0DA1- FF FF FF			
0DA4- FF D6			
0DA6- FF FF C4	1300	.BY \$FF \$FF \$C4 \$FF \$FF \$D0 \$FF \$FF	
0DA9- FF FF D0			
0DAC- FF FF			
0DAE- C8 FF FF	1310	.BY \$C8 \$FF \$FF \$FF \$FF \$FF \$FF \$FF	
0DB1- FF FF FF			
0DB4- FF FF			
0DB6- FF FF FF	1320	.BY \$FF \$FF \$FF \$FF \$FF \$FF \$FF \$FF	
0DB9- FF FF FF			
0DBC- FF FF			
0DBE- C8 EE CA	1330	.BY \$C8 \$EE \$CA \$E8 \$CA \$FA \$E8 \$CA	
0DC1- E8 CA FA			
0DC4- E8 CA			

```

ODC6- DE E4 F8 1340 .BY $DE $E4 $F8 $E2 $D8 $C6 $F8 $FA
ODC9- E2 D8 C6
ODCC- F8 FA
ODCE- EC EE E6 1350 .BY $EC $EE $E6 $C2 $D4 $E0 $EA $CE
ODD1- C2 D4 E0
ODD4- EA CE
ODD6- CC F0 DC 1360 .BY $CC $F0 $DC $DC $DE $FC $E4 $F2
ODD9- DC DE FC
ODDC- E4 F2
ODDE- E8 C6 F2 1370 .BY $E8 $C6 $F2 $DC $D2 $C2 $DA $F4
ODE1- DC D2 C2
ODE4- DA F4
ODE6- E8 CC D6 1380 .BY $E8 $CC $D6 $DE $E4 $F8 $D8 $F0
ODE9- DE E4 F8
ODEC- D8 F0
ODEE- EC EE D4 1390 .BY $EC $EE $D4 $CA $E0 $CE $FC
ODF1- CA E0 CE
ODF4- FC

1400 :
1410 .EN

```

SOURCE FILE: SCREEN DUMP/BP

```

0000: 1 ; *** APPLE'S TEXT SCREEN COPIER ***
0000: 2 ;
0000: 3 ; AFDruk VAN HET SCHERM
0000: 4 ; MET PRINTER MATE PARALLEL INTERFACE IN SLOT#1
0000: 5 ; ZONDER GEBRUIK VAN PR#1
0000: 6 ; BLANKO REGELS KUNNEN WORDEN OVERGESLAGEN
0000: 7 ;
0000: 8 ; DOOR HANS BOSCH
0000: 9 ; REELAAN 35
0000: 10 ; 7522LS ENSCHEDE
0000: 11 ;
0000: 12 ; GEHEUGEN RUIMTE
0028: 13 BASL EQU $28 BASIS ADRES
00D7: 14 VLAG EQU $D7 SPATIE MARKER
FBC1: 15 BASCALC EQU $FBC1 BEREKEN BASIS ADRES VOOR LINE# IN A
0000: 16 ;
----- NEXT OBJECT FILE NAME IS SCREEN DUMP/BP.OBJO
0300: 17 ORG $300
0300: 18 ;
0300:A2 00 19 START LDX #0 EERSTE REGEL
0302:A0 00 20 NXTLIN LDY #0 EERSTE KARAKTER
0304:84 D7 21 STY VLAG (RE)SET VLAG
0306:8A 22 TXA
0307:20 C1 FB 23 JSR BASCALC BEREKEN BASIS ADRES
030A:B1 28 24 GCHAR LDA (BASL),Y HAAL KARAKTER VAN SCHERM
030C:C9 A0 25 CMP #$A0 IS DIT EEN SPATIE?
030E:F0 02 26 BEQ PRINT
0310:85 D7 27 STA VLAG NEE, ASCII CODE IN VLAG
0312:18 28 PRINT CLC
0313:48 29 PHA
0314:8D 00 CC 30 STA $CC00 PRINTER BUFFER
0317:AD 98 CO 31 BUSY LDA $C098
031A:30 FB 32 BMI BUSY
031C:AD 90 CO 33 LDA $C090
031F:AD 98 CO 34 LDA $C098
0322:68 35 PLA
0323:C9 0D 36 CMP #$0D CR?
0325:F0 12 37 BEQ TRYNXT
0327:2A 38 ROL A EINDE VAN REGEL ($0A)?
0328:90 0B 39 BCC CR
032A:C8 40 INY VOLGENDE KARAKTER
032B:C0 28 41 CPY #$28 DEZE REGEL KLAAR?
032D:30 DB 42 BMI GCHAR
032F:A9 0A 43 LDA #$0A
0331:26 D7 44 ROL VLAG JA, STUUR LF
0333:B0 DD 45 BCS PRINT ALS VLAG EEN ASCII CODE BEVAT
0335:A9 0D 46 CR LDA #$0D NU NOG EEN CR
0337:10 D9 47 BPL PRINT
0339:E8 48 TRYNXT INX VOLGENDE REGEL
033A:E0 18 49 CPX #$18 ALLE REGELS GEDAAN?
033C:30 C4 50 BMI NXTLIN
033E:60 51 RTS KLAAR

```

*** SUCCESSFUL ASSEMBLY: NO ERRORS

GAME

High-Low game

The object of this game is to guess a number between 1 and 9999. The computer will tell you if your guess is too high or too low. If you want to give up, type in 0 for your guess or just press RETURN.

GOOD LUCK !!!

The game starts if the <ESC> key is hit.....

Your guess? 5000	Too low
Your guess? 7000	Too low
Your guess? 9000	Too high
Your guess? 8000	Too low
Your guess? 8500	Too high
Your guess? 8300	Too low
Your guess? 8400	Too low
Your guess? 8450	Too low
Your guess? 8470	Too high
Your guess? 8460	Too high
Your guess? 8457	Too low
Your guess? 8458	Too low
Your guess? 8459	You found it !!

You needed 13 turns.

Another game (Y/N) ? Y

```

SCR # 40
0 ( HIGH-LOW GAME  DECLARATIONS ) 1 .
1 ( Simple game written by A.G.Megens 861024 )
2 0 VARIABLE NUM 0 VARIABLE INP
3 0 VARIABLE RND 0 VARIABLE TS
4 0 VARIABLE LEN 0 VARIABLE TURNS
5 HERE RND !
6 : HIGH 10 LEN @ - SPACES ." Too high" ;
7 : LOW 10 LEN @ - SPACES ." Too low" ;
8 : OK 10 LEN @ - SPACES ." You found it !" 7 EMIT CR CR
9 ." You needed " TURNS @ ." turns." CR ;
10 : QUEST CR ." Your guess? " 7 EMIT ;
11 : ANOTHER CR CR ." Another game (Y/N) ? " 7 EMIT
12 BEGIN KEY TS ! TS @ 78 = TS @ 89 = OR UNTIL TS @ EMIT ;
13 : RANDOM RND @ 31421 * 6927 + DUP RND ! ;
14 : CHOOSE ( U1 ---U2) RANDOM U* SWAP DROP ;
15 -->

SCR # 41
0 ( HIGH-LOW GAME  INPUT ) 2 .
1 ( This input routine accepts digits 0-9, RUBOUT and RETURN )
2 ( Call INPUT, result is stored in variable INP, length is )
3 ( stored in LEN, variable TS is last typed character. )
4 ( Note: FORTH works with 16 bits numbers, so maximum number )
5 ( you may type in is 32767, else it will be negative. )
6
7 : INPUT 0 INP ! 0 LEN !
8 BEGIN KEY TS !
9 TS @ 95 = IF LEN @ 1 < IF 7 EMIT 0 TS !
10 ELSE 95 EMIT -1 LEN +!
11 INP @ 10 / INP ! THEN
12 THEN
13 TS @ 47 > TS @ 58 < AND IF TS @ EMIT -48 TS +!
14 1 LEN +! INP @ 10 * TS @ + INP ! THEN
15 TS @ 13 = UNTIL ; -->

```

```

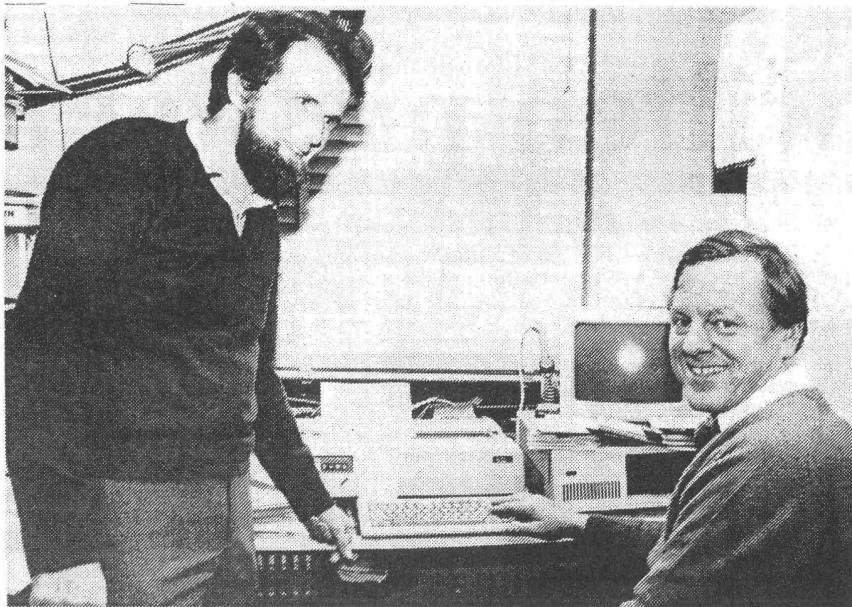
SCR # 42
0 ( HIGH-LOW GAME      INFO ) 3 .
1 : INFO CR CR
2   10 SPACES ." High-Low game" CR
3   9 SPACES ." -----" CR CR CR
4   ." The object of this game is to guess a number " CR
5   ." between 1 and 9999. The computer will tell you " CR
6   ." if your guess is too high or too low." CR
7   ." If you want to give up, type in 0 for your guess" CR
8   ." or just press RETURN." CR CR
9   ." GOOD LUCK !!!" CR CR
10  ." The game starts if the <ESC> key is hit....."
11  7 EMIT ;
12
13 : STUPID CR ." Maximum is 9999, DUMMY!!" 7 EMIT ;
14 : GIVEUP CR CR ." The secret number was " NUM @ . CR ;
15 -->

SCR # 43
0 ( HIGH-LOW GAME      MAIN LOOP ) 4 .
1 : GAME BEGIN
2   0 TURNS ! INFO
3   BEGIN RANDOM DROP KEY 27 = UNTIL CR CR
4   BEGIN 9999 CHOOSE NUM ! NUM @ 0 > UNTIL
5   BEGIN
6     QUEST INPUT 1 TURNS +!
7     LEN @ 4 > IF STUPID 10000 INP ! THEN
8     INP @ 0 > IF INP @ NUM @ < IF LOW THEN
9     INP @ NUM @ > IF HIGH THEN
10    THEN INP @ NUM @ = INP @ 0 = OR
11    UNTIL INP @ 0 = IF GIVEUP ELSE OK THEN
12    ANOTHER TS @ 78 =
13    UNTIL CR ;
14 ;S
15
OK

```

HET VRIJE VOLK

DINSDAG 11 NOVEMBER 1986



De heren van Opbroek en Van Pelt in het zenuwcentrum van hun computerclub.
(Foto Persbureau Jota)

COLUMNS PRINT

Page: 0001

```

-----
I          COLUMNS PRINT          I
I                                  I
I  for printing in two columns (f.ex. I
I  from Wordprocessor or ASS114). The I
I  printer-initialising-routine must: I
I  1. send condensed pitch, set left I
I  margin (04), set horizontal tabu- I
I  lator (70). 2. change printer out- I
I  vector to this routine, reset buf- I
I  ferpointer, set flag=00.           I
I  The processor must send $0C at the I
I  end of each page, and finish the I
I  last page with lf's and send $0C, I
I  $03 at the end of text.           I
I  Leif Rasmussen, Parkvej 1 I
I  4534 Hørve Danmark              I
-----

229B          ORG  $229B
              TTL  COLUMNS PRINT

2363          AHOLD EQU  $2363  holds character
F7B0          CENTRON EQU $F7B0 send chr. to printer
00DD          FLAG  EQU  $DD    =00 if save,=ff if print
00DE          MEPTR EQU  $DE    pointer to buffer
C800          ME    EQU  $C800  buffer for left column

-----
I          MAIN ENTRANCE          I
I  (character in A)              I
-----

229B A4 DD    LDY  FLAG  =00 when saving
229D F0 03    BEQ  LCTM  left column to memory.
229F C8       INV          flag=ff when printing
22A0 F0 35    BEQ  RCTP  columns to printer.

-----
I          LEFT COLUMN TO MEMORY  I
-----

22A2 91 DE    LCTM  STAIY MEPTR  save char. in buffer
22A4 C9 0C    CMPIM $0C  end of columns 1,3,5,,?
22A6 F0 07    BEQ  PTSP

-----
I          INCREMENT MEMORY-POINTER I
-----

22A8 E6 DE    INCRPTR INC  MEPTR
22AA D0 02    BNE  RETUR
22AC E6 DF    INC  MEPTR  +01
22AE 60       RETUR  RTS

-----
I          PREPARE TO START PRINTING I
-----

22AF C6 DD    PTSP  DEC  FLAG  to $FF = printing
22B1 20 EB 22 JSR  RESETME reset memory-pointer
22B4 D0 06    BNE  LCTP  print first left line

-----
I          PREPARE FOR NEXT LEFT LINE I
-----

22B6 20 AB 22 PFNLL JSR  INCRPTR skip 0D,0A
22B9 20 AB 22 JSR  INCRPTR

-----
I          LEFT COLUMN TO PRINTER  I
-----

22BC B1 DE    LCTP  LDAIY MEPTR  get char. from buffer

```

COLUMNS PRINT

Page: 0002

```

22BE C9 0D    CMPIM $0D  end of line ?
22C0 F0 0C    BEQ  PFRL
22C2 C9 0C    CMPIM $0C  end of left column ?
22C4 F0 EB    BEQ  RETUR
22C6 20 D0 22 JSR  CENTRO  print character
22C9 20 AB 22 JSR  INCRPTR  increment pointer
22CC D0 EE    BNE  LCTP  get next chr.

-----
I          PREPARE FOR RIGHT LINE  I
-----

22CE A9 09    PFRL  LDAIM $09  send tab. (70)

-----
I          SEND CHARACTER TO PRINTER I
-----

22D0 BD 63 23 CENTRO STA  AHOLD  send to printer
22D3 20 B0 F7 JSR  CENTRON
22D6 60       RTS

-----
I          RIGHT COLUMN TO PRINTER  I
-----

22D7 C9 03    RCTP  CMPIM $03  end of text ?
22D9 F0 17    BEQ  PROLC
22DB 48       PHA
22DC 20 D0 22 JSR  CENTRO
22DF 68       PLA
22E0 C9 0A    CMPIM $0A  new line ?
22E2 F0 D2    BEQ  PFNLL
22E4 C9 0C    CMPIM $0C  new page ?
22E6 F0 01    BEQ  PFNP
22E8 60       RTS  return to processor

-----
I          PREPARE FOR NEXT PAGE  I
-----

22E9 E6 DD    PFNP  INC  FLAG  to $00 = save left column

-----
I          RESET MEMORY-POINTER  I
-----

22EB A9 C8    RESETME LDAIM ME  /256
22ED 84 DE    STY  MEPTR  (Y allways = 0)
22EF 85 DF    STA  MEPTR  +01
22F1 60       RTS

-----
I          PRINT REST OF LEFT COLUMN I
-----

22F2 B1 DE    PROLC  LDAIY MEPTR
22F4 C9 0C    CMPIM $0C  if end of text, print
22F6 F0 D8    BEQ  CENTRO  form feed and return.
22F8 20 D0 22 JSR  CENTRO
22FB 20 AB 22 JSR  INCRPTR
22FE D0 F2    BNE  PROLC  allways, get next char.

```

>>> Error in 0000 statement(s)

>>> Op-Code: \$B000 - \$B064 / \$229B - \$22FF / 0101 Bytes / 01 Pag.

>>> Assembled by ASS114 / 3.4

```

10 ;Hardcopy routine for Octopus
20 ;By: Coen Boltjes 015-136812 Vidibus 400029830
30 ;For Junior: Use the (addresses)
40 ;The routine PRCHA is printer dependent.
50 ;The first prints only ASCII characters, and
60 ;can be used with all ASCII printers. The
70 ;second is an example of a routine which also
80 ;can print the VIDEOTEX block graphics.
90
100 ;
110 ;VARIABLES
120 XHOLD=$235F
130 YHOLD=$2361
140 AHOLD=$2363
150 FLN=$E7CD
160 RAMBEG=$E7DB
170 CHAPLN=$E7DD
180 LPSCR=CHAPLN+1
190 RAMPTR=$6C
200 CENTRO=$F780
210 B000
220 ;
230 B000 18 HRDCO CLC
240 B001 ADCDE7 LDA FLN
250 B004 6DDBE7 ADC RAMBEG
260 B007 856C STA RAMPTR
270 B009 ADCDEE7 LDA FLN+1
280 B00C 6DDCE7 ADC RAMBEG+1
290 B00F 856D STA RAMPTR+1
300 B011 AEDEE7 LDX LPSCR
310 B014 2047B0 HRDCO1 JSR CRLF
320 B017 201EBO JSR PRTLN
330 B01A CA DEX
340 B01B DOF7 BNE HRDCO1
350 B01D 60 RTS
360 ;
370 B01E 8E5F23 PRTLN STX XHOLD
380 B021 ACDDDE7 LDY CHAPLN
390 B024 8C6123 STY YHOLD
400 B027 A000 PRTLN1 LDY #000
410 B029 B16C LDA (RAMPTR),Y
420 B02B 205EBO JSR PRCHA
430 B02E E66C INC RAMPTR
440 B030 DOOC BNE PRTLN2
450 B032 E66D INC RAMPTR+1
460 B034 A56D LDA RAMPTR+1
470 B036 2907 AND #07
480 B038 18 CLC
490 B039 6DDCE7 ADC RAMBEG+1
500 B03C 856D STA RAMPTR+1
510 B03E CE6123 PRTLN2 DEC YHOLD
520 B041 DOE4 BNE PRTLN1
530 B043 AE5F23 LDX XHOLD
540 B046 60 RTS
550 ;
560 B047 8E5F23 CRLF STX XHOLD
570 B04A A90A LDA #0A
580 B04C 8D6323 STA AHOLD
590 B04F 20B0F7 JSR CENTRO
600 B052 A90D LDA #0D
610 B054 8D6323 STA AHOLD
620 B057 20B0F7 JSR CENTRO
630 B05A AE5F23 LDX XHOLD
640 B05D 60 RTS
650 ;
660 ;Only prints codes $20-$7F. Others as a space.
670 B05E 3004 PRCHA BMI PRCHA1
680 B060 C920 CMP #20
690 B062 1002 BPL PRCHA2
700 B064 A920 PRCHA1 LDA #20
710 B066 8D6323 PRCHA2 STA AHOLD
720 B069 20B0F7 JSR CENTRO
730 B06C 60 RTS
740 ;
750 ;Prints all codes using EPSON MX-80 format
760 ;and VIDEOTEXT-modifications of issue 47
770 ;May be adjusted for other configurations
780 ;
790 B06D 10F7 PRCHA BPL PRCHA2
800 B06F 297F AND #7F
810 B071 C920 CMP #20
820 B073 DOE4 BNE PRCHA1
830 B075 A9DF LDA #DF
840 B077 D012 BNE PRCHA3
850 B079 1010 PRCHA1 BPL PRCHA3
860 B07B 49FF EOR #FF
870 B07D 291F AND #1F
880 B07F 09C0 ORA #11000000
890 B081 D008 BNE PRCHA3
900 B083 C920 PRCHA2 CMP #20
910 B085 1004 BPL PRCHA3
920 B087 291F AND #1F
930 B089 09A0 ORA #10100000
940 B08B 8D6323 PRCHA3 STA AHOLD
950 B08E 20B0F7 JSR CENTRO
960 B091 60 RTS

```

```

100 REM File merge & -split program version 2.1 (860904)
110 REM For Octopus/DOS-Junior
120 REM By:C.J. Boltjes Tel. 015-136812
130 REM Vidibus 400029830
140 REM Why a new File-Merge program?
150 REM 1)This program don't use an objectcode file,
160 REM so it can be run as a standalone program.
170 REM 2)This program list only the first file in
180 REM memory. So larger files can be merged, and
190 REM the execution is faster.
200 REM 3)A file split option is implemented too.
210 REM 4)By using DOS-Commands instead of POKE's
220 REM the program is easier to understand, and
230 REM to adapt.
240 REM When the program is used on a Junior "GO F71D"
250 REM in line 630 must be changed into "GO FE1B".
260 REM The "ME D800,...." must be changed into a value
270 REM where RAM is present.
280 REM When using the SPLIT option you have to give the
290 REM first and last line seperated by a "-"
300 DISK!"ME D800,D800":REM SET PRINT POINTER
310 DI$="DISK!"+CHR$(34)
320 INPUT "S)plit or M)erge ";A$
330 MO=0
340 IF (A$="S"OR A$="s") THEN MO=1
350 IF (A$="M"OR A$="m") THEN MO=2
360 IF MO=0 THEN PRINT CHR$(7):GOTO 320
370 GOSUB 510: REM LOAD FIRST FILE

```

```

380 PRINT#5, "LIST#5";
390 IF MO=2 THEN GOTO 420
400 GOSUB 560: REM ENTER DELIMITERS
410 GOTO 440
420 PRINT#5, :REM CLOSE LINE
430 GOSUB 600:GOSUB 510:REM SECOND FILE
440 PRINT#5, "PRINT#5, ";CHR$(34);"DISK!";
450 PRINT#5, CHR$(34);CHR$(34);";
460 PRINT#5, CHR$(34);"IO 01,01";CHR$(34)
470 PRINT#5, DI$;"ME 9000,9000";CHR$(34)
480 DISK!"ME D800,9000":REM SET START
490 DISK!"IO 10,01"
500 END
510 INPUT"Which drive ";D$
520 INPUT"Which file ";F$
530 PRINT#5, DI$;"SE ";D$
540 PRINT#5, DI$;"LO ";F$
550 RETURN
560 INPUT "List FROM-TO ";A$
570 PRINT#5, ", ";A$
580 PRINT#5, "NEW"
590 RETURN
600 PRINT#5, :PRINT#5, "REM ";CHR$(34);
610 PRINT#5, "*** Place next disk, and press ";
620 PRINT#5, "any key to continue ***"
630 PRINT#5, "DISK!";CHR$(34);"GO F71D"
640 RETURN

```

DE 6502 KENNER

SHOPPING AROUND

Anyone of us who is building and expanding his own computersystem will face a moment that he (or she ?) has to buy or order the necessary components. As we here in Holland will always try to pay less and get more I spend some time to check whether my local shop was the best deal for a larger project. As example I used the EC 65K card together with the new I/O card as published by Elektor in special 4 and 5. For a good reference I selected two shops each in Holland and Germany and one shop in England and Belgium.

In the first column you can find the part list as published by Elektor and in the other columns the 6 different shops.

In the Netherlands NL1:TIMTRONIX, tel 050-140937
NL2:MEEX IT/GDRIS, tel 015-130489

In Germany D1 :RATEV ELEKTRONIK, tel 02102-42051/52
D2 :SIMONS ELEKTRONIK, tel 02272-81619

In Great Britain GB1:TECHNOMATIC Ltd, tel 01-2081177

In Belgium B1 :TRIAC , TEL 03-2382352/53

All prices in local currency

CPU CARD		NL1	NL2	D1	D2	GB1	B1
R1,R2	:470	0.20	0.20	0.18	0.20	n.a.	2.4
R3,R9,R10,R16	:3K3	0.40	0.40	0.36	0.40	n.a.	4.8
R4,R8,R11...R14,R17:2K2		0.70	0.70	0.63	0.70	n.a.	8.4
R5,R6,R7	:1M	0.30	0.30	0.27	0.30	n.a.	3.6
R15	:1K	0.10	0.10	0.10	0.10	n.a.	1.2
C1	:12 PF Ker.	0.15	0.30	0.25	0.08	n.a.	2.5
C2,C4,C5	:330 N	1.95	1.80	1.35	1.47	n.a.	30.0
C3,C6...C15	:100 N	3.85	5.50	4.95	3.19	n.a.	110.0
C16	:10u/12V Tantal	0.70	0.50	1.65	0.39	n.a.	20.0
D1...D3	:HSCH 1001(hp)	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
T1	:2N2222	0.90	1.25	0.90	0.35	0.30	19.0
IC1	:74LS04	0.95	0.95	0.90	0.59	0.24	25.0
IC2	:74LS93	1.75	1.95	1.05	1.24	0.54	42.0
IC3	:74F74	n.a.	9.50	1.85	1.78	0.70	65.0
IC4,IC7	:74F00	n.a.	2.40	3.50	2.56	1.00	112.0
IC5	:74F04	n.a.	1.20	1.75	1.28	0.50	56.0
IC6	:74LS125	1.50	1.85	1.15	1.18	0.50	39.0
IC8...IC10	:74F245	n.a.	12.00	23.70	3.54	14.25	1785.0
IC11	:ME555	1.10	1.10	1.10	0.78	0.22	30.0
IC12	:74LS74	1.35	1.40	1.00	0.88	0.35	32.0
IC13	:74LS00	0.95	0.75	0.65	0.55	0.24	25.0
IC14	:74S30	1.65	2.50	n.a.	1.28	0.50	38.0
IC15	:74F373	n.a.	3.00	6.45	3.68	4.00	295.0
IC16	:74F138	n.a.	2.00	3.55	1.98	1.80	132.0
IC17	:74S260	1.85	2.00	2.40	n.a.	1.00	95.0
IC18	:6558C16P2	n.a.	79.50	n.a.	n.a.	n.a.	n.a.
IC19	:2716 EPROM	11.95	14.95	8.90	9.95	3.50	365.0
X1	:8 MHz	4.95	6.95	2.10	2.28	1.50	150.0
64 pin male din 41612		5.95	4.95	2.15	2.98	2.25	90.0
Connectorstrip 1row,12pin		n.a.	3.95	2.50	3.48	1.10	35.0
Connectorstrip 2row,30pin		3.95	4.50	5.00	6.98	1.95	180.0
10 shortcutconnector		2.30	8.00	2.70	2.20	n.a.	n.a.



PAPERWARE SERVICE (ENGLISH VERSION) LIST OF PRICES

All prices including forwarding charges, etc.
All prices only for European (C.E.P.T.)-countries.

NEW :
MICRO-WARE Assembler/Disassembler/Editor for 6502
Manual for Elektor's OCTOPUS/EC65-computer.
Documented by Marc Lachaert, Belgium.
Send cheque of Hfl. 49,50 to W.L.v.Pelt (eurocheque 40,00)

NEW:
MICRO-WARE Assembler/Disassembler/Editor for 6502
Complete Source-listing for Elektor's Octopus/EC65-comp.
Documented by Marc Lachaert, Belgium
Send cheque of Hfl. 74,50 to W.L.v.Pelt (eurocheque 65,00)

EXTENSIONS ON OS-65D V3.3 FOR ELEKTOR'S JUNIOR-computer.
Gert Klein, The Netherlands.
Dutch version published in DE 6502 KENNER nr.39, Aug.'85.
With the new DOS-commands :
DI to supply the directory of a diskette, SD to replace the old DI command in OS-65D, FO to format a diskette and to write an empty directory to track 12, CR to write a filename into the directory with trackrange to match, DE to delete a filename from the directory, CH to change filename1 into filename2, NU to supply the number of tracks by the Basic of assembly sourcefile. With four commands for further extensions: FI to load the FORTH interpreter/compiler, WF to warmstart the FORTH, MA to load the macro assembler by C. Moser, WS to warmstart the Moser-assembler.
Complete English assembly source listing with introduction for users of JUNIOR, but easy to adapt on your OCTOPUS.
Send cheque of Hfl. 17,50 to W.L.v.Pelt (eurocheque 8,00)

MICRO-WARE Assembler/Editor with bank-switching
Complete source-listing (Disassembler not included)
Author: Fernando Lopes, Portugal.
System: Elektor's JUNIOR-computer with VDU and OS65-D DOS.
With new commands: PUT FILE, PUT TT, LOAD FILE, LOAD TT, toggle centronics flag, return to kernel, and modified error routine, and new subroutines DECBPF, INCBPF, SETPM-SETSC-SETPF, PRDAT to read data and time from the Real Time Clock and to print it in the header lines, SETBK0-SWAPDN-SAVBNK-RSTBNK to manage the bankswitching, LINE NUMBERS.
Refer : DE 6502 KENNER, May 1986, p.44-47
Hardware article "Bank Switching for the JUNIOR-computer" by Fernando Lopes, Portugal.
Send cheque of Hfl. 45,00 to W.L.v.Pelt (eurocheque 35,50)

DATBAS. A database-program written in Basic for Elektor's JUNIOR-computer with VDU-card and OHIO Scientific OS-65D V3.3 disk operating system.
Jan van Heuven, The Netherlands. Modified by Fernando Lopes, Portugal.
Send cheque of Hfl. 30,00 to W.L.v.Pelt (eurocheque 20,50)

And for the I/O card the following:

R1...R8	:2K2	0.80	0.80	0.72	0.80	n.a.	9.6
R9	:330K	0.10	0.10	0.09	0.10	n.a.	1.2
R10	:20M	0.10	0.10	0.09	0.10	n.a.	1.2
R11...R14,R17,R18	:100K	0.80	0.80	0.54	0.80	n.a.	7.2
R15 trim	:10K	0.60	0.80	0.65	0.55	n.a.	12.0
R16	:30K	0.10	0.15	0.12	0.10	n.a.	1.2
R19...R21	:2K2	0.30	0.30	0.27	0.30	n.a.	3.6

C1	:22pf ker.	0.15	0.30	0.25	0.08	n.a.	2.5
C2	:10...40 pf	0.90	1.25	1.00	1.00	n.a.	23.0
C3,C5...C7,C9,C10	:100 M	2.10	3.00	2.70	2.94	n.a.	60.0
C4,C8,C11,C12	:10u/16v tant.	2.80	2.00	6.60	1.56	n.a.	80.0

D1...D3	:1N4148	0.15	0.12	0.09	0.10	0.12	30.0
D4	:AA136	n.a.	0.50	0.45	n.a.	n.a.	n.a.
D5	:HSC1001(hp)	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
T1...T3	:BC130C	n.a.	1.80	1.29	0.57	n.a.	n.a.
IC1	:74F245	n.a.	4.00	7.90	3.98	4.75	595.0
IC2	:74LS02	0.95	0.95	0.65	0.55	0.24	25.0
IC3	:74LS688	6.95	8.00	5.55	4.98	3.50	199.0
IC4,IC6	:74LS00	1.90	1.80	1.30	1.10	0.24	25.0
IC5	:74LS27	1.05	1.10	0.65	0.64	0.24	25.0
IC7	:MC146818	26.60	27.95	24.80	n.a.	n.a.	495.0
IC8	:6551	16.30	15.95	13.40	13.50	6.00	495.0
IC9	:1488	2.75	1.45	2.17	1.68	0.60	50.0
IC10	:1489	2.75	1.45	2.17	1.68	0.60	50.0
IC11,IC12	:6521 or 6821	14.30	19.00	9.00	8.68	5.60	290.0

X1	:32.768 KHz	n.a.	1.70	4.95	1.75	1.00	150.0
X2	:1.8432 MHz	5.95	7.15	2.10	11.50	2.25	150.0
64 pin male din 41612		5.95	4.95	2.15	2.98	2.75	90.0
connectorstrip 2 row 72 pin		7.90	9.00	5.00	3.92	3.90	200.0
shortcutconnector		0.25	0.80	0.27	0.22	n.a.	n.a.
Ni/Cd battery 3.6V/100 mAh		n.a.	29.30	n.a.	n.a.	n.a.	230.0

Not all the shops could offer me the complete list special the Skotky diode is not available. The 65SC816 can be obtained at the GTE resalesoffice in Holland or Germany.

As You can see I left the price of the two componentplates out of this list because You can order them directly at Elektor in Your own country. Also the 65SC816 is not yet available in most of the stores, if You are interested best to call GTE office in Your own country.

The shops selected where choisen because of the advertisements in Elektor wich made the writing of the story easier. As everybody has realized during the last years the prices of electronic spareparts are fluctuating very strongly. Date of the a.m. prices is July 1986.

All mentioned prices are included VAT. For ordering parts from a different country subtract VAT and add shipping and custom taxes.

Will Cuijpers

CENTRONICS PRINTER INTERFACE DEVICE 4 OR 5 ON COMMODORE 64
Ruud Uphoff, The Netherlands. Machinecode.
Send cheque of Hfl. 15,00 to W.L.v.Pelt (eurocheque 5,50)

THE VDU CARD

Article to assure that Elektor's VDU card works well.
Software (machinecode) with introduction.
Many members of the club use this software instead of the software from Elektor.
Authors: J.J.A. and J.A.J. Janssen, The Netherlands.
Transl.: Willem van Asperen.
System: Elektor's JUNIOR computer with VDU card.
Publ.: DE 6502 KENNER 31, Apr. 1984, p.17-26.
Send cheque of Hfl. 19,00 to W.L.v.Pelt (eurocheque 9,50)

THE GRAPHIC DISPLAY

Article with schematic diagram to do graphical work (plotting of graphs etc.) together with Elektor's VDU card.
Authors: J.J.A. and J.A.J. Janssen
Transl.: Willem van Asperen
System: Elektor's JUNIOR computer with VDU card.
Publ.: DE 6502 KENNER 36, Febr. 1985, p.5-11
Send cheque of Hfl. 14,50 to W.L.v.Pelt (eurocheque 5,00)

MON65/DOS65 is the new system of our club, build with Elektor's CPU, VDU, RAM-cards and our professional FDC controller card, for 6502 or 65C02.
For more information, write to E.J.M. Visschedijk
Drakensteyn 299
NL - 7608 TR ALMELO

Source-listings of the monitor and the disk operating system, complete with comments, already in English language.
Hardware description, English version, translated by Albert v.d. Beukel, The Netherlands, and corrected by Andrew Gregory, England.

DOS-manual to be translated soon. Helpfile available on diskette.

Monitor-manual to be translated soon.

Editor-manual to be translated soon.

For those who want to be informed about the hardware first it is possible to order for the hardware description, only the English version. If wanted:

Send cheque of Hfl. 29,50 to W.L.v.Pelt (eurocheque 20,00)

TOKENIZED Microsoft Basic Keywords and addresses VIC-20 Commodore Basic V2.

Nico de Vries, The Netherlands.

Send cheque of Hfl. 14,50 to W.L.v.Pelt (eurocheque 5,00)

PATCHES ON KIM-1 MICROSOFT BASIC FOR ELEKTOR'S JUNIOR-comp
Koen van Nieuwenhove, Belgium/W.L. van Pelt, The Netherl.
With PMODE, TRACE, STEP RESET, PRINTER PG., VIDEO, AUTO, RENUMBER, DATASAVE, EDIT LINE, APPEND.
Send cheque of Hfl. 24,50 to W.L.v.Pelt (eurocheque 15,00)

HINTEXT Basic Texteditor

for Elektor's JUNIOR and OCTOPUS-computers

Author: M.R. van Hintum, The Netherlands

Transl.: Maarten van Lieshout (Introduction)

Willem van Asperen (Basic-progr.)

The Dutch version was published in DE 6502 KENNER nr. 41, Dec. 1986, p.6-14.

Send cheque of Hfl. 14,50 to W.L.v. Pelt (eurocheque 5,00)

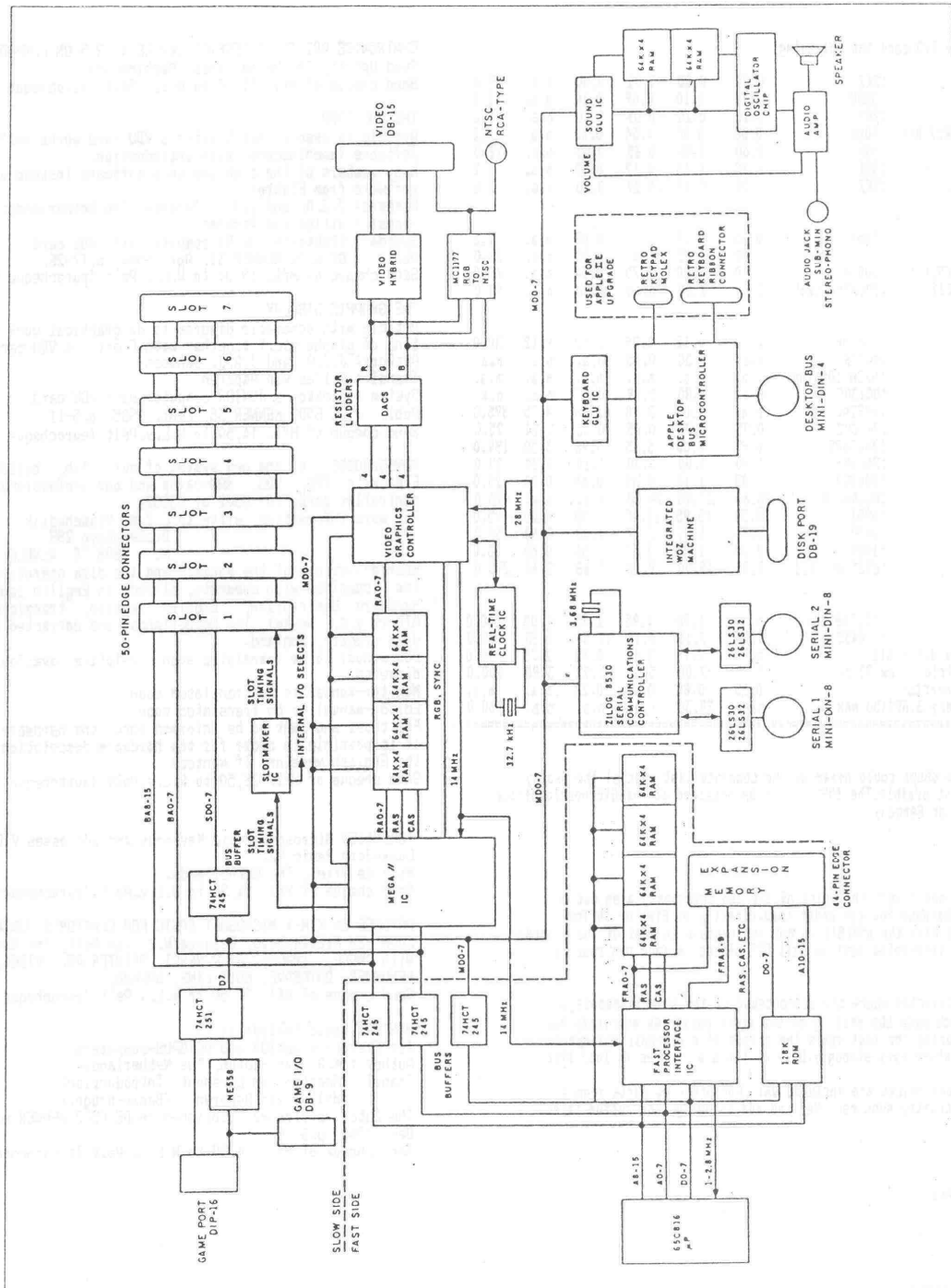


Figure 1: A block diagram of the Apple II GS.

OCTOPUS-FORTH 1.2

OCTOPUS-FORTH 1.2 is a Forth-system, specially build for Elektor's Octopus/EC65 6502-computer as published in the Elektor's Special Editions. It uses the OHIO-DOS which functions as a host for the Forth-system. OCTOPUS-FORTH 1.2 is based on the model of the Forth Interest Group as published in their F16-Forth Installation Manual and the F16-Forth 6502 Assembly Source Listing (both of them can be ordered by our club). Several bugs in this model are removed and a lot of high level words are rewritten in code to increase speed. Although a lot of other publications about Forth have been used, most of this system is written by our clubmember Fridus Jonkman.

OCTOPUS-FORTH 1.2 consists of:

- A precompiled section:
 - * kernel
 - * I/O
 - * assembler
 - * full screen editor
 - * directory-system
 - * copy-facilities
 - * words to support the VDU-card
 - * etc.
- Utilities:
 - * recursive decompiler
 - * tracer for debugging colon-definitions
 - * breakpoint-setting for debugging
 - * vectoring
 - * strings
 - * sysgen for generating Forth-diskettes
 - * etc.
- A meta-compiler:

The source screens of the precompiled section are included; they can be loaded to build a new precompiled Forth elsewhere in memory. This powerful tool allows full control to build a customized system.
- Documentation screens (English):

All words used in the precompiled section are fully documented as far as they are not included in the F16-Forth model. Also utilities and the use of the meta-compiler are described.

Memory map:

- \$0000-0100 : zero page
- \$0100-0200 : computation-stack
- \$0200-0200 : terminal input buffer
- \$0200-2200 : kernel, I/O, assembler
- \$2200-3200 : OHIO-DOS
- \$3200-5042 : extensions
- \$5042-BF60 : word-, textbuffers, free for compilation
- \$BF60-DF80 : 8 diskbuffers, each 1 Kbyte
- \$DF80-E000 : user-area

Diskettes:

The system as described here will be available on 80 track systems, which are single sided (2 diskettes).

Prices and distribution:

Because OHIO-DOS is part of the system and is not placed in the public domain, you must prove you bought it yourself before we can deliver the Forth-system. For OCTOPUS-FORTH 1.2 : all rights reserved by our club. Send 2 blank diskettes with labels and R/W-protectors and a cheque of Hfl. 59,50 (eurocheque Hfl. 50,-) to W.L. van Pelt at the editorials office, Krimpener a.d. IJssel.

THE APPLE II GS: A 16-BITS APPLE II

By : Gert van Opbroek, The Netherlands.

The Apple II-series has a new member: the Apple II GS which stands for Graphic and Sound. The machine has the following specifications:

- CPU: 65C816
- Memory: 256 kB RAM expandable to 4 MB; 128 kB ROM
- Graphics: five modes ranging from 40 by 48 points in 16 colours to 640 by 200 points in four colours; total number of colours: 4096
- Sound: synthesizer with 32 oscillators for 15 simultaneous, independent tones
- Connections: RGB-port, monochrome Video, Audio stereo headphone, 2 RS422 serial ports, joystick or paddles, connector for 5 1/4 and/or 3 1/2 inch disk drives, connector for the separate keyboard and on this keyboard a connector for a mouse
- 7 Apple II compatible expansion slots

The machine is hard- and software compatible with the other members of the series. This means that most of the old Apple II software will run on the Apple II GS and that the Apple II expansion boards can be used. The old software will run a factor 2.5 to 3 faster on this new machine

In Holland, the machine can be bought from 1 February 1987. When ordered before this date, a system containing:

- An Apple II GS with keyboard and mouse
- An Unified 800 kB 3.5 inch disk
- An Apple II GS B&W monitor
- An Apple II GS 256 kB expansion card

will cost Dfl. 4794.00 incl. 20% VAT.

Information about this machine:

- In Holland: Apple Computer B.V.
Postbus 7
3712 ZB Huis ter Heide
03404 - 86922

Refer to our clubs magazine.

- A Porcut Review can be found in BYTE October 1986.

ML BASICODE UTILITY-DISK FOR ELEKTOR'S OCTOPUS/EC65

Our member Marc Lachaert developed the new bootable disk for 40 trs, SS, DD on your Octopus. Because OHIO-DOS is part of the system and not is placed in the public domain, you must prove you bought it yourself, before we can deliver the diskette.

The diskette boots up with a menu to choose!

1. Loading the standard BASICODE-2 subroutines from DISK and the BASICODE program of your choice from TAPE
 2. Loading the BASICODE program of your choice from TAPE
 3. Loading a program from DISK and MERGE it with a BASICODE program from TAPE
 4. Saving the program of your choice from DISK in BASICODE format on TAPE
 5. Enable/Disable the RSEQ/EDIT commands
 6. End of program. (EXTRA: BASIC GAME "SLIDING GRID")
- To order the diskette send cheque of Hfl. 22,00 (eurocheque Hfl. 12,50) and a blank diskette to W.L. van Pelt.