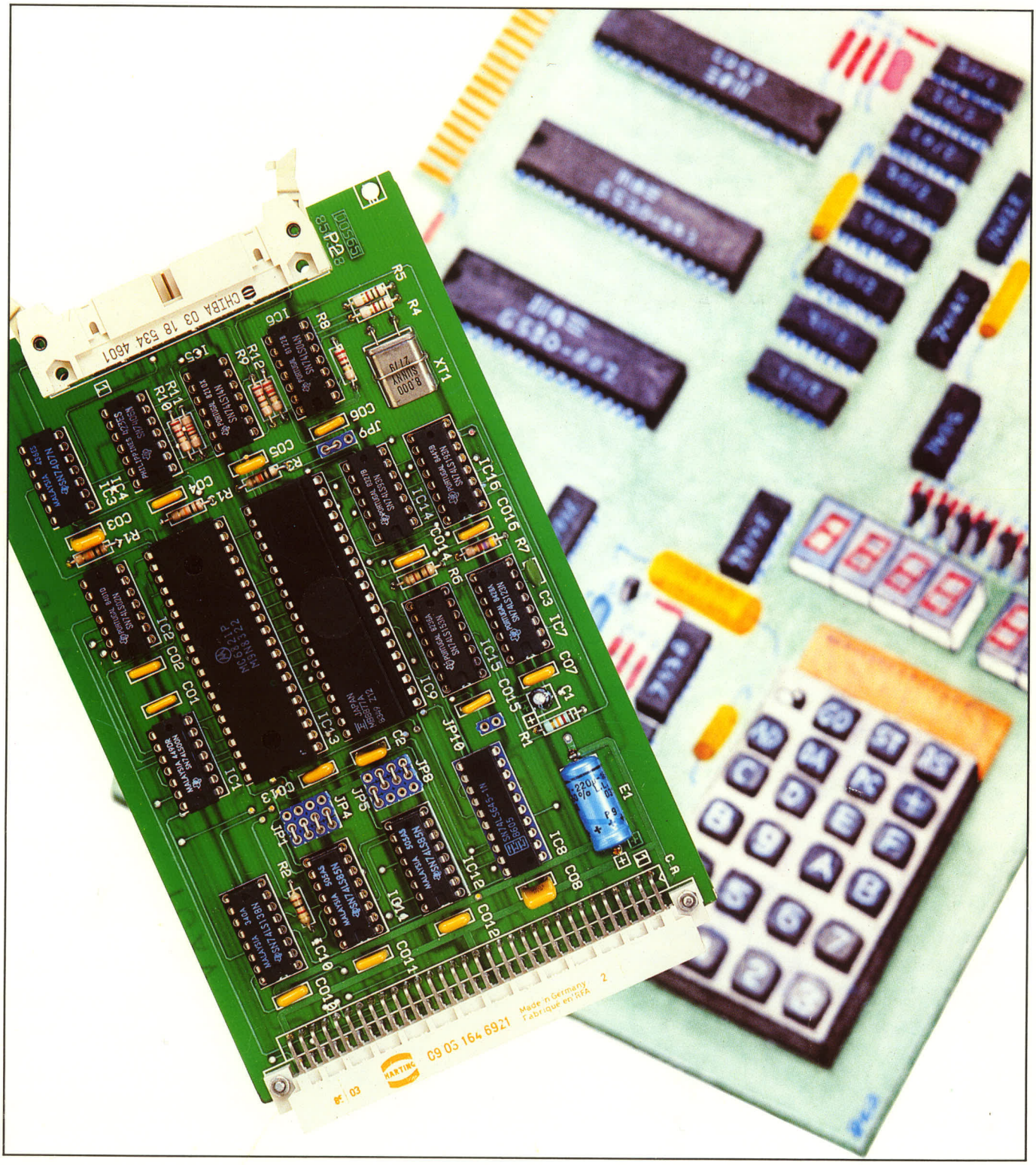
DE6502KENNER 53

Elfde Jaargang, nr 6, december 1987



INFORMATIE.

De 6502 Kenner is een uitgave van de KIM Gebruikersclub Nederland. Deze vereniging is volledig onafhankelijk, is statutair opgericht en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305.

Het doel van de vereniging is het bevorderen van de kennisuitwisseling tussen gebruikers van computers die zijn opgebouwd rond een microprocessor uit de 6500-familie. Voorbeelden hiervan zijn onder andere: Elektuur EC-65, Commodore 64, Apple IIc, Elektuur Junior, Atari 600 en 800.
De eerder genoemde kennisuitwisseling komt onder andere tot stand door 6 maal per jaar de 6502 Kenner te publiceren, door de organisatie van landelijke bijeenkomsten voor de leden en het instandhouden van een softwarebibliotheek op cassette, floppy disk en papier.

Verschijningsdata:

De 6502 verschijnt op de derde zaterdag van de maanden februari, april, juni, augustus, oktober en december.

Landelijke bijeenkomsten:

Deze worden gehouden op bij voorkeur de derde zaterdag van de maanden januari, maart, mei, september en november. De exacte plaats en datum worden steeds in De 6502 Kenner bekend gemaakt in de rubriek Uitnodiging.

Het bestuur.

Het bestuur bestaat uit de voorzitter, secretaris en penningmeester, die samen het dagelijks bestuur vormen. Verder hebben thans vijf leden in het bestuur zitting.

Voorzitter:
Rinus Vleesch Dubois
Florence Nightingalestraat 212
2037 NG HAARLEM
Telefoon 023-330993

Secretaris:
Gert Klein
Diedenweg 119
6706 CM WAGENINGEN
Telefoon 08370-23646

Penningmeester:
John van Sprang
Tulp 71
2925 EW KRIMPEN A/D IJSSEL
Telefoon 01807-20589

Leden:
Willem van Pelt
Jacob Jordaenstraat 15
2923 CK KRIMPEN A/D IJSSEL
Telefoon 01807-19881

Adri Hankel
Willem Kloosstraat 32
7606 BB ALMELO
Telefoon 05490-51151

Erwin Visschedijk
Dillelaan 11
7641 CX WIERDEN
Telefoon 05496-76764

Gert van Opbroek
Bateweg 60
2481 AN WOUBRUGGE
Telefoon 01729-8636

Nico de Vries
Mari Andriessenrade 49
2907 MA CAPELLE A/D IJSSEL
Telefoon 010-4517154

Lees verder op pagina 4

Inhoudsopgave.

Informatie.....2-4
Van het bestuur.....5
Van de penningmeester.....39
Voorlopige bijeenkomstenagenda 1988.....49
Uitnodiging bijeenkomst januari 1988.....6
Vraag en Aanbod.....6,18
Huisbezoek.....18
Verantwoording auteurs De 6502 Kenner no. 52.....20
In den beginne.....26
Datasheetservice.....39
Oproep 1.....34
Oproep 2.....43
Stand van zaken Bulletinboard.....43
Spreekbeurt EPROMs programmeren.....44
Harddiskcontroller.....49
Status van de EPROMprogrammer.....49
Echt waar.....49
Database voor BBC model B.....50
DOS65: Aansluiting IBM toetsenbord op DOS65.....7
Het fijne van HELP.....17
Terminalprogramma.....19
Bugs Communication-65 V2.2.....19
Communicatiepakket V2.2.....20
Search file string.....21
Drive steprate in DOS65.....25
DOS65-bug.....49
EC65: Modified version of Elektor's listing stopper.....30
Lineare Gleichungssysteme.....31
16 kanaals 12 bit A/D Convertor.....33
Octopus WP: een oproep.....34
Algemeen: Zeer intensieve geheugentest.....35
Datum naar weekdag conversie.....40

YEP! I bought this computer to get rid of all that paperware..... AM

Naast het bestuur zijn er een aantal ereleden, die zich in het verleden bijzonder verdienstelijk voor de club hebben gemaakt:
Erevoorzitter: Siep de Vries
Ereleden: Mevr. H. de Vries-van der Winden en Anton Mueller

De 6502 Kenner is een uitgave van de KIM Gebruikersclub Nederland. Het blad wordt bij verschijnen gratis aan alle leden toegezonden. De kopij voor het blad dient bij voorkeur van de leden afkomstig te zijn. Alle kopij wordt door de redactie op bruikbaarheid en publicatiewaarde beoordeeld. Deze twee criteria, in samenhang met de actualiteit, bepalen of een stuk al of niet, en zo ja, wanneer het gepubliceerd wordt. De redactie streeft ernaar de kopij zoveel mogelijk in de oorspronkelijke vorm te plaatsen. In principe wordt Nederlandstalige kopij niet vertaald naar een andere taal; indien de schrijver ook anderstaligen wenst te bereiken, wordt hem aangeraden de kopij van een samenvatting in een tweede taal te voorzien. De redactie streeft ernaar, een Nederlandstalig blad te maken.
Helaas kan de redactie, noch het bestuur, enige aansprakelijkheid aanvaarden voor de toepassing(en) van gepubliceerde kopij.

Software Coördinatie.

Indien uw software heeft geschreven voor een systeem, is het verstandig dit naar de softwarecoördinator voor dat systeem te sturen. Hij zorgt ervoor, dat uw software wordt gecatalogiseerd, en geschikt gemaakt wordt voor verdere distributie. Uw ontvangt uw schijf of cassette natuurlijk retour.

Elektuur EC-65(K) en Junior met OHIO OS-65D:
Vacant

Elektuur Junior (cassette):
Vacant

DOS-65 (ook Junior):
Erwin Visschedijk
Dillelaan 11
7641 CX WIERDEN

Kopij inzenden.

In principe kan kopij in elke vorm naar de redactie gestuurd worden. Het onderwerp is geheel vrij, het moet echter wel met computers of hun toepassing te maken hebben.
Om redenen van efficiency geeft de redactie er de voorkeur aan, kopij te ontvangen in een vorm die per computer verwerkbaar is. Dit verhoogt de kans op publicatie, en geeft de redactie meer mogelijkheden voor een nette opmaak, etc. Op dit moment kan de redactie de volgende media lezen en verwerken:

EC-65, 40 track 5.25" disk
Elektuur Junior cassette tape
DOS-65, 40/80 tr. 5.25" disk
Apple][CP/M
MS-DOS, 360k 5.25" disk

De volgende tekstverwerkerformaten kunnen verwerkt worden:

MS-DOS: WordStar 3.30, WordPerfect 4.1 en 4.2
Apple CP/M: WordStar 3.0

Voor de andere informatiedragers is een zogenaamde schone ASCII-file het gemakkelijkst. Een file is een schone ASCII-file, als hij zonder meer op uw scherm of printer kan worden afgedrukt zonder tussenkomst van een programma.

Indien u een schijf of cassette inzendt, dan streeft de redactie ernaar u die schijf op korte termijn te retourneren. Ook hoopt de redactie u te belonen met wat extra's op die schijf.
Zoals vanouds blijft kopij op papier natuurlijk ook welkom!

Uw kunt al uw kopij, in welke vorm dan ook, altijd sturen aan:

Gert van Opbroek
Bateweg 60
2481 AN WOUBRUGGE

++++++++++++++++++++++++++++++++++

Van het bestuur.

Nog even een terugblik en dan.... voorwaarts! Ja, we hebben als bestuur geleerd en zeker tijdens de in Almelo gehouden ledenvergadering. De affaire 'redactie' is daar uitvoerig aan de orde geweest. Het bestuur heeft een toelichting gegeven op het waarom van het naar huis sturen van de redacteur.
De heer Coen Boltjes trad op als woordvoerder voor diegenen die W. van Pelt graag in zijn oude functie als redacteur zagen terugkomen. De medestanders van woordvoerder Coen Boltjes bestonden hoofdzakelijk uit door de heer C. Boltjes aangeschreven, in het buitenland woonachtige, leden.
Door de zeer geringe opkomst van de leden ontstond de situatie, dat de heer C. Boltjes met de hem gegeven volmachten zelfs een meerderheid had kunnen behalen. Het bestuur heeft gesteld dat bij gebruikmaking van deze meerderheid door middel van volmachten om W. van Pelt in zijn functie van redacteur te herstellen, het voltallige bestuur zou aftreden.
Zover is het echter niet gekomen. De heer C. Boltjes heeft geen optimaal gebruik kunnen maken van de aan hem toevertrouwde volmachten. Naar alle waarschijnlijkheid om de volgende twee redenen:

1e. Een grote meerderheid van de aanwezige leden stemden tegen C. Boltjes en zijn medestanders.
2e. C. Boltjes en zijn medestanders zouden bij terugkeer van W. van Pelt de bestuursverantwoordelijkheid op zich hebben moeten nemen.

Naar onze mening heeft vooral het laatste punt de heer C. Boltjes er van weerhouden voet bij stuk te houden.

Nu vooruit! Hoe vond u bijvoorbeeld de leesbaarheid en de nieuwe lay-out van de laatste twee 6502 Kenners? Het bestuur heeft er weer veel zin in, mede dankzij het aanblijven als penningmeester van de heer J. van Sprang, die, en dat mag best wel eens gezegd worden, de centjes uitstekend beheert en registreert.
Inmiddels heeft zich een nieuwe redacteur aangediend in de persoon van drs. Gert van Opbroek. Gert, namens het bestuur en de leden wensen we je veel succes. Ook mogen we de inmiddels ook gevonden operator voor het te starten bulletinboard niet vergeten: Jacques Banser. Ook jou Jacques wensen we veel succes toe.
De eerste redactionele inzending op floppy disk is inmiddels ook binnen! Bij deze roepen wij de bezitters van de Octopus/EC-65 machines op, om meer kopij in te sturen. De redactie wil namelijk graag een blad maken waar iedereen wat aan heeft.
Tenslotte wensen wij een ieder fijne feestdagen toe. Tot de volgende bijeenkomst in Krommenie.

Namens het bestuur,

R. Vleesch Dubois.

UITNODIGING.

Uitnodiging voor de landelijke bijeenkomst op zaterdag 16 januari 1988.

Locatie: nieuwe kantine FORBO-Krommenie
Industrieweg 12
1566 JP ASSENDELFT
Telefoon 075-291911

Routebeschrijving:

Per auto, komende uit de richting Amsterdam:

Coentunnel door, en Coentunnelweg helemaal afrijden. Aan het einde rechtsaf (water aan de linkerzijde). Dan de 1e afslag rechtsaf, richting Uitgeest-Alkmaar. Doorrijden tot aan stoplichten. Linksaf de spoorbaan over. Na 75 meter linksaf: Industrieweg. Links aanhoudende komt men dan op het FORBO-terrein.

Per auto, komende uit de richting Alkmaar:

Snelweg Alkmaar-Haarlem. Afslag Uitgeest-Zaandam. Bij kruising linksaf. Bij 3e stoplichten rechtsaf, de spoorbaan over. Na 75 meter linksaf: Industrieweg. Links aanhoudende komt men dan op het FORBO-terrein.

Per trein:

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

PROGRAMMA:

09.30 - Zaal open. Ontvangst met koffie.
10.00 - Opening door de voorzitter en verwelkoming door de gastheer Co Filmer. Nieuwe diashow over FORBO-Krommenie.
10.30 - Behandeling Jaarverslag 1987.
10.45 - Koffiepauze.
11.00 - Gert Klein houdt een spreekbeurt over het nieuwe bulletinboard.
12.00 - Lunch, aangeboden door FORBO-Krommenie.
13.00 - Forum, aansluitend Markt.
13.15 - Informeel gedeelte, bedoeld om kennis, ervaring en Public Domain software uit te wisselen met uw medeleden.
BRENG DAAROM OOK UW EIGEN SYSTEEM MEE!!
(En vergeet de snoeren niet.....)
17.00 - Sluiting.

ENTREE: fl. 10.-

::::::::::::::::::::::::::::::::

Vraag en Aanbod.

Te koop:

6502 Elektuur CPU-kaart
Floppy disk controllerkaart
Elektuur VDU-kaart
Elektuur statische RAMkaart (2)
Eigenbouw geheugenkaart
Eigenbouw I/O kaart
5.25" disk drive (met manuals/schema's)
19 inch kast
Monochroom monitor
Een hoop documentatie

Prijs: t.e.a.b.

Te bevragen onder:
Telefoon 01899-22460 (na 18.00 uur)

Aansluiten van een IBM toetsenbord op DOS65.

Door:
Wim Schimmel
Calslaan 28b-62
7522 MC Enschede
tel: 053-895040

Vooralsnog was het niet zondermeer mogelijk een IBM (-compatible) toetsenbord aan te sluiten op het DOS65 systeem. Dit is nu mogelijk gemaakt.

1 Het probleem

Een IBM toetsenbord wijkt nogal af van de gebruikelijke ASCII parallel toetsenborden. De verschillen zijn:

- Synschrone serie le output (zie figuur 2)
- Geen ASCII output maar een toetswaarde (="scan-code") output, plus een extra bit dat aangeeft of de toets wordt ingedrukt of losgelaten. B.v. het indrukken van de ESCAPE toets geeft de code 01, het weer loslaten code 81. (zie voor de scan-code figuur 3)

2 De oplossingen

De volgende drie oplossingen zijn gevonden:

1) Volledige hardware aanpassing.
   voor: - Eenvoudige software aanpassing, simpelweg inpluggen van het IBM toetsenbord en klaar.
   na: - Een zeer uitgebreide (en dure) schakeling.

2) Serieel/Parallel aanpassing d.m.v. een ACIA en de toetsvertaling softwarematig.
   voor: - De software is slechts een eenvoudig "tabel-programma".
   - Er is een 1 karakter buffer, dus er hoeven geen toetsen te worden gemist.
   na: - Geen kant en klare ACIA op de markt die het vreemde signaal kan inlezen en een TTL schakeling wordt veel te complex.

3) Volledige software aanpassing. Dus een programma die het serie le signaal inleest en vertaald naar ASCII.
   voor: - Zeer flexibel.
   - Geen hardware.
   na: - Geen 1 karakter buffer, zodat het kan voorkomen dat er een toets wordt gemist.

Ondanks het vervelende nadeel is er toch gekozen voor het derde alternatief. Mede ter voorkoming van hardware problemen.

4 Uitwerking van de oplossing

Ook de oplossing is gesplist in de twee problemen, we beginnen met het serie le probleem.

4.1 Serieel

De DOS65 invoer werkt op interrupt basis. Bij een interrupt wordt in een routine gesprongen die de toets uit de VIA haalt en in een "toets buffer" plaatst. Zodra de toetsen nodig zijn worden ze uit die buffer gehaald en verwerkt. Hoe het IBM toetsenbord moet worden aangesloten staat in figuur 1.

figuur 1. Aansluiting van het toetsenbord aan VIA1.

Op een negatieve flank van CA1 wordt een interrupt gegenereerd. Zodra naar de interrupt routine is gesprongen wordt begonnen met het softwarematig schuiven van de bits. Probleem is nu, dat te laat naar de interrupt routine kan worden

gesprongen (later dan 560us) zodat er bits worden gemist. In de omzettings routine moet een "time-out" procedure worden toegevoegd om het "oneindig" wachten op bits die verloren zijn gegaan te voorkomen.

4.2 Toetswaarden

Elke toets van het toetsenbord geeft een code af (zowel bij maak als bij verbreek). Dit geldt ook voor de toetsen SHIFT, CONTROL, ALT, CAPSLOCK en NUMLOCK. De ASCII waarden van toetsen is echter wel afhankelijk van de "toestand" van deze controle toetsen. Daarom is een toestandregister gebruikt, dat bijhoudt of deze controle toetsen in de actieve of non-actieve toestand zijn. Voor de omzetting van de toetswaarden naar ASCII waarden wordt gebruik gemaakt van 3 tabellen namelijk de ASCII-, SHIFT- en NUMLOCK-tabel. De ALT toets is (nog) niet ge mplementeerd (in de zin van andere codes althans).

5 Listing

De listing van het programma zal nu waarschijnlijk weinig vraagtekens oproepen. Duidelijk is wel de enorme flexibiliteit van het geheel, daar het programma op schijf staat. Wil men een andere tabel of een (uitgebreider) programma dan kan dit simpel worden veranderd. Bij het opbooten wordt het programma, via login.com, ingelezen en gestart.

6 Aanpassingen aan IO65

De veranderingen in de IO65 EPROM zijn zeer gering. De enige verandering die moet worden aangebracht is dat niet op de letter B moet worden gewacht, maar op een willekeurige toets (of beter gezegd: op een negatieve flank op CA1). Aangezien het toetsenbord programma nog niet geladen is zal de IO65 nooit een ASCII B kunnen lezen op poort A van de VIA, maar wel de CA1 interrupt!

7 Aanpassingen utility software

Voorlopig is alleen de EDITOR aangepast, dankzij de enorme flexibiliteit van het EDITOR programma zijn de veranderingen zeer eenvoudig. De file "ibmeddcmd.mac" bestaat uit allemaal tabellen. In deze tabellen staat aangegeven welk commando bij welke toets hoort. Een tabel is alsvolgt opgebouwd:

```
fcc  ascii-code, tabel-offset
```

Wil men nu ASCII-waarde $08 (BS) als DELETE laten fungeren, dan ziet het er dus zo uit:

```
fcc $08, offset-delete (=48)
```

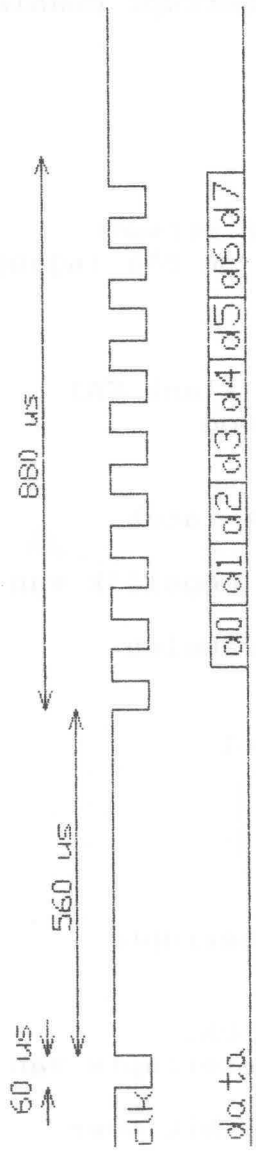
Wat die tabel offset is kun je vinden in de nu al bestaande tabel en blijft gewoon staan. De file 'ibmeddcmd.mac' bestaat uit twee soorten tabellen, namelijk de directe en de second tabellen. De directe codes zijn voor toetsen waarbij control tegelijk ingedrukt is, de secondcodes zijn voor de toetsen in een escape sequence. Is op deze manier 'ibmeddcmd.mac' aangepast, dan start je het commando 'makeed' die een nieuwe (aangepaste) EDITOR gegenereerd. Deze moet dan alleen nog naar de systeemschijf worden gecopieerd en klaar is de nieuwe EDITOR.

N.b.: Van de files makeed en ibmeddcmd.mac is ook een listing afgedrukt.

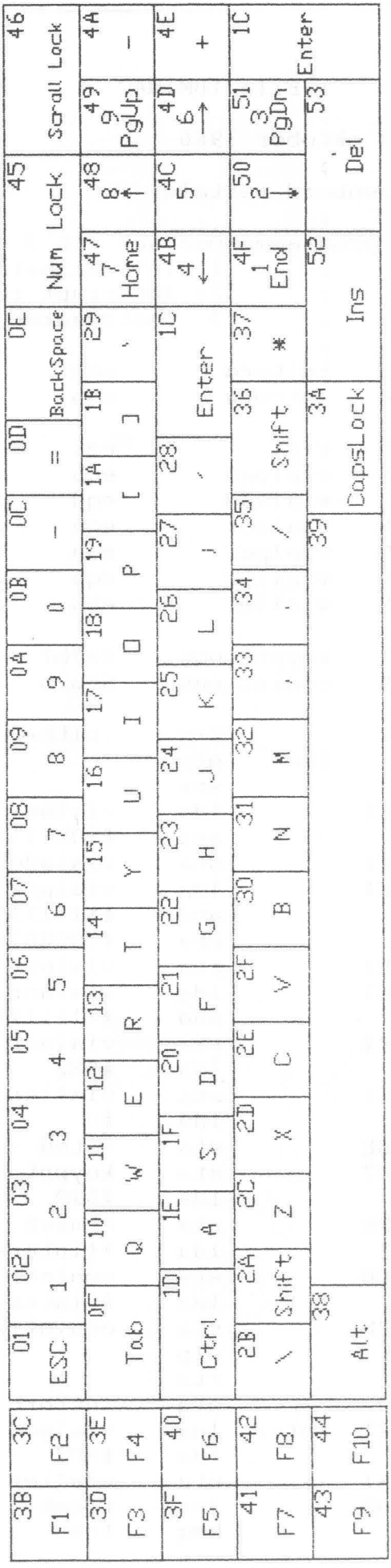
****

MAKEED is een command-file die de bestaande editor ombouwt, zodat van IBM-like toetsenbord de cursortoetsen e.d. gebruikt kunnen worden. Roep MAKEED als commando aan. De optie c moet wel met 'setmode -c MAKEED' gezet zijn. Copieer eerst de EDITOR van de systeemschijf naar de user drive onder de naam EDITOR.DFT. De nieuwe EDITOR zal dan ook op de user drive worden aangemaakt.

```
ECHO Assembling ed commands file.
AS IBMEDDCMD
ECHO Loading the binary files.
LO EDITOR.DFT
LO IBMEDDCMD.BIN
ECHO Appending the binary files.
SAVE -B EDITOR 1000,2FFF
SAVE -A EDITOR 400,4FF
SAVE -A EDITOR 600,6FF
SAVE -A EDITOR 580,5FF
SAVE -A EDITOR 700,7FF
SETMODE -C EDITOR
RENAME EDITOR TEMPORY
RENAME TEMPORY ED$ITOR
ECHO Copy file EDITOR from the user
ECHO drive to your system diskette.
ECHO  First  try this editor version
ECHO by typing u:ed [filename] <ret>
```



Figuur 2. Output van een IBM toetsenbord.



Figuur 3. Voorbeeld van een van de IBM-PC compatibele toetsenborden.

```
                ; File IBM.MAC
                ;
; Datum: 23 oktober 1986
                ;
; IBM toetsenbord vertaler
                ;
;       Dit programma bestaat uit 3 delen, te weten
                ;       1) Initialisatie routine
                ;       2) Interrupt routine (sequentieel-> parallel)
                ;       3) Toetswaarde naar ASCII vertaler
                ;
        A000    initorg         equ     $a000
        BD00    introrg         equ     $bd00

        E100    via1            equ     $e100
        E101    via1pad         equ     via1+1          Data poort A
        E103    via1add         equ     via1+3          Data direction poort A
        E10B    via1acr         equ     via1+11         Auxilary control register
        E10C    via1pcr         equ     via1+12         Peripheral control register
        E10D    via1ifr         equ     via1+13         Interrupt flag register
        E10E    via1ier         equ     via1+14         Interrupt enable register

        E7B7    keypnt  equ     $e7b7
        CB2B    conint  equ     $cb2b

        A000            org     initorg
A000 08         ibm     php
A001 78                 sei                     Disable interrupts
A002 AD 03E1            lda     via1add         Poort PA7 en PA6 ingang
A005 29 3F              and     #%00111111
A007 8D 03E1            sta     via1add
A00A AD 0CE1            lda     via1pcr
A00D 29 FE              and     #%11111110      Negatieve flank CA1
A00F 09 0E              ora     #%00001110      Set CA2 hoog
A011 8D 0CE1            sta     via1pcr
A014 AD 0BE1            lda     via1acr
A017 29 FE              and     #%11111110      Geen input latch
A019 8D 0BE1            sta     via1acr
A01C A9 82              lda     #$82            Interrupt mogelijk van CA1
A01E 8D 0EE1            sta     via1ier
A021 A9 00              lda     #0              Reset variabelen
A023 8D 6ABE            sta     tstnd
A026 8D B7E7            sta     keypnt
A029 A9 20              lda     #$20            jsr inter-1
A02B 8D 2BCB            sta     conint
A02E A9 FF              lda     #(inter-1)&255
A030 8D 2CCB            sta     conint+1
A033 A9 BC              lda     #(inter-1)>>8
A035 8D 2DCB            sta     conint+2
A038 28                 plp                     Enable interrupts
A039 60                 rts
        BD00            org     introrg
BD00 AD 01E1    inter   lda     via1pad         Reset irq CA1
BD03 A9 02              lda     #$02            Disable interrupts van CA1
BD05 8D 0EE1            sta     via1ier
BD08 20 30BD            jsr     flank           Sla eerste bit over
BD0B F0 1B              beq     fout
```

```
)0D A9 80           lda    #$80          8 bits per byte
)0F 8D 6BBE         sta    tmp
)12 20 30BD  1      jsr    flank
)15 F0 11           beq    fout
)17 AD 01E1         lda    via1pad       Lees bit en reset CA1 flag
)1A 0A              asla
)1B 6E 6BBE         ror    tmp
)1E 90 F2           bcc    1.b
)20 AD 6BBE         lda    tmp
)23 20 40BD         jsr    ascii         Vertaal toets naar ASCII
)26 90 02           bcc    tstuit        Test op geldige data
)28 68       fout   pla                   Neem het laatste terugkeeradres
)29 68              pla
)2A A2 82    tstuit ldx    #$82          Enable interrupts van CA1
)2C 8E 0EE1         stx    via1ier
)2F 60              rts
)30 AD 01E1  flank  lda    via1pad       Reset CA1 flag
)33 A2 46           ldx    #700/10       Maximaal 700us geen flank
)35 A9 02           lda    #%00000010    anders time-out
)37 2C 0DE1  1      bit    via1ifr
)3A D0 03           bne    2.f
)3C CA              dex
)3D D0 F8           bne    1.b
)3F 60       2      rts                  (z=1: time-out)

             ; File ASCII.MAC
             ;
Date : 22 oktober 1986
             ;
ASCII converter voor IBM toetsenbord.
             ;   Het programma ascii vertaald toetswaarden
             ;   uit een IBM toetsenbord naar normale ASCII
             ;   waarden. De carry-flag wordt gebruikt om
             ;   aan te geven of er een "geldige" ASCII
             ;   waarde is vertaald of niet (C=0, geldig).

     00EF    nmlck  equ    $ef
     00F7    cplck  equ    $f7
     00FB    alt    equ    $fb
     00FD    shift  equ    $fd
     00FE    cntrl  equ    $fe
     0000    nul    equ    $00

             ; Register gebruik:
             ;  Accu  - Algemeen rekenregister
             ;  X-reg - ASCII waarde (voorzover bekend)
             ;  Y-reg - Toetswaarde

D40 08       ascii  php                  m/v bit in N-flag
D41 29 7F           and    #$7f
D43 A8              tay                  (y-reg toetswaarde)
D44 BE C3BD         ldx    (tascii-1),y  Haal ASCII waarde uit de tabel
D47 28              plp
D48 10 1C           bpl    maak
D4A E0 E0           cpx    #$e0          Is het een control toets?
D4C 90 16           bcc    brkend
D4E E0 F8           cpx    #$f8          Caps- en Num-lock (toggle)
```

```
BD50 90 09           bcc    lockkey
BD52 8A              txa                    Reset het corresponderende bit
BD53 2D 6ABE         and    tstnd
BD56 8D 6ABE         sta    tstnd
BD59 B0 09           bcs    brkend
BD5B 8A       lockkey txa                   Inverteer het "toggle" bit
BD5C 49 FF           eor    #$ff
BD5E 4D 6ABE         eor    tstnd
BD61 8D 6ABE         sta    tstnd
BD64 38       brkend sec                    Geen geldige ASCII waarde
BD65 60              rts
BD66 E0 E0    maak   cpx    #$e0
BD68 90 0F           bcc    gnctrl         Spring bij speciale toets ingedrukt
BD6A E0 F8           cpx    #$f8
BD6C 90 09           bcc    spclend        Shift, Cntrl en Alt: a/u functie
BD6E 8A              txa                   Zet het bijbehorende bit aan
BD6F 49 FF           eor    #$ff
BD71 0D 6ABE         ora    tstnd
BD74 8D 6ABE         sta    tstnd
BD77 38       spclend sec                   Geen geldige ASCII waarde
BD78 60              rts
BD79 E0 61    gnctrl cpx    #'a'
BD7B 90 11           bcc    gnlttr
BD7D E0 7B           cpx    #'z'+1
BD7F B0 0D           bcs    gnlttr         Het is een letter
BD81 A9 0A           lda    #~cplck|~shift
BD83 2C 6ABE         bit    tstnd
BD86 F0 24           beq    mkend          Spring bij Capslock en shift uit
BD88 8A              txa                   Conv. naar de hoofdletter waarde
BD89 29 5F           and    #$5f
BD8B AA              tax
BD8C 10 1E           bpl    mkend
BD8E C0 47    gnlttr cpy    #$47
BD90 90 10           bcc    gnmrk          Numeriek pad toets?
BD92 AD 6ABE         lda    tstnd
BD95 29 12           and    #~nmlck|~shift
BD97 F0 13           beq    mkend
BD99 C9 12           cmp    #~nmlck|~shift
BD9B F0 0F           beq    mkend          En is de num-lock functie aan?
BD9D BE 16BE         ldx    (tnmlock-$47),y      (d.i. num-lock ex-or shift)
BDA0 10 0A           bpl    mkend
BDA2 A9 02    gnmrk  lda    #~shift
BDA4 2C 6ABE         bit    tstnd
BDA7 F0 03           beq    mkend          Spring als shift niet ingedrukt
BDA9 BE 16BE         ldx    (tshift-1),y
BDAC A9 01    mkend  lda    #~cntrl
BDAE 2C 6ABE         bit    tstnd
BDB1 F0 0E           beq    einde          Spring als control niet ingedrukt
BDB3 E0 A0           cpx    #$a0
BDB5 B0 06           bcs    1.f            Als ASCII waarde >= cursor-control
BDB7 8A              txa                   Maak control code van (0-31)
BDB8 29 1F           and    #$1f
BDBA AA              tax
BDBB 10 04           bpl    einde
BDBD 8A       1      txa
BDBE 69 0F           adc    #$10-1         Maak cursor control code van
BDC0 AA              tax
```

```
BDC1 8A       einde  txa
BDC2 18              clc                    Geldige ASCII waarde
BDC3 60              rts

             ; Tabellen

BDC4 1B31323334 tascii fcc  $1b,'1','2','3','4','5','6','7'
BDC9 353637
BDCC 3839302D3D      fcc    '8','9','0','-','=',$7f,$09,'q'
BDD1 7F0971

BDD4 7765727479      fcc    'w','e','r','t','y','u','i','o'
BDD9 75696F
BDDC 705B5D0DFE      fcc    'p','[',']',$0d,cntrl,'a','s','d'
BDE1 617364

BDE4 6667686A6B      fcc    'f','g','h','j','k','l',';','\''
BDE9 6C3B27
BDEC 60FD5C7A78      fcc    '`',shift,'\\','z','x','c','v','b'
BDF1 637662

BDF4 6E6D2C2E2F      fcc    'n','m',',','.','/',shift,$12,alt
BDF9 FD12FB
BDFC 20F7818283      fcc    ' ',cplck,$81,$82,$83,$84,$85,$86
BE01 848586

BE04 8788898AEF      fcc    $87,$88,$89,$8a,nmlck,$80,$a7,$a8
BE09 80A7A8
BE0C A92DA4A5A6      fcc    $a9,'-',$a4,$a5,$a6,'+',$a1,$a2
BE11 2BA1A2

BE14 A3A0AA          fcc    $a3,$a0,$aa

BE17 1B21402324 tshift fcc  $1b,'!','@','#','$','%','^','&'
BE1C 255E26
BE1F 2A28295F2B      fcc    '*','(',')','_','+',$7f,$08,nul
BE24 7F0800

BE27 0000000000      fcc    nul,nul,nul,nul,nul,nul,nul,nul
BE2C 000000
BE2F 007B7D0D00      fcc    nul,'{','}',$0d,nul,nul,nul,nul
BE34 000000

BE37 0000000000      fcc    nul,nul,nul,nul,nul,nul,':','"'
BE3C 003A22
BE3F 7E007C0000      fcc    '~',nul,'|',nul,nul,nul,nul,nul
BE44 000000

BE47 00003C3E3F      fcc    nul,nul,'<','>','?',nul,$12,nul
BE4C 001200
BE4F 2000919293      fcc    ' ',nul,$91,$92,$93,$94,$95,$96
BE54 949596

BE57 9798999A00      fcc    $97,$98,$99,$9a,nul,$03
BE5C 03

BE5D 3738392D34 tnmlock fcc '7','8','9','-','4','5','6','+'
```

```
BE62 35362B
BE65 313233302E       fcc    '1','2','3','0','.'

BE6A 00         tstnd fcc    0       Initieele waarde is 0 (alles uit)

BE6B            tmp   res    1
       A000           end    ibm
```

```
                ; File          IBMEDDCMD.MAC
                ;
; Programma      EDD
                ;
; Function       Editor overlayer voor het IBM toetsenbord
                ;
; Door           Ad Brouwer en Erwin Visschedijk
                ;
; Datum                   mei 1984,  16 oktober 1986
                ;
         0600           org    $600

0600            ibmeddcmd
0600 04         setlu  fcc    4                scroll naar beneden pos
0601 0D                fcc    13               centrum regel
0602 15                fcc    21               scroll naar boven pos
0603 14                fcc    20               aantal regels om te scrollen

0604 80                fcc    $80             default overschrijf/insert mode
0605 00                fcc    0               default wraparound flag
0606 50                fcc    80              maximaal aantal tekens op een regel

0607 0000000000        fcc    0,0,0,0,0,0,0,0,0 reserved
060C 00000000

0610 00080A000C conky  fcc    0,8,$a,0,$c,0,0,0 toegestane kontrole karakters
0615 000000
0618 1B00000000        fcc    $1b,0,0,0,0,$12,
061D 12

061E 0911192129 tabtb  fcc    9,17,25,33,41,49,57,65 default tab posities
0623 313941
0626 4900000000        fcc    73,0,0,0,0,0,0,0
062B 000000

062E 00000000 funstr   fcc    0,0,0,0          functie toets string
0632 1B       seckey   fcc    $1b              <esc> second key
0633 11       repkey   fcc    $11              ^Q herhaal laatste commando
0634 8A       comkey   fcc    $8a              F10 verlaat edit mode
0635 00       comopt   fcc    0                geen tweede verlaat toets
0636 4D       mackey   fcc    'M'              2de M macro toets
0637 80       macopt   fcc    $80              na de tweede toets
```

```
0638 0000000000        fcc     0,0,0,0,0,0     gereserveerd
063D 00

        ;direct
063E A400       functb fcc     $a4,0           links
0640 A602              fcc     $a6,02          rechts
0642 A204              fcc     $a2,04          beneden
0644 A806              fcc     $a8,06          boven
0646 0920              fcc     $09,32          ^I tab
0648 161C              fcc     $16,28          ^V controle karakter insert
064A 1022              fcc     $10,34          ^P spring tab posities
064C 0F24              fcc     $0f,36          ^O spring tab posities terug
064E 0D1E              fcc     $0d,30          <ret> splits regel
0650 123A              fcc     $12,58          ^R reprint scherm (oude regel)
0652 B608              fcc     $b6,08          woord vooruit
0654 B40A              fcc     $b4,10          woord terug
0656 A10E              fcc     $a1,14          einde regel
0658 A70C              fcc     $a7,12          begin regel
065A A310              fcc     $a3,16          scherm vooruit
065C A912              fcc     $a9,18          scherm terug

065E AA28              fcc     $aa,40          delete onder de cursor
0660 7F30              fcc     $7f,48          delete voor de cursor
0662 022A              fcc     $02,42          ^B delete woord
0664 0E2C              fcc     $0e,44          ^N delete tot eol
0666 156E              fcc     $15,110         ^U regel naar de delete buffer
0668 8160              fcc     $81,96          F1 start selecteren
066A 823C              fcc     $82,60          F2 insert paste-buffer

066C B714              fcc     $b7,20          ga naar begin van file
066E B116              fcc     $b1,22          ga naar einde van file

0670 8354              fcc     $83,84          F3 def zoek string en zoekt vw
0672 9358              fcc     $93,88          shift F3 zoekt voorwaards
0674 844E              fcc     $84,78          F4 def zoek en vervang string
0676 945C              fcc     $94,92          shift F4 verv en zoekt voorwaards
0678 A046              fcc     $a0,70          Ins set insert mode
067A B048              fcc     $b0,72          shift ins set overschrijf mode

067C 0C1A              fcc     $0c,26          ^L insert (form feed)
067E 00                fcc     0               beeindig normaal


        ;commando
067F 00                fcc     0

        ;second
0680 5570              fcc     'U',112         2de U ins regel uit delete buffer
0682 4818              fcc     'H',24          2de H centreer huidige regel
0684 5714              fcc     'W',20          2de W ga naar begin van de file
0686 5A16              fcc     'Z',22          2de Z ga naar einde van de file

0688 4936              fcc     'I',54          2de I Insert regel
068A 4A38              fcc     'J',56          2de J join regels

068C 2E2A              fcc     '.',42          2de . delete woord
068E 2F2C              fcc     '/',44          2de / delete rest van de regel
0690 7C2E              fcc     '|',46          2de | delete tot einde file
```

```
0692 2C32               fcc     ',',50          2de , delete woord terug
0694 3F34               fcc     '?',52          2de ? delete tot begin file

0696 4F4E               fcc     'O',78          2de O def zoek- en vervang-string
0698 5050               fcc     'P',80          2de P verw zoek- door      ,,
069A 4C52               fcc     'L',82          2de L vervang zoek- door   ,,
069C 4654               fcc     'F',84          2de F def zoek-string en zoek vw
069E 4156               fcc     'A',86          2de A  ,,    terugwaards
06A0 3A58               fcc     ':',88          2de : zoek vw
06A2 3B5A               fcc     ';',90          2de ; zoek tw
06A4 2A5C               fcc     '*',92          2de * vervang en zoek vw
06A6 2B5E               fcc     '+',94          2de +  ,,    tw
06A8 423E               fcc     'B',62          2de B definieer macro start
06AA 4E40               fcc     'N',64          2de N definieer macro einde
06AC 5442               fcc     'T',66          2de T teken positie
06AE 5944               fcc     'Y',68          2de Y spring naar teken

06B0 5E3C               fcc     '^',60          2de ^ insert paste-buffer
06B2 5B60               fcc     '[',96          2de [ start selecteren

06B4 514A               fcc     'Q',74          2de Q type info over de tekstfile
06B6 434C               fcc     'C',76          2de C start commando regel
06B8 00                 fcc     0               einde second normaal


                ; select direct
06B9 A400               fcc     $a4,0           links
06BB A602               fcc     $a6,02          rechts
06BD A204               fcc     $a2,04          beneden
06BF A806               fcc     $a8,06          boven
06C1 B608               fcc     $b6,08          woord voorwaards
06C3 B40A               fcc     $b4,10          woord terug
06C5 A10E               fcc     $a1,14          einde regel
06C7 A70C               fcc     $a7,12          begin regel

06C9 9166               fcc     $91,102         shift F1 plaats in paste-buffer
06CB 9268               fcc     $92,104         shift F2 voeg toe aan paste-buffer
06CD 1262               fcc     $12,98          ^R breek selectie mode af
06CF 00                 fcc     0               einde selectie mode

                ; select commando
06D0 00                 fcc     0

                ; select second
06D1 4876               fcc     'H',118         2de H verander zaak
06D3 4678               fcc     'F',120         2de F afbreken tot recher kantlijn
06D5 537A               fcc     'S',122         2de S regel uitvullen (spaties)
06D7 447C               fcc     'D',124         2de D delete overtollige spaties
06D9 5C68               fcc     '\\',104        2de \ voeg aan de paste-buffer toe
06DB 7C64               fcc     '|',100         2de | verwijder het geselecteerde

06DD 514A               fcc     'Q',74          2de Q typ info over de textfile
06DF 434C               fcc     'C',76          2de C start commando regel
06E1 00                 fcc     0               einde second selectie

        0600            end     ibmeddcmd

Errors detected: 0
```

Het fijne van HELP.

Door : Erwin Visschedijk.

Op de systeemschijf van DOS65 staat de utility HELP. Als dit commando wordt aangeroepen met als parameter een van de DOS65 commando's (bv. 'help type') dan verschijnt er een verhaal over het gebruik van dat commando. Als alleen 'help' wordt ingetypt dan verschijnt een overzicht van alle tot dan toe bekende DOS65 commando's. Als dan het gewenste commando wordt ingetypt, dan verschijnt hierover de informatie op het beeldscherm. De tekst die op het beeldscherm verschijnt staat eigenlijk in het commando zelf en wordt door het commando help opgezocht in het gevraagde commando.

Als men zelf een utility schrijft, is het ook mogelijk hierin een 'help' gedeelte op te nemen. Het is dan mogelijk met het help commando de 'help-tekst' op het beeldscherm te zetten. Het is vooralsnog niet mogelijk de naam van dat commando in de lijst te laten opnemen van het help commando zelf.

Men gaat als volgt te werk om 'help' zo gek te krijgen informatie te geven over het zelf gemaakte commando.

- Zet voorin het programma een jump over de nu volgende tekst naar het eigenlijke begin van het programma.
- Zet 4 bytes op deze plaats die een label vormen waarnaar het commando help naar kan zoeken. Dit label is 'HELP' waarvan de meest significante bits per letter hoog zijn. Er komt dus te staan $c8,$c5,$cc,$d0.
- Dan volgen er drie bytes die wel aanwezig moeten zijn maar die niet altijd gebruikt hoeven te worden door de gebruiker. Wil men namelijk vanuit het commando zelf de help-tekst ook op het beeldscherm kunnen zetten, dan komen hier de drie bytes te staan die een jump subroutine naar PRINT STRING tot gevolg hebben, dus 203BC0. Wil men niet vanuit het commando zelf deze tekst ook kunnen printen, zet hier dan 3 dummy bytes neer bv. $4c,$4c, $4c.
- Nu volgt de eigenlijke tekst die men wil weergeven. Om een uniforme beeldschermopmaak te krijgen worden de volgende zaken, ook in deze volgorde, van het commando beschreven. Bekijk bv. de help-teksten van de commando's TYPE, PRINT en RS232.
  - Een regel met de beschrijving van het commando.
  - Een regel die begint met 'Syntax :' waarin de syntax van dit commando wordt gedefinieerd. Optionele zaken uit de syntax specificatie worden voorafgegaan door een [en afgesloten door een]. Opties als -CELMN bij TYPE staan altijd tussen haken. Een parameter die optioneel is staat ook tussen haken. (LOAD)
  - Dan volgt een regel die begint met 'Options :' waarin de eventuele opties beschreven worden. Zijn er geen opties dan staat hier 'No options.'
  - Als voor het commando een afkorting bestaat die zelf gemaakt kan worden (zie DOS65 manual pag. 32,33), dan wordt deze abbreviation op deze regel als volgt vermeld: 'Abbreviation : '. Is er geen afkorting, dan staat hier 'No abbreviation.'
  - Als laatste kan er nog een voorbeeld gegeven worden.
  - Sluit de complete tekststring af met een nul byte: $00.

Hier volgt een assembler voorbeeld.
\r is een return in de assembler.

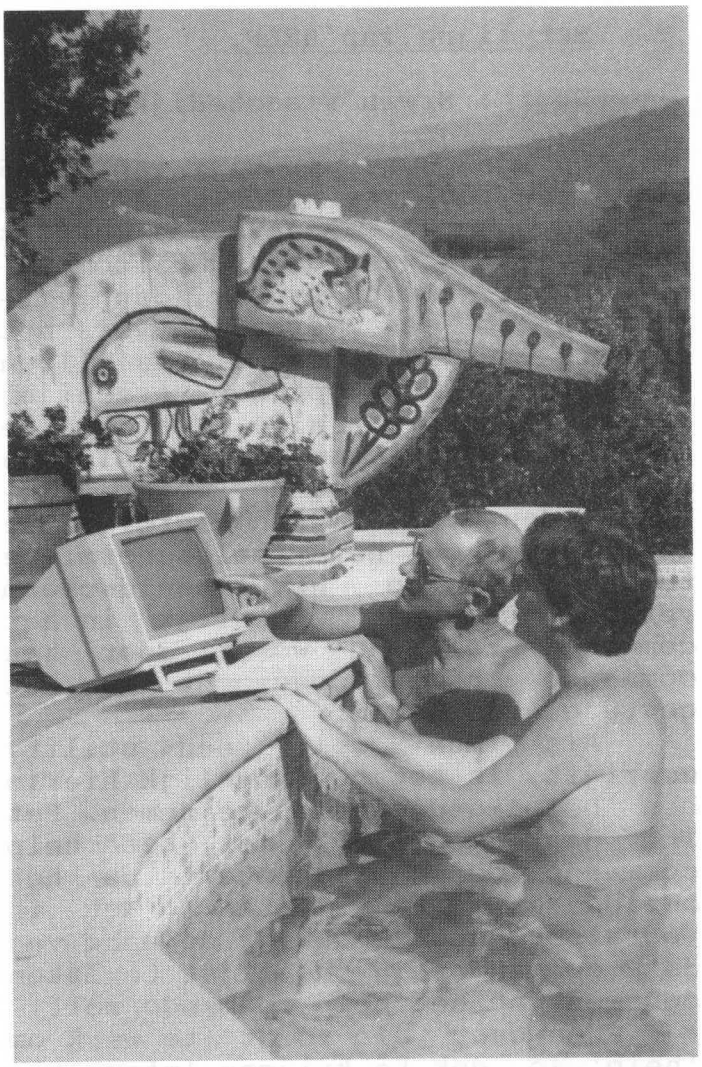
```
     lib DVAR  ;Hierin staat print
     org  $a000

vbeeld
jmp  start

     fcc  $c8, $c5, $cc, $d0
prhelptxt
jsr  print     ;Print text
                    ;until $00
     fcc  '\rFunctie beschrijving.'
     fcc  '\rSyntax  : VBEELD [-AB]'
     fcc  '\rOptions : -A : blabla.'
     fcc  '\r          -B : blabla.'
     fcc  '\rAbbreviation : VB'
     fcc  '\rExample : VBEELD -A'
     fcc  '\rGeeft een voorbeeld.\r'
     fcc  $00

start
lda  #00
enz.
```

****



Due to a lack of Club activities during the summer, the Committee members of our '6502 Kenners' get the opportunity to travel abroad to inspect the equipment of the Club's foreign members and make sure that it does in fact contain a 6502 processor.

I had the pleasure of being inspected by Erwin and Erna Visschedijk. As computers and their operators tend to overheat here during the hot season, I had my system installed in a cooler place.

Unfortunately the picture does not show Erna Visschedijk. As she was not dressed according to 6502 fashion, she had to stay at the other side of the pool where she was generating IRQ waves.

Conrad Kleipool, Cogolin, France.

Vraag en Aanbod.

Te koop:

Apple][/e (enhanced met:
- Amberkleurige Apple monitor
- GRAPPLE printerinterface met kabel
- 80 kolomskaart met 64k RAM
- Z80 kaart voor CP/M
- 128k RAMkaart
- FDC floppy disk controller
- 2 Apple compatible slimline drives

Totaalprijs: fl. 1450.-

Alsmede:

OKI microline 80 met tractorfeed
Prijs fl. 450.-

Te bevragen bij:
A. Mueller
Sinjeur Semeÿnstraat 78'
Amsterdam

TERMINALPROGRAMMA.

Dit terminalprogramma, geschreven in DOS65 basic V2.00, maakt het mogelijk voor mensen die nog geen communicatieprogramma bezitten voor hun modem, om toch contact te leggen met een Fido, bijvoorbeeld de clubfido.
Starten op de normale wijze met RUN
Stoppen kan met het apestaartsymbool "$".

```
1 REM Bram de Bruine terminal.
10 POKE &E133,&0016 :REM Init acia to 300/300
20 POKE &E132,&0005
25 I=0:P=&A000 :REM Poke acia irqroutine in memory
30 READ K
40 POKE P+I,K:I=I+1:IF K<>&00FF THEN 30
50 GOTO 90 :REM Next data contains acia irqroutine
60 DATA 0,0,96,120,173,49,225,41,8,240,8,173,48,225,141,0
70 DATA 160,88,96,173,1,160,240,8,141,48,225,169,0,141
80 DATA 1,160,88,96,255
90 POKE &E76C,&0002:POKE&E76D,&00A0 :REM Swap irq vector
95       PRINTCHR$(12):CURSOR       TO       1,23:PRINT"TERMINAL       300/300
Baud":PRINT:PRINT"TERM>";
100 RB=PEEK(P)
110 IF RB<>0 THEN PRINTCHR$(RB);:POKEP,0:REM Anything received, then print
120 I$="":GETI$
130 IF I$<>""THEN POKEP+1,ASC(I$) :REM Key pressed, then transmit
140 IF I$<>"$"THEN GOTO 100
150 POKE&E76C,&0027:POKE&E76D,&00F3:PRINT:PRINT"EXIT TERMINAL":END
```

Toelichting: De regels 10 en 20 programmeren de ACIA. Regel 30-80 bevatten de data voor een machinecode IRQroutine, die in het geheugen word gepoked. Het eerste byte (P) fungeert als ontvangstbuffer. P+1 is het zendbuffer.
Regel 90 laat de interruptvector naar bovengenoemde IRQroutine wijzen.
De regels 100-150 bevatten het werkelijke terminalgedeelte. Als het ontvangstbuffer niet leeg is (RB <> 0), wordt het ontvangen karakter geprint.
Is RB=0 dan wordt gekeken of er een toets is ingedrukt. Is dit het geval, dan wordt de toetscode via de RS232 poort verzonden naar het modem.

BUGS COMMUNICATION-65 V2.2

In viditel en astrid V2.2 zitten de volgende fouten:
1. Breektoets is niet te wijzigen met MCONFIG.
2. Bij ASCII-upload in half-duplexmode wordt de data niet geprint. (Alleen astrid)

ad. 1.

Omdat het definieren van een breektoets in de regel een eenmalige zaak is, kan men die het beste als volgt wijzigen:

```
VIDITEL                          ASTRID
KBRK EQU $1844                   KBRK EQU $18A4
Zet met MON de gewenste waarde
op deze locatie. (bv $04 voor ^D)
```

```
MON
$ 1844
1844 04
Q
Save -ab s:VIDITEL 1844,1844
```

```
MON
$ 18A4
18A4 04
Q
Save -ab s:ASTRID 18A4,18A4
```

Hobbyisten, die de breektoets ongewijzigd op ESC laten staan hoeven dit niet te doen.

ad. 2

Daar half-duplex bedrijf zelden wordt gebruikt, en de data wel correct naar het modem wordt verzonden, heeft deze bug het gevolg dat men niet zo comfortabel werkt. (Geen data op het scherm).
Voor de meesten zal het niet noodzakelijk zijn deze bug te elimineren.

Bij een volgende versie zullen deze fouten verholpen zijn.
Indien u bugs (of ongemakken) ontdekt, laat het a.u.b. weten!

COMMUNICATIEPAKKET V2.2

Voor DOS65 is een compleet communicatiepakket beschikbaar, bestaande uit:
-Communication-65 (astrid), bedoeld voor communicatie met fido's.
-Viditel-65, speciaal ontworpen voor viditel databanken. (Bv Comnet, Elektuur)
-Configuratieprogramma
-Manuals
-Helputilities

De software is te bestellen bij de DOS65 software coördinator.

Verantwoording Auteurs De 6502 Kenner 52.

In de laatste 6502 Kenner hebben we verzuimd de namen van de auteurs van een aantal artikelen te vermelden. Omdat hiernaer gevraagd werd op de ledenvergadering maken we dat alsnog in orde. Hier zijn ze:

De 6502 Kenners gaan de lucht in	Gert Klein
DOS65, laatste nieuws	Erwin Visschedijk
Subdirectories (pagina 16)	Erwin Visschedijk
Elektuur keyboard (pagina 16)	Erwin Visschedijk
Trage printerspooler	Erwin Visschedijk
Software (pagina 23)	Erwin Visschedijk
Manuals (pagina 23)	Erwin Visschedijk
Hardware (pagina 23)	Erwin Visschedijk
De SmartWatch: 'n slim klokje	Nico de Vries
Tips, of leve de Taiwan-klonen!	Nico de Vries
Nieuw van Acorn: de Archimedes	Adri Hankel
Nieuwe chips van GTE	Adri Hankel
65C02 Tip(je)	Nico de Vries

Nico de Vries

```
                        4 ;*************************************************
                        5 ;* SEARCH [-FLNTU] FILE STRING                   *
                        6 ;*-----------------------------------------------*
                        7 ;* New DOS65 command to search file(s) for a string *
                        8 ;* A.Megens, august 1987      (c) DE 6502 KENNERS  *
                        9 ;*************************************************
          A000          10         org     $a000
          000D          11 cr       equ     13
          0009          12 tab      equ     9
          005F          13 ulmask   equ     %01011111       ;upper/lowercase mask
          00FC          14 cbp      equ     $fc             ;string pointer
          00FF          15 mode     equ     $ff
          C297          16 calpoin  equ     $c297           ;compute pointer
          AA00          17 inbuf    equ     $aa00           ;inputbuffer
          D09F          18 setfiln  equ     $d09f           ;define file name
          D0BD          19 prfile   equ     $d0bd           ;print file name
          D03F          20 open     equ     $d03f           ;open file
          D000          21 sread    equ     $d000           ;read single char.
          D048          22 close    equ     $d048           ;close file
          D0B7          23 ermes    equ     $d0b7           ;print error messsage
          ABE0          24 namb     equ     $abe0           ;file spec. buffer
          C020          25 inchar   equ     $c020           ;get key in A
          C023          26 prchar   equ     $c023           ;print char. in A
          C038          27 hexout   equ     $c038           ;print A in hex
          C03B          28 print    equ     $c03b           ;print string until 0
          C02F          29 crlf     equ     $c02f           ;print crlf (newline)
                        30 ;
A000 4C 82A2            31 start    jmp     search          ;start address
                        32 ;-----------------------------------
                        33 ; enables short view of possibilities
                        34 ; with the DOS "HELP SEARCH" command
                        35 ;-----------------------------------
A003 C8C5CCD04C         36         fcc     $c8,$c5,$cc,$d0,'LLL' ;HELP label
A00A 0D                 37 info     fcc     cr
A00B 46756E6374         38         fcc     'Function: search file(s) for string.',cr
A030 53796E7461         39         fcc     'Syntax  : SEARCH [-FLNWU] file(s) string.',cr
A05A 4F7074696F         40         fcc     'Options : -F : Do not print filename(s).',cr
A083 2020202020         41         fcc     '          -L : Do not print matched line(s).',cr
A0B0 2020202020         42         fcc     '          -N : Do not print line numbers.',cr
A0DA 2020202020         43         fcc     '          -W : Wait after start to insert diskettes.',cr
A10F 2020202020         44         fcc     '          -U : Ignore upper/lowercase',cr
A135 4162627265         45         fcc     'Abbreviation : SE',cr
A147 57696C6420         46         fcc     'Wild card characters are allowed in the source filespec.',cr
A180 537472696E         47         fcc     'String may contain control codes or several words',cr,cr
A1B3 4578616D70         48         fcc     'Examples : SEARCH filename1,filename2  print',cr
A1E0 7365617263         49         fcc     'search 2 files for occurance of string print.',cr
A20E 5345415243         50         fcc     'SEARCH -u drv:[dir/]*.mac test',cr
A22D 636865636B         51         fcc     'check all .mac files on specified drive for occurance of string test',cr
A272 6F72207374         52         fcc     'or string TEST',cr
A281 00                 53         fcc     0
                        54 ;-----------------------------------
                        55 ; Search (-options) (file) (string)
                        56 ;-----------------------------------
A282 A9 00              57 search   lda     #0
A284 AA                 58         tax
A285 9D D8A4            59 clropt   sta     opend,x         ;clear options
A288 E8                 60         inx
A289 E0 05              61         cpx     #opend-options
A28B D0 F8              62         bne     clropt
A28D A9 AA              63         lda     #inbuf>>8       ;point to start of inputbuffer
A28F A0 00              64         ldy     #inbuf&255
A291 85 FD              65         sta     cbp+1
A293 84 FC              66         sty     cbp
A295 20 9BA4            67         jsr     nextpar         ;point to first parameter (filename/options)
A298 90 06              68         bcc     cont            ;ok, continue
A29A A9 12              69 nofile   lda     #$12            ;else filename missing
A29C 38                 70 opterr   sec
A29D 4C 91A4            71 lerr     jmp     error
                        72
                        73 ;-----------------------------------
                        74 ; check options
                        75 ;-----------------------------------
A2A0 A0 00              76 cont     ldy     #0
A2A2 B1 FC              77         lda     [cbp],y         ;check for options
A2A4 C9 2D              78         cmp     #'-'
A2A6 D0 29              79         bne     defile          ;no options, define filename
A2A8 C8                 80 getopt   iny
A2A9 B1 FC              81         lda     [cbp],y         ;get options
A2AB C9 20              82         cmp     #' '            ;reached end of optionfield?
A2AD F0 1D              83         beq     pfile           ;yes, point to filename
```

```
A2AF C9 0D      84          cmp     #cr             ;reached EOL?
A2B1 F0 E7      85          beq     nofile          ;then no filename given
A2B3 29 5F      86          and     #ulmask         ;accept upper/lowercase
A2B5 A2 00      87          ldx     #0
A2B7 DD D3A4    88  sopt    cmp     options,x       ;check option table
A2BA F0 0A      89          beq     enable          ;enable this one
A2BC E8         90          inx
A2BD E0 05      91          cpx     #opend-options  ;all options checked?
A2BF D0 F6      92          bne     sopt            ;no, try next
A2C1 A9 15      93          lda     #$15            ;else 'Option error'
A2C3 4C 9CA2    94          jmp     opterr
A2C6 9D D8A4    95  enable  sta     opend,x         ;set option flag
A2C9 4C A8A2    96          jmp     getopt          ;and get next option
                97  ;-----------------------------
                98  ; get file definition
                99  ;-----------------------------
A2CC 20 A8A4   100  pfile   jsr     wfnonsp         ;search filename
A2CF B0 C9     101          bcs     nofile          ;not found
A2D1 AD DAA4   102  defile  lda     waitflg         ;check -W option
A2D4 F0 1E     103          beq     10.f            ;not set, continue
A2D6 20 3BC0   104          jsr     print           ;else give opportunity to swap diskettes
A2D9 496E736572 105         fcc     'Insert diskette..... ',0
A2EE 20 20C0   106          jsr     inchar          ;wait for key
A2F1 20 2FC0   107          jsr     crlf
A2F4 A5 FD     108  10      lda     cbp+1
A2F6 A4 FC     109          ldy     cbp
A2F8 20 9FD0   110          jsr     setfiln         ;define file-name
A2FB B0 A0     111          bcs     lerr            ;error in name
A2FD 8A        112          txa
A2FE 8D E0AB   113          sta     namb
A301 A5 FD     114          lda     cbp+1
A303 A4 FC     115          ldy     cbp
A305 A2 81     116          ldx     #$81            ;open file for ASCII reading
A307 20 3FD0   117          jsr     open
A30A B0 91     118          bcs     lerr            ;error opening file
A30C 8E D1A4   119          stx     fmode
               120  ;-----------------------------
               121  ; get searchstring
               122  ;-----------------------------
A30F 20 9BA4   123          jsr     nextpar         ;point to next parameter <searchstring>
A312 90 1C     124          bcc     mloop           ;start search
A314 20 3BC0   125          jsr     print
A317 5365617263 126         fcc     'Searchstring missing',cr,0
A32D 4C 8EA4   127          jmp     xerror
               128  ;-----------------------------
               129  ; search opened file
               130  ;-----------------------------
A330 20 3BC0   131  mloop   jsr     print
A333 2A2A2A2A2A 132         fcc     '*****************************',cr,0
A353 AD D8A4   133          lda     fileflg         ;print filename?
A356 D0 11     134          bne     11.f            ;no, skip it
A358 20 3BC0   135          jsr     print
A35B 46696C653A 136         fcc     'File: ',0
A362 38        137          sec                     ;show current file
A363 20 BDD0   138          jsr     prfile          ;with drive spec.
A366 20 2FC0   139          jsr     crlf
A369 A9 00     140  11      lda     #0              ;reset line counter
A36B 8D CAA4   141          sta     linecnt
A36E 8D CBA4   142          sta     linecnt+1
A371 8D CCA4   143          sta     linecnt+2
A374 8D D2A4   144          sta     match
               145  ;-----------------------------
               146  ; get one line in linebuf
               147  ;-----------------------------
A377 A0 00     148  rline   ldy     #0              ;read one line from file
A379 8C CDA4   149  loop    sty     ysav
A37C AE D1A4   150          ldx     fmode
A37F 20 00D0   151          jsr     sread           ;read single char.
A382 B0 76     152          bcs     lbr             ;end_of_file
A384 AC CDA4   153          ldy     ysav
A387 99 DDA4   154          sta     linebuf,y
A38A C8        155          iny
A38B C9 0D     156          cmp     #cr             ;CR?
A38D D0 EA     157          bne     loop            ;no, get next char
               158  ;-----------------------------------
               159  ; check this line for matched strings
               160  ;-----------------------------------
A38F AD CAA4   161  nline   lda     linecnt         ;update line counter
A392 18        162          clc
A393 F8        163          sed
A394 69 01     164          adc     #1
```

```
A396 8D CAA4    165          sta    linecnt
A399 AD CBA4    166          lda    linecnt+1
A39C 69 00      167          adc    #0
A39E 8D CBA4    168          sta    linecnt+1
A3A1 AD CCA4    169          lda    linecnt+2
A3A4 69 00      170          adc    #0
A3A6 8D CCA4    171          sta    linecnt+2
A3A9 D8         172          cld
A3AA A2 00      173          ldx    #0
A3AC BD DDA4    174 sloop1   lda    linebuf,x       ;get char. from line buffer
A3AF C9 0D      175          cmp    #cr             ;reached end?
A3B1 F0 44      176          beq    rnext           ;yes, read next line
A3B3 8E D0A4    177          stx    xpos
A3B6 A0 00      178          ldy    #0
A3B8 B1 FC      179 sloop2   lda    [cbp],y         ;get char. from searchstring
A3BA C9 0D      180          cmp    #cr             ;if EOL then matched!
A3BC F0 3F      181          beq    found
A3BE 48         182          pha
A3BF BD DDA4    183          lda    linebuf,x
A3C2 8D CEA4    184          sta    asav
A3C5 68         185          pla
A3C6 8E CFA4    186          stx    xsav
A3C9 AE D9A4    187          ldx    caseflg         ;ignore upper and lowercase?
A3CC F0 17      188          beq    check           ;no, check the way it is
A3CE C9 41      189          cmp    #'A
A3D0 90 13      190          bcc    check           ;also digits and controlchar.
A3D2 29 5F      191          and    #ulmask         ;else convert to uppercase
A3D4 48         192          pha
A3D5 AE CFA4    193          ldx    xsav
A3D8 BD DDA4    194          lda    linebuf,x       ;buffer too
A3DB C9 41      195          cmp    #'A
A3DD 90 02      196          bcc    6.f
A3DF 29 5F      197          and    #ulmask
A3E1 8D CEA4    198 6        sta    asav
A3E4 68         199          pla
A3E5 AE CFA4    200 check    ldx    xsav
A3E8 CD CEA4    201          cmp    asav            ;compare buffer and string
A3EB D0 04      202          bne    nextch          ;not matched, try next in buffer
A3ED E8         203          inx                    ;else point to next char. in buffer
A3EE C8         204          iny                    ;and searchstring
A3EF D0 C7      205          bne    sloop2
A3F1 AE D0A4    206 nextch   ldx    xpos
A3F4 E8         207          inx                    ;next char. in linebuffer
A3F5 D0 B5      208          bne    sloop1
A3F7 4C 77A3    209 rnext    jmp    rline           ;read next line of file
A3FA 4C 42A4    210 lbr      jmp    nfile           ;extend branch, open next file
                211
                212 ;-------------------------------
                213 ; matched string found, show it
                214 ;-------------------------------
A3FD A9 01      215 found    lda    #1              ;a matched string is found
A3FF 8D D2A4    216          sta    match
A402 AD DCA4    217          lda    numflg          ;print line number?
A405 D0 21      218          bne    prline          ;no, skip this part
A407 20 3BC0    219          jsr    print
A40A 4C6F6E653A 220          fcc    'Line: ',0      ;print line number
A411 AD CCA4    221          lda    linecnt+2
A414 20 BDA4    222          jsr    prnum
A417 AD CBA4    223          lda    linecnt+1
A41A 20 BDA4    224          jsr    prnum
A41D AD CAA4    225          lda    linecnt
A420 20 38C0    226          jsr    hexout
A423 A9 09      227          lda    #tab
A425 20 23C0    228          jsr    prchar
A428 AD DBA4    229 prline   lda    lineflg         ;print line?
A42B D0 0F      230          bne    pdone           ;no, skip this part
A42D A0 00      231          ldy    #0
A42F B9 DDA4    232 ploop    lda    linebuf,y
A432 20 23C0    233          jsr    prchar
A435 C9 0D      234          cmp    #cr
A437 F0 03      235          beq    pdone
A439 C8         236          iny
A43A D0 F3      237          bne    ploop
A43C 20 2FC0    238 pdone    jsr    crlf
A43F 4C 77A3    239          jmp    rline           ;and read next line
                240 ;-------------------------------
                241 ; EOF detected, open next file
                242 ;-------------------------------
A442 20 48D0    243 nfile    jsr    close           ;close current file
A445 AD D2A4    244          lda    match
A448 D0 1F      245          bne    1.f
```

```
A44A 20 3BC0      246           jsr     print
A44D 4E6F206D61   247           fcc     'No matched string(s) found',cr,0
A469 E6 FF        248  1        inc     mode
A46B A0 00        249           ldy     #0
A46D B1 FC        250           lda     [cbp],y
A46F 8D CEA4      251           sta     asav
A472 A9 0D        252           lda     #cr             ;put temp. EOL at start of searchstring
A474 91 FC        253           sta     [cbp],y
A476 A9 FA        254           lda     #$fa            ;next file
A478 A2 81        255           ldx     #$81            ;open file for ASCII reading
A47A 20 3FD0      256           jsr     open
A47D 48           257           pha
A47E A0 00        258           ldy     #0
A480 AD CEA4      259           lda     asav            ;restore searchstring
A483 91 FC        260           sta     [cbp],y
A485 68           261           pla
A486 B0 0D        262           bcs     exit            ;error opening next file, return to DOS
A488 8E D1A4      263           stx     fmode
A48B 4C 30A3      264           jmp     mloop           ;search this file
                  265  ;
A48E 20 48D0      266  xerror   jsr     close           ;close file
A491 20 B7D0      267  error    jsr     ermes           ;print error
A494 60           268  done     rts                     ;return to DOS
A495 C9 13        269  exit     cmp     #$13
A497 F0 FB        270           beq     done            ;'file not found' error, then all files done
A499 D0 F6        271           bne     error           ;else other error occured
                  272  ;-------------------------------
                  273  ;point to next parameter
                  274  ;-------------------------------
A49B A0 FF        275  nextpar  ldy     #-1             ;search next parameter
A49D C8           276  wfspace  iny
A49E B1 FC        277           lda     [cbp],y         ;get char. from inputbuffer
A4A0 C9 0D        278           cmp     #cr
A4A2 F0 14        279           beq     parerr          ;if EOL parameter missing
A4A4 C9 20        280           cmp     #' '            ;space?
A4A6 D0 F5        281           bne     wfspace         ;no, try next
A4A8 C8           282  wfnonsp  iny                     ;skip spaces
A4A9 B1 FC        283           lda     [cbp],y         ;get char. from inputbuffer
A4AB C9 0D        284           cmp     #cr
A4AD F0 09        285           beq     parerr          ;if EOL parameter missing
A4AF C9 20        286           cmp     #' '            ;space?
A4B1 F0 F5        287           beq     wfnonsp         ;yes, try next
A4B3 20 97C2      288           jsr     calpoin         ;update pointer
A4B6 18           289           clc
A4B7 60           290           rts
A4B8 38           291  parerr   sec
A4B9 60           292           rts
A4BA 84 FC        293           sty     cbp
A4BC 60           294           rts
                  295  ;-------------------------------
                  296  ;print linenumber, convert
                  297  ;leading zero's to spaces
                  298  ;-------------------------------
A4BD D0 08        299  prnum    bne     prnum1
A4BF A9 20        300           lda     #' '
A4C1 20 23C0      301           jsr     prchar
A4C4 4C 23C0      302           jmp     prchar
A4C7 4C 38C0      303  prnum1   jmp     hexout
                  304  ;-------------------------------
                  305  ; variables and tables
                  306  ;-------------------------------
A4CA 000000       307  linecnt  fcc     0,0,0           ;line counter
A4CD 00           308  ysav     fcc     0
A4CE 00           309  asav     fcc     0
A4CF 00           310  xsav     fcc     0
A4D0 00           311  xpos     fcc     0
A4D1 00           312  fmode    fcc     0               ;save fd
A4D2 00           313  match    fcc     0               ;match flag
A4D3 4655574C4E   314  options  fcc     'FUWLN'         ;possible options
        A4D8      315  opend    equ     $               ;for every option a flag
A4D8 00           316  fileflg  fcc     0               ;-F option
A4D9 00           317  caseflg  fcc     0               ;-U option
A4DA 00           318  waitflg  fcc     0               ;-W option
A4DB 00           319  lineflg  fcc     0               ;-L option
A4DC 00           320  numflg   fcc     0               ;-N option
A4DD              321  linebuf  res     256             ;line buffer
                  322
        A000      323           end     start

Errors detected: 0
```

Drive Steprate in DOS65

De steprate van een drive is de snelheid waarmee de kop van de ene track naar de volgende stapt. Het is de tijd tussen twee step-pulsen.

Default staat de steprate van de DOS65 drives op 12 ms. Dit is de snelheid die gekozen wordt door de boot routine. Daarna neemt de DOS van schijf het over en dan hangt het af van de boot die geïnstalleerd is welke steprate er genomen wordt (12, 6 of 3 ms).

Als men echter oudere drives op DOS65 wil aansluiten dan kan het zijn dat de steprate van 12 ms te snel is. Het is mogelijk zelfs 20 of 30 ms te kiezen. Als het niet de drive is van waaruit moet worden opgestart, dan kan men volstaan met alleen de DOS op schijf te wijzigen. Moet er echter van de trage drive worden opgestart, dan moet ook de eprom (IO65) worden aangepast en omgeprogrammeerd.

Het installatie programma op de systeemschijf voorziet niet in een keuze van 20 of 30 ms steprate. Het een en ander moet dan zelf worden gewijzigd.

De 3 ms steprate wordt met een truuk uit de 6 ms gehaald. DOS65 schakelt dan softwarematig even over naar een 8 inch drive die een dubbele clock snelheid heeft. De floppy disk controller krijgt dan tijdens het steppen een dubbele clock snelheid, zodat de steprate verdubbeld wordt.

Er staan in de programmatuur twee waarden die moeten worden aangepast om de steprate te wijzigen. In de listings heten die REST en SEEKT. Hieronder volgt een tabel voor de waarden die REST en SEEKT moeten hebben voor de verschillende snelheden. De snelheden kunnen niet voor iedere drive afzonderlijk worden ingesteld.

	6ms	12ms	20ms	30ms
REST	08	09	0A	0B
SEEKT	18	19	1A	1B

In de eprom IO65 staat REST op adres FEF2 en staat SEEKT op FEBF.

Om de systeemschijf aan te passen voor steprates van 20 of 30 ms moet het programma DISKDOCTOR gebruikt worden.

In DISKDOCTOR kan een track gekozen worden met het T commando en een sector met het S commando. In de nieuwere DISKDOCTOR's kunnen deze hexadecimaal worden ingegeven. In de oudere moet dat decimaal worden gedaan. De informatie op de schijf is ook hexadecimaal, dus de gebruikers van de oudere DD moeten de hex waarden omrekenen naar decimaal.
Men gaat als volgt te werk:

- Copieer eerst een systeemschijf om op te experimenteren. (Vergeet niet het BOOTLINK commando.) De file BOOT wordt gebruikt als DOS65 operating system en hierin staan de waarden die gewijzigd moeten worden.
- Start DISKDOCTOR op en kies de drive waarin de zojuist gecopieerde diskette zit.
- Ga naar track 0, sector 3. De laatste 2 bytes op de regel waar een filenaam staat geven aan op welke track en sector het TSL blok van die file staat.
- Ga naar de track en sector die aangegeven worden bij de file BOOT.
- Op deze sector staat vanaf adres $20 per 2 bytes de track en sector nummers van de data blokken. Adres $40 en $41 geven de track en sector aan van het 17e data blok. Ga naar het 17e blok door de juiste track en sector te kiezen.
- Zoek in deze track naar A019 en A009. De A019 en A009 staan alleen in deze sector als de file BOOT gecopieerd is van de gedistribueerde systeemschijf als hierop nog geen installatie programma is losgelaten. Anders wordt namelijk de file BOOT overschreven door de inhoud van BOOTB (6ms) of BOOTC (3ms). BOOTA is de 12ms versie.
- Wijzig de 19 en de 09 volgens de tabel.
- Schrijf deze sector weer weg met het commando W.

Succes, Erwin Visschedijk.

In den beginne.....en hoe het verder ging.

(Of: de rol van Commodore).

Door: N. de Vries

Het volgende nummer van dit blad zal het eerste nummer van de twaalfde jaargang zijn. Onwillekeurig denk je dan, zo rond de jaarwisseling, terug naar hoe het ooit eens was. Zo ook over onze club, de 6502, de KIM, de andere eerste machines. Daarom eens een verhaal over het begin. Voor diegenen die al tien jaar of langer lid zijn: leuk om te lezen misschien. Voor onze nieuwere leden: zo is het dus gekomen.....

MOS Technology.

In 1974 gebeurde er iets, dat waarschijnlijk iedere dag wel gebeurt: een bedrijf neemt een ander bedrijf over. Of preciezer: Commodore, fabrikant van rekenmachines en kantoormeubelen neemt de chipfabriek MOS Technology over, met als doel zelf chips voor haar rekenmachines te maken en minder afhankelijk van anderen te zijn.
MOS Technology maakte toen wat RAM- en ROM-geheugens, vrijwel allemaal second sources van andere fabrikanten, zoals Motorola en Intel. Het was (en is) een van de kleinere chipfabrieken. Een van de directeuren, Chuck Peddle genaamd hield zich ook bezig met microprocessors, en wel de 6800 van Motorola. Chuck kreeg het gevoel dat het beter kon en begon na te denken. Na enige tijd ontwierp hij een nieuwe microprocessor die wij allemaal kennen: de 6502.

Als uitgangspunt nam Chuck de 6800 die hij immers kende. De instructieset was naar zijn smaak te groot. Hij ontwierp een kleinere set, die echter wel dezelfde mogelijkheden had. De 6800 was verder voorzien van twee accu's, een 16 bit index register en een 16 bit stackpointer. Het viel Chuck op, dat in veel gevallen 8 bits voor deze registers meer dan voldoende waren. De 6502 kreeg dan ook een strikte 8-bit architectuur met 1 accu. Om het verlies van een accu en de verkleining van de 16 bit index naar 8 bit een beetje goed te maken cre erde hij er een tweede index register Y bij. Ook de adressering kon verbeterd worden. Zo kreeg de 6502 indirecte adresseermodes (kunt u nog zonder?) die toendertijd zeer uniek waren. Later zouden de 6809 en de Z80 als enige 8-bit CPU's ook indirecte modes krijgen. Wat bleef van de 6800 was de businterface en het bijbehorende protocol en de pinning. Op die manier kon de 6502 gebruik maken van de reeds bestaande 6800-serie bouwstenen, waaronder het 6810 RAM, de 6820 PIA en de 6840 PTM. Deze feiten waren overigens voor Motorola aanleiding tot een proces, dat zij uiteindelijk verloren.
Een apart verhaal vormden de prestaties van de processor. In die tijd was 1 MHz zo ongeveer de enig praktische clocksnelheid omdat geheugens domweg niet sneller waren. Prestatiewinst moest dus komen uit een zo klein mogelijk aantal clockcycli per instructie. De 6502 kreeg een zogenaamde pipe-line structuur. In een dergelijke structuur doet de CPU in principe steeds meer dan 1 ding tegelijk. Een voorbeeld: terwijl de instructiedecoder het opgehaalde opcodebyte decodeert, haalt de busbesturing al vast het volgende byte op dat mogelijk de operand of de volgende opcode is. Op die manier bespaart men clockcycli. Dit verklaart bijvoorbeeld waarom JMP en JSR relatief veel cycles vereisen: de pipe-line faalt hierbij zodat er tenminste 1 extra geheugencyclus nodig is. Dankzij de pipe-line is de 6502 een opvallend snelle CPU geworden. Hij is zelfs zo snel dat een 4 MHz 6502 geen gek figuur slaat ten opzichte van 68000 en 8086 CPU's die op 4 of 6 MHz lopen!

Dankzij Chuck Peddle kreeg Commodore dus zijn eigen microprocessor, die in

ontwerperskringen voor enige opschudding zorgde. In die tijd hadden ontwerpers niet veel te kiezen: Intel maakte de 8080 en Motorola de 6800. De 6502 was dus een welkome aanvulling. Met deze drie microprocessors werd heel anders omgesprongen dan nu. Hogere programmeertalen waren nog niet beschikbaar, veel geheugen ook niet, en niemand had eigenlijk een idee wat de werkelijke mogelijkheden waren. Het gebruik van microprocessors bleef daarom in eerste instantie beperkt tot een aantal techneuten die in hobbykamertjes mini-systeempjes in elkaar prutsten. Dit leidde soms tot het uitbrengen van zo'n systeempje in bouwdoosvorm: de kit.

Bij MOS Technology was het al niet anders. De eerste kit die gemaakt werd rond de 6502 was de KIM (Dit betekent: Keyboard Input Monitor) en deze werd in 1975 door MOS Technology op de markt gebracht. De KIM zag er net zo uit als alle andere kits: een CPU (de 6502 op 1 MHz), wat geheugen (256 byte was heel gewoon, 1kbyte al een luxe), een nogal primitieve monitor in ROM en de noodzakelijke I/O. De I/O bestond uit een tweetal RRIOTs die ten dele gebruikt werden om het hexadecimale toetsenbord en de zes zevensegment displays aan te sturen. Verder was de KIM voorzien van een cassetteinterface om de programma's te kunnen bewaren. Op sommige KIMs is de invloed van Commodore waar te nemen: zij dragen het Commodore-embleem (zie bijvoorbeeld de aquarel op de oudere nummers).

Na de introductie van de KIM liep het storm. Er werden er duizenden verkocht: de markt was er kennelijk rijp voor. De kopers kwamen uit drie typische groepen: de industrie, die er kleine besturinkjes mee bouwde, het onderwijs, die er de hedendaagse schare computergekken mee opvoedde en de technisch ge nteresseerde hobbyist. De aantrekkingskracht was groot: men sluite een voeding en een cassetterecorder aan en men heeft een hele echte computer! Niet alleen de aantrekkingskracht was groot: ook de problemen. Een hele generatie technici werd plotsklaps geconfronteerd met iets heel nieuws: de microprocessor. Iedereen worstelde met in principe dezelfde problemen, veelal ontstaan uit dezelfde oorzaak: gebrek aan informatie en ervaring. Dit was een ideale voedingsbodem voor allerlei clubs en clubjes waarin van gedachten werd gewisseld over de nieuwe speeltjes. Overal ter wereld schoten de gebruikersclubs dan ook als paddestoelen uit de grond. Zo werd in Nederland een eerste computerclub opgericht door een aantal vooruitziende lieden: de KIM Gebruikers Club Nederland. Het clubblad van deze vereniging heeft u nu in handen. Het blad Radio Bulletin zorgde via allerlei leerzame artikelen voor een nog grotere populariteit van de KIM.

De ontwikkelingen stonden echter niet stil. Chuck Peddle had nog meer idee n. De KIM en zijn broertjes waren allesbehalve gebruikersvriendelijk: geen kast, een losse voeding, een losse cassetterecorder, programmeren in machinetaal, en last but not least: een primitieve gebruikersinterface met klein hex display en het kleine toetsenbord. Het kon allemaal mooier. Het moest om te beginnen in 1 enkele kast. Dit was voor Commodore geen probleem: men had ruime ervaring met stalen kantoormeubelen. Het toetsenbord moest een schrijfmachine-indeling krijgen. Zo geschiedde. De displays werden vervangen door een heus TV-scherm. Ook voor het onvriendelijke programmeren in machinetaal was een oplossing. Bij een klein bedrijfje, dat de naam Microsoft droeg, had een 17-jarig jochie een BASIC-interpreter geschreven die op de 6502 liep. Al deze ingredi nten werden in een grote pot gekiept. Toen het gekookte gaar was, was het midden 1977 geworden. Het product heette:

<u>De PET.</u>

In die dagen was het beeld van een computer heel anders dan nu: computers, zo dacht de gewone man, vullen grote zalen en kosten zeer, zeer veel geld, zowel in aanschaf als in gebruik. Commodore introduceerde de PET (afkorting van: Personal Electronic Transactor, zegt niets over de kwaliteit van het apparaat!). Voor $495 (toen ca. fl. 2200.-) kreeg je een complete computer in een kast die er futuristisch uitzag. De PET had 4 kbyte geheugen, standaard een BASIC interpreter met floating point, een scherm met een werkelijk uitstekend beeld van 25 regels met ieder 40 tekens en bezat verder standaard een ingebouwde cassetterecorder die niet alleen snel was maar ook zeer betrouwbaar bleek. De gewone man kon hem kopen en gemakkelijk leren bedienen, mede dankzij de nog altijd niet overtroffen schermeditor.

De bom barstte. Commodore werd overspoeld met orders. De verkoop ging zelfs zo goed dat men niet eens tijd had om andere dingen te doen zoals goede documentatie schrijven, of een printer ontwikkelen of zelfs gebruikersvragen beantwoorden. Dit probleem was zo groot, dat men er tot in 1979 last van zou hebben.

Ongeveer tegelijkertijd was er een technisch iemand, die Steve Wozniak heette. Hij bezat een oude Volkswagenbus en de garage van zijn ouderlijk huis was zijn hobbykamer. In zijn vrije tijd bouwde Steve een 6502 computer. De machine was voorzien van een aantal speciale zaken: 16 kbyte dynamisch geheugen (zeer veel voor die tijd) dat met een zeer forse investering kon worden uitgebreid tot 48 kbyte, een geschakelde voeding, high-resolution graphics, en een aantal uitbreidingsslots waarop de 6502 bus (gedeeltelijk) beschikbaar was. Verder bezat de machine net als de PET een BASIC interpreter (weliswaar alleen integers) en 40 koloms video. Uniek in de Apple was de standaard aanwezige machinetaalmonitor, die nota bene ook nog kon disassembleren. In tegenstelling tot de PET, wiens toetsenbord duidelijk Commodore's calculatorachtergrond verraadde, had Steve's machine een heus schrijfmachinetoetsenbord dat echter alleen hoofdletters kon produceren. Steve Wozniak noemde zijn machine de Apple. Net als Commodore is Apple heden ten dage een computerfabrikant die nog steeds meetelt.

Commodore.

Terug naar Commodore. Omdat men het zo druk had met het verkopen van PETs, bleven noodzakelijke dingen als een printer en een diskdrive (te) lang uit. Ook het verhelpen van de bugs in de ROMs van de PET duurde lang. In 1979, toen men de grote klap te boven begon te komen, verschenen eindelijk de langverwachte printer en de diskdrive. Tegelijkertijd werd de PET op de helling gezet. Er werd een volwaardig toetsenbord gemonteerd (de cassetterecorder werd extern), terwijl er ook een vrijwel bugvrije ROMset verscheen, die naar het voorbeeld van Apple was uitgebreid met een machinetaal monitor. Ook werden de machines nu leverbaar met geheugens tot 32 kbyte. Voor het eerst kon er met de PET (ook Apple was inmiddels zover en introduceerde een DOS en drives) aan serieuze zaken gedacht worden zoals zakelijke toepassingen. Een tekstverwerker verscheen. De verkopen bleven stijgen, terwijl de prijzen nauwelijks daalden. De computer gebaseerd op de microprocessor had zijn intrede gedaan in het kantoor.

Er ontstond vraag naar machines met 80 koloms schermen, grotere diskdrives en snellere printers. In 1980 kwam Commodore uit met de 8032, die voorzien was van 80 koloms video met nog krachtiger edit-mogelijkheden. Deze machine was zodanig geprijsd en zo goed uitgedacht, dat de 8032 gedurende de VIJF jaar daarna de meest verkochte personal computer werd in Europa. Intussen werd de gewone man niet vergeten. Men had de technisch ge nteresseerde hobbyist al een PET verkocht en de kantoren een 8032. De

man in straat was nu aan de beurt.

In het najaar van 1981 verscheen de VIC-20. Er waren twee dingen nieuw aan deze computer: het was de eerste Commodore computer zonder scherm. Dat moest vanwege het tweede nieuwtje: kleur. De VIC-20 ontleende zijn naam aan de Video-Interface-Chip, die een kleurenbeeld van 25 regels van 22 karakters genereerde. De 22 werd afgerond naar 20. De VIC-20 was opgezet als spelletjescomputer pur sang: kleur, geluid, weinig geheugen en een cartridge poort. Als scherm kon de kleuren TV in de huiskamer worden benut. Door z'n aantrekkelijke prijs (ca. fl. 1000.-) en een grote reclame campagne werd het een succes. De VIC-20 was bewust beperkt gehouden: er moest een instap op lage niveau mogelijk zijn en er moest ruimte zijn voor een opvolger.

Die opvolger kwam er. Na iets meer dan een jaar verscheen de klapper van de eeuw: de Commodore 64, zo genoemd vanwege het feit dat deze machine 64 kbyte geheugen had. De 64 kon goedkoop in massa worden geproduceerd omdat MOS Technology (die inmiddels Commodore Semiconductor Group heette) al het moeilijke in een drietal speciaal voor de 64 gemaakte chips had gebakken: de VIC-II (een 6567) maakte het schitterende hi-res kleuren beeld (compleet met 8 afzonderlijk te definieren en te bewegen objecten, de sprites) en zorgde en passant voor de (niet eenvoudige) besturing van de DRAMs. De tweede innovatie was de CIA (de 6526); een soort luxe VIA. De derde speciale chip was de 6581 SID: zo ongeveer een halve synthesizer die voor fascinerende geluidseffecten kan zorgen. De ontwikkelingskosten voor de 64 zijn zeer laag geweest: vrijwel alles stamde uit de VIC-20 terwijl deze machine op zijn beurt grotendeels stoelde op de tweede serie PETs. De verkopen daarentegen overtroffen de stoutste verwachtingen: Commodore begon weer aan dezelfde groeistuipen te lijden als in het begin. Het antwoord van de concurrentie kwam ook: MSX, met de Japanners die immers nooit van de computerkoek hadden meegesnoept, voorop. De 64 is door de enorme aantallen die ervan zijn verkocht uitgegroeid tot een standaard: je kunt hem nog steeds kopen, zij het in een nieuw jasje.

De opvolger van de 64, de 128 maakte zijn verwachtingen niet meer waar: de markt was verzadigd geraakt, mede door dumptoestanden met de MSX en MSX-2 machines en de trucs van meneer Clive Sinclair met de ZX-81 en de Spectrum.

Vandaag.

Vandaag kun je nog steeds de invloeden van het begin aanwijzen. Zo wordt er nog altijd in machinetaal geprogrammeerd. De 6502 is nog steeds redelijk populair voor kleine besturingen: de KIM en later de Apple zijn hier verantwoordelijk voor omdat deze machines de 6502 introduceerden bij diegenen die nu in het management zitten en er hun ervaring met microprocessors mee opdeden. De andere 8-bit CPU's (behalve misschien de Z-80) zijn nu een langzame dood aan het sterven, of zijn al overleden zoals de 2650, de 1802 en de SC/MP.

Als je terug kijkt kun je zien dat IBM precies de goede politiek heeft gevolgd om de markt voor zich op te eisen: wacht tot er standaards gaan ontstaan, combineer deze, en pas een modernere CPU toe. Het recept? Neem een goede BASIC, (die van Microsoft dus) combineer deze met een 1:1 kopie van de PET scherm editor, voorzie de machine van uitbreidingsslots in Apple-stijl, bouw het geheel op met standaardcomponenten die zich allang bewezen hebben, gebruik een operating system dat compatible is met het immens populaire CP/M en laat dit draaien op een redelijk moderne CPU van Intel, nota bene de uitvinders van de microprocessor, en wat krijg je? Juist, de IBM PC. En de Taiwanezen maar kopieren.....

```
0001 0000                .TITLE 'OCTOFATE ASSEMBLER FOR EC65/OCTOPUS V2.4'
0002 0000        ;===== LISTING START/STOP AND DELAY ===
0003 0000        ;===== FOR  EC65/OCTOPUS COMPUTER =====
0004 0000        ;
0005 0000        ; G.A. PETERUSMA ----- LEERDAM ----- HOLLAND
0006 0000        ;
0007 0000        ;===== MODIFIED VERSION OF ELEKTOR'S =====
0008 0000        ;=====       LISTING STOPPER        =====
0009 0000        ;
0010 0000        ;ELEKTOR COMPUTING ISSUE 5 PAGE 80 DUTCH EDITION
0011 0000        ;
0012 0000        ;
0013 0000                 *=$E10D
0014 E10D        VAIFR    *=$E101
0015 E101        VAPAD    *=$F000
0016 F000        VIDEO    *=$2325
0017 2325        KPDO     *=$F71D
0018 F71D        RECCHA   *=$2311
0019 2311        OUT      *=$24B2
0020 24B2        ;
0021 24B2 2000F0 START    JSR VIDEO
0022 24B5 AD0DE1          LDA VAIFR       ;KEY PRESSED ?
0023 24B8 2902            AND #$02
0024 24BA D001            BNE CHECK
0025 24BC        ;
0026 24BC 60     BY       RTS             ;IF NOT, RETURN
0027 24BD        ;
0028 24BD AD01E1 CHECK    LDA VAPAD       ;GET THE CHARACTER...
0029 24C0 2907            AND #$07        ;... AND MASK IT
0030 24C2 C915            CMP #$15        ;WAS IT FUNCTION KEY F2 ?
0031 24C4 F026            BEQ PRPDEL
0032 24C6 C911            CMP #$11        ;WAS IT FUNCTION KEY F1 ?
0033 24C8 F008            BEQ PRPST
0034 24CA C903            CMP #$03        ;WAS IT A BREAK ?
0035 24CC D0EE            BNE BY
0036 24CE 8D2523 ABORT    STA KPDO
0037 24D1 60              RTS
0038 24D2        ;
0039 24D2 A9D7   PRPST    LDA #$D7        ;CHANGE OUTPUT BYTE TO
0040 24D4 8D1123          STA OUT         ;STOP ROUTINE
0041 24D7 60              RTS
0042 24D8        ;
0043 24D8 2000F0 STOP     JSR VIDEO
0044 24DB C90A            CMP #$0A        ;DO WE HAVE A LINEFEED ?
0045 24DD D0DD            BNE BY          ;IF NOT, RETURN
0046 24DF 201DF7 GETKEY   JSR RECCHA
0047 24E2 C911            CMP #$11        ;START AGAIN ?
0048 24E4 D0F9            BNE GETKEY
0049 24E6 A9B1            LDA #$B1        ;RESTORE OUTPUT BYTE
0050 24E8 8D1123          STA OUT
0051 24EB 60              RTS
0052 24EC        ;
0053 24EC A9F1   PRPDEL   LDA #$F1        ; CHANGE OUTPUT BYTE TO
0054 24EE 8D1123          STA OUT         ;DELAY ROUTINE
0055 24F1 60              RTS
0056 24F2        ;
0057
0058 24F2 2000F0 DELAY    JSR VIDEO
```

```
0059 24F5 AD0DE1          LDA VAIFR       ;KEY PRESSED ?
0060 24F8 2902            AND #$02
0061 24FA D003            BNE CONT
0062 24FC 4C0E25          JMP DEL         ;IF NOT, DELAY AGAIN
0063 24FF AD01E1 CONT     LDA VAPAD
0064 2502 297F            AND #$7F
0065 2504 C915            CMP #$15        ;FUNCTION KEY F2 ?
0066 2506 D005            BNE RETURN
0067 2508 A9B1            LDA #$B1        ;RESTORE OUTPUT BYTE
0068 250A 8D1123          STA OUT
0069 250D 60      RETURN  RTS
0070 250E                 ;
0071 250E A240    DEL     LDX #$40        ;DELAY IME
0072 2510 A07F    LOAD    LDY #$7F
0073 2512 88      DECY    DEY
0074 2513 D0FD            BNE DECY
0075 2515 CA              DEX
0076 2516 D0F8            BNE LOAD
0077 2518 60              RTS
0078 2519                 ;
0079 2519                 ;-----------------------------
0080 2519                 ;To use this routine
0081 2519                 ;change address $2311 into $B1
0082 2519                 ;F1 = Function key 1 or CTL-S
0083 2519                 ;F2 = Function key 2 or CTL-U
0084 2519                 ;BREAK = Break key   or CTL-C
0085 2519                 ;-----------------------------
0086 2519                       .END

END OF ASSEMBLY        ERRORS DETECTED: 0000                    <0000>
```

```
10 REM *************     PROGRAMM LINGLS     *************
20 REM DIESES PROGRAMM LOEST LINEARE GLEICHUNGSSYSTEME
30 REM NACH DEM GAUSS-JORDAN ALGORITHMUS
40 REM VEROEFFENTLICHUNG INDER FUNKSCHAU 1980, HEFT20/82
50 REM AUTOR: DIETER SMODE 18-12-1978
60 REM
70 REM AUF DEN EC-65 ANGEPASST DURCH W.TIETSCH, MAERZ 1986
80 REM ************************************************
90 :
100 :
110 T1=PEEK(2888):T2=PEEK(8722):POKE2888,0:POKE8722,0
120 DISK!"GO F32F":REM CLEAR SCREEN AND HOME CURSOR
130 DISK!"IO ,01:REM SCREEN ONLY OUTPUT
140 REM ALLOW <CR> ONLY INPUT --> ""
150 FORI=1TO5:PRINT:NEXTI
160 PRINTTAB(20)"*******************************************"
170 PRINTTAB(20)"*                                         *"
180 PRINTTAB(20)"*        Dieses  Programm loest           *"
190 PRINTTAB(20)"*                                         *"
200 PRINTTAB(20)"*      LINEARE  GLEICHUNGSSYSTEME         *"
210 PRINTTAB(20)"*         ( bis zur Ordnung 15)           *"
220 PRINTTAB(20)"*             und berechnet               *"
230 PRINTTAB(20)"*           INVERSE MATRIZEN              *"
240 PRINTTAB(20)"*                                         *"
250 PRINTTAB(20)"*******************************************"
260 PRINT:PRINTTAB(20)"Lineare Gleichungssysteme  --> 1"
270 PRINTTAB(20)"Inverse Matrizenberechnung --> 2"
280 PRINTTAB(20)"Beende Programm durch      --> <CR>"
290 PRINT:PRINTTAB(55);:INPUTW$
300 IFW$=""THEN1240
310 IFW$="1"THEN350
320 IFW$="2"THEN860
330 FORI=1TO2:PRINTCHR$(27);"5";:NEXTI:GOTO290
340 :
```

```
350 REM BERECHNUNG DES LINEAREN GLEICHUNGSSYSTEMES
360 DISK!"GO F32F":REM CLEAR SCREEN
370 PRINTTAB(20)"***  LINEARE GLEICHUNGSSYSTEME  ***"
380 PRINTTAB(20)"     ------------------------"
390 PRINT:DISK!"IO ,01":REM SCREEN-OUTPUT ONLY
400 PRINTTAB(15)"Dieses Programm loest lineare Gleichungssysteme"
410 PRINTTAB(15)"nach der Methode des Gauss-Jordan-Algorithmus"
420 PRINTTAB(15)"Die maximale Anzahl der N der Spalten muss <16 sein!"
430 PRINT:PRINT:PRINTTAB(15)"Das lineare Gleichungssystem hat die ";
440 PRINT"folgende Form:":PRINT
450 PRINTTAB(15)"a11x1 + a12x2 + ... + a1nxn = a1"
460 PRINTTAB(15)"a21x1 + a22x2 + ... + a2nxn = a2"
470 PRINTTAB(15)"  .                          ."
480 PRINTTAB(15)"  .                          ."
490 PRINTTAB(15)"an1x1 + an2x2 + ... + annxn = an"
500 FORI=1TO7:PRINT:NEXTI:PRINTTAB(15)"Weiter --> <CR>";:INPUTW$
510 DISK!"GO F32F":REM CLEAR SCREEN
520 PRINTTAB(15);:INPUT"Input/Output auf Drucker J/N ";W$
530 IFW$="J"THENDISK!"IO ,09"
540 PRINTCHR$(27);"3";
550 CLEAR:D=1:PRINTTAB(16);:INPUT"Ordnung des Gl.-Systems   N = ";N
560 IFN=0ORN>15THENPRINTCHR$(27);"5";:GOTO550
570 PRINT:PRINT:N=N-1:DIMA(N,N+1)
580 PRINTTAB(15)"Gebe die Koeffizienten der Matrix ein"
590 PRINTTAB(15)"-----------------------------------":PRINT
600 FORI=0TON:FORJ=0TON+1
610 IFJ=N+1THENPRINTTAB(35)"A (";I+1;") = ";:INPUTA(I,J):GOTO630
620 PRINTTAB(15)"A (";I+1;",";J+1;") = ";:INPUTA(I,J)
630 NEXTJ:PRINT:PRINT:NEXTI
640 PRINTTAB(15)"-----------------------------------":PRINT
650 PRINTTAB(15)" Die Loesung des Gleichungssystemes :":PRINT
660 PRINT:PRINT:FORK=0TON:IFK=NTHEN710
670 AB=ABS(A(K,K)):FORI=K+1TON:IFABS(A(I,K))<=ABTHEN700
680 AB=ABS(A(I,K)):D=-D
690 FORJ=KTON+1:Z=A(K,J):A(K,J)=A(I,J):A(I,J)=Z:NEXTJ
700 NEXTI
710 IFABS(A(K,K))<1E-7THEN820
720 FORI=0TON:IFI=KTHEN740
730 F=A(I,K)/A(K,K):FORJ=K+1TON+1:A(I,J)=A(I,J)-A(K,J)*F:NEXTJ
740 NEXTI,K:IFC<>0THEN1150
750 FORI=0TON:D=D*A(I,I):NEXTI
760 PRINTTAB(15)"Die Determinante Det (A) = ";D:PRINT
770 PRINTTAB(15)"Der Loesungsvektor ist :":PRINT
780 FORI=0TON:PRINTTAB(27)"X";I+1;"=";A(I,N+1)/A(I,I):NEXTI
790 PRINT
800 PRINTTAB(15)"-----------------------------------":PRINT
810 PRINTTAB(15)"Weiter --> <CR>";:INPUTW$:GOTO120
820 PRINTTAB(15)"Das Gleichungssystem ist nicht eindeutig loesbar"
830 PRINTTAB(15)"A (";K+1;",";K+1;")";" verschwindet annaehernd!"
840 GOTO800
850 :
860 REM BERECHNUNG DER INVERSEN MATRIX
870 DISK!"GO F32F":REM CLEAR SCREEN
880 PRINTTAB(18)"***  MATRIZEN - INVERSIONEN  ***"
890 PRINTTAB(18)"     ----------------------":PRINT
900 PRINT:DISK!"IO ,01":REM SCREEN-OUTPUT ONLY
910 PRINTTAB(15)"Dieses Programm berechnet die inverse Matrix"
920 PRINT:PRINTTAB(15)"Die Ordnung der Matrix muss <16 sein !!"
930 PRINTTAB(15)"Die Matrix hat die folgende Form:":PRINT:PRINT
940 PRINTTAB(20)"I  d11   d12   ...   d1n  I"
950 PRINTTAB(20)"I  d21   d22   ...   d2n  I"
960 PRINTTAB(20)"I   .                  .   I"
970 PRINTTAB(20)"I   .                  .   I"
980 PRINTTAB(20)"I   .                  .   I"
990 PRINTTAB(20)"I  dn1   dn2   ...   dnn  I"
1000 FORI=1TO7:PRINT:NEXTI:PRINTTAB(15)"Weiter --> <CR>";:INPUTW$
1010 DISK!"GO F32F":REM CLEAR SCREEN
1020 PRINTTAB(15);:INPUT"Input/Output auf Drucker J/N ";W$
1030 IFW$="J"THENDISK!"IO ,09"
1040 PRINTCHR$(27);"3";
1050 CLEAR:PRINTTAB(16);:INPUT"Ordnung des Gl.-Systems   N = ";N
1060 IFN=0ORN>15THENPRINTCHR$(27);"5";" ";:GOTO1050
1070 N=N-1:DIMA(N,N+1):DIMD(N,N+1)
1080 PRINTTAB(15)"Gebe die Koeffizienten der Matrix ein"
1090 PRINTTAB(15)"-----------------------------------":PRINT
1100 FORI=0TON:FORJ=0TON
1110 PRINTTAB(15)"D (";I+1;",";J+1;") = ";:INPUTD(I,J)
```

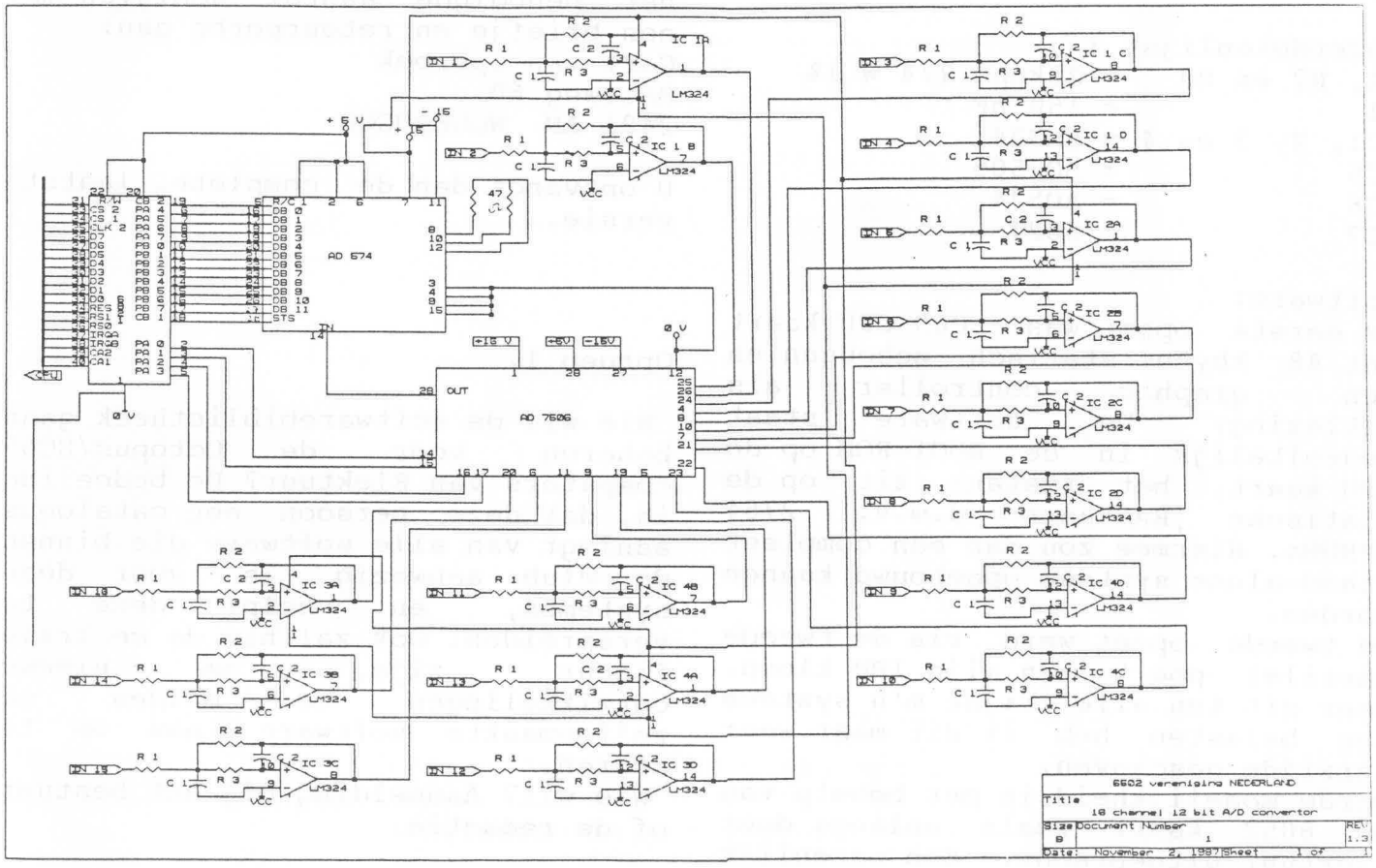
```
1120 NEXTJ:PRINT:NEXTI
1130 FORI=0TON:FORJ=0TON:A(I,J)=D(I,J):NEXTJ:NEXTI
1140 J=N+1:FORI=0TON:A(I,J)=0:NEXTI:A(C,J)=1:C=C+1:GOT0660
1150 PRINTTAB(15)C;". Spalte":PRINT
1160 FORI=0TON:PRINTTAB(15)"A (";I+1;",";C;") = "A(I,N+1)/A(I,I)
1170 NEXTI:PRINT
1180 IFC<=NTHENPRINTTAB(15)"Naechste Spalte --> <CR> ";:GOT01200
1190 GOT01220
1200 INPUTW$:IFW$=""THEN1130
1210 GOT0120
1220 PRINT:PRINTTAB(15)"----------------------------------------":PRINT
1230 PRINTTAB(15)"Weiter --> <CR>";:INPUTW$:GOT0120
1240 DISK!"GO F32F":POKE2888,T1:POKE8722,T2:DISK!"IO, 01":NEW
```

<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>

W.Cuypers
Wingerdstraat 22
5682-CL Best
tel nr: 04998-9627

16 KANAALS 12 BIT A/D CONVERTOR VOOR CPU SYSTEEM.

Opzet was om met de restanten van mijn oude Octopus een nuttig geheel samen te bouwen wat op eender welk systeem aansluitbaar was. Als I/O is daarom gekozen voor de bij iedereen bekende 6821 tesamen met een A/D convertor die op dit moment door enige zaken in Nederland voor aantrekkelijke prijzen verkocht wordt.
De AD 574 heeft de mogelijkheid om direct aan een 8, 12 of 16 bit systeem gehangen te worden zonder externe buffers. In het schema is de AD 574 getekend in de 12 bit mode. Door pen 2 van het IC aan massa te hangen converteert de chip de ingevoerde spanning in een 8 bit signaal dat parallel wordt uitgevoerd. In de 8 bit mode kunnen de 8 hoogste bits of de 4 laagste bits worden gelezen afhankelijk van de status van A0 (pen 4). Voor zo'n



applicatie moeten echter de 4 LSB's (pin 16-19) doorverbonden worden met de 4 MSB's (pin 24-27). Als tijdens lezen A0 laag is, worden de 8 lagere bits ge-enabled en staat de te lezen data op pin 20 tot 27. Is A0 echter hoog, dan worden de 8 hogere bits gedisabled en staan de 4 LSB's op pin 24-27. De 4 middelste bits worden genegeerd zodat er geen data op de pennen 20 tot 23 staat. Tot zover de 8/12 bit mogelijkheid. Voor de dieper ge nteresseerde verwijs ik naar het ANALOG DEVICES data sheet.
Omdat 1 analoge ingang te weinig was om echt mee te kunnen stoeien is er een stukje hardware tussengeplakt in de vorm van de AD 7506. Deze chip kent 16 analoge ingangen, die alle ingangsspanningen tot 24 Volt kunnen verwerken. De chip schakelt d.m.v. een 4 bit code een van de 16 ingangen door naar zijn uitgang. Als ingangscircuit is de LM 324 gekozen i.v.m. zijn compacte opbouw en lineariteit.
Van het geheel is in de tussentijd een printtekening gemaakt die ik eventueel kan copi ren. (printformaat 100*160 mm, Eurokaart)

Onderdelenlijst :

R1, R2 en R3	= 22kohm 1/4 W 1%
C1	= 150 nF
IC1, 2, 3 en 4	= LM324
IC5	= AD7506
IC6	= AD574
IC7	= 6821

Software:
De eerste opzet was EC65 CPU kaart met 48 kbyte statisch geheugen en een graphic controller als uitlezing. De software staat gedeeltelijk in de BOOT-ROM op de CPU kaart, het restant zit op de statische RAMkaart d.m.v. 2764 EPROMs. Hiermee zou dan een compleet stand-alone systeem opgebouwd kunnen worden.
De tweede opzet werd via de tweede parallel poort aan mijn IBM kloon. Daar dit ten alle tijde m'n systeem zou belasten heb ik dit maar weer terzijde geschoven.
Derde mogelijkheid is met behulp van de 8052 kaart zoals onlangs door Elektuur uitgebracht. Een wezenlijk voordeel is, dat het een redelijk zelfstandig systeem wordt wat ten alle tijden d.m.v. de seri le poort te be nvloeden is.
Zoals te merken is, heb ik mijn keus nog niet gemaakt. Een ieder die hierin mee wil denken verzoek ik daarom contact met mij op te nemen.

Octopus WP: een oproep.

Door: R.T. Overakker
Laan van Poot 458
2566 DP DEN HAAG
Telefoon 070-250281

Voor diegenen die zich de WP voor de Octopus van Leif Rasmussen aanschaften.
Het is mogelijk, dat de versies 3.0 en 3.1 door elkaar zijn geraakt, en dat in sommige systeemschijven een bug is binnengeslopen. Het adres $01DF moet niet $6C, maar $7C zijn.
Daarnaast zijn nu ook de complete sources en de manual op schijf beschikbaar.
Indien u een correct werkende versie wenst te ontvangen, stuur dan het benodigde aantal schijven met een briefje en retourporto aan:
Gert van Opbroek
Bateweg 60
2481 AN WOUBRUGGE

U ontvangt dan de complete, laatste versie.

Oproep 1.

Wie wil de softwarebibliotheek gaan beheren voor de Octopus/EC65 computers van Elektuur? De bedoeling is, dat deze persoon een catalogus aanlegt van alle software die binnen de club aanwezig is voor deze systemen, en helpt deze te verspreiden. Ook zal hij de centrale figuur zijn, om nieuwe ontwikkelingen te melden en zelfgemaakte software aan op te sturen.
Wie wil? Aanmelding bij het bestuur of de redactie.

Zeer intensieve geheugentest.

N. de Vries.

De listing die na dit verhaaltje is afgedrukt is een buitengewoon intensieve geheugentest. Het programma werd oorspronkelijk geschreven door Carl Moser (Ja, die van de assembler..).

Bij geheugenproblemen kunnen de volgende dingen de oorzaak zijn:
- Een chipselect werkt niet.
- Een adreslijn is kortgesloten naar de nul of +5 Volt.
- Een datalijn is kortgesloten naar de nul of +5 Volt.

De laatste oorzaak wordt door vrijwel iedere primitieve geheugentest gevonden. De eerste twee oorzaken echter meestal niet. Het programma bestaat uit twee tests, die na elkaar worden uitgevoerd. Het is de bedoeling, dat het programma vanuit een monitoromgeving gerund wordt. Na be indiging van de eerste test volgt een BRK instructie. In de meeste monitoren kan voor de tweede test gewoon een G (CR) of E (CR) gegeven worden om de tweede test te starten. Aan de inhoud van de accu is te zien, welke test zojuist voltooid werd. De accu bevat E1 aan het eind van test 1 en E2 aan het eind van test 2. Indien aan het eind van test 2 weer een E/G (CR) wordt gegeven, dan wordt test 1 weer uitgevoerd, enzovoort.

Test 1 is bedoeld om het geheugen te testen op dataretentie en op onderbroken/kortgesloten data- en adreslijnen. Dit wordt gedaan door 00 01 02 03 04 ..... FE FF in de opvolgende geheugenlocaties te schrijven en deze data vervolgens weer terug te lezen. Nadat het gehele geheugenblok gevuld is, volgt nogmaals een teruglezing. Met deze laatste teruglezing wordt de echte geheugenfunctie getest: bij wat grotere blokken duurt het langer dan een refreshperiode voor de tweede teruglezing wordt uitgevoerd. Bij de volgende loop wordt er 01 02 03 04 05 ..... FE FF 00 in de locaties geschreven. Dit gaat zo door totdat iedere locatie alle mogelijke bytes heeft geschreven en gelezen heeft gekregen.

Test 2 is bedoeld om de Chip Select lijnen te testen. De test verloopt analoog aan test 1, met dit verschil dat de laatste data niet FF, maar F2 is. Omdat de cyclus nu korter is dan 256 bytes komt hetzelfde testpatroon niet steeds een pagina verder terecht, maar eerder. Hierdoor kunnen overlappingen in Chip-Selects worden gevonden.

Gedurende de test worden er puntjes op het scherm geprint om de gebruiker te laten weten, dat het programma nog loopt. Ieder puntje is een complete doorgang van 1 patroon. Er worden dus altijd 256 puntjes afgedrukt. De grootte van het geteste blok bepaalt hoe snel dit gebeurt. Dit afdrukken is wel noodzakelijk, omdat op 1 MHz een test van 1 kbyte al 38 seconden duurt. Als er een fout wordt gevonden, dan wordt het adres waar de fout optrad afgedrukt, gevolgd door de data die er had moeten staan en de data die werd teruggelezen. Alle informatie verschijnt in hex.

Fouten die in de eerste test worden gevonden zijn vrijwel altijd te wijten aan een defecte geheugenchip. Aan het adres is te zien in welk geheugen de fout optrad, aan de data eventueel ook als er andere geheugens dan byte-brede typen zijn toegepast. Verloopt test 1 goed, maar test 2 niet, dan is er iets mis met de Chip-Select functie. In dat geval is er waarschijnlijk iets aan de hand met de decoders of met de adreslijnen A8 tot en met A15.

Als er fouten optreden in adressen die steeds een bepaalde macht van 2 uit elkaar liggen (2, 4, 8, 16, enzovoort), dan is er iets aan de hand met de adreslijnen. De volgende tabel kan dan helpen:

Data bit met fout:	Controleer adreslijn:
D0	A0 of A8
D1	A1 of A9
D2	A2 of A10
D3	A3 of A11
D4	A4 of A12
D5	A5 of A13
D6	A6 of A14
D7	A7 of A15

Voorbeeld:
Indien u een array met 4116's (16k x 1) over een bereik van 32 kbyte controleert en er treden fouten op over het hele bereik bij test 1, dan is er iets mis met: de refresh, een of meer datalijnen, een of meer adreslijnen of de voeding. Treden er echter alleen fouten op in test 2, dan is de generatie van de CAS (=CS) de oorzaak.
Bij fouten in test 1 zal evaluatie van de verwachte en de teruggelezen data, in samenhang met de foute adressen tot de precieze oorzaak moeten leiden.

OPMERKING: de listing is gemaakt op een DOS65 machine met de assembler AS. Het programma is echter universeel; het kent maar 1 extern label te weten de routine die het ASCII karakter in A afdrukt. Deze routine is voor iedere machine wel te vinden. Door de ORG statements aan te passen, kan het programma zo ongeveer overal neergezet worden; het is minder dan een pagina lang.

```
                ;
                ;****************************************
                ;*                                      *
                ;* MEMTEST                              *
                ;*                                      *
                ;* A universal memory test program,     *
                ;* originally written by Carl Moser.    *
                ;* Comments by N. de Vries.             *
                ;*                                      *
                ;****************************************
                ;
        0000            ORG     0
0000            ADDRS   RES     2       ;current memory address
                ;
        0200            ORG     $0200   ;start of program
                ;
0200 A2 00      MEMTEST LDX     #0
0202 8E E102            STX     TSTTYPE ;set test type 1
0205 20 1B02            JSR     TESTPGM ;perform test
0208 A9 E1              LDA     #$E1    ;get flag
020A 00                 BRK             ;and break into monitor
020B EA                 NOP             ;to eliminate
020C EA                 NOP             ;BRK overhead
020D EE E102            INC     TSTTYPE ;set test type 2
0210 20 1B02            JSR     TESTPGM ;perform test
0213 A9 E2              LDA     #$E2    ;flag test 2 ended
0215 00                 BRK             ;and break into monitor
0216 EA                 NOP             ;skip
0217 EA                 NOP             ;BRK overhead
0218 4C 0002            JMP     MEMTEST ;and test again
                ;
021B 20 A402    TESTPGM JSR     CRLF    ;start on new line
021E A0 00              LDY     #0      ;pattern register
0220 A2 00              LDX     #0      ;get offset zero
0222 8E E002            STX     TSTPTRN ;intialize test pattern
0225 4C 3302            JMP     NX_PASS ;and start testing
```

```
0228 A9 2E      NX_PTRN LDA     #'.'     ;get a period
022A 20 CB02            JSR     WRITE    ;print it
022D EE E002            INC     TSTPTRN  ;adapt test pattern
0230 D0 01              BNE     NX_PASS  ;if all patterns done
0232 60                 RTS              ;exit
                        ;
0233 AC E002    NX_PASS LDY     TSTPTRN  ;get current pattern
0236 20 AD02            JSR     INIADDR  ;move start to current
0239 98         LOOP1   TYA              ;move pattern to A
023A 81 00              STA     [ADDRS,X] ;move to memory
023C C1 00              CMP     [ADDRS,X] ;if read back is wrong
023E F0 03              BEQ     NO_ERR1  ;
0240 20 8702            JSR     ERROR    ;print address and values
0243 20 7A02    NO_ERR1 JSR     CHKEND   ;else check if at end
0246 F0 06              BEQ     CK_PTRN  ;if so, test data retention
0248 20 6702            JSR     INC_RY   ;adapt pattern+address
024B 4C 3902            JMP     LOOP1    ;and repeat
                        ;
024E AC E002    CK_PTRN LDY     TSTPTRN  ;get current pattern
0251 20 AD02            JSR     INIADDR  ;move start to current
0254 98         LOOP2   TYA              ;get test pattern in A
0255 C1 00              CMP     ÄADDRS,XÜ ;check if still there
0257 F0 03              BEQ     NO_ERR2  ;if not,
0259 20 8702            JSR     ERROR    ;print address and values
025C 20 7A02    NO_ERR2 JSR     CHKEND   ;check if at end address
025F F0 C7              BEQ     NX_PTRN  ;if so, do next pattern
0261 20 6702            JSR     INC_RY   ;adapt pattern+address
0264 4C 5402            JMP     LOOP2    ;and loop
                        ;
                        ;*** Adapt Y and increment address
                        ;
0267 C8         INC_RY  INY              ;adapt current pattern
0268 AD E102            LDA     TSTTYPE  ;get test number
026B F0 06              BEQ     INCADDR  ;if test 1, exit
026D C0 F3              CPY     #$F3     ;if at chip select test
026F 90 02              BCC     INCADDR  ;value
0271 A0 00              LDY     #0       ;reset Y
0273 E6 00      INCADDR INC     ADDRS    ;adapt address low
0275 D0 02              BNE     SKIP_HI  ;if on page boundary
0277 E6 01              INC     ADDRS+1  ;adapt address high
0279 60         SKIP_HI RTS              ;and exit
                        ;
                        ;*** Check if at end address
                        ;
027A AD DF02    CHKEND  LDA     ENDAD+1  ;get end address high
027D C5 01              CMP     ADDRS+1  ;if high bytes
027F D0 05              BNE     EXIT1    ;match,
0281 AD DE02            LDA     ENDAD    ;get end address low
0284 C5 00              CMP     ADDRS    ;if at end address, set Z
0286 60         EXIT1   RTS              ;and exit
                        ;
                        ;*** Print failing address and data
                        ;
0287 48         ERROR   PHA              ;save A
0288 20 A402            JSR     CRLF     ;start on a new line
028B A5 01              LDA     ADDRS+1  ;get address high
028D 20 B802            JSR     TBYT     ;print it in hex
0290 A5 00              LDA     ADDRS    ;get address low
```

```
0292 20 B802            JSR     TBYT    ;and complete address
0295 20 D502            JSR     SPACE2  ;print two spaces
0298 68                 PLA             ;restore test data
0299 20 B802            JSR     TBYT    ;print it
029C 20 D502            JSR     SPACE2  ;print two spaces
029F A1 00              LDA     ÄADDRS,XÜ ;get memory contents
02A1 20 B802            JSR     TBYT    ;print in hex, and:
                        ;
                        ;*** Print CRLF
                        ;
02A4 A9 0D      CRLF    LDA     #$0D    ;get a return
02A6 20 CB02            JSR     WRITE   ;print it
02A9 A9 0A              LDA     #$0A    ;get a line feed
02AB D0 1E              BNE     WRITE   ;print it and exit
                        ;
                        ;*** Get start address in current address
                        ;
02AD AD DC02    INIADDR LDA     START   ;get start low
02B0 85 00              STA     ADDRS   ;in current address low
02B2 AD DD02            LDA     START+1 ;get start high
02B5 85 01              STA     ADDRS+1 ;and complete address
02B7 60                 RTS             ;then exit
                        ;
                        ;*** Print A in hex
                        ;
02B8 48         TBYT    PHA             ;save A
02B9 4A                 LSRA            ;get
02BA 4A                 LSRA            ;high
02BB 4A                 LSRA            ;nibble into
02BC 4A                 LSRA            ;low nibble
02BD 20 C102            JSR     NIBBLE  ;print nibble in hex
02C0 68                 PLA             ;restore A
02C1 29 0F      NIBBLE  AND     #%00001111 ;get low nibble only
02C3 09 30              ORA     #$30    ;convert to ASCII
02C5 C9 3A              CMP     #$3A    ;if letter
02C7 90 02              BCC     WRITE   ;add
02C9 69 06              ADC     #6      ;letter offset
                        ;
                        ;*** Print the character in A
                        ;
02CB 8C E202    WRITE   STY     SAVEY   ;save Y
02CE 20 23C0    ROMLINK JSR     OUTPUT  ;print A
02D1 AC E202            LDY     SAVEY   ;restore Y
02D4 60                 RTS             ;and exit
                        ;
                        ;*** Print two spaces
                        ;
02D5 20 D802    SPACE2  JSR     SPACE   ;print a space, and:
02D8 A9 20      SPACE   LDA     #$20    ;get a space
02DA D0 EF              BNE     WRITE   ;and go print it
                        ;
                        ;*** Start and end addresses
                        ;
02DC 0003       START   FDB     $0300   ;default start address
02DE FF8F       ENDAD   FDB     $8FFF   ;default end address
                        ;
02E0            TSTPTRN RES     1       ;current test pattern
02E1            TSTTYPE RES     1       ;current test type
```

```
02E2            SAVEY   RES     1       ;save place for Y
                        ;
                        ;*** External label
                        ;
        C023    OUTPUT  EQU     $C023   ;print the character in A
                        ;
                        END
                                global labels

ADDRS   0000    CHKEND  027A    CK_PTRN 024E    CRLF    02A4
ENDAD   02DE    ERROR   0287    EXIT1   0286    INCADDR 0273
INC_RY  0267    INIADDR 02AD    LOOP1   0239    LOOP2   0254
MEMTEST 0200    NIBBLE  02C1    NO_ERR1 0243    NO_ERR2 025C
NX_PASS 0233    NX_PTRN 0228    OUTPUT  C023    ROMLINK 02CE
SAVEY   02E2    SKIP_HI 0279    SPACE   02D8    SPACE2  02D5
START   02DC    TBYT    02B8    TESTPGM 021B    TSTPTRN 02E0
TSTTYPE 02E1    WRITE   02CB
```

Datasheetservice.

Een van de voorstellen op de laatste ledenvergadering was, om binnen de club een datasheetservice te beginnen. In principe kunt u voor een kopie van een datasheet contact opnemen met Nico de Vries (zie voorin dit nummer).

Hij zal trachten het gevraagde datasheet op te sporen, en het tegen een redelijk bedrag te kopieren. De voorgestelde gang van zaken is voorlopig: bel Nico op en vertel wat u hebben wilt. Nico zoekt uit, hoeveel pagina's het datasheet lang is, en belt/schrijft u terug met de verwachte kosten. Na overmaking van het bedrag ontvangt u vervolgens het gekopieerde datasheet. Het origineel gaat in een bibliotheek en in een catalogus waaruit de leden kunnen putten. Waarschijnlijk zullen de kosten fl. 0.10 per pagina zijn, vermeerderd met de portokosten.

Eveneens kunt u op deze manier in het bezit komen van de catalogus van voorradige datasheets.

Attentie: Contributie voor 1988.

U vergemakkelijkt de taak van de penningmeester enorm, door tijdig de contributie voor het verenigingsjaar 1988 over te maken. Gebruik hiervoor bij voorkeur de u toegezonden acceptgirokaart. Mocht u geen kaart ontvangen hebben, of hem kwijt zijn, maak de contributie (hfl. 50.-) dan over op:

Postbank rekening no. 3757649, t.n.v.

KIM-Gebruikers Club Nederland
Krimpen a/d IJssel.

Attention: Membership-fee for 1988.

Please pay your membership-fee for 1988 as soon as possible: this will ensure the timely delivery of the first issue of this magazine in the next year. If you are living outside Holland or Belgium, the fee is hfl. 59.50. The easiest way of paying is sending an Eurocheque for the amount specified above to:

J. van Sprang
Tulp 71
NL-2925 EW KRIMPEN A/D IJSSEL
The Netherlands

```
/*************************************************************************
        D A T U M   - - >   W E E K D A G   C O N V E R S I E
 *************************************************************************/
/*
        De hier beschreven routines kunnen gebruikt worden om uitgaande
        van de datum de bijbehorende dag in de week, het weeknummer
        van de datum en het jaar waarin de bepaalde week valt te bepalen.

        Als bijprodukt bevat dit pakket een functie waarmee het aantal
        dagen tussen twee data bepaald kan worden.

        Auteur: G. van Opbroek
                Bateweg 60
                2481 AN Woubrugge

        Versie: 1.0 - 8 oktober 1987

        N.B. Wie helpt mij aan een algoritme voor het berekenen van
             de paasdagen?
*/

#include <stdio.h>
#include <strings.h>

/*      DAG_FACTOR:

        Parameters:     dag,maand,jaar
        Returns:        Dagnummer
*/

int DAG_FACTOR(dag,maand,jaar)

int dag,maand,jaar ;

/*
        In dagfactor wordt aan de ingevoerde datum een dagnummer
        toegekend. Door dit dagnummer modulo 7 te nemen, krijgen
        we een getal dat de dag in de week aangeeft. Hierbij heeft
        de zondag code 0.

        Het verschil tussen de dagnummers van twee data geeft het
        verschil in dagen tussen deze twee data.

        Om eventueel in een integer-2 formaat te kunnen werken, is
        het nulpunt van de dagnummers in 1985 gelegd. Dit heeft
        echter geen gevolgen voor de berekening van data voor
        deze datum.
*/

{
    int fact1,dag_factor    ;

    fact1 = 365 * (jaar - 1985) - 4 + dag + 31 * (maand - 1) ;

    if (maand <= 2)
        dag_factor = fact1 + (jaar - 1) / 4 -           /* Schrikkeljaren */
                     3 * ((jaar - 1) / 100 + 1) / 4; /* Eeuwjaren      */
    else
        dag_factor = fact1 - (4 * maand + 23) / 10 +
                     jaar / 4 -                         /* Schrikkeljaren */
                     3 * (jaar / 100 + 1) / 4;          /* Eeuwjaren      */

    return(dag_factor);
}
```

```
/*      DATUM_WEEKNR:

        Parameters:     dag,maand,jaar
                        Weekdag (pointer)
        Returns:        Weeknummer
*/

int DATUM_WEEKNR(dag,maand,jaar,weekdag)

int dag,maand,jaar ;
int *weekdag       ;

/*
        Deze functie bepaalt bij een datum het weeknummer in het
        huidige, voorgaande of volgende jaar. In de parameter "weekdag"
        wordt de code voor de dag in de week teruggegeven. Hierbij
        telt de maandag als eerste dag in de week. Voor de bepaling
        van het weeknummer is het aantal donderdagen sinds 1-1 bepalend.
*/

{
    int  klad,klad1,klad2,correctie,klad_weekdag ;

    klad = DAG_FACTOR(1,1,jaar)    ;     /* Bepaal de dag waarop 01/01 viel */
    correctie = 4 - (klad % 7 + 7) % 7; /* Dagfactor kan < 0 zijn */

    /* Corrigeer voor 01/01 op of na donderdag */

    if (correctie < 0)
        correctie = 0 ;
    else
        correctie = 1 ;

    /* Bepaal het aantal weken sinds 01/01 */

    klad2 = DAG_FACTOR(dag,maand,jaar)     ;
    klad  = klad2 / 7 - klad / 7 ;

    /* Bepaal de huidige weekdag en corrigeer eventueel voor de zondag */

    klad2 = (klad2 % 7 + 7) % 7;    /* Dagfactor kan < 0 zijn */
    if (klad2 == 0)                 /* Zondag -> verlaag het weeknummer    */
    {
        klad-- ;
        klad2 = 7 ;
    }

    *weekdag = klad2 ;

    /* Beschouw een uitkomst week 53 apart: */

    if (klad + correctie == 53)
    {
        klad1 = (DAG_FACTOR(1,1,jaar + 1) % 7 + 7) % 7 ;
        if (klad1 < 5)                  /* zo, ma, di, wo */
            return(1);
        else                            /* do, vr, za     */
            return(53);
    }
    else
        /* Voor week 0 geldt: dit is dezelfde week als afgelopen 31/12 */

        if (klad + correctie == 0)
            return(DATUM_WEEKNR(31,12,jaar - 1 ,&klad_weekdag)) ;
        else
            return(klad + correctie) ;
}
```

```
/*      CONVERTEER_DATUM:

        Parameters:     Datum in dd/mm/jj of dd/mm/jjjj - formaat
                        Datum in dwwjjjj - formaat
        Returns:        1 -> datum bestaat; 0 -> datum bestaat niet
*/
int CONVERTEER_DATUM(datum,c_datum)

char datum[]   :    /* dd/mm/jj of dd/mm/jjjj -formaat */
char c_datum[] :    /* dwwjjjj -formaat                */

{
    int dag,maand,jaar ;
    int week,weekdag   ;
    int i              ;

    /* Formaat-controle */

    if (strlen(datum) != 8 && strlen(datum) != 10) return(0) ;

    for (i = 0 ; i < strlen(datum) ; i++)
    {
        if (i != 2 && i != 5)
        {
            if (datum[i] < '0' || datum[i] > '9') return(0) ;
        }
        else
            if (datum[i] != '/') return(0) ;
    }

    /* Conversie naar dag, maand, jaar */

    dag     = 10 * (datum[0] - '0') + datum[1] - '0' ;
    maand   = 10 * (datum[3] - '0') + datum[4] - '0' ;
    jaar    = 10 * (datum[6] - '0') + datum[7] - '0' ;

    if (datum[8] != '\0')
        jaar = 100 * jaar +
               10 * (datum[8] - '0') + datum[9] - '0' ;
    else
        jaar += 1900 ;

    /* Controle of de datum bestaat */

    if (dag     <  1 || dag     > 31                                    ||
        maand   <  1 || maand   > 12                                    ||
        dag     > 30 && (maand == 2 || maand == 4 || maand == 6 ||
                         maand == 9 || maand == 11)                     ||
        dag     > 28 && maand  == 2 &&
                        DAG_FACTOR(dag,maand,jaar)  >=
                                DAG_FACTOR(1,3,jaar))
        return(0);

    /* Conversie */

    week = DATUM_WEEKNR(dag,maand,jaar,&weekdag) ;

    if (week > 50 && maand < 2)         /* Week 52 of 53 van vorig jaar */
        jaar-- ;
    else if (week < 2 && maand > 11)    /* Week 1 van volgend jaar */
        jaar++ ;

    c_datum[0] = weekdag +   '0' ;
    c_datum[1] = '-'             ;
    c_datum[2] = week / 10 + '0' ;
    c_datum[3] = week % 10 +'0'  ;
    c_datum[4] = '-'             ;
```

```
    for (i = 8 ; i >= 5 ; i--)
    {
        c_datum[i] = jaar % 10 + '0' ;
        jaar = jaar / 10 ;
    }
    c_datum[9] = '\0' ;
    return(1);
}

main()
{
    char datum[10]  , dwwjjjj[9] ;

    printf("Geef de datum (dd/mm/jj of dd/mm/jjjj) : ");

    while (scanf("%s",datum) != EOF)
    {
        if (CONVERTEER_DATUM(datum,dwwjjjj))
        {
            printf("De geconverteerde datum is : %s\n",dwwjjjj);
            printf("De datum valt op een ") ;

            switch(dwwjjjj[0])
            {
                case    '1':
                        printf("maandag \n") ;
                        break ;
                case    '2':
                        printf("dinsdag \n") ;
                        break ;
                case    '3':
                        printf("woensdag \n") ;
                        break ;
                case    '4':
                        printf("donderdag \n") ;
                        break ;
                case    '5':
                        printf("vrijdag \n") ;
                        break ;
                case    '6':
                        printf("zaterdag \n") ;
                        break ;
                case    '7':
                        printf("zondag \n") ;
                        break ;
            }
        }
        else    printf("Onjuiste datum \n");
        printf("Geef de datum (dd/mm/jj of dd/mm/jjjj) : ");
    }
}
```

Oproep 2.

Net als voor de Octopus/EC65 is er behoefte aan iemand die de in de club aanwezige Junior-software op cassette gaat catalogiseren en beheren. Als u hiervoor belangstelling heeft, neem dan even contact op met het bestuur. Alvast bedankt.

Stand van zaken bulletinboard.

Tijdens de ledenvergadering heeft zich een kandidaat gemeld voor de functie van sysop voor het bulletinboard. Zijn naam is Jacques Banser. Verder heeft het bestuur inmiddels een IBM/PC-kloon met harde schijf aangeschaft. We hopen in de loop van januari 1988 te kunnen starten. Meer in het volgende nummer.

Spreekbeurt EPROMs programmeren.

Hieronder is de tekst afgedrukt die is uitgereikt bij de spreekbeurt over het programmeren van EPROMs. De spreekbeurt werd gehouden door Nico de Vries op de bijeenkomst van 19 september jongstleden te Haarlem. De aanleiding tot de spreekbeurt was het gereedkomen van de EPROMprogrammer EP en de software hiervoor.

1. Wat is een FET en hoe werkt zo'n ding?

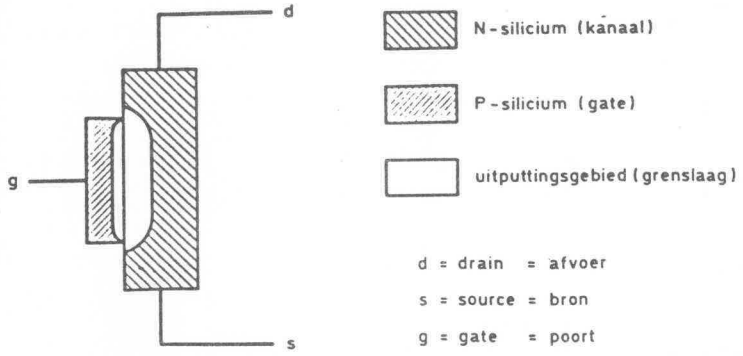
In afbeelding 1 is het schemasymbool van een zogenaamde junction-FET (kort: JFET) getekend. FET betekent: Field Effect Transistor, of in gewoon Nederlands: Veldeffecttransistor.

De FET is als volgt opgebouwd. Het belangrijkste deel van de FET, het kanaal, wordt gevormd door N-silicium. Dit is silicium dat verontreinigd is met 5-waardige materialen, zodat het een overschot aan vrije elektronen heeft. Dit N-silicium is elektrisch geleidend. Aan de uiteinden van het kanaal vinden we de drain en source aansluitingen. De source is de bron van electronen, de drain voert ze af. De elektronenstroom loopt dus van source naar drain. De werkelijke stroom loopt dus van drain naar source.

Het kanaal is verder aan de zijkant voorzien van een elektrode, die van P-silicium gemaakt is. Deze elektrode wordt de gate genoemd. De gate vormt met het kanaal een PN-overgang ofwel een junctie, vandaar de naam junction-FET. Zoals misschien bekend, is een PN-overgang niets anders dan een diode.

In normaal bedrijf wordt de FET zodanig gebruikt, dat de gate-kanaaldiode in sper staat. Er loopt dan ook geen stroom in de gate, of anders gezegd: de FET is spanningsgestuurd.

Sluiten we de source en de gate kort, zodat de gate-source spanning (U_{GS}) nul volt is, dan heerst er rond de gate geen electrisch veld. Het N-silicium is een geleidend materiaal, zodat er een stroom tussen drain en source kan lopen.



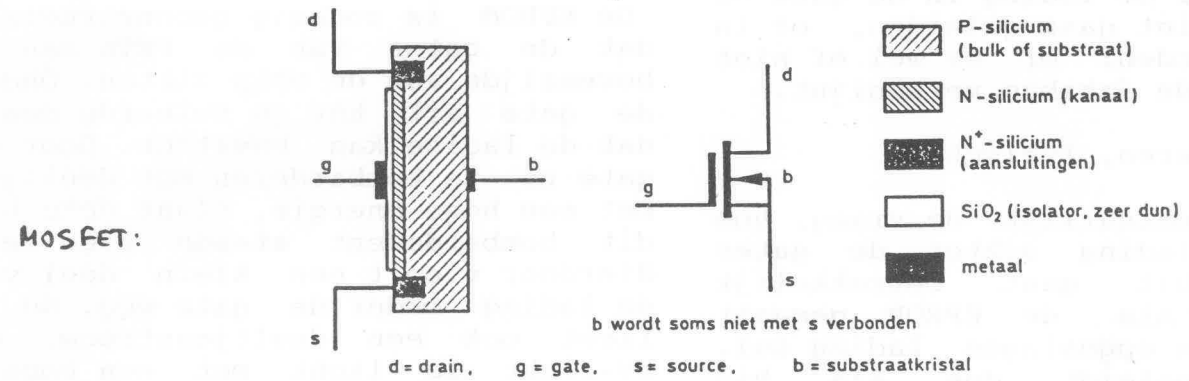
Schematische voorstelling van een grenslaag-FET

Afb. 1

Het blijkt dat deze stroom een maximum waarde heeft, die alleen bepaald wordt door het beschikbare aantal vrije ladingsdragers (elektronen) in het kanaal. Deze stroom is dus onafhankelijk van de spanning over het kanaal. De FET gedraagt zich in deze toestand dus als een stroombron. De verzadigingsstroom wordt I_{DSS} genoemd.

Maken we de gate negatief ten opzichte van de source, dan ontstaat er een electrisch veld in het kanaal. Dit veld drijft een deel van de vrije electronen weg uit de buurt van de gate, waardoor het nog geleidende deel van het kanaal smaller wordt. Hierdoor neemt de weerstand van het kanaal toe, waardoor er minder stroom gaat lopen.

Dit negatief maken van de gate kunnen we voortzetten, totdat het



Afb. 2

veld rond de gate zo sterk is, dat alle electronen uit het kanaal verdreven worden. Op dat moment loopt er geen stroom meer door het kanaal; of anders gezegd: de FET staat afgeknepen. De gate- source spanning waarbij dat gebeurt heet de pinch-off spanning, of U_P.

2. De FET met stacked (gestapelde) gate.

Tot dusver hebben we het alleen maar over standaard JFETs gehad. In EPROMs wordt een variant op de standaard FET gebruikt, namelijk de MOSFET (MOS betekent Metal Oxide Screen). Deze FETs hebben een andere gate-constructie. Tussen de gate, die meestal van metaal is, is een laagje Siliciumoxide (SiO_2) aangebracht, dat zeer goede isolatieeigenschappen heeft. De werking van deze FETs is gelijk aan die van de JFETs met dit verschil dat er in het geheel geen stroom in de gate loopt. Verder leent deze FET zich zeer goed voor integratie in IC's. De FETs in een EPROM zijn vrijwel altijd van het MOS-type, en hebben een extra groot isolatiegebied in de gate, waarin een geleidend deel silicium is opgesloten.

Het is nu mogelijk, de pinch-off spanning te verschuiven, en wel door een permanent veld(je) aan te brengen. Dit gebeurt, door in het geleidende gebied een lading aan te brengen. Omdat het materiaal om het gebied een goede isolator is, kan de lading niet weglekken en zit dus opgesloten.

Als er geen lading is aangebracht, dan hebben we met een gewone FET te maken, d.w.z. de FET geleidt bij $U_{GS}=0$ en spert bij $U_{GS}=-U_P$.

Brengen we echter een negatieve lading aan in het ge soleerde gebied in de gate, dan verschuift de waarde van de pinch-off spanning naar boven, omdat het veld dat door de gate wordt opgewekt wordt opgeteld bij het reeds aanwezige stationaire veld. De aangebrachte lading kan zo groot worden dat de pinch-off spanning zelfs positief wordt. In dat geval staat de FET dus afgeknepen als U_{GS} nul volt is.

Omdat de gate nu bestaat uit een stapeling van de aansluiting, een laagje oxide, een laagje geleidend silicium en weer een laagje oxide, worden dergelijke FETs wel stacked gate FETs (=gestapelde gate FET)genoemd.

3. EPROM-opbouw.

Zoals reeds gezegd, bestaan de geheugenelementen in een EPROM uit FETs met een gestapelde gate. Iedere FET stelt 1 bit in de EPROM voor. De gates van de FETs zijn verbonden met de adresdecoder in de EPROM. Alle sources liggen aan de nul, en de drains zijn steeds in stellen van acht met de databusbuffers verbonden. De adresdecoder activeert de acht FETs die bij een bepaald adres horen, door de gates aan de nul te leggen. Van de niet geselecteerde FETs ligt de gate aan een negatieve spanning, die op de chip gemaakt wordt (door de zgn. substraat bias generator). Alle niet-geselecteerde FETs blijven dus

altijd gesperd. Van de geselecteerde FETs bepaalt de lading in de gate of ze wel of niet gaan geleiden, of in andere woorden: of er wel of niet een nul op de databus verschijnt.

4. Programmeren, fysisch.

Nu is het natuurlijk de vraag, hoe we die lading achter de gates krijgen. Dit gaat betrekkelijk eenvoudig. Als de EPROM gemaakt wordt, is de opgeslagen lading nul. De FET geleidt dus als hij geselecteerd wordt. Het aanbrengen van de lading geschiedt door het aanleggen van een (relatief) hoge spanning tussen de drain en de gate van de FET. Hierdoor treedt doorslag op, waardoor een aantal electronen naar het ge soleerde deel kunnen stromen. Als de spanning weer wordt weggehaald kunnen de elektronen echter niet weer weg, ze zitten opgesloten, en de FET blijft nu afgeknepen als U_{GS} nul volt is. Het materiaal tussen drain en gate is zodanig gemaakt, dat het doorslagproces geen blijvende schade aanricht. Wel is er sprake van een zekere 'slijtage' waardoor de EPROM niet een oneindig aantal keren geprogrammeerd kan worden.

De relatief hoge spanning is de programmeerspanning. Omdat de maten op chip-niveau in microns worden uitgedrukt, zal het duidelijk zijn, dat doorslag al optreedt bij enkele tientallen volts. In de praktijk ligt de programmeerspanning tussen 12.5 en 25 Volt.

De programmeerpuls schakelt de programmeerspanning naar de te programmeren FET(s). De duur van deze puls bepaalt dus, hoeveel elektronen, of anders gezegd, hoeveel lading er onder de gate 'geschoten' wordt.
Merk op, dat er bij het programmeren nergens sprake is van verbranding of zelfs verhitting van materiaal. De uitdrukking EPROMs 'branden' is dan ook volkomen onjuist.

5. Wissen, fysisch.

Iedereen is wel op de hoogte, dat EPROMs ook weer gewist kunnen worden. Dit gebeurt op de volgende manier.

De EPROM is zodanig geconstrueerd, dat de gates van de FETs aan de bovenzijde van de chip zitten. Onder de gate zit het ge soleerde deel, dat de lading kan bevatten. Door de gate nu te bombarderen met deeltjes met een hoge energie, slaat deze bij dit bombardement steeds even lek. Hierdoor vloeit een klein deel van de lading onder de gate weg. Nu is licht ook een deeltjesstroom, en UV-licht is licht met een hogere energie-inhoud. Vandaar dat EPROMs gewist kunnen worden door ze met UV-licht te beschijnen. Om beschijning van de chip zelf mogelijk te maken, zijn alle EPROMs voorzien van een venster van kwartsglas in het huisje.

6. EPROM programmeren.

Het programmeerproces zal nu inmiddels duidelijk zijn geworden: de programmeerspanning zorgt ervoor, dat er doorslag tussen drain en gate van de FETs in de EPROM kan optreden, terwijl de lengte van de programmeerpuls bepaalt, hoeveel lading er onder een gate verdwijnt. Het gemak waarmee de drain-gate overgang doorslaat bepaalt ook hoeveel lading er per eenheid programmeerpuls onder de gate verdwijnt, en ook hoe gemakkelijk de EPROM weer gewist kan worden. De kwaliteit van de isolatie rond de gate bepaalt hoelang de EPROM de geprogrammeerde data vast kan houden. Naarmate de isolatie slechter is, vloeit er meer lading weg. In de praktijk is het een en ander zo uitgevogeld, dat een EPROM die op de juiste manier geprogrammeerd is, de data tenminste 10 jaar kan vasthouden.

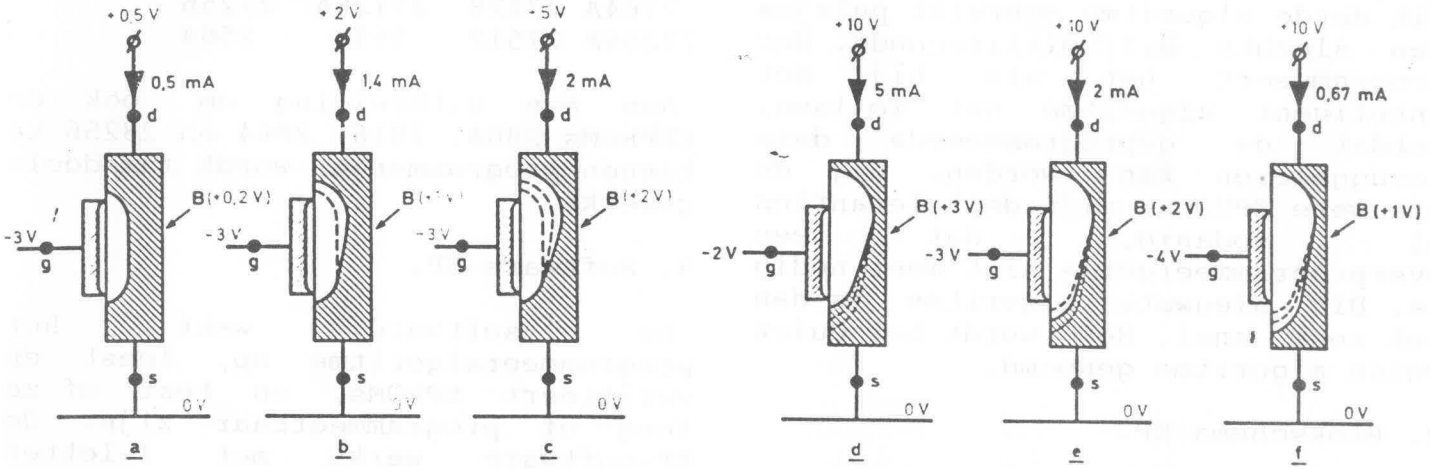
7. Programmeeralgoritmes.

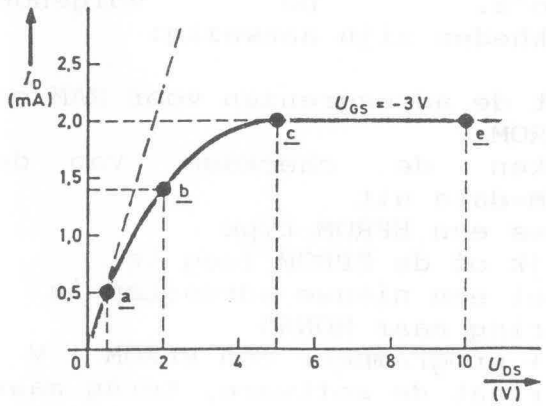
Het programmeeralgoritme dient zodanig in elkaar te zitten, dat er een zodanige lading onder de gate wordt aangebracht, dat de EPROM de data tenminste 10 jaar vast kan houden. Het aanbrengen van te weinig

lading bekort deze tijd; teveel lading echter betekent onnodige doorslag (hierbij treedt slijtage op in het materiaal) en ook een langere wistijd.

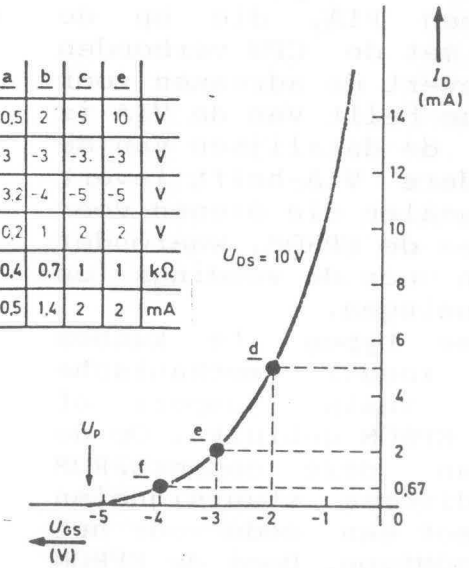
De eerste algoritmen zijn nogal bot op dit punt. Ieder te programmeren adres ontvangt een puls die zodanig van lengte is dat de slechtste FET nog betrouwbaar geprogrammeerd wordt. Bij de oudste EPROMs is de lengte van de programmeerpuls dus vast, en bedraagt 50 milliseconde. Vrijwel alle 2716's en 2732's moeten op deze manier geprogrammeerd worden.

Intel, de uitvinder van de EPROM, kwam in het begin van de jaren '80 met een algoritme dat zodanig in elkaar zit, dat iedere FET in principe de juiste hoeveelheid lading krijgt toegevoerd. Het algoritme programmeert meerdere malen met korte pulsjes van 1 milliseconde, totdat de geprogrammeerde data uit de EPROM kunnen worden teruggelezen. Op dat moment is de pinch-off spanning van de te programmeren FETs zodanig verschoven, dat deze juist afgeknepen staan als ze geselecteerd worden. Het tot dusver benodigde aantal pulsjes wordt vervolgens met drie vermenigvuldigd. De FETs worden nu overgeprogrammeerd met een puls die de drievoudige pulsduur heeft van de tot dusver toegevoerde pulsduur. De FETs bevatten nu een redelijk nauwkeurig bepaald overschot aan lading, die ervoor zorgt dat de pinch-off spanning ver genoeg boven nul volt ligt. Dit algoritme werd door Intel het inteligent algoritme genoemd. Het





	a	b	c	e	
U_{DS}	0,5	2	5	10	V
U_{GS}	-3	-3	-3	-3	V
U_{GB}	-3,2	-4	-5	-5	V
U_{BS}	0,2	1	2	2	V
R_{BS}	0,4	0,7	1	1	kΩ
I_D	0,5	1,4	2	2	mA



	d	e	f	
U_{DS}	10	10	10	V
U_{GS}	-2	-3	-4	V
U_{GB}	-5	-5	-5	V
U_{BS}	3	2	1	V
R_{BS}	0,6	1	1,5	kΩ
I_D	5	2	0,67	mA

Afb 3
FET karakteristieken

werd voor het eerst toegepast op de 2764 EPROM.
Door het voortschrijden der techniek worden EPROMs steeds beter, dat wil zeggen dat er steeds minder lading onder de gates moet worden opgeslagen om dezelfde data bewaartijd te verkrijgen. Hierdoor daalt in principe de pulsduur van de programmeerpuls. Ook worden de isolatie-eigenschappen van de gates steeds beter. Hierdoor wordt de isolatielaag rond de gate kleiner, en daarmee daalt ook de programmeerspanning. Vandaar dat nieuwere versies van bestaande EPROMs soms met een lagere programmeerspanning geprogrammeerd moeten worden.
Dit alles heeft geleid tot een eveneens door Intel ontwikkeld algoritme, dat uitsluitend bij EPROMs met 12.5 Volt programmeerspanning wordt gebruikt. Dit derde algoritme gebruikt pulsjes van slechts 0.1 milliseconde. Het programmeert net als bij het inteligent algoritme net zo lang, totdat de geprogrammeerde data teruggelezen kan worden. Bij de nieuwere EPROMs zijn de toleranties al zodanig, dat een overprogrammeeractie niet meer nodig is. Dit nieuwste algoritme is dan ook zeer snel. Het wordt het Quick Pulse algoritme genoemd.

8. Blokschema EP.

De EP-hardware is opgebouwd rond een VIA en een PIA, die op de normale manier met de CPU verbonden zijn. De PIA levert de adressen voor de EPROM, de ene helft van de VIA is verbonden met de datalijnen van de EPROM. De andere VIA-helft levert een aantal signalen die dienen voor de besturing van de EPROM, waaronder schakelsignalen voor de voedings- en programmeerspanningen.
Om de diverse typen te kunnen programmeren zonder mechanische omschakelingen zoals jumpers of relais is een EPROM gebruikt. Op de adreslijnen van deze codeerEPROM worden de diverse stuursignalen gezet, samen met een code voor het onderhavige EPROMtype. Door de EPROM nu slim te programmeren, verschijnen op de datalijnen direct de gewenste signalen voor de diverse pinnen op de EPROM programmeervoet.
Een aantal pinnen op die programmeervoet vereisen wat meer dat alleen TTL-signalen. Dit zijn bijvoorbeeld de voedingspinnen en de pinnen waarop de programmeerspanning kan voorkomen. Deze pinnen worden vanuit de codeerEPROM gestuurd via L200 instelbare spanningsregelaars. Voor de keuze van de programmeerspanning is een zelfde weg bewandeld als voor de signalen: een PROMmetje. Door de codeerEPROM en de PROM opnieuw te programmeren kan de hardware eventueel worden aangepast aan nieuwe typen. De huidige versie van de hardware is geschikt om de volgende typen te programmeren:

2716	2732	2732A	2764
2764A	27128	27128A	27256
27256A	27512	2532	2564

Aan een uitbreiding om ook de EEPROMs 2804, 2816, 2864 en 28256 te kunnen programmeren wordt inmiddels gewerkt.

9. Software EP.

De software wekt het programmeeralgoritme op, leest en verifieert EPROMs, en test of ze leeg of programmeerbaar zijn. De EP-software werkt met 1-letter commando's. De volgende mogelijkheden zijn aanwezig:

B	zet de adresgrenzen voor RAM en EPROM
C	reken de checksum van de RAM-data uit
D	kies een EPROM type
E	kijk of de EPROM leeg is
I	stel een nieuwe adresstap in
M	spring naar MON65
P	T + programmeer een EPROM + V
Q	verlaat de software, terug naar DOS
R	lees een EPROM in RAM
S	laat de EPROMinhoud zien zonder RAM te laden
T	test of de EPROM programmeerbaar is

V vergelijk EPROM- en RAMdata met elkaar

De software is in staat de drie meest gangbare algoritmen op te wekken: 50 ms, inteligent en quick pulse. Verder zijn de parameters van het inteligent algoritme eenvoudig te veranderen vanuit MON65.
De adresstap laat toe, ook met 16- en 32-bit data te werken.

EP is op dit moment alleen beschikbaar voor DOS65. Er is inmiddels iemand bezig het om te schrijven naar OHIO-DOS.

Harddiskcontroller.

Er zijn wat plannen, om een harddiskcontrollerkaart te gaan ontwikkelen voor de Elektuurbus. De kaart zal worden opgezet rond de Western Digital chip WD1010 en is bedoeld voor de pseudo-standaard interface ST506, waarmee de meeste gangbare drives zijn uitgerust. Laat eens weten of u hiervoor eventueel belangstelling heeft.
N. de Vries.

Status van de EPROMprogrammer.

De eerste proefprint is klaar en wordt thans volgebouwd. Als alle bugs eruit zijn, wordt de kaart in productie genomen. Omdat de kaart enkelzijdig is geworden kan hij voor een bescheiden prijs worden aangeboden. In totaal bestaat EP uit twee printen: de eurokaart met de electronica, en een klein printje voor de ZIF-voet. Inmiddels is Ton Smits begonnen de software om te schrijven naar OHIO-DOS, zodat gebruikers van dit DOS ook aan hun trekken kunnen komen.
De club zal ervoor zorgen, dat de PAL en de PROM die in de schakeling voorkomen geprogrammeerd en wel in combinatie met de print zullen kunnen worden gekocht. Meer definitief nieuws in het volgende nummer. Als u haast heeft, belt u 010-4517454 (Nico de Vries).

DOS65 bug.

Het blijkt, dat de positioneersequence (control-T, gevolgd door de gewenste coördinaten) niet altijd goed werkt, als dit via de standaard DOS-routine CHROUT geschiedt. Dit komt omdat DOS een aantal karakters uitfiltert. Zo werkt bijvoorbeeld een positionering op 13,13 niet goed. De oplossing is, om niet de DOS-entry voor CHROUT, maar de IO65-entry te gebruiken. Ook kan de speciale entry POS, eventueel gevolgd door UNPOS, (beide in IO65) gebruikt worden. Deze werken altijd goed.

Voorlopige bijeenkomstenagenda 1988.

Onder voorbehoud zullen dit de plaatsen en data zijn waarop de bijeenkomsten zullen worden gehouden:

16 januari	Krommenie
19 maart	Geldrop
21 mei	Krimpen a/d IJssel??
17 september	Haarlem
19 november	Almelo

Wij streven ernaar u op iedere bijeenkomst een aantrekkelijk programma te bieden. Suggesties voor thema's voor bijeenkomsten zijn steeds welkom. Indien u een spreekbeurt wilt verzorgen kunt u dit bij het bestuur aanmelden.
Wie organiseert er een bijeenkomst in Belgie??

Echt waar:

On an empty disk, you can seek forever.....

Korte cursus Vlaams:

The system has crashed -
Uw machien is de natuur in, of:
Uw machien is in panne gevallen.

Indirect addressing:
onbereikbaar adresseren

Database for the BBC model B.

Our club-member H.K. Hemminga has developed a simple database system for the BBC model B micro computer. Here are some quotes from the documentation we received from him.

What is SIDAP?

SIDAP is an acronym for Simple Indexed Data Package that enables the BBC micro computer equipped with a floppy disk drive to access datasets (files) indexed by an ASCII key.

Hard- and software requirements

SIDAP is designed to work on a BBC model B (32 kbyte RAM) microcomputer fitted with BBC OS 1.2 or higher and the BBC DFS (disk file system), which can be either single or dual density and use either 40 or 80 tracks disk drives.
The application programs written in BASIC can simply CALL SIDAP with some parameters that specify the function requested and the BASIC variables involved.

Main features of SIDAP

SIDAP has the following features:

1) Support for up to five indexed datasets open simultaneously.
2) Key lengths from 1 to 82 bytes of almost any ASCII value.
3) Data record lengths up to 255 bytes subdivided in fields.
4) A SIDAP dataset is simply a BBC file, which can be copied etc.
5) Fast access through the use of an integrated MASTER INDEX with BITMAP.
6) The datasets never need reorganization, this is realized by comprehensively using the index and data areas.
7) The error handling is well defined to ensure that the integrity of the dataset is maintained.
8) It is written in assembler to assure speed and minimal memory requirements.
9) It is supported by a service program that is easy to use.

List of supported functions

The online data manager of SIDAP supports the following functions:

'O' Logical Open the dataset.
'C' Logical Close the dataset.
'I' Get information about the number of records in the dataset.
'S' Start notless by key or generic key, no data is returned.
'R' Read by key or generic key.
'W' Write new key and associated data.
'U' Update the data by key.
'D' Delete the data by key.
'F' Read the First record of the dataset.
'N' Read the Next record of the dataset.
'P' Read the Previous record of the dataset.
'L' Read the Last record of the dataset.

Package contents

The package consists of:

1) Online datamanager (written in 6502 assembler) in object form.
2) A service program (written in BBC BASIC and assembler).
3) Documentation.

How to obtain SIDAP free of charge

The SIDAP package can be obtained free of charge and under no warranty:
Send an audio cassette and a self addressed, stamped envelope to:

H.K. Hemminga
Hoofdweg 152-3
NL-1057 DB Amsterdam
The Netherlands.

VERZEKERING VAN PERSONAL COMPUTERS

Verzekerd kunnen worden dé micro-computers, bestemd voor prive gebruik.

Deze apparatuur is ook buiten de woning of de bedrijfsruimte verzekerd.
De verzekering is namelijk van kracht in de Benelux en West-Duitsland.

VERGOED WORDEN:

1. schaden aan de apparatuur veroorzaakt door:
   - brand, ontploffing en kortsluiting (incl. eigen gebrek)
   - een van buiten komend onheil.

2. verlies van de apparatuur door diefstal.
   - diefstal van de apparatuur buiten het in de polis genoemde pand is verzekerd na braak.

3. de bereddingskosten die zijn gemaakt tot vermindering van schade aan de apparatuur.

4. de opruimingskosten tot 10 % van de op apparatuur verzekerde som.

SCHADEVERGOEDING

De reparatiekosten worden vergoed:
Bij totaal verlies wordt de nieuwwaarde uitgekeerd. De vergoeding geschiedt echter naar dagwaarde indien de waarde minder dan 40 % van de nieuwwaarde is.

NADERE BIJZONDERHEDEN

De verzekering wordt gesloten voor de duur van 1 jaar en wordt stilzwijgend verlengd voor dezelfde termijn.

De premie voorwaarden worden op aanvraag verstrekt voor:
- apparatuur die in beroep of bedrijf voor andere dan administratieve doeleinden wordt gebruikt.
- apparatuur in gebruik bij onderwijsinstellingen en verenigingen.

LET OP !!

Leden van de Kim gebruikers club nederland betalen geen poliskosten.

INLICHTINGEN: ADMINISTRATIEKANTOOR A. METZLAR
VOORBURGGRACHT 479
1724 RH OUDKARSPEL
TEL.02260-16889

TECHNITRON TLP-12 LASER PRINTER
— U HEEFT EIGENLIJK GEEN ANDERE KEUZE!

- 12 pagina's per minuut (max.)
- tot 10.000 afdrukken per maand
- 8 ingebouwde lettertypes; 32 afdruk-combinaties
- unieke "FontMaker" service
- unieke "FormsMaker", formulier- en logo service
- 3 ingebouwde hardware-emulaties
- flexibele in- en uitvoer van papier

Technitron DATA

Technitron Data B.V.
Zwarteweg 110, Postbus 14,
1430 AA Aalsmeer
tel. 02977-22456
telefax 02977-40968
telex 13301

Vestigingen in:
BONDSREPUBLIEK DUITSLAND – DENEMARKEN – ENGELAND – FRANKRIJK – ITALIË – NOORWEGEN – VERENIGDE STATEN – ZWEDEN