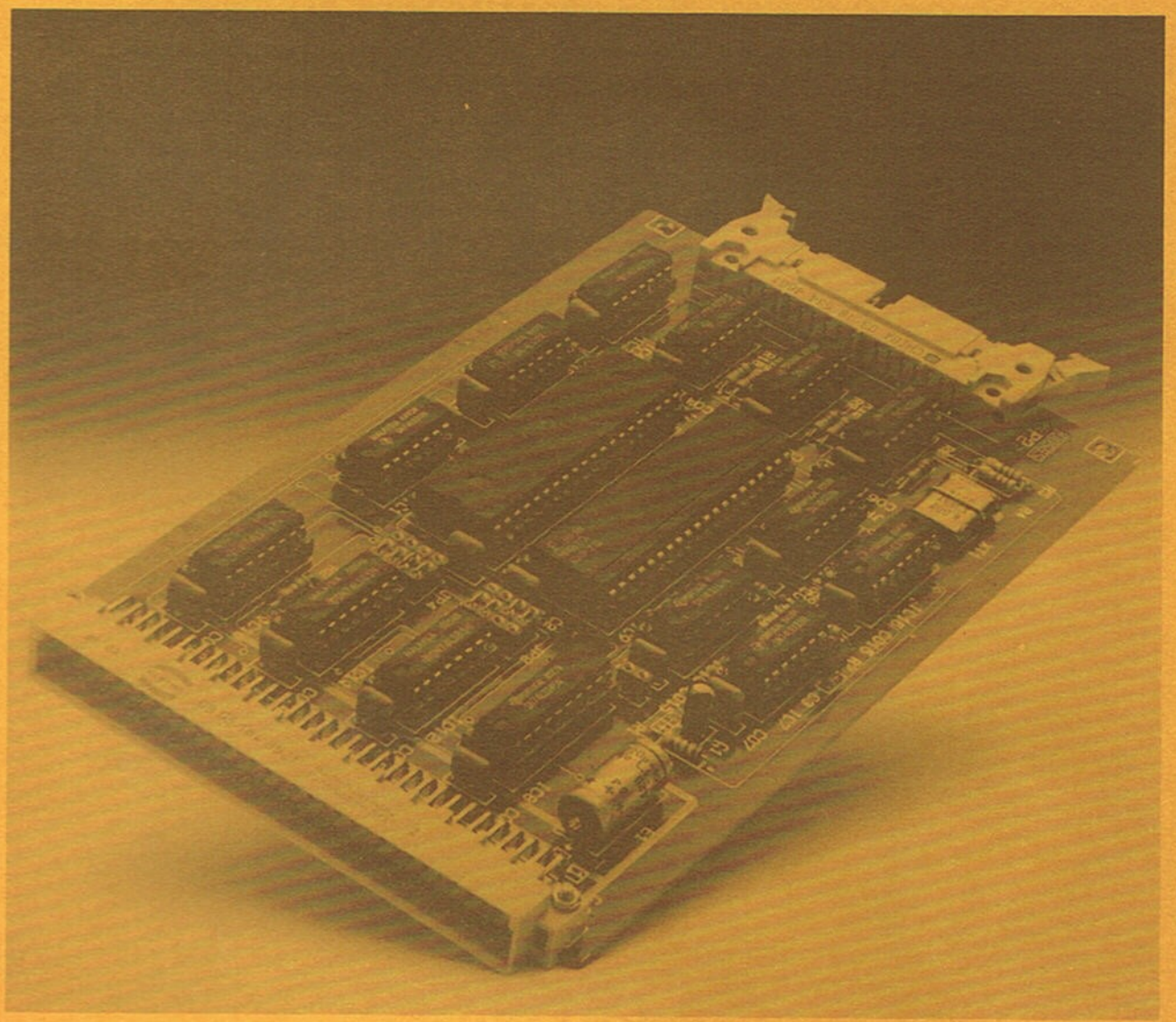
DE 6502 KENNER 43



Tiende Jaargang, Nr. 2
april 1986

DE 6502 KENNER

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

Adres voor het inzenden van en reakties op artikelen voor DE 6502 KENNER:
Willem L. van Pelt
Jacob Jordaensstraat 15
2923 CK Krimpen a/IJssel
Tel.: 01807 - 19881

Vaste medewerkers:
Willem L. van Pelt
Gerard van Roekel
Frans Smeehuijzen

Freelance medewerkers:
Rob Banen
Fred Behringer, (Germany)
Fridus Jonkman
Gert Klein
Roger Langeveld
Fernando Lopez, (Portugal)
Frank Manshande
Ruud Uphoff
Frans Verberkt
Herman Zondag

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

De voorpagina is de DOS65-controllerkaart. ontwikkeld door Ad Brouwer.
CAD/CAM: E. Visschedijk.
I.s.m. : A. Hankel
Fotogr.: Fr. Visschedijk.

INHOUDSOPGAVE DE 6502 KENNER NR. 43 MEI 1986

PAPERWARE-SERVICE

DATBAS for Elector's JUNIOR with VDU-card and OS-65D

Our member Jan van Heuven published in our issue nr. 37 from page 25 the PRINT&(X,Y) command adaptation and the Basic-program DATBAS to define databases, to add records, to edit, to erase and to print. The article of Jan van Heuven was published then in the Dutch language.
With some little modifications our Portugese friend Fernando Lopez translated the PRINT&(X,Y) adaptation and the DATBAS-program into the English language to serve other members of our club.
Send eurocheque of HFL. 20,== to Mr. W.L. van Pelt at Krimpen a.d. IJssel. If not paying with eurocheques: HFL. 7,50 extra transfers!!!

De OCTOPUS, Elektuur's nieuwste DOS-computer, begint in onze club een plaats te veroveren. In Duitsland zijn er al vele gebruikers, met als prettige bijkomstigheid dat wij daardoor ook enkele leden konden begroeten. De meeste van die leden kennen de betekenis van publiceren in DE 6502 KENNER. Dr. Tietsch is van hen de eerste die met een omvangrijke inzending, welke in delen moet worden geplaatst, een belangrijk utility-programma aanbiedt. Hij is iemand die als vele buitenlandse leden echter wel graag ook van ons zou willen leren. Onze mooie nederlandse taal is echter voor hen nog erger dan chinees, heb ik af en toe het gevoel. Willen onze nederlandse en belgische leden echter profiteren van kennis die ons van buiten aangereikt kan worden, dan lijkt het toch niet onverstandig de artikelen toch zoveel mogelijk in het engels te schrijven; immers de voertaal onder doorgewinterde programmeurs. Het is geen eis, de leden die er moeite mee hebben, leveren als gewoonlijk in het nederlands in, maar het is goed te beseffen dat engelstalige artikelen in het buitenland de aantrekkingskracht naar onze club kan vergroten. Ik heb bemerkt dat de kontakten die ik in het buitenland wist te bewerkstelligen eerder konden worden omgezet in een lidmaatschap als ik kon overtuigen dat er voldoende engelstalige artikelen in stonden. Laten we dat met z'n allen proberen waar te maken! Nederland zelf wordt voor ons een te klein gebied om daaruit voldoende leden te blijven trekken, vrees ik, zelfs niet met het verbreden van ons gebied van aandacht door er de 6800, 6809 en 68XXX aan toe te voegen.
De club is doende maatregelen te nemen om de bezitters van OCTOPUS met 80 tracks drives ook aan de benodigde schijven te helpen. Het is nu alleen nog mogelijk 40 tracks te leveren op mijn eigen drives. De rijkdom van de club staat niet toe dat we al onze wensen in kunnen vullen, maar er gloort wat hoop als we onze schouders zetten, en U dus ook, onder het werven van nieuwe leden. Maak er wat werk van. Denk niet: dat doet een ander wel. Doe Uw best. Het is ook Uw eigen belang.
Nog een verzoek. Hoewel afhankelijk van de omvang van de instroom van artikelen, zullen we toch nader onderzoeken of de informatiedichtheid in DE 6502 KENNER nog verder kunnen opvoeren. Dat is geen eenvoudige zaak. Niet uit een redaktioneel oogpunt, maar zeker niet als we denken aan het evenwicht dat er moet bestaan tussen het binnenstromen van de artikelen en het plaatsen ervan, bovendien nog getoetst tegen de achtergrond van de samenstellings- en variatiemogelijkheden vanuit de copy-buffer. Eenvoudig gezegd: er moeten wel voldoende artikelen binnen blijven stromen wil er van enige haalbaarheid sprake zijn. Gelukkig zijn er weer wat toezeggingen uit de hoek van de Applelaars in onze club. De pogingen om uit die hoek materiaal binnen te halen slaagden alweer een beetje, dank zij een nieuw lid, t.w. Frank Manshande, een van de jongere, maar veelbelovende leden, die ook kritische noten plaatst op het redaktionele gebied. Ik ben daar erg blij mee. Aan U wil ik in dit verband vragen Uw inzendingen zoveel mogelijk op papier aan te bieden, het liefst 8 LPI (8 lijnen per inch), in een harde zwart/wit-verhouding, ca. 65 lijnen per pagina.
Het is prettig als ik tussen alle post ook nog Commodore-artikelen of Atari-artikelen aan kan treffen. Tot nu toe blijft dat maar magertjes, terwijl we toch voldoende van deze machines in huis hebben. Het is heus niet zo dat alleen in machinetaal mag worden ingeleverd, Basic, Comal, Pascal, Forth, etc., alles is welkom; we kunnen ook daarvan een hoop leren. Kom, zet Uw beste beentje eens voor.
Nog twee opmerkingen. In de eerste plaats dat er al zo'n veertig DOS65 gebruikers zijn. Niet te verwarren met OCTOPUS, al zijn alle Elektuur-kaarten, m.u.v. de DOS-kaart daarvan, daarin betrokken. Dat is veelbelovend. Het is dan ook met plezier dat we daarvoor de ook het eerste echte programma kunnen plaatsen. Als ik zo op de bijeenkomst zie wat uit dat DOS65 groeit, dan mag best gezegd dat Ad Brouwer, de geestelijke vader ervan, in onze club, met de inzet van Erwin Visschedijk en Adri Hankel, een groot succes weet te bereiken. In de tweede plaats wil ik nog kwijt dat FATE, de 12K Format Lister/Assembler/Tape-utilities/Editor van Rob Banen, eerst door Jaap de Hoop in het engels vertaald, nu door Fred Behringer uit Muenchen naar het duits werd vertaald, beiden v.w.b. het manual want de source blijft in het engels. Fred Behringer heeft inmiddels de eerste kladvertaling naar het duits ingeleverd van het manual van Micro-ADE. Wat minder zichtbaar wordt dat is het vele documentatiewerk dat Fernando Lopez uit Portugal voor ons doet. Ik wil graag van deze plaats af, naar U toe, laten weten dat die inspanningen van grote betekenis kunnen zijn in relatie met Uw inzendingen en Uw inspanningen om van onze club iets geweldigs te maken.

Willem van Pelt

U I T N O D I G I N G B I J E E N K O M S T

Datum : zaterdag 24 mei 1986
Lokatie : Speeltuingebouw Beeklust/Ossenkoppelerhoek
Jan Steenstraat 69, ALMELO. Tel.: 05490 - 19443

Reisroute :

- per auto - Vanuit het westen en het zuiden via de A1/A35.
1. Aan het einde van de snelweg rechtsaf. Bij de eerstvolgende stoplichten rechtdoor. Na ca. 100 m. kruispunt zonder stoplichten. Ga hier linksaf.
2. Deze straat, de Jan Vermeerstraat, maakt een flauwe bocht naar rechts. Aan het einde van deze bocht ziet U links de lichtmasten van een voetbalveld. Ga linksaf. Het speeltuingebouw vindt U na ca. 150 m. aan de linkerkant.

3. Vanuit het noorden via de N36.
Rij door tot ANWB-borden richting Wierden/Zwolle. Ga hier rechtsaf. Blijf deze weg volgen. U komt dan over een spoorwegovergang.
Ga na ca. 150 m. linksaf, weer richting Wierden/Zwolle. Na ca. 200 m. ziet U rechts een Texaco-benzinestation. Ga hier rechtsaf. Verder als beschreven bij punt 2.

TOEGANGSPRIJS : FL. 10,==. Lunchpakket eventueel zelf mee te nemen.
Lunch is ook tegen vergoeding verkrijgbaar.
Konsumpties tegen betaling.

PROGRAMMA :
09.30 Zaal open.
10.15 Opening door de gastheren Erwin Visschedijk en Adri Hankel en door diens echtgenoten, die voor een uiterst vriendelijke bediening en begeleiding zorgen.
10.30 Lezing over transition-(toestands-)diagrammen
Met demonstratie 6502-gestuurde telefooncentrale door Erwin Visschedijk
11.30 Koffiepauze.
11.45 Forum. Aan het forum kunnen vragen gesteld worden van allerlei aard.
12.00 Lunchpauze.
13.00 INFORMEEL GEDEELTE.
Tijdens het informeel gedeelte kunnen leden vrij met elkaar en met elkaars ervaringen kennis maken. Leden brengen hun systemen mee en demonstreren dit aan de aanwezigen. NEEM DAAROM UW COMPUTER MEE !!!
M A R K T. Op eigen tafel(s) te regelen.
17.00 Sluiting.

DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65

RS232 ACIA (6551) instel programma.

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

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

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

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

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

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

DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65 DOS65

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


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

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

```
80BD 9D 3A88            STA     MPOS,X
80C0 E8                 INX                     ;pointer=8
80C1 AD 51CE            LDA     CMD             ;transmitter control
80C4 4A                 LSRA
80C5 4A                 LSRA
80C6 29 03              AND     #%00000011
80C8 9D 3A88            STA     MPOS,X
                ;
                ;compute index table
                ;
80CB A2 00      RCINDEX LDX     #0              ;set index zero
80CD 18         CMPOS13 CLC
80CE BD 3A88            LDA     MPOS,X          ;fetch value from table
80D1 7D 3A88            ADC     MPOS,X          ;double it
80D4 9D 4488            STA     INDEX,X         ;save in index table
80D7 E8                 INX                     ;increase pointer
80D8 E0 09              CPX     #9              ;until end of table
80DA D0 F1              BNE     CMPOS13
80DC 60                 RTS
                ;
                ;sub set inverse flag 'i' on computed locations
80DD A0 00      SETINV  LDY     #0              ;index Y zero
80DF A2 00              LDX     #0              ;index X zero
80E1 18         SETINV2 CLC                     ;prepare for add.
80E2 BD 0D88            LDA     OFFSET,X        ;fetch low byte offsettable
80E5 79 4488            ADC     INDEX,Y         ;add index to this value
80E8 85 52              STA     INDPNT          ;set low byte index pointer
80EA E8                 INX
80EB BD 0D88            LDA     OFFSET,X        ;fetch high byte offsettable
80EE 69 00              ADC     #$0             ;add carry
80F0 85 53              STA     INDPNT+1        ;set high byte index pointer
80F2 98                 TYA                     ;save Y-reg.
80F3 48                 PHA
80F4 A0 00              LDY     #$0
80F6 B1 52              LDA     [INDPNT],Y      ;fetch address pointer
80F8 85 54              STA     ADPNT
80FA C8                 INY
80FB B1 52              LDA     [INDPNT],Y
80FD 85 55              STA     ADPNT+1
80FF A0 04              LDY     #4              ;fourth character in string
8101 A9 69              LDA     #'i             ;inverse flag
8103 91 54              STA     [ADPNT],Y
8105 68                 PLA                     ;restore Y-reg
8106 A8                 TAY
8107 E8                 INX
8108 C8                 INY
8109 C0 0A              CPY     #10             ;done all offsets ?
810B D0 D4              BNE     SETINV2
810D 60                 RTS
                ;
                ;sub print string pointered by PNT
810E A0 00      PNTOUT  LDY     #0
8110 B1 50      PNTOUT2 LDA     [PNT],Y
8112 F0 06              BEQ     PNTOUT3         ;$00: end-ofstring
8114 20 23A0            JSR     OUT
8117 C8                 INY
8118 D0 F6              BNE     PNTOUT2
811A 60         PNTOUT3 RTS
                ;
                ;next sub contains two functions:
                ;a-reset 'n' flags in BD00 file
                ;b-recompute acia CMD & CTL register
                ;  from MPOS variables
811B A9 7E      RECOM   LDA     #BD00&255
811D 85 56              STA     SPNT
811F A9 85              LDA     #BD00>>8
8121 85 57              STA     SPNT+1
8123 A2 04              LDX     #4
8125 A9 6E              LDA     #'n
8127 9D 7E85            STA     BD00,X          ;reset first flag
```

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

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

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

```
82A4 20 CB80    COMBSB  JSR     RCINDEX         ;recompute index
82A7 4C 6A82            JMP     COMLFB          ;continue elsewhere
82AA 60         COMBSEND RTS
                ;end BS command
                ;
                ;HT command
82AB AD 4388    COMHT   LDA     VPOS            ;fetch current V pos
82AE AA                 TAX                     ;use as index
82AF BD 3A88            LDA     MPOS,X          ;fetch current H pos
82B2 DD 1F88            CMP     MAX,X           ;compare with max value
82B5 F0 06              BEQ     COMHTEND        ;no movement possible
82B7 FE 3A88            INC     MPOS,X          ;else increase allowed
82BA 4C A482            JMP     COMBSB          ;continue elsewhere
82BD 60         COMHTEND RTS
                ;end HT command
                ;
                ;H command (help)
82BE A2 1D      COMH    LDX     #HELPSCR&255    ;print HELP-screen
82C0 A9 84              LDA     #HELPSCR>>8
82C2 20 0380            JSR     STRING
82C5 20 20A0            JSR     IN              ;wait for a key
82C8 20 1B81            JSR     RECOM           ;reset string's and reg.'s
82CB 68                 PLA                     ;reset stackpointer
82CC 68                 PLA
82CD 4C E581            JMP     MAIN1           ;continue in mainloop
                ;end H command
                ;
                ;Q command
82D0 20 1B81    COMQ    JSR     RECOM
82D3 A2 36              LDX     #END&255        ;ask for confirmation
82D5 A9 85              LDA     #END>>8
82D7 20 0380            JSR     STRING
82DA 20 20A0    COMQB   JSR     IN
82DD 48                 PHA
82DE 20 23A0            JSR     OUT
82E1 68                 PLA
82E2 C9 4E              CMP     #'N
82E4 F0 0B              BEQ     COMQEND
82E6 C9 59              CMP     #'Y
82E8 D0 07              BNE     COMQEND
82EA A0 5E              LDY     #SCOM&255
82EC A9 85              LDA     #SCOM>>8
82EE 20 06A0            JSR     COMMAND1
82F1 A2 79      COMQEND LDX     #END1&255       ;print END message
82F3 A9 85              LDA     #END1>>8
82F5 20 0380            JSR     STRING
82F8 68                 PLA                     ;reset stackpointer
82F9 68                 PLA
82FA 60                 RTS                     ;back to DOS65
                ;end Q command
                ;
                ;end availlable commands
                ;
                ;
                ;********** Tables etc. **********
                ;
                ;startscreen
82FB 0C1401011B STARTSCR FCC    CLS,FB,1,1,ESC,'i- RS232 -  '
8300 692D205253
8305 323332202D
830A 2020
830C 4143494120         FCC     'ACIA 6551 set & check utility '
8311 3635353120
8316 7365742026
831B 2063686563
8320 6B20757469
8325 6C6974920
832A 2020566572         FCC     '  Version 1.01   Oct. 1985  HaViSoft',ESC,'n'
832F 73696F6E20
8334 312E303120
```

```
8339 20204F6374
833E 2E20313938
8343 3520204861
8348 5669536F66
834D 741B6E
8350 140C034261         FCC     FB,12,3,'Baudrate :'
8355 7564726174
835A 65203A
835D 140A05576F         FCC     FB,10,5,'Wordlength :'
8362 72646C656E
8367 677468203A
836C 140E065061         FCC     FB,14,6,'Parity :'
8371 7269747920
8376 3A
8377 140C075374         FCC     FB,12,7,'Stopbits :'
837C 6F70626974
8381 73203A
8384 1406095265         FCC     FB,6,9,'Receiver clock :'
8389 6365697665
838E 7220636C6F
8393 636B203A
8397 14070A5265         FCC     FB,7,10,'Receiver mode :'
839C 6365697665
83A1 72206D6F64
83A6 65203A
83A9 14020B5265         FCC     FB,2,11,'Receiver interrupt :'
83AE 6365697665
83B3 7220696E74
83B8 6572727570
83BD 74203A
83C0 14010C4461         FCC     FB,1,12,'Data terminal ready :'
83C5 7461207465
83CA 726D696E61
83CF 6C20726561
83D4 6479203A
83D8 14230D5F5F         FCC     FB,35,13,'___'
83DD 5F
83DE 140C0E5472         FCC     FB,12,14,'Transmit : Interrupt:  RTS:  Transmitter:'
83E3 616E736D69
83E8 74203A2049
83ED 6E74657272
83F2 7570743A20
83F7 205254533A
83FC 2020547261
8401 6E736D6974
8406 7465723A
840A 1401155479         FCC     FB,1,21,'Type H for help'
840F 7065204820
8414 666F722068
8419 656C70
841C FF                 FCC     $FF             ;end-of-file marker
                        ;
                        ;helpscreen
841D 140102190A HELPSCR FCC     FB,1,2,$19,10,10,10,SP,SP,SP,SP
8422 0A0A202020
8427 20
8428 417661696C         FCC     'Availlable commands :',13,10,10
842D 6C61626C65
8432 20636F6D6D
8437 616E647320
843C 3A0D0A0A
8440 2020202020         FCC     SP,SP,SP,SP,SP,SP,SP,SP,SP,SP,SP
8445 2020202020
844A 20
844B 2020202020         FCC     SP,SP,SP,SP,SP,SP,SP,SP,SP,SP,SP
8450 2020202020
8455 20
8456 48203A2070         FCC     'H : prints this message',13,10
845B 72696E7473
8460 2074686973
```

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

```
857E 1417031B6E BD00    FCC     FB,23,3,ESC,'nExternal',ESC,'n',0
8583 4578746572
8588 6E616C1B6E
858D 00
858E 1420031B6E BD01    FCC     FB,32,3,ESC,'n50',ESC,'n',0
8593 35301B6E00
8598 1423031B6E BD02    FCC     FB,35,3,ESC,'n75',ESC,'n',0
859D 37351B6E00
85A2 1426031B6E BD03    FCC     FB,38,3,ESC,'n109.92',ESC,'n',0
85A7 3130392E39
85AC 321B6E00
85B0 142D031B6E BD04    FCC     FB,45,3,ESC,'n135.58',ESC,'n',0
85B5 3133352E35
85BA 381B6E00
85BE 1434031B6E BD05    FCC     FB,52,3,ESC,'n150',ESC,'n',0
85C3 3135301B6E
85C8 00
85C9 1438031B6E BD06    FCC     FB,56,3,ESC,'n300',ESC,'n',0
85CE 3330301B6E
85D3 00
85D4 143C031B6E BD07    FCC     FB,60,3,ESC,'n600',ESC,'n',0
85D9 3630301B6E
85DE 00
85DF 1417041B6E BD08    FCC     FB,23,4,ESC,'n1200',ESC,'n',0
85E4 313230301B
85E9 6E00
85EB 141C041B6E BD09    FCC     FB,28,4,ESC,'n1800',ESC,'n',0
85F0 313830301B
85F5 6E00
85F7 1421041B6E BD0A    FCC     FB,33,4,ESC,'n2400',ESC,'n',0
85FC 323430301B
8601 6E00
8603 1426041B6E BD0B    FCC     FB,38,4,ESC,'n3600',ESC,'n',0
8608 333630301B
860D 6E00
860F 142B041B6E BD0C    FCC     FB,43,4,ESC,'n4800',ESC,'n',0
8614 343830301B
8619 6E00
861B 1430041B6E BD0D    FCC     FB,48,4,ESC,'n7200',ESC,'n',0
8620 373230301B
8625 6E00
8627 1435041B6E BD0E    FCC     FB,53,4,ESC,'n9600',ESC,'n',0
862C 393630301B
8631 6E00
8633 143A041B6E BD0F    FCC     FB,58,4,ESC,'n19200',ESC,'n',0
8638 3139323030
863D 1B6E00
                ;wordlength
8640 1417051B6E WL05    FCC     FB,23,5,ESC,'n5',ESC,'n',0
8645 351B6E00
8649 1419051B6E WL06    FCC     FB,25,5,ESC,'n6',ESC,'n',0
864E 361B6E00
8652 141B051B6E WL07    FCC     FB,27,5,ESC,'n7',ESC,'n',0
8657 371B6E00
865B 141D051B6E WL08    FCC     FB,29,5,ESC,'n8',ESC,'n',0
8660 381B6E00
                ;parity
8664 1417061B6E PAN0    FCC     FB,23,6,ESC,'nNo',ESC,'n',0
8669 4E6F1B6E00
866E 141A061B6E PAOD    FCC     FB,26,6,ESC,'nOdd',ESC,'n',0
8673 4F64641B6E
8678 00
8679 141E061B6E PAEV    FCC     FB,30,6,ESC,'nEven',ESC,'n',0
867E 4576656E1B
8683 6E00
8685 1423061B6E PAMA    FCC     FB,35,6,ESC,'nMark',ESC,'n',0
868A 4D61726B1B
868F 6E00
8691 1428061B6E PASP    FCC     FB,40,6,ESC,'nSpace',ESC,'n',0
8696 5370616365
```

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

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

BACKNUMBERS "DE 6502 KENNER"

Following backnumbers of DE 6502 KENNER are available:

NR. 16	MAY	1981	NR. 30	FEB	1984
NR. 17	AUG	1981	NR. 31	APR	1984
NR. 18	OCT	1981	NR. 32	JUN	1984
NR. 19	DEC	1981	NR. 33	AUG	1984
			NR. 34	OCT	1984
NR. 22	AUG	1982	NR. 35	DEC	1984
NR. 23	OCT	1982			
NR. 24	DEC	1982	NR. 36	FEB	1985
			NR. 37	APR	1985
NR. 25	FEB	1983	NR. 38	JUN	1985
NR. 26	MAY	1983	NR. 39	AUG	1985
NR. 27	AUG	1983	NR. 40	OCT	1985
NR. 28	OCT	1983	NR. 41	DEC	1985
NR. 29	DEC	1983			

Send Hfl. 9,= per edition on eurocheque to Mr. John van Sprang, Tulp 71, 2925 EW Krimpen a/d IJssel, The Netherlands. Six backnumbers of 1984 or 1985 only Hfl. 45,= !!
The secretary, Gert Klein, will send you the backnumbers.

PRICES OF THE MON65/DOS65 SYSTEM

- MANUAL MON65
  MANUAL DOS65
  MANUAL EDITOR
  DESCRIPTION OF THE HARDWARE
  This all, about 110 pages FL.50,=
- 2764 MONITOR EPROM FL.35,=
- 2732 CHAR.GENERATOR EPROM FL.25,=
- DISKETTE WITH I.E. MICRO-ADE, FORTH, EDITOR, UTILITIES FL.15,=
- FDC-CARD FL.50,=
- SOURCE MON65 FL.25,=
- SOURCE DOS65 FL.25,=
- SOURCE UTILITIES FL.25,=

All text in Dutch language. We are hard working on translations into English. Want informations (English)? Please ask the editorial office.
All prices are without HFL. 10,= transports.
Only for members of our club.
Send your orders to:
E.J.M. Visschedijk (Havisoft)
Drakesteyn 299
7608 TR ALMELO
The Netherlands
Payments on eurocheque. Don't forget the number and your signature.
If not paying with eurocheque HFL. 7,50 extra transfers!

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

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

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

Jan Vernimmen
Telf: 02990 - 21739

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

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

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

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

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

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

```
**********************************************************
*                  APPLE FILES OPGESLAGEN                *
*                        DOOR :                          *
*                   FRANK MANSHANDE                      *
**********************************************************
```

Heeft U zich ooit wel eens afgevraagd wat er precies gebeurd, als je een programma 'saved' of 'load'? En waarom een programma, hoe klein ook, minstens twee sectoren op een diskette in beslag neemt? Nou, ik wel! Maar ik weet nu hoe, en welke de reden is. Dus laat ik het U maar snel uitleggen ...
Om uit te leggen waarom een programma minstens 2 sectoren inneemt, moet ik U eerst een speciale sector uitleggen die bij een programma hoort: de Track/Sector List. Dit is de eerste sector van een programma. Hierin staat opgeslagen welke sectoren op welke track gebruikt. Meestal is er maar 1 zo'n sector, maar als het programma meer dan 122 data sectoren inneemt, zal er nog een Track/Sector List sector zijn.
De Track/Sector List sector is dus de eerste sector van een programma, en hierna worden de data sectoren opgeslagen. Dus als je een heel kort programma hebt, zal het toch minstens twee sectoren innemen!
Wat staat er nu allemaal in een Track/Sector List? Dat staat hieronder in een tabel weergegeven:

BYTE :	FUNKTIE :
$00	Niet gebruikt.
$01	Hier staat het tracknummer, als er nog een Track/Sector List sector nodig is, dus als het programma meer dan 123 sectoren in beslag neemt.
$02	Hier staat het sectornummer van de volgende Track/Sector List sector (als die er is).
$03-$04	Niet gebruikt.
$05-$06	Hier staat welke de eerste data sector is, die in deze Track/Sector List staat. Als hier dus $00/$00 staat, dan is de eerste sector in deze Track/Sector List de eerste data sector van het programma.
$07-$0B	Niet gebruikt.
$0C-$0D	Track- en sectornummer van de eerstvolgende data sector.
$0E-$0F	Track- en sectornummer van de volgende data sector.

```
   |          |
   V          V
```

$FE-$FF Track- en sectornummer van de volgende data sector. (Als er op een van die Track/Sector-paren $00/$00 staat, dan zijn er niet meer sectoren !)

Maar hoe zijn de programma's nu verder opgeslagen? We weten nu, dat de eerste sector de Track/Sector List sector is.
We beginnen met een Basic-file :
Een Basic-programma begint op $0800, dus het startadres hoeft DOS niet op te slaan. De lengte van het programma moet DOS echter wel opslaan, en dat gebeurt op byte 0 en byte 1. In byte 0 staat de 'low byte' en in byte 1 de 'high byte'. Stel dat er op byte 0 de waarde $03 staat en op byte 1 $02, dan is de lengte dus $0203 bytes lang, d.i. decimaal 515. De rest van de sector is dan opgevuld met hexadecimale getallen die de commando's, regelnummers, enz. voorstellen.
Ten tweede de Integer-Basic file :
Dit is bijna hetzelfde als bij de Basic file, alleen bij de regelnummer zit het iets anders.
Ten derde de machinetaal file :
In byte 0 en byte 1 staat het startadres, op byte 2 en 3 staat de lengte van de machinetaal file. De rest is opgevuld met het programma in de sector.
En tenslotte de text file :
Dit is de gemakkelijkste. De text van een file staat in hexadecimale code, en elk stuk tekst is een zogenaamde record. Een record is bijvoorbeeld een regel uit een tekst (net zoals bij deze tekst, elke regel is een record). De records zijn onderling gescheiden door $8D (return-code). Het einde van een textfile wordt aangegeven door een $00 (break-code).

Als je een programma 'runt' of 'load' zoekt DOS eerst in de VTOC, waar de Catalog ergens staat. Dan laadt hij de eerste Catalog sector in en zoekt naar de opgegeven naam. Als hij die niet vinden kan, kijkt hij of er nog meer Catalog sectoren zijn en zoekt die ook af naar de naam. Kan DOS de naam niet vinden, dan volgt de foutmelding : File Not Found.
Vindt DOS de naam echter wel, dan zorgt hij dat het programma geladen en eventueel uitgevoerd wordt d.m.v. ingebouwde routines!
Als een programma opgeslagen moet worden zal Dos eerst in de Catalog kijken of de file misschien ge'locked' is. Is dit niet het geval, dan zal DOS het programma opslaan.
Bij een textfile wordt de file eerst geopend. Is de file niet aanwezig, dan maakt DOS hem als een 1 sector lang programma zonder inhoud. Als de file wel aanwezig is, dan wordt deze geopend, en kunnen er data gelezen of geschreven worden.

```
==========================================================
| ELEKTOR'S OCTOPUS (= EC) DISKETTES AVAILABLE
| ============================================
| How to order your diskettes :
|
| 1. check your membership of our club
| 2. send 11 new diskettes, with stickers, to the following
|    address :
|    Editorial Office DE 6502 KENNERS
|    c/o Willem L. van Pelt
|    Jacob Jordaensstraat 15
|    2923 CK Krimpen a.d. IJssel
|    The Netherlands
|    (Phone : 01807 - 19881)
| 3. send eurocheque to the amount of HFl. 125,== to pay 10 discs
|    (1 is free), no returnporti needed
|
| We only can write discs for the Octopus on SSDD 40 tracks.
|
==========================================================
| VRAAG EN AANBOD
| ===============
| Te koop: Wegens overschakeling op PC een 6502 uP-systeem (7 eu-
|          rokaarten) in prof. 19" rack incl. voeding + 1 * BASF
|          6106 drive, alles met zeer uitgebreide documentatie.
|          Vr.pr. HFl. 500,=. Bijbehorende Terminal Beehive B-150
|          Vr.pr. HFl. 400,=. Voor de hardware-specialist nog an-
|          dere zaken in aanbouw t.e.a.b.
|          Tel.: 071-142488 (na 14.00 uur). P. Wieberneit.
|
| Gevr.  : overdruk van het artikel "A Computerchess Tutorial"
|          van N.D. Whaland in BYTE vol. 3 blz. 168 - 181, okt 78
|          Opsturen naar het redaktie-adres s.v.p.
|
| Te koop: Twee "Twente"-inbouw mono recorders zonder voeding ge-
|          bruikt bij de Elektuur JUNIOR-computer voor het aanma-
|          ken van bandjes voor de cassette-service van de redak-
|          tie. T.e.a.b. Tel.: 01807 - 19881 W.L. van Pelt.
|
==========================================================
| VAN DE REDAKTIE
| ===============
| Indien U van plan bent oude computertijdschriften weg te gooien,
| bedenk dan dat de redaktie er nog veel aan heeft. Stuur de over-
| bodige tijdschriften op naar het redaktie-adres. Wij zijn er erg
| blij mee.
| Wegens gebrek aan tijd verzoeken we ook Basicode-opnamen naar de
| redaktie op te sturen. Ook daarmee helpt U ons.
```

```
1 REM CØMPETITIESTANDEN
3 REM BY GERARD KEET
5 REM RØDENBURG NØ.3
7 REM 1965BL HEEMSKERK
9 REM TEL.02510-39763
10 GØTØ 20010
3000 REM INPUT TEAMNAAM
3010 PRINT:INPUT "TEAM: ";Z$
3020 IF LEN(Z$)=12 THEN 3090
3030 IF LEN(Z$)>12 THEN 3080
3040 Z$=Z$+" ":GØTØ 3020
3080 Z$=LEFT$(Z$,12)
3090 RETURN
3100 REM INPUT KLASSENR
3110 INPUT "KLASSENR: ";S$(0)
3120 IF LEN(S$(0))>5 THEN 3110
3130 IF LEN(S$(0))=5 THEN 3190
3140 S$(0)=S$(0)+" ":GØTØ 3130
3190 RETURN
5000 REM INITIALISEREN STANDEN (NULLEN NAAR INPUTBUFFER)
5010 INPUT "ZEKER WETEN DAT JE WILT INITIALISEREN <J/N>: ";K$
5015 IF K$<>"J" THEN 5090
5020 FØR I=AI TØ LA:PØKE I,48:NEXT
5090 RETURN
6000 REM TEAM WIJZIGEN
6010 GØSUB 3010:REM INPUT TEAMNAAM
6060 I=1
6070 IF S$(I)=Z$ THEN 6110
6080 I=I+1:IF I<=AT THEN 6070
6100 PRINT:PRINT "TEAM NIET IN KLASSE":GØTØ 6150
6110 INPUT "GESPEELD: ";S(I,1)
6120 INPUT "  PUNTEN: ";S(I,2)
6130 INPUT "    VØØR: ";S(I,3)
6140 INPUT "   TEGEN: ";S(I,4)
6150 RETURN
7000 REM TEAM VERWIJDEREN
7010 GØSUB 3010:REM INPUT TEAMNAAM
7020 I=1
7030 IF S$(I)=Z$ THEN 7060
7040 I=I+1:IF I<=AT THEN 7030
7050 PRINT:PRINT "TEAM NIET IN KLASSE":GØTØ 7090
7060 FØR J=I TØ AT-1:S$(J)=S$(J+1):FØR L=1 TØ 4:S(J,L)=S(J+1,L)
7070 NEXT L:NEXT J
7080 S$(AT)="000000000000":FØR L=1 TØ 4:S(AT,L)=0:NEXT L:AT=AT-1
7090 RETURN
8000 REM TEAM TØEVØEGEN
8010 IF AT<12 THEN 8030
8020 PRINT:PRINT "GEEN RUIMTE VØØR NIEUW TEAM":GØTØ 8090
8030 GØSUB 3010:REM INPUT TEAMNAAM
8032 NG=TRUE
8035 FØR I=1 TØ AT
8040 IF S$(I)<>Z$ THEN 8045
8043 NG=FALSE
8045 NEXT
8050 IF NG THEN 8080
8060 PRINT:PRINT "TEAM BESTAAT AL":GØTØ 8090
8080 AT=AT+1:S$(AT)=Z$
8090 RETURN
```

```
10000 REM LEES KLASSE IN TABELLEN S EN S$
10010 SKL=KL:FØR I=1 TØ 12:S$(I)="":FØR J=1 TØ 4:S(I,J)=0
10020 NEXT J:NEXT I
10070 KL=KL+5:AT=1
10090 FØR I=1 TØ 12:S$(AT)=S$(AT)+CHR$(PEEK(KL)):KL=KL+1:NEXT I
10130 IF S$(AT)="000000000000" THEN 10300
10135 H$="":FØR I=1 TØ 10:H$=H$+CHR$(PEEK(KL)):KL=KL+1:NEXT I
10170 G$=LEFT$(H$,2):P$=MID$(H$,3,2):V$=MID$(H$,5,3):T$=RIGHT$(H$,3)
10180 S(AT,1)=VAL(G$):S(AT,2)=VAL(P$):S(AT,3)=VAL(V$):S(AT,4)=VAL(T$)
10190 AT=AT+1:IF AT<13 THEN 10090
10300 AT=AT-1:KL=SKL:AA=2*(AT-1)
10310 RETURN
11000 REM ZØEK KLASSE IN BUFFER
11010 I=0
11020 H$="":FØR K=0 TØ 4:H$=H$+CHR$(PEEK(AI+I*(D+B*(C+E))+K)):NEXT K
11030 IF H$=S$(0) THEN 11100
11040 I=I+1:IF I<36 THEN 11020
11050 IF NWKL THEN 11080
11060 PRINT:PRINT:PRINT "KLASSE BESTAAT NIET":PRINT
11070 FØR R=1 TØ 400:NEXT
11080 NG=TRUE
11090 GØTØ 11190
11100 NG=FALSE:KL=AI+I*(D+B*(C+E))
11190 RETURN
12000 REM SCHRIJF KLASSE NAAR BUFFER
12010 H$=S$(0)
12020 FØR J=1 TØ 5:PØKE KL,ASC(H$):H$=RIGHT$(H$,5-J):KL=KL+1:NEXT J
12030 FØR I=1 TØ AT:H$=S$(I):FØR J=1 TØ 12:PØKE KL,ASC(H$)
12060 H$=RIGHT$(H$,12-J):KL=KL+1:NEXT J
12070 G$=STR$(S(I,1)):P$=STR$(S(I,2)):V$=STR$(S(I,3)):T$=STR$(S(I,4))
12080 G$=RIGHT$(G$,LEN(G$)-1):P$=RIGHT$(P$,LEN(P$)-1)
12100 V$=RIGHT$(V$,LEN(V$)-1):T$=RIGHT$(T$,LEN(T$)-1)
12300 IF LEN(G$)=2 THEN 12320
12310 G$="0"+G$:GØTØ 12300
12320 IF LEN(P$)=2 THEN 12340
12330 P$="0"+P$:GØTØ 12320
12340 IF LEN(V$)=3 THEN 12360
12350 V$="0"+V$:GØTØ 12340
12360 IF LEN(T$)=3 THEN 12380
12370 T$="0"+T$:GØTØ 12360
12380 H$=G$+P$+V$+T$
12390 FØR J=1 TØ 10
12400 PØKE KL,ASC(H$):H$=RIGHT$(H$,10-J):KL=KL+1:NEXT J
12410 NEXT I
12420 IF AT=12 THEN 12480
12430 FØR I=AT+1 TØ 12:FØR J=1 TØ 22:PØKE KL,48:KL=KL+1:NEXT J:NEXT I
12480 RETURN
13000 REM STANDEN PRINTEN
13010 PRINT CHR$(12);
13020 GØSUB 3110:REM INPUT KLASSE
13025 PRINT
13030 GØSUB 11010:REM ZØEK KLASSE IN BUFFER
13040 IF NG THEN 13280
13050 GØSUB 10010:REM LEES KLASSE IN TABELLEN S EN S$
13060 FØR I=1 TØ AT-1:FØR J=I+1 TØ AT
```

```
13070 REM STAND BEPALEN
13080 IF S(I,2)<S(J,2) THEN 13160
13090 IF S(I,2)>S(J,2) THEN 13200
13100 IF S(I,1)>S(J,1) THEN 13160
13110 IF S(I,1)<S(J,1) THEN 13200
13120 IF S(I,3)-S(I,4)<S(J,3)-S(J,4) THEN 13160
13130 IF S(I,3)-S(I,4)>S(J,3)-S(J,4) THEN 13200
13132 IF S(I,4)<>0 AND S(J,4)<>0 THEN 13140
13134 IF S(I,4)<>0 THEN 13160
13136 GØTØ 13200
13140 IF S(I,3)/S(I,4)<S(J,3)/S(J,4) THEN 13160
13150 GØTØ 13200
13160 H$=S$(I):S$(I)=S$(J):S$(J)=H$:REM SWAP S$(I),S$(J)
13170 FØR K=1 TØ 4:H=S(I,K):S(I,K)=S(J,K):S(J,K)=H
13175 NEXT K:REM SWAP S(I),S(J)
13200 NEXT J
13210 NEXT I
13220 FØR J=1 TØ AT:IF J<10 THEN PRINT " ";
13232 PRINT J;TAB(3) S$(J);TAB(25);:IF S(J,1)<10 THEN PRINT " ";
13234 PRINT S(J,1);TAB(30);:IF S(J,2)<10 THEN PRINT " ";
13236 PRINT S(J,2);TAB(35);:IF S(J,3)<100 THEN PRINT " ";
13238 IF S(J,3)<10 THEN PRINT " ";
13240 PRINT S(J,3);TAB(38) "-";:IF S(J,4)<100 THEN PRINT " ";
13242 IF S(J,4)<10 THEN PRINT " ";
13244 PRINT S(J,4)
13250 NEXT J:PRINT
13256 INPUT "KAMPIØENSINFØRMATIE? <J/N>: ";K$
13257 IF K$<>"J" THEN 13260
13258 GØSUB 30010:REM KAMPIØENSINFØ
13260 INPUT "NØG EEN KLASSE PRINTEN? <J/N> : ";K$
13270 IF K$="J" THEN 13010
13280 RETURN
14000 REM UITSLAGEN INVØEREN
14010 PRINT CHR$(12)
14020 INPUT "KLASSE <*=STØP> : ";S$(0)
14030 IF S$(0)="*" THEN 14650
14032 IF LEN(S$(0))>5 THEN 14020
14035 IF LEN(S$(0))=5 THEN 14040
14037 S$(0)=S$(0)+" ":GØTØ 14035
14040 GØSUB 11010:REM ZØEK KLASSE IN INPUTBUFFER
14050 IF NG THEN 14010
14060 GØSUB 10010:REM LEES KLASSE IN TABELLEN S EN S$
14070 INPUT "THUIS: ";T$:IF RIGHT$(T$,1)="*" THEN 14070
14090 INPUT "UIT  : ";U$:IF RIGHT$(U$,1)="*" THEN 14090
14110 INPUT "VØØR : ";V:INPUT "TEGEN: ";T
14130 IF LEN(T$)=12 THEN 14170
14140 IF LEN(T$)<12 THEN 14160
14150 T$=LEFT$(T$,12):GØTØ 14170
14160 T$=T$+" ":GØTØ 14130
14170 IF LEN(U$)=12 THEN 14210
14180 IF LEN(U$)<12 THEN 14200
14190 U$=LEFT$(U$,12):GØTØ 14210
14200 U$=U$+" ":GØTØ 14170
14210 I=1:J=1
14230 IF S$(I)=T$ THEN 14280
14240 I=I+1:IF I<=AT THEN 14230
```

```
14260 PRINT:PRINT "THUISTEAM NIET IN KLASSE":GØTØ 14070
14280 IF S$(J)=U$ THEN 14330
14290 J=J+1:IF J<=AT THEN 14280
14310 PRINT:PRINT "UITTEAM NIET IN KLASSE":GØTØ 14070
14330 IF S(I,1)+1<=AA THEN 14360
14340 PRINT:PRINT "GESPEELDE WEDSTRIJDEN THUISTEAM TE HØØG"
14350 GØTØ 14070
14360 IF S(J,1)+1<=AA THEN 14390
14370 PRINT:PRINT "GESPEELDE WEDSTRIJDEN UITTEAM TE HØØG"
14380 GØTØ 14070
14390 IF S(I,3)+V<=999 THEN 14420
14400 PRINT:PRINT "TØTAAL DØELPUNTEN-VØØR THUISTEAM TE HØØG"
14410 GØTØ 14070
14420 IF S(I,4)+T<=999 THEN 14450
14430 PRINT:PRINT "TØTAAL DØELPUNTEN-TEGEN THUISTEAM TE HØØG"
14440 GØTØ 14070
14450 IF S(J,3)+T<=999 THEN 14480
14460 PRINT:PRINT "TØTAAL DØELPUNTEN-VØØR UITTEAM TE HØØG"
14470 GØTØ 14070
14480 IF S(J,4)+V<=999 THEN 14510
14490 PRINT:PRINT "TØTAAL DØELPUNTEN-TEGEN UITTEAM TE HØØG"
14500 GØTØ 14070
14510 S(I,1)=S(I,1)+1:S(J,1)=S(J,1)+1
14520 S(I,3)=S(I,3)+V:S(I,4)=S(I,4)+T
14530 S(J,3)=S(J,3)+T:S(J,4)=S(J,4)+V
14540 IF V<T THEN 14580
14550 IF V>T THEN 14600
14560 S(I,2)=S(I,2)+1:S(J,2)=S(J,2)+1:GØTØ 14610
14580 S(J,2)=S(J,2)+2:GØTØ 14610
14600 S(I,2)=S(I,2)+2
14610 INPUT "NØG MEER VØØR DEZE KLASSE? <J/N> : ";K$
14620 IF K$="J" THEN 14070
14630 GØSUB 12010:REM SCHRIJF KLASSE NAAR BUFFER
14640 GØTØ 14010
14650 RETURN
15000 REM KLASSE VERWIJDEREN
15010 PRINT CHR$(12);:GØSUB 3110:REM INPUT KLASSE
15030 GØSUB 11010:REM ZØEK KLASSE IN BUFFER
15040 IF NG THEN 15090
15050 INPUT "ZEKER WETEN DAT DEZE KLASSE WEG MØET? <J/N> : ";K$
15060 IF K$<>"J" THEN 15090
15070 FØR I=KL TØ LA-F:PØKE I,(PEEK(I+F)):NEXT I
15080 FØR I=LA+1-F TØ LA+1:PØKE I,48:NEXT I
15090 RETURN
16000 REM KLASSE WIJZIGEN
16010 PRINT CHR$(12);:GØSUB 3110:REM I•PUT KLASSE
16030 GØSUB 11010:REM ZØEK KLASSE IN BUFFER
16040 IF NG GØTØ 16010
16050 GØSUB 10010:REM LEES KLASSE IN TABELLEN S EN S$
16060 PRINT CHR$(12);"KLASSE: ";S$(0):PRINT:PRINT
16070 PRINT "**************************"
16080 PRINT "* KEUZESCHERM WIJZIGEN *"
16090 PRINT "*                      *"
16100 PRINT "* 1 KLASSENR           *"
16110 PRINT "* 2 TEAM TØEVØEGEN     *"
16120 PRINT "* 3 TEAM VERWIJDEREN   *"
16130 PRINT "* 4 TEAM WIJZIGEN      *"
16140 PRINT "* 5 STØP                *"
16150 PRINT "**************************".
```

```
16160 PRINT:INPUT "KEUZE: ";K
16170 ØN K GØSUB 3110,8010,7010,6010
16180 IF K>=5 THEN 16230
16190 INPUT "ANDERE KLASSE WIJZIGEN? <J/N> :";K$
16200 IF K$<>"J" THEN 16060
16210 GØSUB 12010:REM SCHRIJF KLASSE NAAR BUFFER
16220 GØTØ 16010
16230 GØSUB 12010:REM SCHRIJF KLASSE NAAR BUFFER
16240 RETURN
17000 REM KLASSE TØEVØEGEN
17010 IF PEEK(LA+1-F)=48 THEN 17050
17020 PRINT "GEEN RUIMTE VØØR NIEUWE KLASSE":PRINT
17030 FØR I=1 TØ 500:NEXT:GØTØ 17250
17050 PRINT CHR$(12);:INPUT "KLASSENUMMER(5 PØS): ";S$(0)
17052 IF RIGHT$(S$(0),1)="*" ØR LEN(S$(0))>5 THEN 17050
17053 IF LEN(S$(0))=5 THEN 17055
17054 S$(0)=S$(0)+" ":GØTØ 17053
17055 NWKL=TRUE:GØSUB 11010:REM ZØEK KLASSE IN INPUTBUFFER
17056 IF NG THEN 17060
17058 PRINT "KLASSE BESTAAT AL":FØRI=1TØ500:NEXT:GØTØ17250
17060 INPUT "AANTAL TEAMS: ";AT:IF AT<=12 THEN 17090
17080 PRINT "TEVEEL TEAMS, MAX 12":GØTØ 17060
17090 FØR I=1 TØ AT
17100 PRINT "TEAM ";I;TAB(8)":";:INPUT " "; S$(I)
17120 IF RIGHT$(S$(I),1)="*" THEN 17100
17130 IF LEN(S$(I))=12 THEN17170
17140 IF LEN(S$(I))=<12 THEN 17160
17150 S$(I)=LEFT$(S$(I),12): GØTØ 17170
17160 S$(I)=S$(I)+" ": GØTØ 17130
17170 NEXT I
17180 I=0
17190 IF PEEK(AI+I*(D+B*(C+E)))=48 THEN 17210
17200 I=I+1:GØTØ 17190
17210 KL=AI+I*(D+B*(C+E)):GØSUB 12010:REM SCHRIJF KLASSE NAAR BUFFER
17230 INPUT 'NØG EEN NIEUWE KLASSE? <J/N> : ";K$
17240 IF K$="J" THEN 17010
17250 NWKL=FALSE
17290 RETURN
18000 REM STANDEN INLEZEN
18010 PØKE 6777,1:X=USR(&"0B02",0):X=USR(&"14BC",0)
18090 RETURN
19000 REM STANDEN WEGSCHRIJVEN
19010 PØKE 6777,1:PØKE 6768,0:PØKE 6769,96:PØKE 6770,0:PØKE 6771,144
19040 X=USR(&"09DF",0):X=USR(&"14BC",0)
19090 RETURN
20000 REM HØØFDPRØGRAMMABLØK
20010 A=36:B=12:C=12:D=5:E=10:AI=24576:DIM S$(12):DIM S(12,4)
20020 F=D+B*(C+E):LA=34259
20090 TRUE=-1:FALSE=0
25010 PRINT CHR$(12);
25020 PRINT "**********************************"
25030 PRINT "**                              **"
25040 PRINT "**   KEUZESCHERM STANDEN ØDIN   **"
25050 PRINT "**                              **"
```

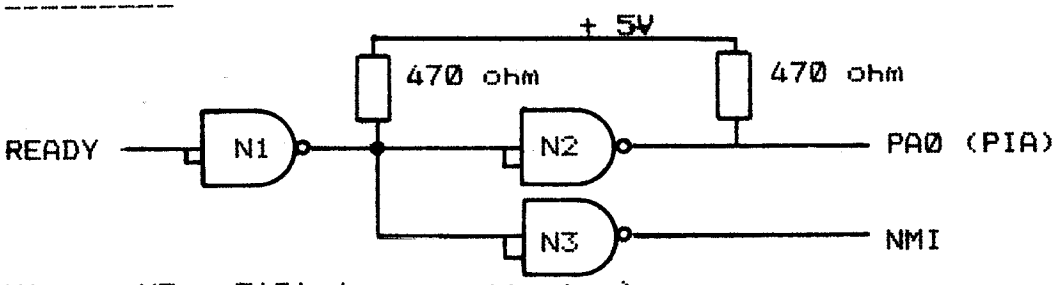
```
25060 PRINT "**    1 STANDEN INLEZEN            **"
25070 PRINT "**    2 KLASSE TØEVØEGEN           **"
25080 PRINT "**    3 KLASSE WIJZIGEN            **"
25090 PRINT "**    4 KLASSE VERWIJDEREN         **"
25100 PRINT "**    5 UITSLAGEN INVØEREN         **"
25110 PRINT "**    6 STANDEN PRINTEN            **"
25120 PRINT "**    7 STANDEN WEGSCHRIJVEN       **"
25130 PRINT "**    8 INITIALISEREN STANDEN      **"
25135 PRINT "**    9 STØPPEN                    **"
25140 PRINT "**                                 **"
25150 PRINT "*************************************"
25160 PRINT:PRINT:INPUT "        KEUZE: ";K
25180 ØN K GØSUB 18010,17010,16010,15010,14010,13010,19010,5010
25190 IF K<9 THEN 25010
25200 END
30000 REM KAMPIØENSINFØRMATIE
30010 I=1:JE=(MID$(S$(0),2,1)="W" ØR MID$(S$(0),2,1)="P")
30020 IF LEFT$(S$(I),4)="ØDIN" THEN 30040
30030 I=I+1:IF I<=AT THEN 30020
30040 ØD=I:NK=FALSE:KA=TRUE:GB=1:PB=0:I=1
30050 INPUT "VØLGENDE TEGENSTANDER ØDIN= ";TS$
30060 IF I>AT ØR NK THEN 30250
30070 IF I<>ØD AND S(I,2)-S(ØD,2)>2*(AA-S(ØD,1)) THEN 30200
30080 IF LEFT$(S$(I),LEN(TS$))=TS$ THEN 30130
30090 IF NØT S(I,2)/S(I,1)>=PB/GB ØR I=ØD THEN 30140
30100 PB=S(I,2):GB=S(I,1):GØTØ 30140
30130 GT=S(I,1):PT=S(I,2)
30140 IF NØT KA ØR I=ØD THEN 30210
30160 IF JE THEN 30180
30170 KA=(S(ØD,2)-S(I,2)>2*(AA-S(I,1))):GØTØ 30210
30180 KA=(S(ØD,2)-S(I,2)>=2*(AA-S(I,1))):GØTØ 30210
30200 NK=TRUE
30210 I=I+1::GØTØ 30060
30250 PRINT: IF NØT NK THEN 30280
30270 PRINT "ØDIN KAN GEEN KAMPIØEN MEER WØRDEN":GØTØ 30450
30280 IF NØT KA THEN 30300
30290 PRINT "ØDIN IS AL KAMPIØEN": GØTØ 30450
30300 IF NØT JE THEN 30340
30310 IF GB<AA AND NØT (S(ØD,2)-1-PB)>=2*(AA-GB-1) THEN 30370
30315 IF GB=AA AND NØT (S(ØD,2)+1-PB)>=0 THEN 30370
30320 IF NØT (S(ØD,2)-PT)>=2*(AA-GT-1) THEN 30370
30330 GØTØ 30360
30340 IF GB<AA AND NØT (S(ØD,2)-1-PB)>2*(AA-GB-1) THEN 30370
30345 IF GB=AA AND NØT (S(ØD,2)+1-PB)>0  THEN 30370
30350 IF NØT (S(ØD,2)-PT>2*(AA-GT-1)) THEN 30370
30360 PRINT "ØDIN KAMPIØEN BIJ GELIJKSPEL":GØTØ 30450
30370 IF NØT JE THEN 30410
30380 IF GB<AA AND NØT (S(ØD,2)-PB)>=2*(AA-GB-1) THEN 30440
30385 IF GB=AA AND NØT (S(ØD,2)+2-PB)>=0 THEN 30440
30390 IF NØT (S(ØD,2)+2-PT)>=2*(AA-GT-1) THEN 30440
30400 GØTØ 30430
30410 IF GB<AA AND NØT (S(ØD,2)-PB)>2*(AA-GB-1) THEN 30440
30415 IF GB=AA AND NØT (S(ØD,2)+2-PB)>0 THEN 30440
30420 IF NØT (S(ØD,2)+2-PT)>2*(AA-GT-1) THEN 30440
30430 PRINT "ØDIN KAMPIØEN BIJ WINST":GØTØ 30450
30440 PRINT "ØDIN KAN NØG GEEN KAMPIØEN WØRDEN"
30450 RETURN
ØK
```

****
********** EEN WAIT-INGANG VOOR DE JUNIOR **********
****

Door : Roger Langeveld
Laan van Middenburg 26 A
2275 CC VOORBURG.
Tel.: 070 - 861128

Dit is een ingang die ik nodig heb bij bepaalde funkties van mijn Graffix 85 Video Board. Bedoelde funkties vragen meer tijd dan de computer wil geven. Worden deze funkties aangesproken, en het videoboard komt in tijdnood, dan maakt het videoboard z'n ready-uitgang laag, waardoor de computer een seintje krijgt om even te wachten met het verzenden van informatie naar het board toe. Op de 6502 microprocessor zit geen "WAIT"-ingang. Voor dit probleem moet er dus een andere oplossing komen.
Het stil zetten van de 6502 microprocessor is niet mogelijk, tenzij door de spanning te verwijderen. Dat lijkt echter niet de meest elegante benadering. Ook het ready-signaal van het videoboard met de NMI verbinden en de NMI-vector gericht op een wachtroutine is wat problematisch, ook al werkt het zonder meer, al moet bekend zijn hoe lang er gewacht moet worden. Voor mij lag toen de oplossing in het kombineren van NMI en PIA-poort. Er is echter wel een stukje hardware nodig.
De hardware en het programma spreken voor zich. Let wel even op dat de NAND-poorten van het type open collector zijn (d.w.z. dat er uitgangen aan elkaar gekoppeld mogen worden, zonder dat er rook komt). De weerstand welke bij de andere NAND-poorten zit (N1 en N2) hoeft bij N3 niet. Deze is er al, namelijk op de basiskaart.

Hardware:



N1 ... N3 = 7401 (open collector)

Software:

```
0200  48          PHA        BRENG INHOUD ACCU NAAR
0201  A9 FE       LDAIM $FE  STACK EN MAAK PA0 INGANG
0203  8D 81 1A    STA PADD
0206  AD 80 1A    LDA PAD    LEES INHOUD DATAPOORT
0209  29 01       ANDIM $01
020B  F0 F9       BEQ        IS READY HOOG ?
020D  20 BC 14    JSR RESET  TTY
                             HAAL INHOUD ACCU
0210  68          PLA        WEER OP EN KEER WEER TERUG
0211  40          RTI        NAAR PROGRAMMA.
```

NMI-vector op $0200.

TOEPASSING "runtime" VOOR DE APPLESOFT COMPILER 'TASC'.
==========

Het bezig zijn met Basic- en Basicodeprogramma's heeft Uw redakteur enigszins geprikkeld. Basic is een hogere programmeertaal, waarvan velen al wel weten dat deze per definitie langzamer is dan machinetaal. Nu kun je het in Basic ontwikkelde programma opnieuw schrijven, en dan in machinetaal. Indien je echter in het bezit bent van aan APPLE-computer met disk-operating en tevens van de Applesoft Compiler TASC, dan is er nog een tussenoplossing denkbaar om de snelheid van machinetaal te bereiken. Met de compiler maak je van het Basic-programma dat in je geheugen staat een machinetaal-file, dat echter nog niet in staat is om te worden gebruikt. Het moet samenwerken met de file RUNTIME, die op de TASC-schijf staat.
Hoe kun je nu te werk gaan? Je maakt een lege schijf aan door deze te formatteren en met INIT meteen een HELLO-programma erop te schrijven. Van de TASC-schijf ga je nu de binaire file RUNTIME overbrengen naar de lege schijf. Compileer daarna met TASC het in machinetaal om te zetten Basic- of Basicode-programma en zet dit eveneens op de nieuwe schijf. Wil je het bewuste programma nu 'runnen', dan moet je wel eerst met BLOAD RUNTIME de zaak voorbereiden. Daarna pas kan men met BRUN (filename) de routine gebruiken. Het verschil in snelheid is erg groot. In sommige van dit soort bewerkingen verloopt de interaktie met de gebruiker zo snel, m.a.w. de op het scherm verschijnende teksten vliegen voorbij en zijn niet te lezen, dat de wens zal ontstaan om binnen de routines weer wachtloops aan te brengen.
TASC is in de handel verkrijgbaar en het is de moeite waard het aan te schaffen.

Om nu de gebruikersvriendelijkheid van de schijf te vergroten heb ik Frans Verberkt gevraagd zodanige voorzieningen te treffen dat bij het opbooten van mijn (nieuwe) schijf RUNTIME meteen wordt geladen, waarna is direkt de machinetaalversie van het Basic- of Basicode-programma dat ik verkies kan 'brunnen'.
Frans bedacht een oplossing, maar stuitte eerst op een probleem: RUNTIME is geschreven in het geheugengebied $0803 t/m $17AF, terwijl in dit gebied het HELLO-programma werkzaam is! Een gevaarlijk gebied dus. Eerst moest hij dit verhelpen, en daarom schreef hij een kleine machinetaalroutine OVERZET RUNTIME, met startadres $0320 (#800) in de assembler van Carl Moser. Het HELLO-programma laadt RUNTIME naar $3803 t/m $47AF. OVERZET RUNTIME zet hem naar $0803 t7m $17AF, en zorgt tevens voor een goede afsluiting. Men kan immers niet meer terug naar Basic (HELLO). Het bijbehorende HELLO-programma zorgt dan voor een goede afwikkeling, waarmee de gecompileerde files op schijf direkt gebruiksgereed klaar staan.

AS L

```
0005            .LS
0010 ;OVERZET RUNTIME  29 DECEMBER 1985
0020
0030 ;Frans Verberkt
0040 ;Hillekensacker 12 - 10
0050 ;6546 KG  NIJMEGEN
0060
0070 ;   080 - 779555
0080
0090
0100            .OS
0110
0120
0130
0140
0150 ;***********************
0160
0170 ;PARA - METERS RUNTIME
0180
0190 ;***********************
0200
0210 BEGIN      .DE $0803         ;WERKGEBIED
0220 EINDE      .DE $17AF
0230
0240 MEM        .DE $3803         ;BEGIN TYDELIJK GEBIED
0250
```

```
0260
0270 ;*************
0280
0290 ;PAGE ZERO
0300
0310 ;*************
0320
0330 VANL        .DE $3C
0340 VANH        .DE $3D
0350
0360 TOTL        .DE $3E
0370 TOTH        .DE $3F
0380
0390 NAARL       .DE $42
0400 NAARH       .DE $43
0410
0420
0430 ;***************
0440
0450 ; APPLE ROUTINES
0460
0470 ;***************
0480
0490 DOSKOUD     .DE $03D3          ;KOUDE START D.O.S. !!
0500
0510 MOVE        .DE $FE2C
0520
0530
0540 ;*****************
0550
0560 ;OVERZET RUNTIME
0570
0580 ;*****************
0590
0600             .BA $0320          ;CALL 800
0610
0620             LDA #L,MEM
0630             STA *VANL
0640             LDA #H,MEM
0650             STA *VANH
0660
0670             LDA #L,MEM+EINDE-BEGIN
0680             STA *TOTL
0690             LDA #H,MEM+EINDE-BEGIN
0700             STA *TOTH
0710
0720             LDA #L,BEGIN
0730             STA *NAARL
0740             LDA #H,BEGIN
0750             STA *NAARH
0760
0770             LDY #$00           ;NOODZAKELIJK VOOR MOVE!!
0780             JSR MOVE           ;OM GEEN 'GEKKE' LIST TE KRIJGEN.
0790             JMP DOSKOUD        ;HEEFT SOORTGELIJK EFFEKT
0800             ;                   ALS COMMANDO (NEW).
0810
0820
```

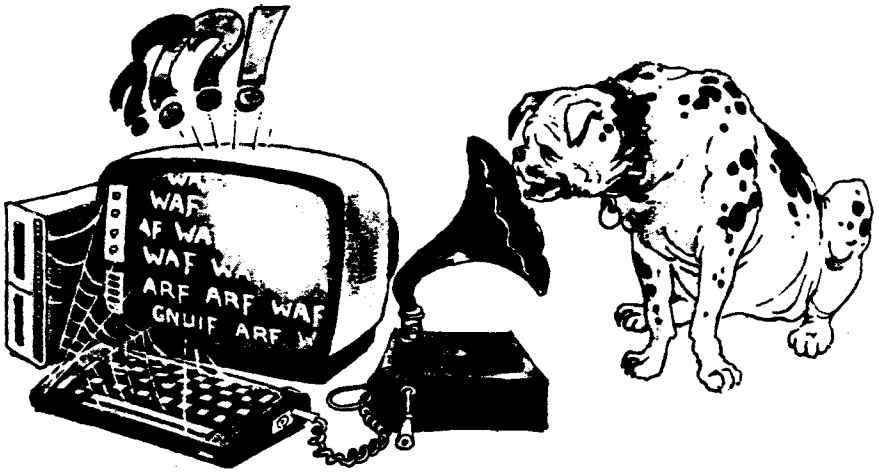
```
0320- A9 03
0322- 85 3C
0324- A9 38
0326- 85 3D

0328- A9 AF
032A- 85 3E
032C- A9 47
032E- 85 3F

0330- A9 03
0332- 85 42
0334- A9 08
0336- 85 43

0338- A0 00
033A- 20 2C FE
033D- 4C D3 03
```

//0000,0340,0340



```
]LIST

1000  REM
 HELLO RUNTIME

1010  GOSUB 2010: REM  INIT
1020  GOSUB 3010: REM  TEST FILES
1030  GOSUB 4010: REM  BOODSCHAP
1040  GOSUB 5010: REM  HAAL RUNTIME
1050  GOSUB 6010: REM  CATALOG
1060  GOSUB 7010: REM  OVERZETTEN
1070  END
2000  REM
 INIT

2010 DI$ =  CHR$ (4)
2020  RETURN
3000  REM
 TEST FILES

3010  HOME
3020 FI$ = "RUNTIME"
3030  GOSUB 3510: REM  TEST FI$
3040 FI$ = "OVERZET RUNTIME"
3050  GOSUB 3510: REM  TEST FI$
3060  RETURN
3500  REM
 TEST FI$

3510  ONERR  GOTO 3710: REM  FOUT FI$
3520  PRINT DI$;"VERIFY";FI$
3530  POKE 216,0: REM  CLEAR ONERR
3540  RETURN
3700  REM
 FOUT FI$

3710  POKE 216,0: REM  CLEAR ONERR
3720 FC =  PEEK (222): REM  FOUTCODE
3730  IF FC = 6 THEN 3770
3740  PRINT
3750  PRINT "DISK FOUT NR. #";FC
3760  GOTO 3520
3770  PRINT "FILE '";FI$;"' IS NIET AANWEZIG"
3780  GOTO 3520
4000  REM
 BOODSCHAP

4010  HOME
4020  PRINT "laad of type geen BASIC programma"
4030  PRINT "Indien U dit per ongeluk doet  CALL800"
4040  RETURN
5000  REM
 HAAL RUNTIME

5005  REM  LAADT RUNTIME OP $3803 T/M $47AF.
5010  PRINT DI$;"BLOAD RUNTIME,A$3803"
5020  PRINT "RUNTIME IS ";
5030  INVERSE
5040  PRINT " GELADEN ";
5050  NORMAL
5060  PRINT " van $3803 t/m $47AF"
5070  RETURN
6000  REM
 CATALOG

6010  POKE 49168,0: REM  CLEAR TOETS
6020  POKE 34,3: REM  BOVENGRENS = REGEL 4
6030  PRINT DI$;"CATALOG"
6040  RETURN
7000  REM
 OVERZETTEN

7010  PRINT : PRINT "RUNTIME ";
7020  INVERSE
7030  PRINT " GAAT NAAR ";
7040  NORMAL
7050  PRINT " $0803 t/m $17AF"
7060  POKE 34,0: REM  HERSTEL BOVENGRENS
7070  PRINT DI$;"BRUN OVERZET RUNTIME"
7080  RETURN
```

Nevenstaande inzending is een Basicprogramma geschreven voor een PRIME computer, en derhalve afwijkend van Microsoft Basic. De IF .. THEN .. DO .. ELSE struktuur behoeft geen nadere verklaring, deze spreekt voor zich.
CHR$(154) wordt gebruikt om het schera schoon te maken. LIN(5) betekent dat op de printer of op het schera 5 regels worden overgeslagen voordat de volgende regel wordt afgedrukt. Bij gebruik van Microsoft Basic kan veelal het LET-statement vervallen.

Het publiceren van dit programma gebeurt om een aantal redenen:
- het wil iets laten zien van en voor beginners;
- er is in zekere zin struktuur te herkennen;
- het nodigt anderen uit verbeterde versies in te leveren;
- het toont dat Basic's niet aan elkaar gelijk zijn;
- het nodigt leerlingen van scholen uit tot het inzenden van hun programma's.

De redaktie ziet graag dit soort materiaal binnenkomen, in Basic, FORTH, COMAL, PASCAL, machinetaal, etc. etc.

COMMODORE-INFO

Er zijn nogal eens mensen die zich geen raad weten met hun Coomodore als deze gerepareerd moet worden. Van ingewijden vernamen wij dat de firma Escom, Antoniuslaan 1, Hendrik Ido Ambacht, deze machines repareert. Naar verluidt zouden er eenheidsprijzen worden berekend.

DISKETTE VERZENDDOOSJES

De firma ATLANTA, een bekende naam voor gebruikers van kantoormaterialen verkoopt - aldus leerde mij Frank Manshande door toezending van een diskette - handige disketteverzenddoosjes voor verzending via PTT. Ze zijn gemaakt van stevig golfkarton, zijn door perforatie voorgevouwen, en kunnen enkele diskettes bevatten.

CALL TO MICRO-ADE USERS

Fernando Lopes, Porto, Portugal.

I'm using Micro-ADE to do listings of long programs. I like to use labels with more characters than those allowed (A-Z), like numbers, punctuation marks, lower case, etc.
These labels work very well, but they seem strang when I do the TABLE listings. The problem is in the PACK and UNPACK routines.
If there is some club member that can write a patch to fix this need, despite the greater wasting of memory by the TABLES, please write to the editor.

KOPY VAN HET BEELDSCHERM VOOR C-64

Veelal zijn het de kleine programma's die zo ongelooflijk veel kunnen. Er bestaan veel variaties om een beeldscherakopy (geen HiRes) op een printer af te drukken. Maar dit goed werkende programma is 't kleinste. Opnemen in het programma waarvan u een beeldscherakopy wilt.

```
20000 OPEN1,4:OPEN2,3:PRINTCHR$(19)
20010 FORT=0TO999
20020 GET#2,A$:PRINT#1,A$;
20030 NEXT:CLOSE2:CLOSE1
```

BASIC PROGRAMMA "POSITIEVE GETALLEN"

Gert Kwetters en Bart van Pelt zijn
1e jaars student aan de
Hogere Economische School, Rotterdam

```
10  REM AUTEURS: GERT KWETTERS/BART VAN PELT
15  REM        : AUTOMATISCHE INFORMATIE VERWERKING (AIV)
16  REM          Kralingse Zoom 91, 3063 ND Rotterdam.
17  REM
20  REM EEN RIJ VAN DERTIG GETALLEN IS GEGEVEN,
30  REM BESTAANDE UIT POSITIEVE EN NEGATIEVE GETALLEN
40  REM DATA STRUKTUUR:   READ/DATA
50  REM INITIALISATIES:
60  REM                   M=MINIMUM
70  REM                   T=TELLER
80  REM                   S=SOM
90  LET M=1E+06
100 LET T=0
110 LET S=0
120 GOSUB 620
130 PRINT "* * * * * * * * * * * * * * * * * * * * * *"
140 PRINT "*DIT PROGRAMMA ZAL U UIT EEN RIJ VAN 30 *"
150 PRINT "*GETALLEN DE VOLGENDE GEGEVENS LEVEREN: *"
160 PRINT "*DE POSITIEVE GETALLEN OP EEN REGEL     *"
170 PRINT "*HET AANTAL POSITIEVE GETALLEN          *"
180 PRINT "*DE SOM VAN DE POSITIEVE GETALLEN       *"
190 PRINT "*HET KLEINSTE POSITIEVE GETAL           *"
200 PRINT "* * * * * * * * * * * * * * * * * * * * * *"
210 PRINT LIN(5)
220 INPUT "WILT U JA EN RETURN INDRUKKEN OM VERDER TE GAAN ?";I$
230 IF I$="JA" THEN DO
240                 PRINT CHAR(154)
250                 DO END
260 ELSE DO
270      GOTO 220
280      DO END
290 REM                   VERWERKING
300 GOSUB 630
310 PRINT
320 PRINT "DE POSITIEVE GETALLEN OP EEN RIJ:";
330 FOR I=1 TO 30
340     READ G
350     IF G)0 THEN DO
360                 PRINT G;
370                 LET T=T+1
380                 LET S=S+G
390                 IF G(M THEN DO
400                             LET M=G
410                             DO END
420                 DO END
430 NEXT I
440 REM                   RESULTATEN AFDRUKKEN
450      IF T=0 THEN DO
460                  PRINT "DEZE RIJ BEVAT GEEN POSITIEVE GETALLEN"
470                  DO END
480 ELSE DO
490 PRINT
500 PRINT
510 PRINT "HET AANTAL POSITIEVE GETALLEN IS: ",T
520 PRINT
530 PRINT "DE SOM VAN DE POSITIEVE GETALLEN IS: ",S
540 PRINT
550 PRINT "HET KLEINSTE POITIEVE GETAL IS: ",M
560 DO END
570 DATA 1,2,3,4,5,6,7,8,9,10,-1,-2,-3,-4,-5,-6,-7,-8,-9,-10,11,12,13,14,15
580 DATA -11,-12,-13,-14,-15
590 PRINT LIN(5)
600 PRINT "                   EINDE PROGRAMMA"
610 END
620 PRINT CHAR(154)
630 PRINT LIN(5)
640 RETURN
```

OCTOPUS PART 1

```
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260                ;
270                ;
280 3A7E                    *=$3A7E
290                ;
300
310
320
330                ;
340 001B=                   INBUF=$001B  BASIC'S INPUT BUFFER
350                                      ;STARTS HERE
360                ;
370 0020=                   ZR00=INBUF+$05 ;DRIVE NUMBER (A=1,B=2, ..)
380 0021=                   ZR01=INBUF+$06 ;DRIVE TO COPY TO
390 0022=                   ZR02=INBUF+$07 ;FIRST TRACK TO BE COPIED
400 0023=                   ZR03=INBUF+$08 ;LAST TRACK + 1
410 0024=                   ZR04=INBUF+$09 ;TEMPORARY STORAGE FOR SUB2
420 0025=                   ZR05=INBUF+$0A ;1. BYTE OF TRACK ZERO
430 0026=                   ZR06=INBUF+$0B ;2. BYTE
440 0027=                   ZR07=INBUF+$0C ;3. BYTE
450 0028=                   ZR08=INBUF+$0D ;DISC FLAG
460 0029=                   ZR09=INBUF+$0E ;PAGE-COUNTER
470 002A=                   ZR0A=INBUF+$0F ;SECTOR-COUNTER
480 002B=                   ZR0B=INBUF+$10 ;STORAGE FOR STACKPOINTER
490 002C=                   ZR0C=INBUF+$11 ;MAX TRACK NUMBER
500 002D=                   ZR0D=INBUF+$12 ;MIN TRACK NUMBER
510                ;
520
530
540
```

```
;
;DISKETTE COPIER VERSION 2.2
;   Version 1.0   : original
;   Version 2.0   : mod 0/Okt.85
;   Version 2.1   : code error
;   Version 2.2   : return->basic
;
;MODIFIED FOR SERIAL SYSTEM
; FOR DOUBLE-SIDED DISK-DRIVES
;
;VALID DRIVE-CONFIGURATION:
;   Drive 1  side  A  and  B
;   Drive 2  side  C  and  D
;
;Attention :
;    Changes have to be made, if
;    sides A  and  B  on differ-
;    ent Disk-Drives !!
;================================
;Wolfgang Tietsch
;Eilenburger Weg 11
;6800 Mannheim 31
;================================
;DECEMBER 1985
;================================
```

```
;====================
;PAGE ZERO LOCATIONS
;====================
```

```
;================================
;ADRESSES OF THE FLOPPY CONTROLLE
;================================
```

```
550                     ;
560 C000=               FLOCON=$C000 ;ADRESS OF THE FLOPPY CONTR.
570                     ;
580                             ;PIA ADRESSES
590                     ;
600 C000=               DRA =FLOCON  ;DATA/DATA DIR REGISTER A
610 C002=               DRB=DRA+2    ;DATA/DATA DIR REGISTER B
620                     ;
630                             ;ACIA ADRESSES
640                     ;
650 C010=               CACIA=DRA+$10 ;ACIA CONTROL REGISTER
660 C011=               DACIA=DRA+$11 ;ACIA DATA REGISTER
670                     ;
680                             ;SPECIAL MEMORY LOCATIONS
690                     ;
700                     ;       TEMP1 = $48  POINTER TO FIRST FREE
710                     ;       TEMP2 = $00  RAM LOCATION
720                     ;       CMPFL1 = FLAG FOR COMPARE-MODE IN SUB7
730                     ;       CMPFL2 = FLAG FOR COMPARE-MODE IN SUB4
740                     ;
750                             ;====================
760                             ;DOS-KERNEL ADRESSES
770                             ;====================
780                     ;
790 00C7=               ZRC7=$00C7   ;BASIC COMMAND POINTER
800 00C8=               ZRC8=$00C8
810 00E1=               ZRE1=$00E1   ;OSIBAD
820 00E2=               ZRE2=$00E2   ;OSIBAD+1
830 00E5=               HSTTK=$00E5 ;MAX TRACK NUMBER
840                     ;
850 00FF=               MEMHI=$00FF ;FLOPPY LOAD POINTER
860 00FE=               MEMLO=MEMHI-1
870                     ;
880 07E0=               EXECUR=$07E0 ;EXECUTE NEXT BASIC STATEMEN
890                     ;
900 2300=               HIRAM=$2300 ;HIGHEST RAM PAGE FOR STORAGE
910 2321=               INDST=$2321 ;INPUT DISTRIBUTOR
920 2322=               OUTDST=$2322 ;OUTPUT DISTRIBUTOR
930 2340=               INECHO=$2340 ;SCREEN OUTPUT
940 265D=               TKNUM=$265D  ;TRACK NUMBER
950 265E=               SECTNM=$265E ;SECTOR NUMBER
960 265F=               PAGCNT=$265F ;PAGE COUNT
970 2AC6=               DSTX=$2AC6   ;CONSOLE TERMINAL NUMBER
980 2779=               MAXTRK=$2779 ;MAX ALLOWED TRACK NUMBER
990 2CE5=               BUFIND=$2CE5 ;INDEX COMMAND DOS BUFFER
1000 2E79=              BUFFER=$2E79 ;1 PAGE BUFFER FOR TR + SEC
1010                    ;
1020                            ;KERNEL SUBROUTINES
1030                    ;
1040 2663=              HOME=$2663   ;HOME HEAD
1050 267A=              DELAY=$267A  ;DELAY ROUTINE
1060 26BC=              SETTK=$26BC  ;MOVE HEAD TO TRACK IN ACCU
1070 271D=              INDEX=$271D  ;WAIT TILL INDEX END
1080 272E=              INACIA=$272E ;INIT ACIA FOR DATA TRANSF
1090 2754=              LDHEAD=$2754 ;LOAD HEAD
1100 2761=              UNLDHD=$2761 ;UNLOAD HEAD  1.ENTRY
1110 2763=              UNLOAD=$2763 ;UNLOAD HEAD  2.ENTRY
1120 2768=              INIT=$2768   ;INIT ALL TRACKS FROM 0-34
```

```
1130 2772=                      INIT1=$2772  ;INIT SOME TRACKS !
1140 27C2=                      BYTWRT=$27C2 ;WRITE A BYTE ON DISK
1150 27E1=                      DSKWRT=$27E1 ;WRITE SECTOR ON DISK
1160 29C6=                      SETDRV=$29C6 ;SET FOR DRIVE IN ACCU
1170 2A7D=                      EXCOM=$2A7D  ;EXECUTE A DOS COMMAND
1180 2C9B=                      OSINP=$2C9B  ;LOAD COM IN DOS BUFFER
1190 2CE4=                      BUFBYT=$2CE4 ;READ BYTE INTO ACCU
1200 2CF7=                      SWAP=$2CF7   ;SWAP PAGE 0 AND 1
1210 2D6A=                      CRLF=$2D6A   ;DO A CR AND LF
1220 2D73=                      STROUT=$2D73 ;PRINT FOLLOWING STRING
1230 2D92=                      PRTAHX=$2D92 ;PRINT BYTE IN ACCU AS ASCII
1240 2D9B=                      PRTHEX=$2D9B ;PRINT LOW NIBBLE IN ACCU
1250                ;
1260                            ;$3A79/7A  =  START OF PROGRAMM
1270                            ;$3A7B/7C  =  END   OF PROGRAMM
1280                            ;$3A7D     =  NUMBER OF TRACKS
1290                ;
1300                ;
1310                                  ;======================
1320                                  ;M A I N  R O U T I N E
1330                                  ;======================
1340                ;
1350                ;
1360 3A7E A92E      START     LDA #$2E      ;LOAD ADRESS OF DOS-COMMAND
1370 3A80 85E2                STA ZRE2      ;BUFFER ($2E1E) INTO PAGE
1380 3A82 A91E                LDA #$1E      ;ZERO POINTER OSIBAD
1390 3A84 85E1                STA ZRE1
1400 3A86 ADC62A              LDA DSTX      ;GET CONSOLE TERMINAL NUMBER
1410 3A89 8D2223              STA OUTDST    ;OUTPUT BYTE
1420 3A8C 8D2123              STA INDST     ;INPUT  BYTE
1430                ;         LDA #$3E      ;IS IN THE ORIGINAL CODE
1440                ;         LDA #$43      ;          "
1450                ;         JSR EXCOM     ;SET POINTER FOR DOS COM
1460 3A8F 20732D              JSR STROUT
1470 3A92 1B                  .BYTE $1B,$1C,$0D,$0A,$0A
1470 3A93 1C
1470 3A94 0D
1470 3A95 0A
1470 3A96 0A
1480 3A97 2D                  .BYTE '- Diskette copier -'
1480 3A98 20
1480 3A99 44
1480 3A9A 69
1480 3A9B 73
1480 3A9C 6B
1480 3A9D 65
1480 3A9E 74
1480 3A9F 74
1480 3AA0 65
1480 3AA1 20
1480 3AA2 63
1480 3AA3 6F
1480 3AA4 70
1480 3AA5 69
1480 3AA6 65
1480 3AA7 72
1480 3AA8 20
1480 3AA9 2D
1490 3AAA 0D                  .BYTE $0D,$0A,$0A
```

```
1490 3AAB 0A
1490 3AAC 0A
1500 3AAD 43                .BYTE 'Copy from which drive'
1500 3AAE 6F
1500 3AAF 70
1500 3AB0 79
1500 3AB1 20
1500 3AB2 66
1500 3AB3 72
1500 3AB4 6F
1500 3AB5 6D
1500 3AB6 20
1500 3AB7 77
1500 3AB8 68
1500 3AB9 69
1500 3ABA 63
1500 3ABB 68
1500 3ABC 20
1500 3ABD 64
1500 3ABE 72
1500 3ABF 69
1500 3AC0 76
1500 3AC1 65
1510 3AC2 20                .BYTE ' (A/B/C/D) ? ',$00
1510 3AC3 28
1510 3AC4 41
1510 3AC5 2F
1510 3AC6 42
1510 3AC7 2F
1510 3AC8 43
1510 3AC9 2F
1510 3ACA 44
1510 3ACB 29
1510 3ACC 20
1510 3ACD 3F
1510 3ACE 20
1510 3ACF 00
1520 3AD0 20D33C            JSR SUB1      ;CHECK VALID DRIVE NUMBER
1530 3AD3 8520              STA ZR00      ;DRIVE NUMBER
1540 3AD5 20732D            JSR STROUT
1550 3AD8 0D                .BYTE $0D,$0A
1550 3AD9 0A
1560 3ADA 43                .BYTE 'Copy to which drive '
1560 3ADB 6F
1560 3ADC 70
1560 3ADD 79
1560 3ADE 20
1560 3ADF 74
1560 3AE0 6F
1560 3AE1 20
1560 3AE2 77
1560 3AE3 68
1560 3AE4 69
1560 3AE5 63
1560 3AE6 68
1560 3AE7 20
1560 3AE8 64
1560 3AE9 72
1560 3AEA 69
```

```
1560 3AEB 76
1560 3AEC 65
1560 3AED 20
1570 3AEE 28                    .BYTE '(A/B/C/D)    ? ',$00
1570 3AEF 41
1570 3AF0 2F
1570 3AF1 42
1570 3AF2 2F
1570 3AF3 43
1570 3AF4 2F
1570 3AF5 44
1570 3AF6 29
1570 3AF7 20
1570 3AF8 20
1570 3AF9 20
1570 3AFA 3F
1570 3AFB 20
1570 3AFC 00
1580 3AFD 20D33C                JSR SUB1        ;CHECK VALID DRIVE NUMBER
1590 3B00 8521                  STA ZR01        ;DRIVE NUMBER
1600 3B02 200E3D                JSR SUBA        ;CHECK IF SAME DISC DRIVE
1610 3B05 20732D  TRACK         JSR STROUT
1620 3B08 0D                    .BYTE $0D,$0A
1620 3B09 0A
1630 3B0A 46                    .BYTE 'First track to copy '
1630 3B0B 69
1630 3B0C 72
1630 3B0D 73
1630 3B0E 74
1630 3B0F 20
1630 3B10 74
1630 3B11 72
1630 3B12 61
1630 3B13 63
1630 3B14 6B
1630 3B15 20
1630 3B16 74
1630 3B17 6F
1630 3B18 20
1630 3B19 63
1630 3B1A 6F
1630 3B1B 70
1630 3B1C 79
1630 3B1D 20
1640 3B1E 20                    .BYTE '       <0-34> ? ',$00
1640 3B1F 20
1640 3B20 20
1640 3B21 20
1640 3B22 20
1640 3B23 3C
1640 3B24 30
1640 3B25 2D
1640 3B26 33
1640 3B27 34
1640 3B28 3E
1640 3B29 20
1640 3B2A 3F
```

```
1640 3B2B 20
1640 3B2C 00
1650 3B2D 207D3D            JSR SUB2      ;CHECK VALID TRACK NUMBER
1660 3B30 8522              STA ZR02      ;FIRST TRACK TO BE COPIED
1670 3B32 852D              STA ZR0D
1680 3B34 20732D            JSR STROUT
1690 3B37 0D                .BYTE $0D,$0A
1690 3B38 0A
1700 3B39 4C                .BYTE 'Last track'
1700 3B3A 61
1700 3B3B 73
1700 3B3C 74
1700 3B3D 20
1700 3B3E 74
1700 3B3F 72
1700 3B40 61
1700 3B41 63
1700 3B42 6B
1710 3B43 20                .BYTE ' (Inclusive)   <0-34> ? ',$00
1710 3B44 28
1710 3B45 49
1710 3B46 6E
1710 3B47 63
1710 3B48 6C
1710 3B49 75
1710 3B4A 73
1710 3B4B 69
1710 3B4C 76
1710 3B4D 65
1710 3B4E 29
1710 3B4F 20
1710 3B50 20
1710 3B51 20
1710 3B52 3C
1710 3B53 30
1710 3B54 2D
1710 3B55 33
1710 3B56 34
1710 3B57 3E
1710 3B58 20
1710 3B59 3F
1710 3B5A 20
1710 3B5B 00
1720 3B5C 207D3D            JSR SUB2      ;CHECK VALID TRACK NUMBER
1730 3B5F 852C              STA ZR0C      ;MAX TRACK NUM
1740 3B61 8523              STA ZR03      ;TR NUMB TO BE COPIED
1750 3B63 E623              INC ZR03      ;TR.NUM +1
1760 3B65 C522              CMP ZR02
1770 3B67 3003              BMI INPERR
1780 3B69 4C9D3B            JMP READY
1790 3B6C 20732D  INPERR    JSR STROUT
1800 3B6F 2A                .BYTE '** Track from > track to !'
1800 3B70 2A
1800 3B71 20
1800 3B72 54
1800 3B73 72
1800 3B74 61
```

```
1800 3B75 63
1800 3B76 6B
1800 3B77 20
1800 3B78 66
1800 3B79 72
1800 3B7A 6F
1800 3B7B 6D
1800 3B7C 20
1800 3B7D 3E
1800 3B7E 20
1800 3B7F 74
1800 3B80 72
1800 3B81 61
1800 3B82 63
1800 3B83 6B
1800 3B84 20
1800 3B85 74
1800 3B86 6F
1800 3B87 20
1800 3B88 21
1810 3B89 20                    .BYTE ' **',$0D,$0A
1810 3B8A 2A
1810 3B8B 2A
1810 3B8C 0D
1810 3B8D 0A
1820 3B8E 54                    .BYTE 'Try again !',$00
1820 3B8F 72
1820 3B90 79
1820 3B91 20
1820 3B92 61
1820 3B93 67
1820 3B94 61
1820 3B95 69
1820 3B96 6E
1820 3B97 20
1820 3B98 21
1820 3B99 00
1830 3B9A 4C053B                JMP TRACK
1840 3B9D 20732D  READY         JSR STROUT
1850 3BA0 0D                    .BYTE $0D,$0A,$0A
1850 3BA1 0A
1850 3BA2 0A
1860 3BA3 43                    .BYTE 'Copy from Drive ',$00
1860 3BA4 6F
1860 3BA5 70
1860 3BA6 79
1860 3BA7 20
1860 3BA8 66
1860 3BA9 72
1860 3BAA 6F
1860 3BAB 6D
1860 3BAC 20
1860 3BAD 44
1860 3BAE 72
1860 3BAF 69
1860 3BB0 76
1860 3BB1 65
1860 3BB2 20
```

```
1860 3BB3 00
1870 3BB4 A520              LDA ZR00
1880 3BB6 6909              ADC #$09
1890 3BB8 209B2D            JSR PRTHEX
1900 3BBB 20732D            JSR STROUT
1910 3BBE 20                .BYTE ' to Drive ',$00
1910 3BBF 74
1910 3BC0 6F
1910 3BC1 20
1910 3BC2 44
1910 3BC3 72
1910 3BC4 69
1910 3BC5 76
1910 3BC6 65
1910 3BC7 20
1910 3BC8 00
1920 3BC9 A521              LDA ZR01
1930 3BCB 6909              ADC #$09
1940 3BCD 209B2D            JSR PRTHEX
1950 3BD0 20732D            JSR STROUT
1960 3BD3 20                .BYTE ' / Tracks ',$00
1960 3BD4 2F
1960 3BD5 20
1960 3BD6 54
1960 3BD7 72
1960 3BD8 61
1960 3BD9 63
1960 3BDA 6B
1960 3BDB 73
1960 3BDC 20
1960 3BDD 00
1970 3BDE A52D              LDA ZR0D
1980 3BE0 20922D            JSR PRTAHX
1990 3BE3 20732D            JSR STROUT
2000 3BE6 20                .BYTE ' thru ',$00
2000 3BE7 74
2000 3BE8 68
2000 3BE9 72
2000 3BEA 75
2000 3BEB 20
2000 3BEC 00
2010 3BED A52C              LDA ZR0C
2020 3BEF 20922D            JSR PRTAHX
2030 3BF2 20732D            JSR STROUT
2040 3BF5 0D                .BYTE $0D,$0A,'Is this okay '
2040 3BF6 0A
2040 3BF7 49
2040 3BF8 73
2040 3BF9 20
2040 3BFA 74
2040 3BFB 68
2040 3BFC 69
2040 3BFD 73
2040 3BFE 20
2040 3BFF 6F
2040 3C00 6B
2040 3C01 61
2040 3C02 79
```

```
2040 3C03 20
2050 3C04 28                    .BYTE '(Y/N) ? ',$00
2050 3C05 59
2050 3C06 2F
2050 3C07 4E
2050 3C08 29
2050 3C09 20
2050 3C0A 3F
2050 3C0B 20
2050 3C0C 00
2060 3C0D 20C53C                JSR GETANS      ;GET ANSWER
2070 3C10 F003                  BEQ LAB01       ;THEN START COPYING
2080 3C12 4C7E3A                JMP START
2090                    ;
2100 3C15 20F73F  LAB01         JSR COPY        ;COPY NOW
2110 3C18 206127                JSR UNLDHD
2120 3C1B 206326                JSR HOME
2130 3C1E 20732D                JSR STROUT
2140 3C21 0D                    .BYTE $0D,$0A,$0A
2140 3C22 0A
2140 3C23 0A
2150 3C24 44                    .BYTE 'Do you want to copy '
2150 3C25 6F
2150 3C26 20
2150 3C27 79
2150 3C28 6F
2150 3C29 75
2150 3C2A 20
2150 3C2B 77
2150 3C2C 61
2150 3C2D 6E
2150 3C2E 74
2150 3C2F 20
2150 3C30 74
2150 3C31 6F
2150 3C32 20
2150 3C33 63
2150 3C34 6F
2150 3C35 70
2150 3C36 79
2150 3C37 20
2160 3C38 61                    .BYTE 'another diskette '
2160 3C39 6E
2160 3C3A 6F
2160 3C3B 74
2160 3C3C 68
2160 3C3D 65
2160 3C3E 72
2160 3C3F 20
2160 3C40 64
2160 3C41 69
2160 3C42 73
2160 3C43 6B
2160 3C44 65
2160 3C45 74
2160 3C46 74
2160 3C47 65
2160 3C48 20
```

```
2170 3C49 28                    .BYTE '(Y/N) ? ',$00
2170 3C4A 59
2170 3C4B 2F
2170 3C4C 4E
2170 3C4D 29
2170 3C4E 20
2170 3C4F 3F
2170 3C50 20
2170 3C51 00
2180 3C52 20C53C                JSR GETANS    ;GET ANSWER
2190 3C55 D003                  BNE MESSA
2200 3C57 4C9D3B                JMP READY
2210 3C5A 20732D  MESSA         JSR STROUT
2220 3C5D 0D                    .BYTE $0D,$0A,$0A
2220 3C5E 0A
2220 3C5F 0A
2230 3C60 50                    .BYTE 'Please, put the TUTORIAL DISC in '
2230 3C61 6C
2230 3C62 65
2230 3C63 61
2230 3C64 73
2230 3C65 65
2230 3C66 2C
2230 3C67 20
2230 3C68 70
2230 3C69 75
2230 3C6A 74
2230 3C6B 20
2230 3C6C 74
2230 3C6D 68
2230 3C6E 65
2230 3C6F 20
2230 3C70 54
2230 3C71 55
2230 3C72 54
2230 3C73 4F
2230 3C74 52
2230 3C75 49
2230 3C76 41
2230 3C77 4C
2230 3C78 20
2230 3C79 44
2230 3C7A 49
2230 3C7B 53
2230 3C7C 43
2230 3C7D 20
2230 3C7E 69
2230 3C7F 6E
2230 3C80 20
2240 3C81 64                    .BYTE 'drive A and depress '
2240 3C82 72
2240 3C83 69
2240 3C84 76
2240 3C85 65
2240 3C86 20
2240 3C87 41
2240 3C88 20
2240 3C89 61
2240 3C8A 6E
```

```
2240 3C8B 64
2240 3C8C 20
2240 3C8D 64
2240 3C8E 65
2240 3C8F 70
2240 3C90 72
2240 3C91 65
2240 3C92 73
2240 3C93 73
2240 3C94 20
2250 3C95 3C                    .BYTE '<RETURN>',$2E,$00
2250 3C96 52
2250 3C97 45
2250 3C98 54
2250 3C99 55
2250 3C9A 52
2250 3C9B 4E
2250 3C9C 3E
2250 3C9D 2E
2250 3C9E 00
2260 3C9F 209B2C                JSR OSINP
2270 3CA2 A901                  LDA #$01
2280 3CA4 20C629                JSR SETDRV    ;SET FOR DRIVE A
2290 3CA7 206326                JSR HOME
2300 3CAA 68                    PLA           ;RESET STACK POINTER
2310 3CAB 68                    PLA
2320 3CAC 20F72C                JSR SWAP      ;REINSTALL BASIC CONDITIONS
2330 3CAF A93C                  LDA #COMADR/$100 ;HI ADR OF BASIC COMMAND
2340 3CB1 85C8                  STA ZRC8
2350 C800=                      ZWI=ZRC8*$100
2360 3CB3 A9BA                  LDA #COMADR-ZWI  ;LO ADRESS
2370 3CB5 85C7                  STA ZRC7
2380 3CB7 20E007                JSR EXECUR    ;PERFORM THE NEXT BASIC COM
2390 3CBA 3A        COMADR      .BYTE $3A     ;STATEMENT DELIMITER ":"
2400 3CBB 89                    .BYTE $89     ;TOKEN FOR "RUN"
2410 3CBC 22                    .BYTE '"BEXEC*"',$00
2410 3CBD 42
2410 3CBE 45
2410 3CBF 58
2410 3CC0 45
2410 3CC1 43
2410 3CC2 2A
2410 3CC3 22
2410 3CC4 00
2420                    ;
2430                    ;
2440                            ;===========
2450                            ;SUBROUTINES
2460                            ;===========
2470                    ;
2480                    ;
2490                    ;
2500 3CC5 209B2C    GETANS      JSR OSINP     ;GET ANSWER
2510 3CC8 A900                  LDA #$00
2520 3CCA 8DE52C                STA BUFIND
2530 3CCD 20E42C                JSR BUFBYT    ;READ 1. CHA OF COM-BUFF
2540 3CD0 C959                  CMP #$59      ;IS IT A YES
2550 3CD2 60                    RTS
2560                    ;
```

```
2570 3CD3 209B2C  SUB1     JSR OSINP     ;CHECK VALID DRIVE NUMBER
2580 3CD6 A900             LDA #$00      ;READ 1. BYTE OF COMMAND
2590 3CD8 8DE52C           STA BUFIND    ;BUFFER INTO ACCU
2600 3CDB 20E42C           JSR BUFBYT
2610 3CDE C941             CMP #$41      ;IS IT A  "A"
2620 3CE0 3007             BMI BADNAM    ;BRANCH IF LESS
2630 3CE2 C945             CMP #$45      ;IST IT A "D"
2640 3CE4 1003             BPL BADNAM    ;BRANCH IF MORE
2650 3CE6 2907             AND #%00000111 ;CONVERT TO 01..04
2660 3CE8 60               RTS           ;DRIVE NUMBER IN ACCU
2670                       ;
2680 3CE9 20732D  BADNAM   JSR STROUT
2690 3CEC 0D               .BYTE $0D,$0A
2690 3CED 0A
2700 3CEE 2A               .BYTE '** Bad name **'
2700 3CEF 2A
2700 3CF0 20
2700 3CF1 42
2700 3CF2 61
2700 3CF3 64
2700 3CF4 20
2700 3CF5 6E
2700 3CF6 61
2700 3CF7 6D
2700 3CF8 65
2700 3CF9 20
2700 3CFA 2A
2700 3CFB 2A
2710 3CFC 0D               .BYTE $0D,$0A
2710 3CFD 0A
2720 3CFE 54               .BYTE 'Try again !  ',$00
2720 3CFF 72
2720 3D00 79
2720 3D01 20
2720 3D02 61
2720 3D03 67
2720 3D04 61
2720 3D05 69
2720 3D06 6E
2720 3D07 20
2720 3D08 21
2720 3D09 20
2720 3D0A 00
2730 3D0B 4CD33C           JMP SUB1
2740                       ;
2750                       ;
2760 3D0E A9FF    SUBA     LDA #$FF      ;IS IT SAME DISC TO COPY ON
2770 3D10 8528             STA ZR08      ;SET DISC FLAG
2780 3D12 A521             LDA ZR01      ;DRIVE TO COPY TO
2790 3D14 C520             CMP ZR00
2800 3D16 F021             BEQ RETD      ;RETURN IF THE SAME DRIVE
2810 3D18 1011             BPL CHE1      ;TO > FROM
2820 3D1A C902             CMP #$02      ;IS TO  "B"
2830 3D1C F01B             BEQ RETD      ;YES RETURN
2840 3D1E A420             LDY ZR00
2850 3D20 88               DEY
2860 3D21 C421             CPY ZR01
2870 3D23 D014             BNE RETD
```

```
2880 3D25 203A3D                JSR SUBB      ;TO IS "A" OR "C"
2890 3D28 4C393D                JMP RETD      ;RETURN
2900 3D2B C903      CHE1        CMP #$03      ;IS TO  "C"
2910 3D2D F00A                  BEQ RETD      ;YES RETURN
2920 3D2F A420                  LDY ZR00
2930 3D31 C8                    INY
2940 3D32 C421                  CPY ZR01
2950 3D34 D003                  BNE RETD
2960 3D36 203A3D                JSR SUBB      ;TO IS "B" OR "D"
2970 3D39 60        RETD        RTS           ;RETURN TO MAIN
2980 3D3A 20732D    SUBB        JSR STROUT
2990 3D3D 0D                    .BYTE $0D,$0A
2990 3D3E 0A
3000 3D3F 49                    .BYTE 'Is it the same diskette, '
3000 3D40 73
3000 3D41 20
3000 3D42 69
3000 3D43 74
3000 3D44 20
3000 3D45 74
3000 3D46 68
3000 3D47 65
3000 3D48 20
3000 3D49 73
3000 3D4A 61
3000 3D4B 6D
3000 3D4C 65
3000 3D4D 20
3000 3D4E 64
3000 3D4F 69
3000 3D50 73
3000 3D51 6B
3000 3D52 65
3000 3D53 74
3000 3D54 74
3000 3D55 65
3000 3D56 2C
3000 3D57 20
3010 3D58 59                    .BYTE 'You want to copy on (Y/N) ? ',$00
3010 3D59 6F
3010 3D5A 75
3010 3D5B 20
3010 3D5C 77
3010 3D5D 61
3010 3D5E 6E
3010 3D5F 74
3010 3D60 20
3010 3D61 74
3010 3D62 6F
3010 3D63 20
3010 3D64 63
3010 3D65 6F
3010 3D66 70
3010 3D67 79
3010 3D68 20
3010 3D69 6F
3010 3D6A 6E
3010 3D6B 20
```

```
3010 3D6C 28
3010 3D6D 59
3010 3D6E 2F
3010 3D6F 4E
3010 3D70 29
3010 3D71 20
3010 3D72 3F
3010 3D73 20
3010 3D74 00
3020 3D75 20C53C            JSR GETANS    ;GET ANSWER
3030 3D78 F002              BEQ RETE      ;DISC FLAG (N) NOT CHANGED
3040 3D7A E628              INC ZR08      ;N-FLAG RESET
3050 3D7C 60       RETE     RTS
3060               ;
3070               ;
3080 3D7D 209B2C   SUB2     JSR OSINP     ;CHECK VALID TRACK NUMBER
3090 3D80 A900              LDA #$00      ;READ 1. BYTE OF COMMAND
3100 3D82 8DE52C            STA BUFIND    ;BUFFER
3110 3D85 8524              STA ZR04
3120 3D87 20E42C   LAB03    JSR BUFBYT
3130 3D8A C90D              CMP #$0D      ;IS IR A "CR" OR IS BUFF
3140 3D8C D007              BNE LAB02     ;FULL - NO THEN BRANCH
3150 3D8E A524              LDA ZR04
3160 3D90 C935              CMP #$35      ;MORE THAN 35 TRACKS ?
3170 3D92 1019              BPL BADTRK
3180 3D94 60                RTS
3190 3D95 C930     LAB02    CMP #$30      ;IS IT < "0" ?
3200 3D97 3014              BMI BADTRK    ;BRANCH IF LESS
3210 3D99 C940              CMP #$40      ;IS IT > "9" ?
3220 3D9B 1010              BPL BADTRK    ;BRANCH IF MORE
3230 3D9D 290F              AND #%00001111 ;CONV ASCII TO HEX
3240 3D9F 0A                ASL A         ;LOW NIBBLE TO HI NIBBLE
3250 3DA0 0A                ASL A
3260 3DA1 0A                ASL A
3270 3DA2 0A                ASL A
3280 3DA3 A204              LDX #$04
3290 3DA5 0A       LOOP1    ASL A         ;HIGH NIBBLE ACCU INTO
3300 3DA6 2624              ROL ZR04      ;LOW NIBBLE ZR04
3310 3DA8 CA                DEX
3320 3DA9 D0FA              BNE LOOP1
3330 3DAB F0DA              BEQ LAB03     ;BRANCH ALWAYS
3340 3DAD 20732D   BADTRK   JSR STROUT
3350 3DB0 0D                .BYTE $0D,$0A
3350 3DB1 0A
3360 3DB2 2A                .BYTE '** Bad track number **'
3360 3DB3 2A
3360 3DB4 20
3360 3DB5 42
3360 3DB6 61
3360 3DB7 64
3360 3DB8 20
3360 3DB9 74
3360 3DBA 72
3360 3DBB 61
3360 3DBC 63
3360 3DBD 6B
3360 3DBE 20
3360 3DBF 6E
3360 3DC0 75
```

```
3360 3DC1 6D
3360 3DC2 62
3360 3DC3 65
3360 3DC4 72
3360 3DC5 20
3360 3DC6 2A
3360 3DC7 2A
3370 3DC8 0D                    .BYTE $0D,$0A,'Try again ! ',$00
3370 3DC9 0A
3370 3DCA 54
3370 3DCB 72
3370 3DCC 79
3370 3DCD 20
3370 3DCE 61
3370 3DCF 67
3370 3DD0 61
3370 3DD1 69
3370 3DD2 6E
3370 3DD3 20
3370 3DD4 21
3370 3DD5 20
3370 3DD6 00
3380 3DD7 4C7D3D                JMP SUB2
3390                    ;
3400                    ;
```

```
10 REM THIS PROGRAM CALCULATES THE DAY OF THE WEEK
20 REM LIMITATION: THE YEAR MUST BE AFTER THE YEAR 1752
30 J$(1)="SUNDAY"
40 J$(2)="MONDAY"
50 J$(3)="THUESDAY"
60 J$(4)="WEDNESDAY"
70 J$(5)="THURSDAY"
80 J$(6)="FRYDAY"
90 J$(7)="SATURDAY"
100 PRINT"INPUT THE DAY, MONTH AND THE YEAR (IN FIGURES)"
110 INPUT D,M,Y
120 IF Y)1752 THEN 150
130 PRINT"THE YEAR MUST BE AFTER THE YEAR 1752"
140 GOTO100
150 K=INT(0,6+(1/M))
160 L=Y-K
170 O=M+12*K
180 P=L/100
190 Z1=INT(P/4)
200 Z2=INT(P)
210 Z3=INT((5*L)/4)
220 Z4=INT(13*(O+1)/5)
230 Z=Z4+Z3-Z2+Z1+D-1
240 Z=Z-(7*INT(Z/7))+1
250 PRINT"DAY OF THE WEEK IS ";J$(Z):PRINT
260 INPUT"NEW DATE ? TYPE YES OR NO ";L$
270 IF L$="YES" THEN 310
280 IF L$="NO" THEN 330
290 PRINT"ILLEGAL ANSWER"
300 GOTO260
310 PRINT
320 GOTO30
330 PRINT"END OF PROGRAM"
340 END
```

C-16 : Listing Only on Key Held Down

Fred Behringer, Muenchen, Germany

What is the use of a long BASIC-listing flushing by in no time? Here is a remedy :

```
10 FORI=0TO9:READA:POKE828+I,A:NEXT:POKE774,60:POKE775,3:DATA72,
   173,67,5,240,251,104,76,110,139:END
```

Use abbreviations such as N- for NEXT in order to keep within the BASIC line limit of 88 characters! (Of course, the program could as well be distributed over two lines, and any line number other than 10 would do as well.)
Initiate this utility program by RUN10.
Now, if LIST is keyed in, the BASIC-listing is emitted only as long as the SHIFT-key is pressed. Keep your fingers off the SHIFT-key, and the listing stops. Press again, and the listing continues.
To return to the usual list operation, type SYS33047 (This initiates the BASIC list vector 0306/0307, which was modified by our utility program.)
On RUN10, the FOR-NEXT loop pokes the following machine code program into the cassette buffer from 828 to 837 :

```
0828: 48              PHA       save CPU register A          72
0829: AD 43 05 SHIFT  LDA $0543 SHIFT-key held down?  173,67,5
082C: F0 FB           BEQ SHIFT if not (=0), wait      240,251
082D: 68              PLA       fetch saved A                104
082E: 4C 6E 8B        JMP $8B6E continue listing    76,110,139
```

The list routine then by-passes to this SHIFT-key waiting loop via the modified list vector 0306/0307.

BANK SWITCHING FOR THE JUNIOR COMPUTER

Three boards (at least) are needed for this implementation:
1- the old (original) 64k dyn RAM card, where a new gate will be added,
2- a new 64k dyn RAM card with 2 tracks cut, and
3- a small decoding board, with a bank register and a few TTL ic's, to be linked with the new 64k board.

How it works:

DECODING BOARD:

In fig. IB is the decoding necessary to implement a new 64k dyn RAM card.
The bank register is the LS 173, a 4-bit D-type register, which latches the nibbles written to it (through the four least significant data lines); with these 4 bits can be defined 16 different banks.
The LS 173 is cleared by the system Reset, so the bank 0 is always selected at start.
To memory-map this chip, a write-only address is chosen, i.e. an EPROM address, like $F7FF. The choice must take into account that the Junior bus buffers aren't enabled in the address range of $E000-E3FF and $FA00-FFFF. An elegant 2x LS 688 decoder is used (fig.IC), but a much simpler $F7xx, for instance, would work.
As this is a write-only register, a spare RAM address should keep track of the bank in use.
Fig. IIIA shows the timing diagram for this writing function.
In the purposed circuit diagram only 8 banks are addressable by the LS 138 decoder; and applying the 4Q output to the G2A input, is assured that only the first 8, of the 16 possible with the LS 173, can be selected.
The Y outputs, mutually exclusive low, select the banks. N103 and N107 activate the 4→B space (bank 0) of the original board. With Y1 and Y2 the 2 "halves" of the new 64k board are selected, being the LS 173 1Q output who defines #1 or #2; furthermore, inverting the A15 address line, the 2 banks are always addressed in the 4→B range. (fig. IIC)

NEW 64K DYN RAM BOARD:

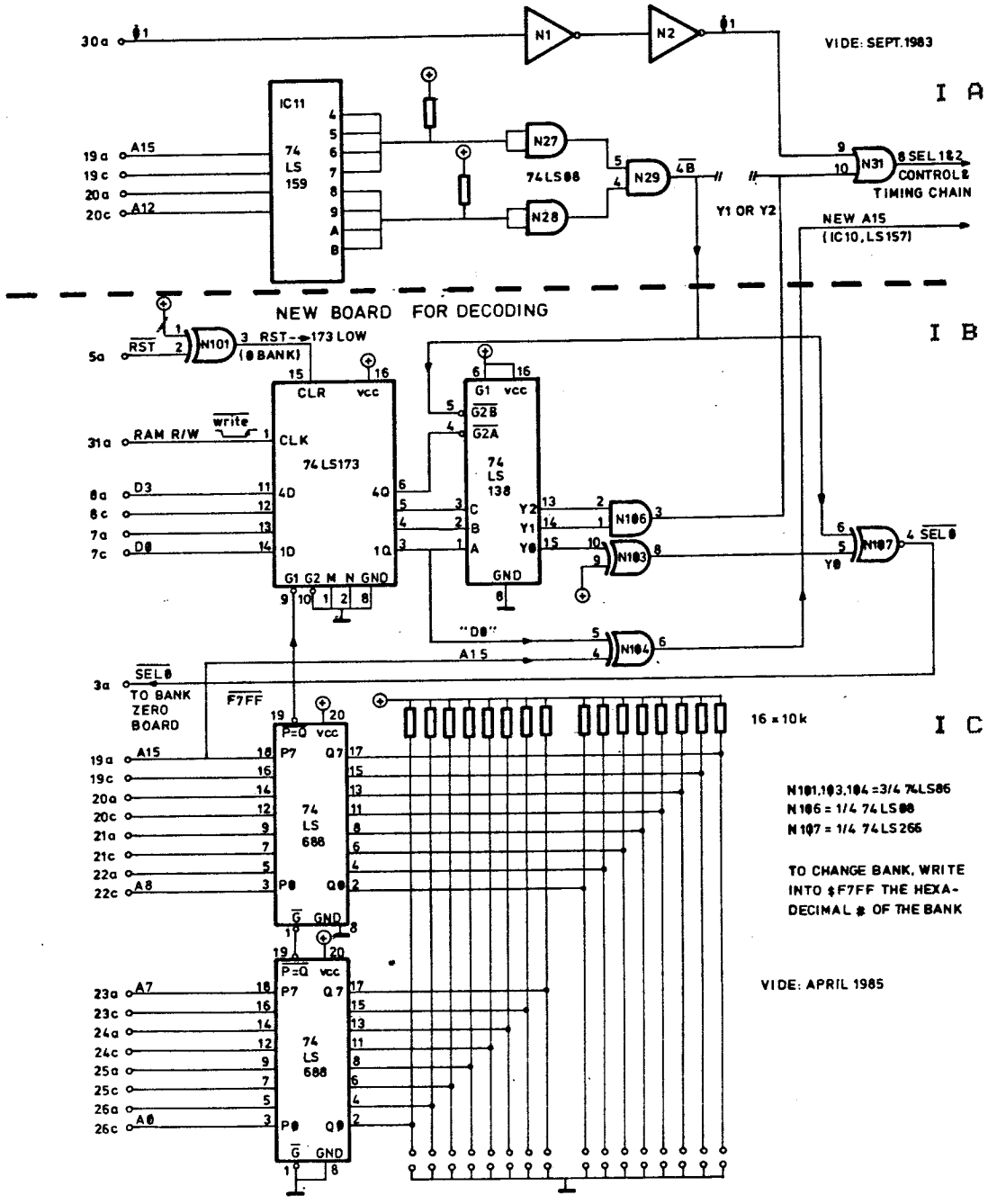
Fig.1A. Please refer to E101, Sep.83. Selecting the outputs 4 through B (4→B) from the LS 159 decoder (ic 11), we get the $\overline{4B}$ line which is active low between addresses $4000 and $BFFF; these are the boundaries of the 32k banks. (fig. IIC)
This new 64k board, itself, needs a simple modification; the line between gate N29 output, and the input of N31, must be broken; the output (from the first) is wired to the G2B of the LS 138 in the decoding board, while the output of N106 is fed to the input of N31. Besides, the new A15 line, from N104 in the decoding board, shall be directed to the memory board, instead of the original A15, into the LS 157 (ic 10) multiplexer.

ORIGINAL 64K DYN RAM BOARD:

Fig.IIB. The original 64k memory board can be disabled only in the 4→B range. This way, always the other $0000 through 3FFF and $C000 through FFFF are addressable. There are no problems, so, with special JSR's or RTS's, or special 4-byte addressing modes, neither any special care with page zero. (This is a base for a data-only enlarged storage space; to work with program code, there, it's different!).
The modification needed to disable this original board is the addition of a new OR-gate, to get the $\overline{SEL}$ line for the board circuitry. This is achieved with a new LS 32 in piggy back instalation on the original (ic 8) one. (fig. IIA)
The SEL0 line is only (active) low when bank zero is selected *and* in the 4→B range; otherwise, when bank 1 or 2 are selected, $\overline{SEL0}$ goes high disabling the board; or, outside 4→B, $\overline{4B}$ goes high, all 138 Y's high (including Y0) and then $\overline{SEL0}$ low, permitting the addressing of the 0→3 and C→F memory in this board. (fig. IB)
Switch S1 enables the use of the original memory board in the absence of any additional board, making input 2 of N31A grounded.

The use of the added 64k memory will give the COPIER program (from BEXEC*) with one drive, more speed and will reduce the disk swapping times!
An Assembler, like MicroAde, should gain extra space for source code. (IIIB)(IV)
All that is needed is the software.
And that's the challenge!

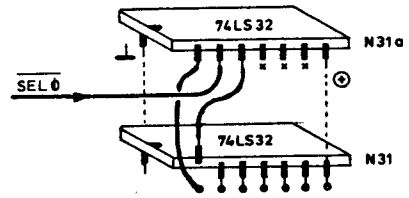
Fernando Lopes
Colégio dos Orfãos
Lg. Baltazar Guedes
4300 PORTO PORTUGAL



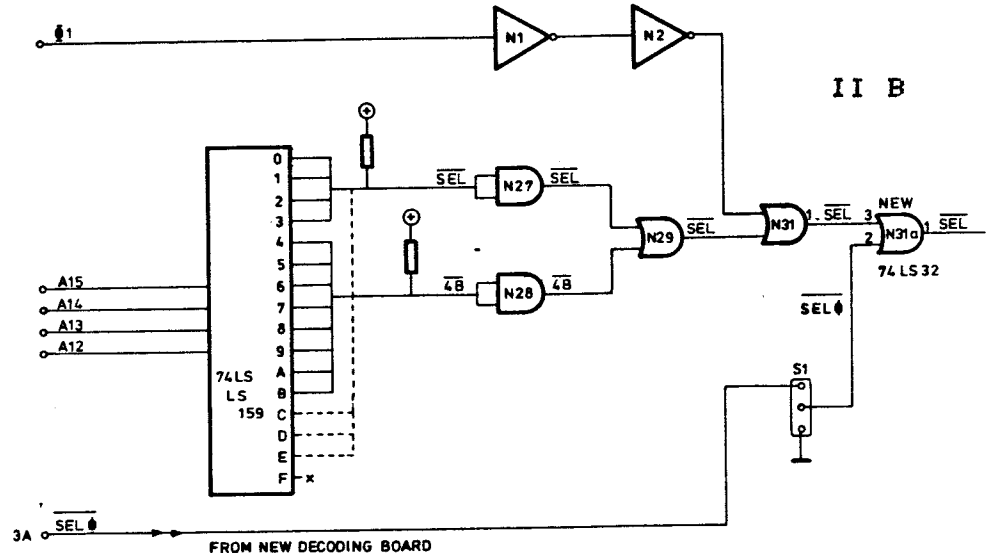
aan de tekentafel:
Herman Zondag

64k DYN. RAM CARD (OLD)
BANK 0 (ONE NEW GATE)

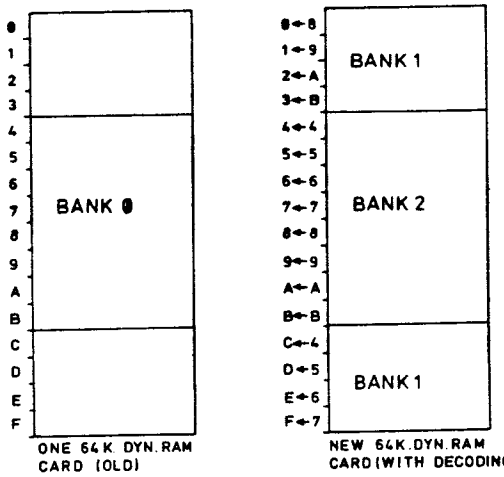
II A

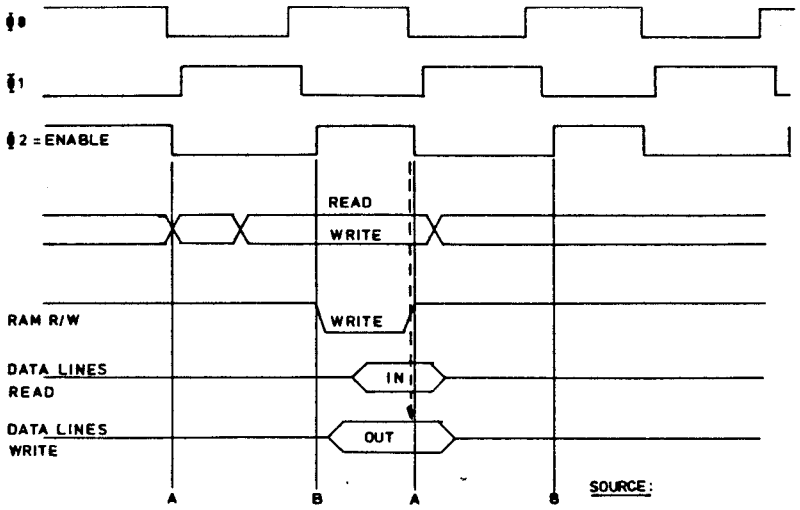


II B



II C

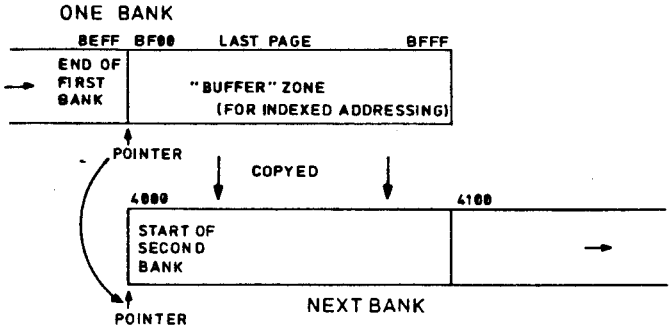




III A

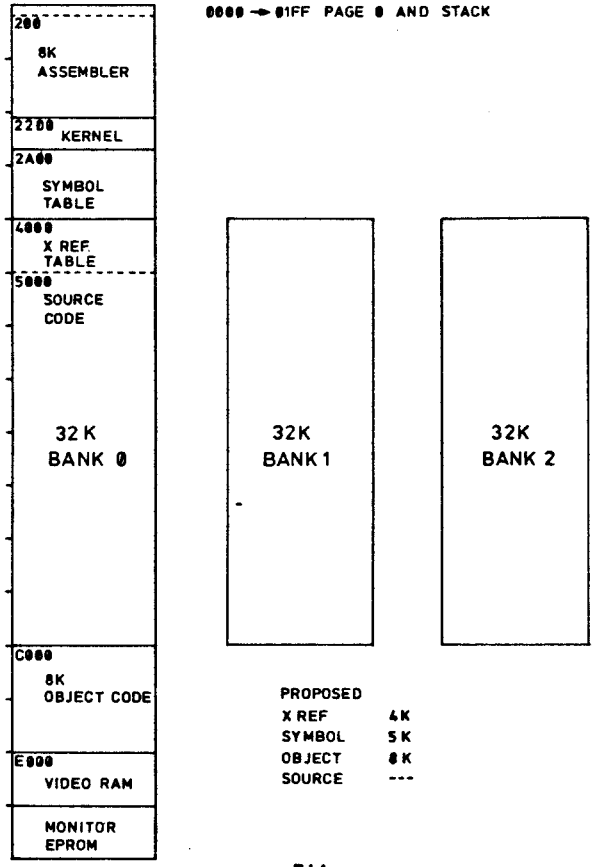
SOURCE:

ROCKWELL, DATA SHEET N:D39
REV. 5AUG.1983
(ADAPTED)
VIDE, MEMORY TIMING
ELEKTOR FEB. 1984



III B

ASSEMBLER (MICRO ADE) WITH BANK SWITCHING



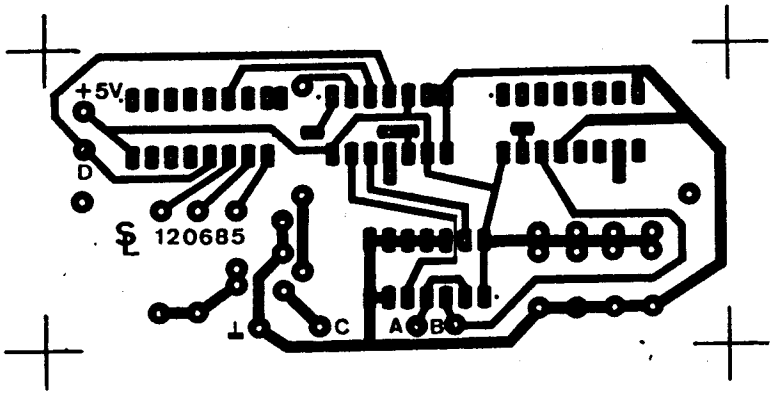
IV

Siegfried Losensky
Wächtersbach 74
D 6114 Gross-Umstadt

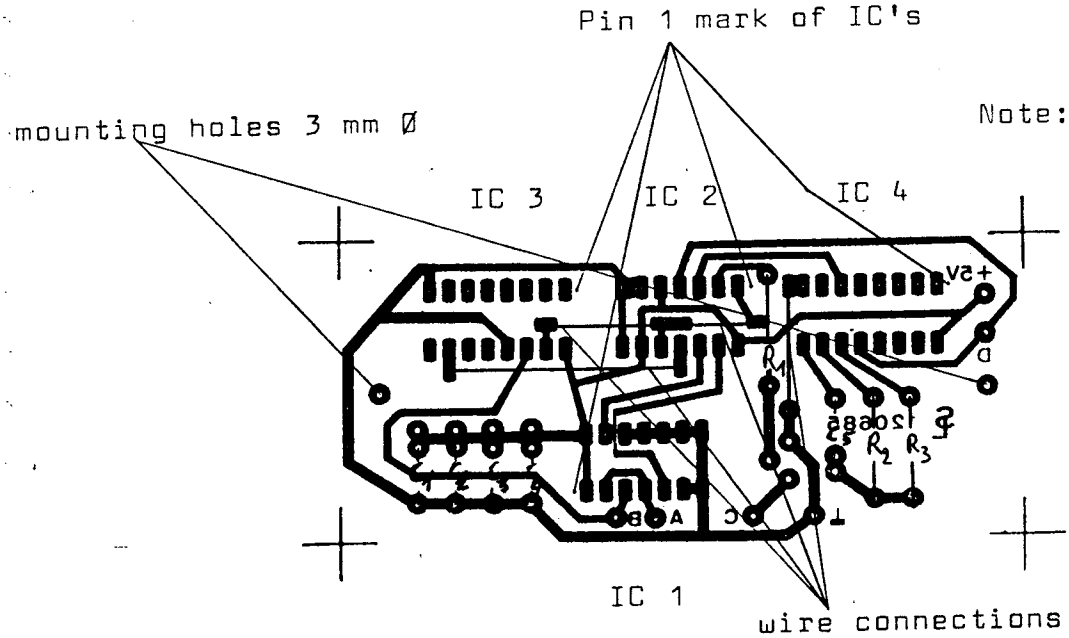
In the german Elektor of April 1985, from page 4-34, a motor-steering for diskette-driver was published, including a scheme. Because I did not want to buy new tools for Vero-thread-technik, I designed a foil-copy on my platine-layout for disk-drive-motor-steering according to Elektor 4/84 for publishing in the club-journal.

for reproduction

this side
towards UV-lamp



Look upon electronic-parts side



****
*
*
*
****

HET APPLE BOOT PROCES

DOOR :

FRANK MANSHANDE

Wanneer je een diskette in de disk-drive doet, en de APPLE daarna aanzet, of als je PR#6 doet of, in de monitor, 6 CTRL-P, dan gaat de disk-drive draaien en wordt het HELLO-programma gerund. Hoe komt dit nou?

Wel, als je de computer aanzet, of een van de andere mogelijkheden gebruikt om de drive te laten opstarten, dan springt de computer naar locatie $C600. Dit is decimaal 50688 (je kunt dus ook de drive laten opstarten met CALL 50688 !). Hier begint namelijk de ROM van de disk-drive interface (als de disk-drive interface in slot 6 zit, anders springt de computer naar een andere locatie; als je hem bijvoorbeeld in slot 4 hebt zitten, zal de computer naar geheugenlocatie $C400 springen). Op $C600 begint dus de interface, en in de ROM van die interface staan een aantal routines om ervoor te zorgen dat de drive opstart.

Wat gebeurt er nu in de disk-interface ROM, oftewel op locatie $C600? Op $C600-$C65B staat een routine om de eerste disk-drive (drive 1) aan te zetten, de drive-arm naar track 0/sector 0 te bewegen (dit hoor je duidelijk) en dan zorgt deze routine er ook nog voor, om op speciale locaties informatie op te slaan. Op $3D wordt het sectornummer gezet, in dit geval dus $00. Op $41 wordt het tracknummer gezet, in dit geval dus ook $00. En dan zet het op de adressen $26/$27 het adres, waar de informatie van de sector moet komen. Hier komt $800 te staan, dus de sector die wordt ingelezen komt op $800 terecht.
Op $C65C-$C6FA staat een routine om een sector te lezen. Het haalt de nodige informatie op uit de geheugenplaatsen $26, $27, $3D en $41 en gaat zoeken op de disk naar de sectorheader. De sectorheader bestaat uit 3 bytes: D5/AA/96, en dient ervoor dat de drive weet, wanneer een bepaalde sector begint. Wanneer het D5/AA/AD tegenkomt, springt het naar $C6A6.
Op $C683 begint een routine om de eerste 6 bytes te lezen. De eerste 2 bevatten het volumenummer, de tweede 2 bevatten het tracknummer en de derde 2 het sectornummer. Deze laatste 4 bytes vergelijkt het met wat er op $41 en $3D staat. Het slaat het gevonden tracknummer op in $40. Eerst vergelijkt het het gevonden sectornummer met wat er op $3D staat. Daarna vergelijkt het het gevonden tracknummer met wat er op $41 staat. Als een van deze beiden niet gelijk is, gaat het terug naar $C65C. Als beiden kloppen, gaat het naar $C65D, en gaat daar de data lezen die op de disk staat.
Op $C6A6 begint een routine om de gevonden data goed in het geheugen te zetten. Ook verhoogt hij de waarde van $27 met 1, zodat de volgende sector data $100 bytes verder komt te staan, oftewel op de volgende pagina. Hierna springt de routine naar $801.
Van dit alles hoef je eigenlijk alleen te onthouden dat deze routines ervoor zorgen dat sector 0 van track 0 wordt ingelezen, en dat dit boot 0 heet.

Wat gebeurt er nu allemaal verder vanaf $801? Hier wordt boot 2 geladen (de routine die op $801 begint heet, zoals te verwachten, boot 1). Er worden 9 sectoren ingelezen en het gebruikt daarvoor de routine op $C65C. In deze 9 sectoren staat onder andere de RWTS-routine (Read Write Track Sector). De informatie op deze 9 sectoren wordt meteen neergezet op het adres waar ze stonden toen de diskette geinitialiseerd werd. Als de diskette op een 48K APPLE geinitialiseerd werd, of op een APPLE IIe of IIc, dan komen de sectoren op $B700 tot $BFFF. Dan springt de routine, die begon op $801, naar $B700 en hier wordt dan het verdere DOS geladen (boot 2). Op een 48K APPLE wordt het verdere DOS op $9D00 gezet.

Nu is dus geheel DOS 3.3 geladen, maar hoe komt het nou dat het HELLO-programma wordt geRUNd?
Wel, dit is heel wat makkelijker uit te leggen. De naam van het HELLO-prgramma wordt namelijk bewaard in DOS, en wel op $AA75. De naam wordt gelezen vanaf $AA75, en DOS runt dan dit programma. Heel gemakkelijk dus !

WIE HEEFT INFORMATIE OMTRENT DE C-16 ?

Fred Behringer
Strassbergerstrasse 9c/519
D - 8000 Muenchen 40

Behalve mijn VIC-20, waarmee ik al sedert ruim 5 jaar werk, heb ik nu nog een Commodore-16 gekocht, omdat die zo goedkoop was (DM 130,--). Wie heeft data sheets (of kan mij vertellen, waar ik deze kan verkrijgen) voor de volgende in de C-16 stekende IC's : CPU 8501 (of 7501), TED 7360 (?) ? Bestaan er becommentarieerde ROM-listings, zoals dat met de DATA-Becker-boeken "VC-20 Intern" en "C-64 Intern" het geval is? En waar zijn deze verkrijgbaar?