De uP Kenner 71

PATH voor DOS65, H. Speksnijder

De uP Kenner 72

RAMTEST voor DOS65, F. Vandekerkhove

De uP Kenner 80

EP revisited, N. de Vries

De uPKENNER 81 – April 1993

Voortgang DOS65

De uPKENNER 82 – Augustus 1993

Hang eens een muis aan je DOS65

Voortgang DOS65 RAM kaart

De uPKENNER 83 – Oktober 1983

DOS65 uit de ijskast!

Hang eens een muis aan je DOS65 (deel twee)

Voortgang DOS65

De uPKENNER 84 – December 1993/Februari 1994

Voortgang DOS65

De uPKENNER 86 – Augustus 1994

DOS65 Virtual disk op de nieuwe RAM kaart

De uPKENNER 87 – oktober 1994

DOS65: RAM(P) kaart?

PATH: het path-mechanisme in DOS-65 gemaakt

Bij sommige computers is het niet nodig dat alle commando's in dezelfde directory zitten. Het is dan bijvoorbeeld mogelijk een lijstje van verschillende directory's op te geven. Als daarna een commando wordt ingetypt, dan zoekt het systeem volgens dat lijstje het commando op. Bij DOS-65 kan een commando in een andere directory wel uitgevoerd worden maar dan moet u bijv. 1:d/DEMO intypen. Daarin is DEMO dan een commando op directory d/ van drive 1. Vooral tijdens het maken van een nieuwe utility is het heel handig als DOS-65 ook het path mechanisme kent. Men dan ook niet langer verplicht om alle utilities in 0: te zetten.

De werking van DOS-65:

a) DOS-65 zoekt z'n commando's eerst op in een lijst, daarin staan bijv. DIR, LOAD en SAVE.
b) als het commando daar niet gevonden is, dan volgt een sprong naar $C832 en wordt het commando op de systeem-drive gezocht. Wordt het commando daar niet gevonden, dan volgt een foutmelding m.b.v. jmp $C847.
c) Als het gevonden is dan wordt het commando uitgevoerd m.b.v. jmp $C849.

De wijziging is niet zo erg moeilijk

Het zoeken van a) blijft bestaan maar als het commando niet gevonden is, volgt een sprong naar een klein extra programma dat staat op $BF00 en in het geheugen blijft staan. Dit extra programma zoekt niet alleen in de systeem-drive, maar ook in andere directory's. Daartoe wordt een lijstje bijgehouden dat direct achter het programma staat: pathlijst. Is het zoeken niet succesvol, dan volgt exact dezelfde sprong als bij DOS-65. Als het zoeken wel succesvol is dan wordt het betreffende commando uitgevoerd, ook weer door te vervolgen op dezelfde plaats waar DOS-65 dat zou doen.

PATH

Om te kunnen instellen welke drive/dir in de lijst staat, is een utility gemaakt waarbij een beetje gekeken is naar ASN. Deze utility test eerst of het path er al is, door te kijken welk adres op orgjmpa staat. Indien het nog niet geïnstalleerd is, dan wordt dat alsnog gedaan. Daarbij wordt wel bewaakt of u de juiste versie van DOS-65 heeft.
Als er geen opties en geen drive/dir zijn opgegeven bij PATH dan volgt een display v.d. waarden die nu in de pathlijst staan. Is er wel drive/dir is opgegeven dan worden die toegevoegd aan de pathlijst. De pathlijst mag echter niet langer dan 8 worden. Dat is voor de veiligheid want met een floppy disk zou het zoeken anders wel erg lang gaan duren.

Wordt als optie -R gegeven, dan wordt alleen de laatste van de pathlijst verwijderd. De rest v.d. pathlijst blijft dus bestaan. Hierbij is ook toegestaan in te geven PATH -R 1a. In dit geval wordt de laatste v.d. pathlijst verwijderd en daarna 1:A/ aan de pathlijst toegevoegd. Ook wordt ervoor gewaakt dat er geen lege lijst ontstaat want dan zouden er helemaal geen commando's meer uitgevoerd kunnen worden. Daarom wordt de eerste in de lijst alleen overschreven door de combinatie: PATH -c 1b [1c]. Daarbij wordt dus de gehele pathlijst uitgewist en dan worden 1:B/ [en evt. 1:C/] erin gezet. Dus PATH -c mag wel, maar dan wordt de lijst niet helemaal uitgewist. Het resultaat van een commando op de pathlijst is gemakkelijk zichtbaar te maken door de -? optie.

Assembleren van path.mac:

De variabele selectproc bepaalt voor welke processor er geassembleerd wordt:

```
selectproc   equ   nmos    standaard NMOS 6502
selectproc   equ   cmos    standaard CMOS 6502
selectproc   equ   rcmos   Rockwell CMOS 6502
```

Enige belangrijke variabelen in PATH:

```
opt       equ $FE     vlag geset door het resident
                      programma
freemem   equ $BF00   vrij gedeelte in DOS-65
                      (max 1 pagina )
csufl     equ $ABD0   vlag voor commando
sdrive    equ $ABD1   system drive
orgjmpa   equ $C66B   de plaats van het originele
                      jmp adres.
orgjmpd   equ $C832   de bestemming v.h originele
                      jmp adres.
comerr    equ $C847   branch in DOS-65 als
                      commando niet gevonden is.
comfnd    equ $C849   vervolg in dos65 als het
                      commando gevonden is.
```

Ter informatie:

De getallen die DOS-65 gebruikt voor de specificatie van de drive en de directory zijn tussen 0 en $1F. Enige voorbeelden:

```
$00 = 0:
$04 = 0:a/
$08 = 0:b/
$0C = 0:c/
$10 = 0:d/
$01 = 1:
$05 = 1:a/
$09 = 1:b/
$0D = 1:c/
$1D = 1:g/
```

H.Speksnijder

```
;file path.mac
;                                       19900815
;                                       Henk Speksnijder
;                                       Capelle a/d IJssel
;
;assembleer voor rcmos, cmos of nmos processor.
;default is RCMOS.
;
freemem     equ     $BF00               een pagina vrij geheugen in Dos65.
;                                       PAS OP: er wordt een klein programma
;                                       resident geinstalleerd op dit adres.
;                                       De rest van deze utility is alleen nodig
;                                       bij het installeren of wijzigen v.h. path.
;
;----------CONSTANTEN----------------------------------------
;
CR          equ     $0D                 carriage return
space       equ     $20                 space
;
;----------dos IO routines-----------------------------------
;
inch        equ     $C020               dos input character
prch        equ     $C023               dos output print character
crlf        equ     $C02F               print cr + lf
prhx        equ     $C038               print een byte hex
prtx        equ     $C03b               print text tot een nul
loupch      equ     $C041               accu lower to upper
sopt1       equ     $C068               get option
spar1       equ     $C06B               get parameter
open        equ     $D03F               open a file, AY wijst naar filenaam
close       equ     $D048               close a file, X is filenummer
ermes1      equ     $D0B7               error messages
;
;----------RCMOS, CMOS, NMOS---------------------------------
;
rcmos       equ     2                   Rockwell cmos
cmos        equ     1                   standard cmos
nmos        equ     0                   gewone nmos
;
selectproc  equ     rcmos
;
            if      selectproc==rcmos
            opt     rc02
            elseif  selectproc==cmos
            opt     c02
            elseif  selectproc==nmos

ldai        macro   ad                  vervang de cmos: lda [ad]
            ldy     #0
            lda     [ad],y
            endm

stai        macro   ad                  vervang de cmos: sta [ad]
            ldy     #0
            sta     [ad],y
            endm
```

```
bra             macro   ad              vervang de cmos: bra ad
                jmp     ad
                endm

inca            macro                   vervang de cmos inc accu
                clc
                adc     #1
                endm

deca            macro                   vervang de cmos dec accu
                sec
                sbc     #1
                endm

                endif

                if      selectproc! = nmos

ldai            macro   ad              gebruik echte cmos instructie
                lda     [ad]
                endm

stai            macro   ad              gebruik echte cmos instructie
                lda     [ad]
                endm

                endif
;
;-------------------algemene MACRO'S-----------------------------
;
cpxa            macro   op              vergelijk woord in XA met operand
                if      !#,op
                cpx     op + 1
                bne     200.f
                cmp     op
200
                else
                cpx     op > > 8
                bne     200.f
                cmp     op&255
200
                endif
                endm

ldxa            macro   op              load met A = low byte en X = high byte
                if      !#,op
                lda     op
                ldx     op + 1
                else
                lda     op&255
                ldx     op > > 8
                endif
                endm

stxa            macro   ad
                sta     ad
                stx     ad + 1
```

```
                endm

;-----------------Dos 65 intern-----------------------------------------
;
                ;page zero
optmask         equ     $A0             byte om de optie's te bewaren
                ;enkele variablen gebruikt in Dos65.
t1              equ     $F0             temp (pointer) voor deze utility
t2              equ     $F2             temp (input byte) voor deze utility
opt             equ     $FE             flag gezet door residente prog. t.b.v. dos65.
csufl           equ     $ABD0           flag voor commando
sdrive          equ     $ABD1           system drive
orgjmpa         equ     $C66B           de plaats van het originele jmp adres.
orgjmpd         equ     $C832           de bestemming v.h originele jmp adres.
comerr          equ     $C847           branch in dos65 als commando niet gevonden is.
comfnd          equ     $C849           vervolg in dos65 als het commando gevonden is.

 ;--------------Opties-------------------------------------------------
;
erase           equ     $80             -E : Verwijdert het path en het residente programma
;                                       en herstelt Dos65 in originele staat.
clear           equ     $40             -C : Maakt de gehele path lijst schoon.
remove          equ     $20             -R : Verwijdert alleen de laatste van de path lijst.
show            equ     $10             -? : Laat actuele pathlijst zien op display.
help            equ     $08             -H : Laat help-text van PATH zien.
;
 ;------------MAIN--hoofdprogramma-----------------------------
;
                org     $A000
begin           jmp     pastart
                fcc     $C8,$C5,$CC,$D0  support Dos65: HELP
seehelp         jsr     prtx
                fcc     '\rDefines the command-path'
                fcc     '\rDefault is the system drive'
                fcc     '\rSyntax : PATH [-ECAR?H] [drive:directory] etc.'
                fcc     '\roptions :'
                fcc     '\r   -E erase entire path'
                fcc     '\r   -C clear the pathlist'
                fcc     '\r   -R remove last entry from the pathlist'
                fcc     '\r   -? show actual pathlist'
                fcc     '\r   -H Show this comment'
                fcc     '\rNo abbreviation.    max. 8 drive/dir specs'
                fcc     '\rthe characters / , : @ are optional'
                fcc     '\rExamples:'
                fcc     '\rPATH 0:@/ 0:d/    drive 0 main dir. and dir d/'
                fcc     '\rPATH 0a 0d 1a     drive 0 dir. a/ + d/ + drive 1 dir. a/'
                fcc     '\rPATH              show the current situation'
                fcc     '\rPATH -? 1a        add 1:a/ to the pathlist and print it'
                fcc     '\rPATH -R           remove last entry from path'
                fcc     '\r'
                fcc     '\rPATH -C 1:a       clear pathlist and then put 1:a/ in it'
                fcc     '\rremark: only this will change the first entry in pathlist'
                fcc     '\r',0
                rts
;
;BEGIN VAN PROGRAMMA
;
```

```
pastart   jsr    sopt1          sopt1 = scan opties
          fcc    'ECR?H',0
          bcc    1.f
          jmp    ermes1         error met optie's
1         sty    t1             save command buffer pointer
          sta    t1+1
          stx    optmask        save opties masker
          txa
          bit    #help
          beq    2.f
          jmp    seehelp        show help-text
          ;
2         ldxa   orgjmpa
          cpxa   #path          test of path al geinitialiseerd is
          beq    p2
          ;
          cpxa   #orgjmpd       test of adresses goed zijn
          beq    p1
          jmp    error1         onjuiste dos65 versie
          ;
p1        ldxa   #path          initializeer path
          stxa   orgjmpa        dos zal jmp uitvoeren naar path routine
          lda    sdrive
          sta    pathlis
          lda    #$FF           clear path list,
          sta    pathlis+1      only the first entry is there now.
          ;
p2        lda    optmask
          bit    #erase         indien nodig erase het path
          beq    1.f
          jmp    stop
1         bit    #clear         indien nodig clear het path
          beq    2.f
          lda    #$FF
          sta    pathlis        (sentinel op eerste positie )
          bra    10.f
2         bit    #remove        indien nodig verwijder laatste v.d. lijst
          beq    10.f
          jsr    findend
          cpx    #1             (er moet er 1 in de lijst blijven )
          beq    10.f
          dex
          lda    #$FF           (nieuwe sentinel )
          sta    pathlis,x
          ;
10        ldy    #0             scan i.v.m. empty string
          jsr    skipspa
          bne    p3
          jsr    tespath
          jmp    seepath
          ;
p3        jsr    loupch         scan voor drive:dir specificatie
          cmp    #'0'
          bcc    31.f           branch if accu < '0'
          cmp    #'4'           drive nummer
          bcc    32.f           branch if accu < '4'
31        jsr    error2
```

```
                jmp     p9
32              eor     #'0'
                sta     t2
                jsr     next
                beq     p4
                cmp     #':'
                bne     33.f
                jsr     next
                beq     p4
33              jsr     loupch          directory
                eor     #'@'
                cmp     #8
                bcc     35.f            branch if accu < 8
                jsr     error3
                jmp     p9
35              asla
                asla
                ora     t2
                sta     t2
                jsr     next
                beq     p4
                cmp     #'/'
                bne     36.f
                jsr     next
                beq     p4
36              jsr     error2
                jmp     p9
                ;
p4              ldx     #0              zoek of deze drive/dir al in de lijst stond
1               lda     pathlis,x
                cmp     #$FF            ( sentinel )
                beq     p5
                cmp     t2
                beq     2.f
                inx
                bpl     1.b
                bra     p5
2               jsr     seedrdi         display drive/directory
                jsr     error4          "drive/dir already in the list"
                bra     p6
                ;
p5              cpx     #8
                bcc     1.f             branch if x < 8
                jsr     error5          path is vol
                jmp     p9
1               lda     t2
                sta     pathlis,x       nieuwe entry in pathlist
                inx
                lda     #$FF
                sta     pathlis,x       (nieuwe sentinel )
                ;
p6              jsr     skipspa
                beq     p9              indien command line niet leeg dan continue
                jmp     p3
p9              jsr     tespath
                lda     optmask         indien nodig display actuele pathlijst
                bit     #show
```

```
                beq     9.f
                jsr     seepath
9               rts

tespath         lda     #$FF            Voor de veiligheid, want het Dos65 system
                cmp     pathlis         kan niet werken zonder drive.
                bne     9.f
                sta     pathlis+1
                lda     sdrive
                sta     pathlis
9               rts

stop            ldxa    #orgjmpd        herstel oude jump adres in dos65
                stxa    orgjmpa
                jsr     prtx
                fcc     'Path erased\r\r',0
                rts
error1          jsr     prtx
                fcc     '\rIncorrect Dos65 version',0
                rts
error2          jsr     prtx
                fcc     'Usage: PATH drive:directory\r',0
                rts
error3          jsr     prtx
                fcc     'Incorrect drive and/or directory assignment\r',0
                rts
error4          jsr     prtx
                fcc     'already in the path list\r',0
                rts
error5          jsr     prtx
                fcc     'path is full\r',0
                rts

seepath         jsr     prtx
                fcc     'Path ',0
                ldx     #0
10              jsr     seedrdi         display deze drive/directory
                inx
                lda     pathlis,x
                cmp     #$FF
                bne     10.b
                jsr     crlf
                rts

                ;display drive/directory
                ; het X-reg. geeft aan welk nummer van de pathlijst.
seedrdi         lda     pathlis,x
                and     #3              neem drive en zet om naar character
                ora     #'0'
                jsr     prch
                lda     #':'
                jsr     prch
                lda     pathlis,x
                and     #$1C            neem dir. en zet om naar character
                beq     11.f
                lsra
                lsra
```

```
                eor     #'@'
                jsr     prch
                lda     #'/'
                jsr     prch
11              lda     #space
                jsr     prch
                lda     #space
                jsr     prch
                rts

1               iny
skipspa         lda     [t1],y
                beq     9.f
                cmp     #space
                beq     1.b
                cmp     #CR
9               rts

next            iny
                lda     [t1],y
                beq     9.f
                cmp     #CR
                beq     9.f
                cmp     #space
9               rts

findend         ldx     #0              vind het einde van de pathlijst
1               lda     pathlis,x
                cmp     #$FF
                beq     2.f
                inx
                bpl     1.b
2               rts

;---------- PATH-----------------------------------------------
;PAS OP: hier komt dan het programma dat resident in het geheugen blijft.

                org     freemem         vrij beschikbare gebiedje in dos65
path            sta     asave
                sty     ysave
                lda     #0
                sta     pacount         tel hoe ver in de pathlijst
                ;
1               ldx     pacount
                lda     pathlis,x       instellen systeem drive
                cmp     #$FF
                beq     6.f             indien $FF dan einde v.d. lijst
                sta     sdrive
                inc     pacount
                bmi     6.f
                lda     asave
                ldy     ysave           oude waarden van a en y terugzetten
                ;
                ldx     #$81            -\
                stx     opt              \
                stx     csufl             \__dit deel is exact als in dos65
                ldx     #$88              /
```

RAMTEST voor DOS65

In een grijs verleden stond er in de μP Kenner een RAMtestprogramma, geschreven voor de Octopus65. Nu heeft een DOS65 machine geen RAM-test: het is zelfs zo, dat IO65 ervan uitgaat, dat het RAM functioneert! Daarom onderstaand programma. Kunt u uw RAM ook eens testen.

Frank Vandekerkhove

```
; file              RAMTEST.mac
; program     RAMTEST
; function    test specific RAM range
; usage             RAMTEST
; by                Frank Vandekerkhove
;                   Sint-Michielsstraat 4
;                   B-2789 Verrebroek-Beveren
;                   + 32 3 773.27.94
; date              20 aug. 1989
; ref.              Elector april 1982
;                   De 6502 kenner nr. 50: RAM test for octo65 by M. Lachaert

                lib     dvar.mac

                org     $200

main            jmp     ramt
                fcc     $c8,$c5,$cc,$d0
                fcc     $4c,$4c,$4c
hlpinfo         fcc     '\rUtility to test a RAM area.'
                fcc     '\rSyntax: RAMTEST start,end (address in hex)'
                fcc     '\rNo options'
                fcc     '\rAbbr.: RAMT',0

ramt            ldx     #$00                    hex parms without +
                jsr     spar                    get parameters
                fcc     t1,t2,0                 two parms
                bcc     1.f
                jmp     ermes                   print error and back to caller

1               jsr     print                   clear and tittle
                fcc     '\f\Ei RAMTEST from $',0
                lda     t1+1
                jsr     hexout
                lda     t1
                jsr     hexout
                jsr     print
                fcc     ' to $',0
                lda     t2+1
                jsr     hexout
                lda     t2
                jsr     hexout
                jsr     print
                fcc     ' \r\r Up \En',0    test from start to end address
                jsr     zero                    zero in test area
```

Fig. 1: sourcetekst van RAMTEST.MAC

```
            jsr     setpnt                  set pnt equal to start address

2           jsr     walk                    test each bit of a cell
            bne     4.f
            lda     #$ff                    test pattern for double addressing
            sta     [t3],y
            jsr     incpnt                  inc the actual address
            bcs     2.b                     next cell to test bits

            jsr     print
            fcc     '\r\Ei Down \En',0 test from end to start
            jsr     zero
            ldx     t2
            stx     t3
            ldx     t2+1
            stx     t3+1

3           jsr     walk
            bne     4.f
            lda     #$ff
            sta     [t3],y
            jsr     decpnt
            bcs     3.b

            jsr     print
            fcc     '\r\Ei Succesfull test. \En',0

exit        jsr     crlf
            jmp     crlf

4           jsr     print
            fcc     '\rError at $',0
            lda     t3+1
            jsr     hexout
            lda     t3
            jsr     hexout
            jsr     print
            fcc     '\rRestart/Continue/Quit ? (R/C*/Q) ',0
            jsr     in
            jsr     loupch
            cmp     #'Q                     quit ?
            beq     exit
            cmp     #'R                     restart ?
            beq     5.f
            jsr     incpnt                  inc. pointer to next cell
            ldy     t3                      restart test at this address
            lda     t3+1
            sty     t1
            sta     t1+1
5           jmp     1.b
;
;*** SUBROUTINES ***
;
zero        jsr     print
            fcc     '\r$00 in area.',0
            jsr     setpnt                  fill test area with $00
            ldy     #$00
```

EP revisited

Het is alweer een tijd geleden dat in dit blad iets te lezen viel over de DOS65 EPROMprogrammer. Dat is een goed teken: kennelijk werken de gebouwde programmers probleemloos en naar tevredenheid. Toch drie puntjes die even de aandacht verdienen.

LET OP: 30 Volt vanaf de bus

Diegenen die dit geprobeerd hebben bliezen hun CRTC kaart gedeeltelijk op. Allereerst excuses hiervoor. Wat was het geval? Pin 31c van de bus is volgens de Elektuurspecificaties vrij. Dus dat lijntje werd gekozen om de +30V voor EP via de bus aan te bieden. Elektuur heeft ooit ook een Z80-bus bedacht, en op die bus wordt pin 31c wel gebruikt. De CRTC-kaart heeft een universele interface voor zowel de 6502 als de Z80, en op pin 31c is een gate van een 74LS00 aangesloten. En die ontploft als je er 30 Volt op zet... Op de andere kaarten is pin 31c niet aangesloten.
Er zijn twee mogelijkheden om deze ramp te voorkomen als u +30V over de bus op EP wilt aansluiten:

1. Het baantje op de print van de CRTC-kaart aan pin 31c onderbreken. Het zit aan de koperzijde van de print.
2. Pin 9 van IC4 doorknippen of naar buiten buigen als IC4 in een voetje zit.

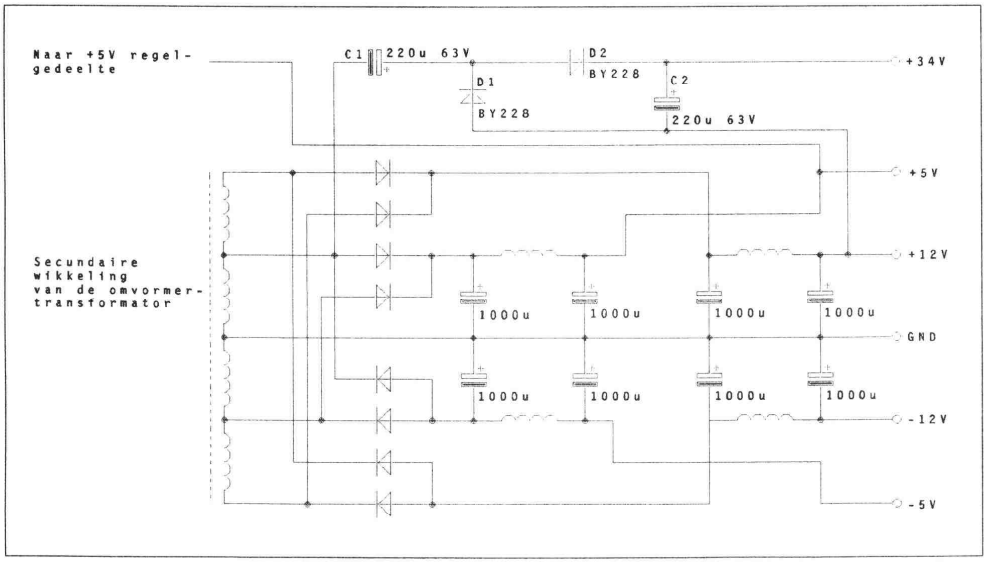
Tekenfoutje in het schema

In het gepubliceerde schema van EP is in de busconnector aangegeven dat het signaal RAM R/W op pin 31c zit. Dit is onjuist. Het zit op pin 31a. De print is goed. Alweer die pin 31c...

Suggestie voor +30 Volt

Ondergetekende gebruikt een ontmantelde PC-voeding om zijn 6502-wonder te voeden. Met een kunstje (4 onderdelen) kan uit zo'n voeding ook +34V getoverd worden. In de afbeelding is een deel van een PC-voeding getekend. Nagenoeg alle voedingen zijn op deze manier opgebouwd: de secundaire van de omvormertrafo is een dubbele wikkeling, met twee aftakkingen op 5/12. Het middenpunt is de nul. Door nu de aftakkingen dubbelzijdig gelijk te richten worden de vier gebruikelijke uitgangsspanningen gemaakt. De truc is nu het punt op te sporen waar de +5V wordt gelijkgericht. De twee gelijkrichtdioden voor de +5V zijn gemakkelijk te vinden: ze zitten met z'n tweeën in 1 behuizing die op een TO-3P transistor lijkt. De behuizing zit naast de twee schakeltransistors op de koelplaat in de voeding. Door de componenten D1, D2, C1 en C2 toe te voegen wordt er een spanning van +34 Volt gefabriekt. Is de spanning te hoog, dan kunnen de anode van D1 en de min van C2 ook op +5V worden aangesloten. Neem voor de elco's goede exemplaren: er lopen behoorlijke stromen bij een frequentie van circa 20 kHz. Goedkope elco's vinden dat helemaal niet leuk.

Nico de Vries



Afb. 1: uitbreiding PC-voeding voor +34 Volt

Een 1 Mbyte RAMkaart voor DOS65

De eerste reactie zal wel zijn: die was er toch al? Dat klopt. Er bestaat een 1 Mbyte RAMkaart voor DOS65 met dynamische 256kx1 RAMs die meestal wordt aangeduid met "de virtual diskkaart". Deze kaart wordt door enkelen met DOS65 gebruikt als RAMdisk. Dit was echter niet het oorspronkelijke doel van de kaart. De virtual diskkaart was namelijk eigenlijk bedoeld voor DOS65 V3.00 en hoger. Deze versie van DOS65 heeft namelijk multi-tasking. En om een beetje normaal met multitasking te kunnen werken, moeten alle actieve taken tegelijkertijd in het geheugen staan. En dan is de normaal beschikbare 56 kbyte veel te weinig. Vandaar die virtual diskkaart.

Problemen

Die waren er helaas ook, met die virtual diskkaart. Als eerste het aantal IC's. Alleen de 1 Mbyte RAM vergde al 32 16-pin IC's. Compleet met refresh logica, de generatie van de extra adreslijnen en de adresmultiplexers werd het aantal zelfs meer dan 50. En dat past niet of nauwelijks op een Eurokaart, als je tenminste ook nog ruimte wilde overhouden voor de bedrading. Dat wordt dubbel duidelijk als je een van weinige gebouwde kaarten ziet: met ontkoppelcondensatoren in de voetjes, en uitgevoerd in wire-wrap of road-runner.

Probleem twee was de refresh: er was nu plotseling een 65C02 nodig want de kaart gebruikte de RDY-lijn van de CPU voor de refresh. En omdat de NMOS 6502 niet duldt dat die lijn op willekeurige momenten wordt geactiveerd, werd een CMOS exemplaar verplicht. Op zich niet zo erg: de meesten hebben toch een CMOS 6502 in hun machine. Maar technisch mooi is anders.

Probleem drie waren de twee RAMmetjes die voor de adresvertaling moesten zorgen. Dat waren 74(LS)189's, in TTL uitgevoerde 16x4 RAMs. Ze werken uitstekend, maar als je naar een onderdelenboer gaat is het antwoord: "Hebben we niet, en ik kan ze ook niet bestellen." Niet verkrijgbaar dus.

Het laatste probleem was de snelheid. Er is ooit beloofd dat de virtual diskkaart op 2 MHz zou kunnen werken. En misschien lukte dat ook nog wel, maar met refresh, de uitgebreide adresdelay vanwege de vertaalslag en de toen niet al te snelle DRAMs was 2 MHz eigenlijk iets te hoog gegrepen, zeker als het technisch verantwoord in elkaar moest zitten.

Het netto resultaat was ernaar: de ware liefhebbers investeerden tijd en (veel) geld in de kaart en bouwden er een op gaatjesbord. Die kaart werd gebruikt als RAMdisk. Omdat de kaart nooit op print werd gezet kwam DOS65 V3.00 er niet. En dat is eigenlijk toch jammer.

Nieuwe ideeën

Voor DOS65 heeft ooit een werkgroep bestaan. Een van de dingen die uit deze werkgroep voortsproot is dat er een vervangend ontwerp voor de virtual diskkaart met DRAMs moest komen. Ondergetekende kreeg de opdracht om een nieuw ontwerp te bedenken, dat in ieder geval op een Eurokaart moest passen. Na een middag brainstormen met de groep werden de uitgangspunten:

- De kaart moet op een Eurokaart passen.
- De kaart moet zonder problemen op 2 MHz kunnen werken. Sneller mag ook natuurlijk.
- Er mogen alleen goed verkrijgbare onderdelen op de kaart voorkomen.

Deze gegevens plus nog wat andere ingevingen worden in een grote pot gestopt, goed geroerd en gaargekookt.

Hiermee werd aan de slag gegaan. Inmiddels was de techniek weer een stukje voortgeschreden. Grotere geheugens waren al heel normaal. De 1 Mbit DRAM was al heel gewoon, en 1 Mbit SRAMs waren aangekondigd respectievelijk mondjesmaat verkrijgbaar. Als de snelheidszijde werd ook winst geboekt: zo was er van Toshiba een 2kx8 RAM verkrijgbaar met een toegangstijd van 50 nanoseconde, terwijl 35 ns was aangekondigd. Deze gegevens plus nog wat andere ingevingen uit de DOS65 werkgroep worden in een grote pot gestopt, goed geroerd en gaargekookt. Dit kwam eruit:

Statisch RAM

Er worden statische geheugens gebruikt. Naar keuze kan de kaart worden benut met de toen gangbare 32kx8 SRAMS of de nieuwe 128kx8 SRAMs. In het eerste geval is de maximale capaciteit van de kaart 256 kbyte, en het tweede 1 Mbyte. Statische geheugens hebben een aantal voordelen:

- Geen refresh, en dat scheelt een hap logica op de kaart.
- Inherent snel. Een SRAM heeft tegenwoordig een toegangstijd van 150 nanoseconde, of zelfs

sneller. Langzamere geheugens worden gewoon niet gemaakt!
- Indien CMOS SRAMs worden gebruikt, is eventueel battery backup mogelijk, zodat hele nieuwe mogelijkheden ontstaan, zoals het booten van DOS vanuit RAM.
- SRAMs plegen "byte-wise" georiënteerd te zijn. Dit heeft het voordeel, dat de kaart niet onmiddellijk volgestampt hoeft te worden, maar naar behoefte en budget gevuld kan worden. De minimum hoeveelheid is namelijk 64kbyte RAM: dan heb je je normale systeem weer beschikbaar. Bij DRAMs moet je minimaal een rijtje van 8 aanschaffen.
- Byte-wide SRAMs hebben bijna dezelfde pinout als EPROMs. Het is dus eventueel mogelijk om de systeemdisk of DOS65 zelf in EPROM te bakken en 1 van de SRAMs door die EPROM te vervangen. Ook dan is booten vanaf de RAMkaart mogelijk.

SRAMs hebben ook een nadeel: ze zijn per bit duurder dan DRAMs. De werkgroep vond dat de voordelen echter ruimschoots tegen dit nadeel opwogen.

De truc is namelijk dat er meer adreslijnen gemaakt moeten worden dan de 65(C)02 levert.

Cachegeheugen

Het probleem van niet verkrijgbaar zijn van de 74(LS)189's werd verplaatst: een onderzoek leerde, dat 2kx8 en zelfs 8kx8 SRAMs met superlage toegangstijden redelijk verkrijgbaar waren uit verschillende bronnen. Normaal worden deze geheugens gebruikt als cachegeheugen voor snelle computers. Deze RAMs waren voor deze toepassing eigenlijk veel te groot. 2kx8 35 nanoseconde zit echter in een "skinny" 24-pin behuizing en neemt minder plaats op de print in als twee 74(LS)189's. Ook de prijs was ten opzichte van de TTL-RAMs beschaafd: de twee 74(LS)189's waren duurder dan het veel te grote snelle 2kx8 RAM. Dus de TMM2018-35 (van Toshiba) erin. UMC, Cypress en Fujitsu maken equivalenten, dus dat loopt wel los.

Werking van de kaart

Eerst even een uitstapje. Het is wellicht nuttig om eerst de werking van de oude virtual diskkaart wat duidelijker te maken. Met name het deel dat de adresvertaling verzorgt. De truc is namelijk dat er meer adreslijnen gemaakt moeten worden dan de 65(C)02 levert. Voor 1 Mbyte zijn er namelijk 20 adreslijnen nodig, terwijl de 65(C)02 er maar 16 levert. Dit werd geregeld door die moeilijk te krijgen 74(LS)189's. De bovenste 4 adreslijnen van de 6502 worden via een multiplexer aangesloten op de adreslijnen van de 74LS189's. De andere helft van de multiplexer laat de adreslijnen A0 tot en met A3 door. De databus wordt met de datalijnen van de 74LS189's verbonden.

In normaal bedrijf vormen A12 tot en met A15 en adreslijnen van de 74LS189's. In de 74LS189's wordt gelezen. De datalijnen (8 in getal) vormen nu de adreslijnen voor het achterliggende blok van 1 Mbyte RAM. A0 tot en met A11 komen rechtstreeks van de processor en duiden een 4 kbyte blok aan. A12 tot en met A19 voor het RAM worden bepaald door de 74LS189's. Deze RAMmetjes bepalen zo voor iedere mogelijke combinatie van CPU-A12 tot en met CPU-A15 een set nieuwe adreslijnen A12 tot en met A19 voor het RAM. Weliswaar is nog steeds slechts 64 kbyte van de 1 Mbyte direct te adresseren, maar door de 74LS189's opnieuw te beschrijven kan een andere set van 16 4 kbyte blokken worden aangewezen.

Dat beschrijven gaat op een bijzondere manier. De 74LS189's kunnen worden geadresseerd op adres $FF00 tot en met $FF0F. Hier zit bij DOS65 de IO65 EPROM. In deze EPROM kan alleen gelezen worden. In de 74LS189's kan alleen worden geschreven. Op het moment dat er geschreven wordt op adres $FF0X klapt de adresmultiplexer voor de 74LS189's om van A12-A15 naar A0-A3 en wordt de databus aangeboden. De processor kan dus het bereikbare blok opnieuw definiëren door 16 bytes weg te schrjven op $FF00-$FF0F, waarbij de 16 bytes A12 tot en met A19 voorstellen voor het RAM, passend bij de 16 mogelijke combinaties van A12 tot en met A15 zoals gegenereerd door de CPU.

De rest van de RAMkaart is normaal, alleen de generatie van de bovenste 8 adreslijnen is bijzonder, zoals hierboven uit de doeken is gedaan. Tot zover het uitstapje.

Task switching

Het laden van een taak bestaat uit drie stappen:
- Redt de huidige registerset in een stuk RAM waarin de taskswitcher zijn huishouding bijhoudt.
- Laadt een nieuwe tabel van 16 aanwijzers in de 74LS189's.
- Laad in het nu beschikbare RAMbereik (theoretisch 64 kbyte, in de praktijk 56 kbyte) het gewenste programma.

- Laadt de oude tabel weer terug in de 74LS189's, zet de programmateller en andere registers weer goed en vervolg de vorige taak.

Het wisselen van een taak gaat nu als volgt:

- Redt de huidige registerset in een stuk RAM waarin de taskswitcher zijn huishouding bijhoudt.
- Laad de tabel voor de tweede taak in de 74LS189's. Het programma van de nieuwe taak is nu in de CPU adresruimte gemapt.
- Laadt de CPU registers met de registerwaarden die bij de nieuwe taak horen.

De taskswitcher van DOS moet nu twee zaken onthouden:

- De huidige registerwaarden van iedere taak die niet actief is.
- De setting van de 74LS189's voor iedere taak.

Het kon eenvoudiger.

Tasknumber register

Hiervoor is al aangegeven, dat de 74LS189's werden vervangen door een 2kx8 SRAM, dat eigenlijk veel te groot is. Er bleven 7 adreslijnen ongebruikt. De brainwave was deze: knoop aan deze 7 adreslijnen een register dat door de CPU bereikt kan worden. Dit register kreeg de naam tasknumber register. Het laden van een taak is nu iets meer werk:

- Redt de huidige registerset in een stuk RAM dat de huishouding van de taskswitcher bijhoudt. Dit is hetzelfde als vroeger.
- Schrijf het nummer van de laden taak in het tasknumber register. Er wordt nu een blokje van 16 bytes in de 2kx8 RAM aangewezen. Dit is nieuw.
- Laadt een nieuwe tabel van 16 aanwijzers in de 2kx8 SRAM. Dit is hetzelfde als vroeger.
- Laad in het nu beschikbare RAMbereik het gewenste programma.
- Schrijf het nummer van de vorige taak terug in het tasknumber register. De oude taak is nu met 1 schrijfoperatie terug in de adresruimte gemapt! Zet de programmateller en andere registers weer goed en vervolg de vorige taak.

Taak wisselen wordt ook eenvoudiger, en dus ook sneller:

- Redt de huidige registerset in het huishoud-RAM van de taskswitcher. Dit is hetzelfde als vroeger.
- Schrijf het nummer van de nieuwe taak in het tasknumber register. Het programma van de nieuwe taak is nu in de CPU adresruimte gemapt met 1 enkele schrijfinstructie.
- Laadt de CPU registers met de registerwaarden die bij de nieuwe taak horen.

Het tasknumberregister levert dus de volgende voordelen op:

- De taskswitcher hoeft niet langer de complete adresmapping van iedere taak te onthouden. Wel moet de switcher het geheugen zodanig beheren, dat er geen overlappingen ontstaan. Het onthouden hoeft niet in sets van 16 bytes, maar bijvoorbeeld als begin- en eindadres in de vorm van RAM-A12 tot en met RAM-A19.
- Taakwisseling verloopt aanzienlijk sneller: 1 schrijfoperatie in het tasknumber register. Er hoeft niet steeds opnieuw de complete mapping geladen te worden.

Het ontwerp in de praktijk

Het tasknumberregister werd dus toegevoegd. Het wordt aangesproken op adres $FF10. Het is net als de 2kx8 SRAM alleen te beschrijven. Zolang de waarde van dit register niet veranderd wordt, werkt de kaart net zo als de oude virtual diskkaart. De adresmultiplexer voor het mapping RAM (de 2kx8 SRAM) werd in een PAL gestampt, evenals een stukje adresdecodering. Na enige optimalisatiepogingen telde het complete ontwerp 17 IC's, waarvan 8 SRAMs (mapping RAM niet meegerekend).

Omdat het ontwerp zo eenvoudig geworden was, is een goede werking op 2 MHz in combinatie met de van nature snelle SRAMs zeker gegarandeerd. De belangrijkste snelheidsbepalende component is naast de SRAMs zelf het mappingRAM, dat 35 nanoseconde vertaging in de adresgeneratie oplevert.

Het ontwerp werd uitgewerkt voor zowel 32kx8 SRAMs als 128kx8 SRAMs. Het verschil werd opgelost in de PAL, en een paar jumpers. Verder zat het ontwerp zodanig in elkaar, dat er meerdere kaarten in 1 systeem mogen voorkomen. De enige limiet was de hoeveelheid SRAM: 1 Mbyte is het maximum want dan zijn de adreslijnen op.

De ijskast

Er werd een prototype opgezet. Halverwege het testen had de werkgroep een onderzoek gedaan naar de behoefte aan DOS65 V3.00 en een vervanger voor de virtual diskkaart. Die bleek buiten de werkgroep zelf exact nul te zijn. En daarmee verdween heel DOS65 V3.00 in de ijskast. Er is 1 systeem dat met V3.00 draait: het systeem van Ad Brouwer, de bedenker van DOS65. Compleet met SCSI harde schijf.

Voorlopig had iedereen het druk met PC's, klonen, MS-DOS, nog meer geheugen, nog grotere harde schijven en nog lagere prijzen. Dankzij die lagere prijzen werd er niet meer geknutseld. Men begon te vergeten hoe dat eruit zag. Dat duurde zo'n twee jaar voort. Toen kwam er een voorstel op een vergadering om DOS65 V3.00 nieuw leven in te blazen.

Een nieuwe werkgroep werd gevormd. Er werden afspraken gemaakt wie wat zou doen. De SRAM-kaart werd weer van stal geplukt. We hadden het druk. Op het werk. Met PC's. Met KGN-68k. Netto resultaat: geen belangstelling buiten de werkgroep zelf, en geen vooruitgang en vooral: geen tijd. IJskast nummer twee werd geopend, en DOS65 V3.00 ging daar moeiteloos in.

Dit najaar verscheen de voorzitter met een plan: DOS65 V3.00 moest maar eens opnieuw bekeken worden. Want slechts 1, zeer uitgebreid project (KGN-68k) was een te smalle basis. Inmiddels was de werkgroep na een bijeenkomst gepolst, en een ieder had er vrede mee, dat de klus gedaan zou worden voor jezelf en jouw mede-werkgroeplid, en niet voor 150 leden die allemaal DOS65 hebben. Het knutselen was terug...

Het knutselen was terug...

Nieuw leven voor de SRAM-kaart

Dus werd de SRAM-kaart weer van stal gehaald. Eerst werd de kaart aan de laatste stand van de techniek aangepast. Dat hield het volgende in:

- De versie met 32kx8 SRAMs komt te vervallen. Inmiddels zijn 128kx8 SRAMs per bit goedkoper dan hun 32kx8 broertjes. Dit scheelde drie jumpers en twee verschillende PALversies. Het ontwerp bevat nu helemaal geen jumpers meer.
- Het mappingRAM wordt nu een 8kx8: deze maat RAMs is thans populair vanwege het gebruik als cacheRAM in snelle 386- en 486-PC's. Verkrijgbaarheid is dus geen probleem, zelfs niet in snelheden van 25 of 20 nanoseconde. Toepassing van 2kx8 blijft mogelijk met een paar draadjes. Voordeel is nu dat er 256 verschillende mappings mogelijk zijn, en dat alle taaknummers nu te gebruiken zijn.
- Het ontwerp is nog verder vereenvoudigd. Zo kon er nog 1 gate worden weggelaten. Ook werden er LS541's in plaats van LS244's en een LS573 in plaats van een LS373 gebruikt. Deze componenten hebben de in- en uitgang tegenover elkaar in plaats van naast elkaar. Dat is gemakkelijker met print ontwerpen.

Verder is er serieus rekening gehouden met het feit dat de kaart op een enkelzijdige Eurokaart moet passen. Dat zal waarschijnlijk wel een toer worden, en ook niet zonder draadbruggen lukken, maar het is het probleren waard. Dubbelzijdig kan altijd nog.

De huidige versie

Het schema en de PAL-inhoud voor de huidige versie staan hierna afgebeeld. De diverse delen zijn eenvoudig te herkennen:

- IC2 is het tasknumber register.
- IC4 is het 8kx8 mappingRAM.
- IC3 is de adresmultiplexer voor het mappingRAM en de adresdecoder voor zowel het mappingRAM als het tasknumber register. De adresdecoder is nog uitgebreid met IC9. Alle adressen worden compleet uitgedecodeerd. De PAL is gegeven als een GAL20V8, maar een PAL/GAL22V10 kan ook gebruikt worden.
- IC5 vormt het datatoegangspad naar het mappingRAM.
- IC6 en IC7 zijn doodordinaire buffers voor de adreslijnen naar de RAMs.
- IC8 is de chip-select decoder voor de RAMs.

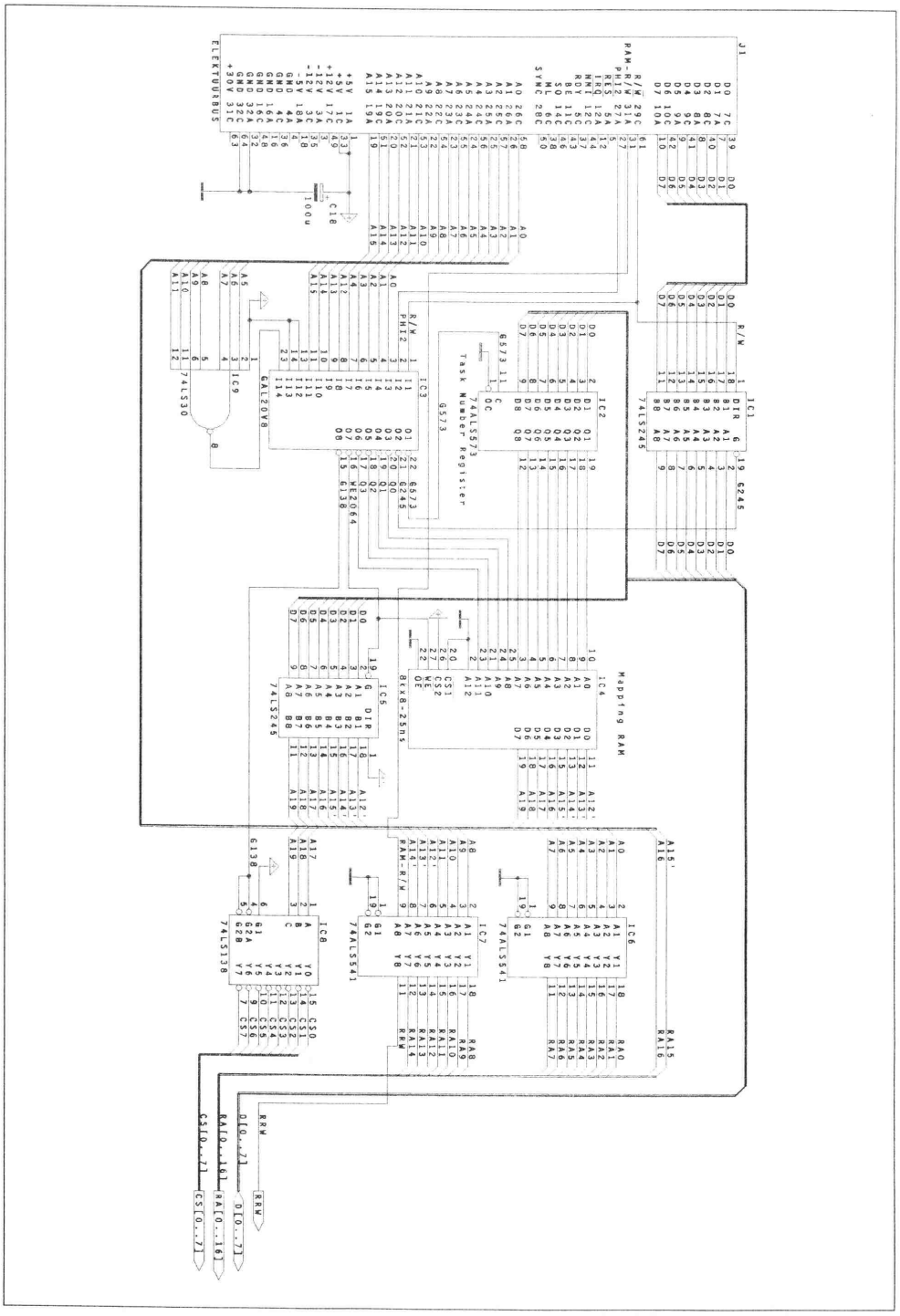
De meeste buslijnen worden belast met 1 TTL-load. PHI2 en A0 tot en met A11 zien twee TTL-loads.

Zowel de CS- als de WE-lijn van de RAMs worden geAND met PHI2 (R/W op de CPU-kaart, CS in de PAL via G138), zodat de RAMs per access het adres inclocken, en er sprake is van WE-controlled write. De delay op de CS-lijn is namelijk langer.

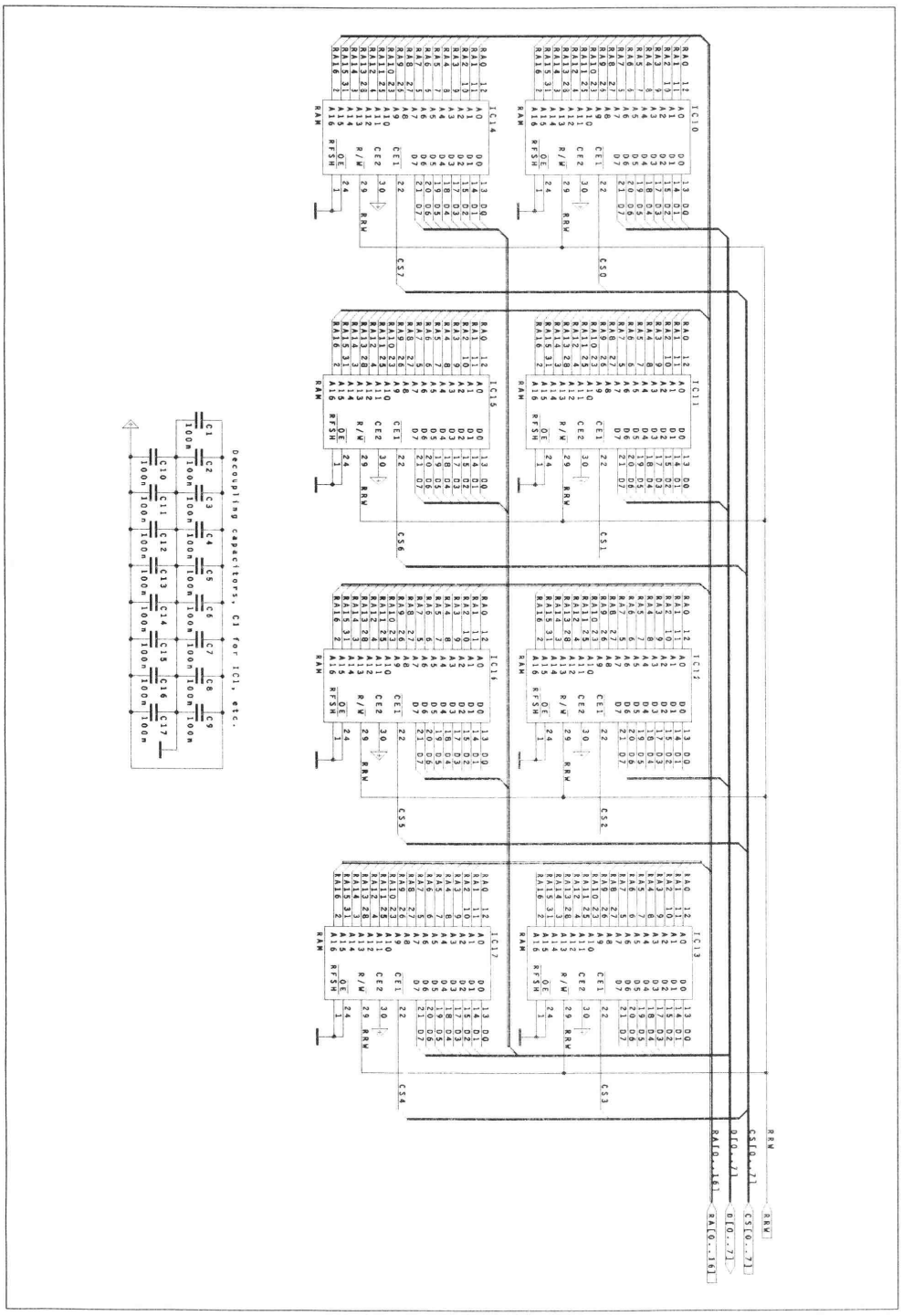
Nu komt het hevige stuk: de print! Gezien de ervaringen met EP doe ik hier nog geen uitspraken over...

Nico de Vries

Opmerking bij het schema: de 1Mbit RAMs zijn getekend als Pseudo Statische RAMs. Gewone CMOS SRAMs kunnen ook gebruikt worden: van deze IC's is pin 1 niet aangesloten.



Afb. 1: stuurgedeelte van de 1 Mbyte RAMkaart



Afb. 2: geheugengedeelte van de 1 Mbyte RAMkaart

Voortgang DOS65

Gisteren (17 april) is de werkgroep DOS65 weer bijeen geweest. Er is weer gebrainstormd over hoe het verder moet. De sfeer was zoals gebruikelijk goed, en de zin om verder te gaan ontbrak evenmin.

De belangrijkste ontwikkeling van de laatste maand is dat er een dubbelzijdige print is getekend van de 1 Mbyte RAMkaart uit het vorige nummer. De print staat op deze bladzijden afgebeeld. Erwin Visschedijk heeft ondergetekende enige tekenvaardigheden bijgebracht, en samen hebben we de print getekend. Speurders die het vorige nummer erbij pakken zullen een paar verschillen ontdekken: de nummering van de IC's is veranderd en er is een batterijtje met bijbehorende chip verschenen. De kaart is nu namelijk ook voorzien van battery-backup, met behulp van de Dallas DS1210 "Nonvolatile controller".

Wat absoluut niet is gelukt is de print enkelzijdig te houden. Dat werd al duidelijk toen we het tekenpakket hadden verteld hoe groot de print moest worden (Eurokaart) en de netlist was ingevoerd: het leek er bijna op dat de print overvol zou worden. Dat bleek mee te vallen, maar om te spreken van een totaal lege print is anders.

De volgende stap is het verder debuggen van het prototype, en het bestellen van een aantal proefprinten. Als deze werken kunnen we aan de slag met DOS65 V3.00. De werkgroep heeft inmiddels de volgende taakverdeling afgesproken:

Tonny Schäffer:	Contact leggen met Ad Brouwer.
Jaap Prenger:	Napeinzen over de combokaart, een kaart met een SCSI harddisk interface, een real time clock, reset- en NMI-generators en misschien wat extra RS-232 poort(en).
Antoine Megens:	IO65 aanpassen.
Henk Speksnijder:	IO65 aanpassen.
Nico de Vries:	1 Mbyte RAMkaart voltooien.

Kortom, er is werk genoeg aan de winkel. Binnenkort hoort u wat het kostenplaatje voor de RAMkaart gaat worden, en hoe u aan de print, de PAL en eventuele andere moeilijke onderdelen kunt komen. Tot de volgende keer.

Nico de Vries

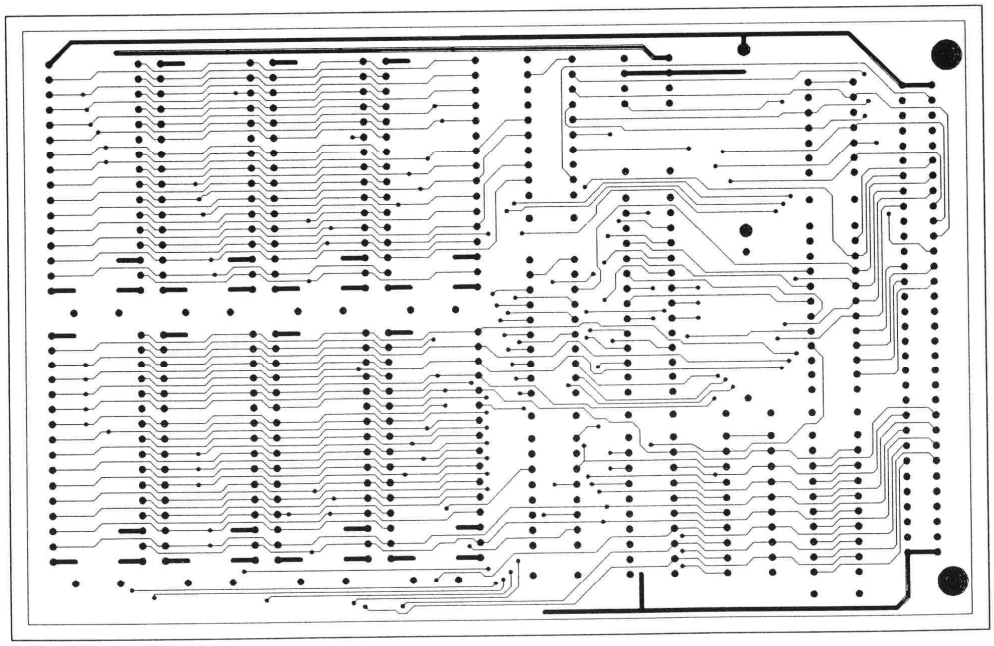


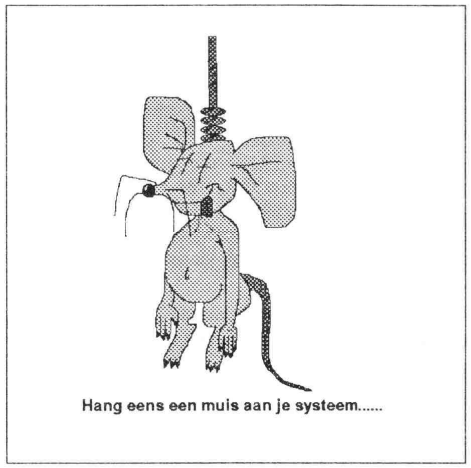
Fig. 1: print, sporen aan soldeerzijde

Hang eens een muis aan je DOS65

De seriële poort van de DOS65 is er niet alleen geschikt voor modem verkeer of een link naar andere machines zoals MS-DOS, maar is ook uitermate geschikt om een muis aan te sturen. De populairste huis-tuin en keuken muis is toch wel de Microsoft compatibele seriële muis. Bij 90% van de PC's zit er zo'n muis aangesloten op een van de COM poorten, bij machines die Windows draaien kun je (bijna) niet zonder.

Zoals de meesten van jullie wel weten dient er een muis driver geladen te worden om met de muis te kunnen werken (onder Windows is die driver 'ingebakken' in het operating system). Meestal heet dat programma MOUSE.COM of zoiets.

Maar wat doen die programma's precies en wat komt er eigenlijk uit zo'n muis? Voorwaar vragen waar je als rechtgeaarde hobbyist behoorlijk mee in je maag kunt zitten. Bovendien, zo'n PC ken je eigenlijk (nog) niet zo goed, of sterker nog, je wilt hem niet eens leren kennen. Maar boven op zolder staat nog je ouwe trouwe DOS65 bakkie en dat ken je door en door, want meestal helemaal zelf in elkaar gefröbeld.



Hang eens een muis aan je systeem.....

Maar boven op zolder staat nog je ouwe trouwe DOS65 bakkie

Aan de slag

Oké, wat hebben we nodig? Een (werkend) DOS65 systeem, een Microsoft compatibele seriële muis, mogelijkerwijs wat RS-232 kabelwerk + soldeerbout, een regenachtige middag en niet te vergeten jezelf.

Standaard is de DOS65 seriële poort van het verkeerde geslacht, vergeleken met zijn MS-DOS broertje (een mannetje). Het beste kun je dus de soldeerbout ter hand nemen (jazeker Nico, het solderen is terug!) en de connector van de DOS65 poort wijzigen in een male 25 polige D connector. Voordeel? Voortaan kun je standaard kabels gebruiken. Voor mensen die dat te ver gaat zijn er allerlei handige geslachtwijzigaars (in slecht Nederlands, of in goed Frans "gender changer") in de handel of eenvoudig zelf te maken. Zorg in ieder geval dat de volgende lijnen zijn aangesloten: TXD, RXD, GND en DTR.

	9-polig sub-D	25-polig sub D
TXD	pin 3	pin 2
RXD	pin 2	pin 3
GND	pin 5	pin 7
DTR	pin 4	pin 20

Let wel, het beestje krijgt de zo broodnodige voeding niet, ik herhaal, niet van zo nu en dan een stukje kaas! Deze wondertjes der techniek nemen genoegen met slechts enkele mA (meestal minder dan 10). In ieder geval geen probleem voor de meeste RS-232 driver IC's (meestal de bekende 1488/1489).

Na wat uitprobeersels kwam ik er al snel achter dat de baudrate op 1200 Bd moest staan. De correcte setting is 1200,8N1 (1200 Baud, 8 bits woord, no parity en 1 stop bit) ook veel gebruikt bij de meeste BBS-en (uit vroeger tijden, want tegenwoordig is 1200Bd niets meer met al die high speed modems van 14K4 of hoger. Maar dit even terzijde).

Ok, die muis hangt eraan en wat nu?

Wat komt er nu eigenlijk uit zo'n muis? En moet je ook antwoord geven? Of moet je er eerst om vragen? Kortom, alweer vragen waar je je suf naar kunt zoeken in allerlei goed bedoelde, maar zich meestal op de vlakte houdende, vakliteratuur.

Uiteindelijk ben ik er toch achter gekomen, een muis pakketje bestaat uit een zogenaamd sync byte (kort voor synchronisatie) en dan 4 bytes die de X,Y beweging beschrijven. Het eerste byte heeft ALTIJD de vorm 10000xxx, waarbij de laatste drie bits de knoppen informatie bevatten. Hierbij geld dat het bit laag is als de knop is ingedrukt. Als er geen knoppen zijn ingedrukt is het sync byte dus 0x87.

Het intelligente beestje stuurt alleen dan een pakketje als er een situatie verandering optreedt. D.w.z. bij het indrukken van een of meer knoppen, bij het loslaten van knoppen of bij een verplaatsing van de muis wordt een volledig pakketje verstuurd.

De vier verplaatsing bytes zijn als volgt gegroepeerd: X1, Y1, X2, Y2. Deze bytes bevatten 8-bits signed integers, zodat de relatieve verplaatsing is uit te reken als volgt:

delta X = X1 + X2
delta Y = Y1 + Y2

Een voorbeeld, oh meester, wij smeken erom...

Maar genoeg droge theorie, het wordt tijd voor een praktisch voorbeeld. De volgende listing bevat een simpel muis test programma, dat je kunt gebruiken

```
;---------------------------------------------------------------------------
; filename      : mouse_test.mac
; version       : 1.02
; last edit     : 31-mar-1993
; author        : Antoine Megens
; module        : DOS65
; description   : mouse test
; copyright     : KGN, 1991,1993
;---------------------------------------------------------------------------
; edit history:
;   date        name    ver. description
; 15-may-1991   am      1.00 started
; 24-jul-1991   am      1.01 new line on sync byte
; 31-mar-1993   am      2.00 added some explanation and graceful break
;---------------------------------------------------------------------------
;
; Use this extremely simple program to test if there is life in your mouse
; Connect a standard MicroSoft compatible serial mouse to the DOS65 serial
; port. This test program shows the raw mouse data.
;
              org     $a000
;
; hardware addresses DOS65 ACIA
;
aciasr        equ     $e131           DOS65 6551 ACIA status register
aciacmd       equ     aciasr+1        DOS65 6551 ACIA command register
aciactl       equ     aciasr+2        DOS65 6551 ACIA control register
;
; DOS65 entries
;
prspace       equ     $c032           print space
hexout        equ     $c038           print A hex
crlf          equ     $c02f           print newline
print         equ     $c03b           print NULL terminated string
inpx          equ     $f009           read from device X (X=3 is serial port)

rdmouse       ldx     #3
              jmp     inpx

mouststst     jsr     print
              fcc     "\fSimple mouse test program\r\r"
              fcc     "Connect MicroSoft compatible serial mouse to DOS65 serial port\r"
              fcc     "Use ctrl-C to abort or press all three mouse buttons\r\r",0
```

Fig. 1: listing voor muisdriver onder DOS65

```
        jsr     iniacia         init ACIA 1200Bd,8N1
loop    jsr     rdmouse
        pha
        and     #%11111000
        cmp     #%10000000
        bne     1.f
        jsr     crlf
1       pla
        cmp     #%10000000      all 3 buttons down?
        beq     2.f
        jsr     hexout
        jsr     prspace
        jmp     loop
2       jsr     aciaoff
        jsr     print
        fcc     "Bye\r",0
        rts

iniacia lda     #%00000101      DTR low, IRQ enabled, no parity
        sta     aciasr          reset ACIA (dummy write to status)
        sta     aciacmd         send command
        lda     #%00011000      1200 Bd internal, 8N1
        sta     aciactl         send control
        rts
aciaoff lda     #%00000010      DTR high, IRQ disabled, XMIT off
        sta     aciasr          reset ACIA (dummy write to status)
        sta     aciacmd         send command
        rts
        end     mouststst
```

om de muis pakketjes zichtbaar te maken op het scherm. Werkt het bij jou niet controleer dan de aansluitingen van de muis. Veel muizen kunnen omgeschakeld worden op een andere mode met een schakelaar, probeer beide modes. Ook komt het nog wel eens voor dat de muis niet goed reset. Los dit op, door de muis uit de RS-232 poort te nemen en opnieuw in te pluggen. Probeer in het uiterste geval de muis uit op een PC met een of ander modem communicatie pakket (zoals Procomm, Telix o.i.d.) op 1200Bd, 8N1. Het scherm moet dan allerlei "troep" weergeven als je met de muis beweegt, of een knop indrukt. Werkt het op de PC wel, maar op DOS65 niet, controleer dan de kabel. Ook een RS-232 break-box kan uitkomst bieden. Soms dondert de spanning voor de voeding van de muis teveel in elkaar mert andere woorden, de driver kan de stroom niet leveren. Maar genoeg van dit alles, mijne dames en heren.....(trom geroffel).....(nog meer trom geroffel).....DE LISTING:

Bovenstaande heb ik inmiddels "ge-upload" (Neerlandici, excusez le mot) op ons club BBS "The Ultimate". De file heet "m_test.mac". Heb je de file op je systeem gekregen (hetzij van het BBS, hetzij door monnikenwerk), dan kun je AS erop los laten. De resulterende .BIN file vervolgens op de gebruikelijke manier naar commando-file vertalen. Muis aansluiten, en opstarten die handel.

Tot slot

Volgende keer hoop ik iets te schrijven over een muis driver waarmee je de cursor over het scherm kunt bewegen. En als ik erg veel zin krijg, ga ik verder met een artikel over de menusoftware die we (Jaap Prenger en ik) besproken hebben op de bijeenkomst in Krommenie. Er bestaat inmiddels een demoversie van deze software die op ieder DOS65 systeem kan draaien, maar deze is gebaseerd op wat verouderde routines. Zodra ik weer wat tijd heb zal ik deze demo onder handen nemen en op ons aller BBS "The Ultimate" uploaden of een floppy sturen. Tot de volgende keer maar weer,

Antoine Megens

Als je zo een soort kasboek bijhoudt, krijg je een goed inzicht waar al het geld blijft. Bovendien ga je op sommige punten kritischer met je geld om waardoor je mogelijk minder geld uitgeeft. Een tweede voordeel is dat je voor je belasting-aangifte alle bedragen netjes op een rij hebt waardoor het invullen van het formulier een fluitje van een cent wordt (zeker als je dat ook nog met de computer doet).

Aangezien ik het iedereen aan kan raden zijn inkomsten en uitgaven bij te houden, is in tabel 1 een overzicht opgenomen van alle rubrieken die wij in het spreadsheet hebben opgenomen. Uiteraard kun je zelf rubrieken toevoegen, weglaten, samenvoegen of verder uitsplitsen. Ik heb in het voorbeeld ook enkele rubrieken die bij ons verder gespecificeerd zijn samengevoegd (o.a. de abonnementen en de lidmaatschappen). Probeer het zo op te zetten dat je ook op papier een overzichtelijk geheel houdt. Wij kunnen op twee pagina's net één kwartaal laten afdrukken.

Als kolommen hebben we de maanden van het jaar plus een kolom waar we de norm-bedragen in hebben staan. Voor de maanden die nog niet geweest zijn, wordt het normbedrag overgenomen. Maandelijks worden de normbedragen vervangen door de werkelijke bedragen. Verder hebben we een kolom voor het jaar-totaal per rij en een kolom waarin het gemiddelde (maand, kwartaal of jaar-) bedrag wordt uitgerekend. Deze bedragen worden volgend jaar uiteraard weer als normbedrag gebruikt.

Om precies het saldo van je giro te krijgen, moet je uiteraard een beginstand invullen. Dat kun je doen in de kolom "Norm" en in de rij Saldo giro-rekening. Verder moet je nog weten dat bij ons de nieuwe maand altijd begint op de datum waarop het salaris binnenkomt (de 28 ste). Het saldo in een maand geeft het saldo aan vlak voordat het salaris wordt bijgeschreven.

Mochten er mensen zijn die nog wat uitleg of verdere hulp nodig hebben, dan kunnen die uiteraard contact met mij opnemen.

Gert van Opbroek

DOS-65 uit de ijskast !

Dankzij of ondanks het mooie weer en met of zonder lentekriebels ben ik naar de bijeenkomst in Geldrop geweest. DOS-65 was reeds uit de ijskast gehaald en stond op tafel (of agenda) ter discussie. Wij "aanwezigen" moeten niet alleen denken aan de thuisblijvers, maar ook aan de nieuwe fijnproevers: wat is DOS-65? Wij weten dat wel en hebben, desnoods met draadjes, al een opvolger staan draaien. Willen we verder, dan moet er hardware aangeboden kunnen worden: nieuw ontwikkelde printen met misschien een moeilijk verkrijgbaar IC-tje uit het club-magazijn. De kracht van DOS-65 is de toegang (binnen het niveau van de hobby-sfeer) tot het laatste bitje van de hardware en de software. Hoewel ik nog niet alles tot op het bot ken, kan ik wel wat hardware met wat software bijeenvoegen. Bovendien kan ik nog met een universeelmeter en foutje opsporen en herstellen. Een MS-DOS machine wordt/is een tool bij het ontwikkelen van projecten. De een wil wat meer dan de andere, maar de meesten hebben geen nood aan een grafische kaart, een andere processor en dergelijke voor de DOS-65. Wel willen we dat alle Centronics-lijnen aanwezig zijn, dat er een tweede seriële poort is voor een muis, dat er wat meer attributen voor de video zijn, ... Wensen waren en zijn er nog genoeg, maar voor wie ? Moeten we met een inschrijving werken voordat er een project gestart wordt? Dat hoeft niet, want "het knutselen is terug" ... . We hebben plezier aan het ontwikkelen en het doorgeven van ons werk. Wie wil, kan meegenieten van DOS-65!

Een dosser die nog geen 65 is ...

Frank Vandekerkhove

Hang eens een muis aan je DOS65 (deel twee)

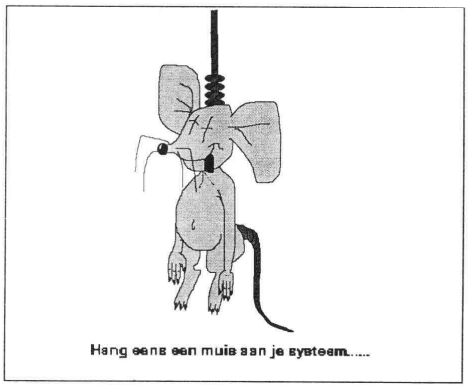
Het voorbeeld programmaatje uit deel een heeft natuurlijk weinig praktische waarde en geschreven volgens het KISS (Keep It Simple Stupid) principe. Ook als debug tooltje, "zit er leven in mijn muis?" heeft het wel enig bestaansrecht.

In dit deel een meer praktische toepassing van de muis data. De meest gebruikte toepassing van een muis, is het om de cursor op een gemakkelijke manier naar een willekeurige plaats op het scherm te bewegen. Zoals uit het vorige artikel al bleek stuurt de muis bij iedere beweging een pakketje data met de relatieve verplaatsing in de X en Y richting. Diegene die het demo programmaatje hebben uitgeprobeerd, zullen gemerkt hebben dat de grootte van de getallen afhangt van de mate van verplaatsing per tijdseenheid, in andere woorden (eerste grondbeginselen van de Natuurkunde) van de snelheid.

Nu dacht ik, simpele ziel, dat het een, zoals de Fransen zeggen, stukje grootmoeders cake zou zijn om deze X,Y verplaatsing over te zetten in een X,Y verplaatsing van de cursor. Helaas, helaas, dat viel even tegen.

Ten eerste is het beeldscherm niet verdeeld in een homogeen vlak in X en Y richting, immers in de X richting hebben we 80 locaties, en in de Y richting slechts 24 (of 25 met de status regel meegeteld). Een verhouding van 1 op 3 dus. De muis 'weet' dat niet en levert voor X en Y vergelijkbare data, d.w.z. je moet de Y richting op een of andere manier 'afzwakken'. Delen door 3 denk je? Helaas, heb ik geprobeerd. Je krijgt dan alleen beweging in de Y richting door met korte,heftige rukken de muis te verplaatsen. De mooie vloeiende cursor beweging die PC applicaties zoals bijvoorbeeld PC-Tools met eveneens een karakter georiënteerd scherm van 80x24, laten zien, begon me steeds meer te ergeren. Hoe deden die lui dat? Natuurlijk is de muis driver op de PC gigantisch (ca 50K) vergeleken met mijn gefröbel op DOS65 (enkele100-den bytes).

Na een aantal dagen vruchteloos worstelen met allerlei 'formules' om een mooie X,Y beweging te krijgen, besloot ik terug te grijpen op een oude truc. Tabellen!. Tabellen? Jazeker, tabellen. Tabellen zoals U en ik. Je vertaalt eenvoudig de X en Y data die uit de muis komt via een tabel in een X en Y verplaatsing voor de cursor. Door voor X en Y een andere tabel te nemen kun je zo de X en Y verschillen voor de cursor opheffen. Geniaal? Helaas, nog niet helemaal. Ook nu kreeg je of een te gevoelige muis, of een die met rukjes bewogen moest worden. De oplossing bleek even eenvoudig als voor de hand liggend. Begin de tabel met nullen, d.w.z. geen verplaatsing naar de cursor. Maar bewaar wèl de muis data. Deze data wordt dan de volgende keer bij de nieuwe muis data opgeteld en levert dan meestal wel een offset in de tabel die een verplaatsing naar de cursor oplevert. In dat geval wordt de muis data weer op nul gezet. Deze oplossing voldoet redelijk d.w.z. met een minimum aan bytes toch een acceptabel muis-cursor "gevoel". Om de tabel niet al te groot te maken, worden alleen de onderste 4 bits van de muis data gebruikt. Dat is meestal voldoende alleen bij een zeer grote, heftige muis beweging (bijv. over de diagonaal van het scherm) begint de cursor na te ijlen (gevoelsmatig).



De mooie vloeiende cursor beweging die PC applicaties laten zien, begon me steeds meer te ergeren. Hoe deden die lui dat?

Met deze muis driver heb ik een kleine demo opgezet waarmee je de DOS65 cursor vrij over het scherm kunt bewegen. Met behulp van een viertal variabelen is een muis gebied te definiëren, zodat de cursor alleen daar kan komen waar JIJ dat wilt. Ook laat de demo zien hoe een z.g. 'hot area' op het scherm kan worden gemaakt. In deze demo is dat het gebied binnen de QUIT box. Kom je met de cursor op één van de letters van het woord QUIT, dan wordt de tekst in inverse video getoond om aan te geven dat het gebied nu actief is. Wordt nu met de linker muis knop geklikt, dan wordt de demo verlaten. In deze demo, wordt de muis beweging bevro-

ren zodra één van de knoppen wordt ingedrukt. Ook wordt de functie die aan de knop gekoppeld is pas uitgevoerd zodra de knop wordt losgelaten.

De demo bestaat uit een aantal files, die op het BBS "The Ultimate" staan. Wegens de niet te geloven beperkingen van MS-DOS filenamen hebben ze een andere naam gekregen: zie de tabel onderaan de bladzijde.

Korte handleiding:

De boel downloaden van "The Ultimate" op een MS-DOS machine, het beste natuurlijk de ge'ZIP'te versie, want dan heb je alles in één keer. De ZIP file uitpakken, deze werd gemaakt met PKZIP 2.04g. Vervolgens de uitgepakte files op een floppy zetten en met het DMC (DOS65 <-> MS-DOS Copy) programma op je DOS-65 zetten. Lukt dat niet dan kun je ze ook via een seriële verbinding op je DOS-65 bakje zetten. Direct downloaden op je DOS-65 pieremegogeltje kan natuurlijk ook, let wel op dat je alle files hebt. De files zijn in MS-DOS formaat op "The Ultimate" gezet, dat kan betekenen dat er nu een Carriage Return code teveel in staat. Is het je gelukt de boel op een DOS65 floppy te pleuren volgens bovenstaande tabel, dan kun je de make script file starten:

```
$ setm -c make
$ make
```

Als alles goed is krijg je nu een executable file met de naam 'mdemo'. Mensen met een 6502 i.p.v. een 65C02 zullen eerst nog wat wijzigingen door moeten voeren. Zo zijn op een aantal plaatsen typische 65C02 instructies gebruikt zoals PHX, BBS e.d. Lukt het echt niet om de boel op een 6502 aan de praat te krijgen, laat me dat dan even weten.

Direct downloaden op je DOS-65 pieremegogeltje kan natuurlijk ook,.

Zo, dat was het weer voor deze keer, genereer je vooral niet en gebruik de demo als basis voor je eigen muis gestuurde software. Ook kan ik me voorstellen, dat je, nadat je je vol walging hebt afgewend van mijn nijvere huisvlijt, de boel overboord gooit en op je eigen manier eens aan de slag gaat met zo'n knaagdiertje aan je DOS65. Al met al was dat ook de reden voor deze artikelen, de DOS65 weer eens van zolder halen en actief aan de slag.

Antoine Megens

DOS65 File name	MS-DOS File name	Description
demo_main.mac	demo.mac	Main module, link all together
demo_misc.mac	d_misc.mac	Misc. routines
demo_map.mac	demo_map.mac	Memory map definitions
do_demo.mac	do_demo.mac	The demo mouse function
mouse_demo.mac	mouse_d.mac	Mouse driver for demo
mouse_def.mac	mous_def.mac	Mouse definitions
make	make	script file to make demo

Fig. 1: niet geloven file-namen (macro-biotisch)

```
; Make batch file for Mouse demo
;
del -y mdemo*
as -bmdemo demo_main
rename mdemo.bin mdemo
setm -c mdemo
```

Fig. 2: make: een script om de boel aan mekaar te knupp'n

```
;---------------------------------------------------------------------
; filename    : demo_main.mac
; last edit   : 09-oct-1992
; author      : Antoine Megens
; description : demo of mouse routine
;---------------------------------------------------------------------

; routines based on 65C02, but easy to adjust for standard 6502

                opt     rc02

                lib     demo_map            ; memory map
                lib     mouse_def           ; mouse definitions

                org     prgrom

start           jsr     iniacia             ; initialize ACIA
                jsr     inimous             ; initialize mouse functions
                jsr     initvdu             ; init CRTC and screen
                stz     oldbut
                stz     oldx
                stz     oldy
                stz     qflag
;
; Main loop starts here
;
main            jsr     mouse               ; mouse function in main loop
                jsr     showbut             ; show button status on screen
                jsr     showxy              ; show X,Y of mouse on screen
                jmp     main                ; not much for a main loop, but hey! it's just
                                            ; a demo you know!

exit            jsr     aciaoff             ; switch off ACIA
                jsr     print               ; clear screen and greet the audience
                fcc     '\fBye, bye\r',0
                rts
;
; Mouse driver
;
                lib     mouse_demo          ; get mouse routines
;
; Mouse functions
;
                lib     do_demo             ; demo function
                lib     demo_misc           ; misc. functions
;
; Low level init routines
;
iniacia         lda     #%00000101          ; DTR low, IRQ enabled, no parity
                sta     aciasr              ; reset ACIA (dummy write to status)
                sta     aciacmd             ; send command
                lda     #%00011000          ; 1200 Bd internal, 8N1
                sta     aciactl             ; send control
```

Fig. 3: demo_main.mac

```
                rts

aciaoff         lda     #%00000010      ; DTR high, IRQ disabled, XMIT off
                sta     aciasr          ; reset ACIA (dummy write to status)
                sta     aciacmd         ; send command
                rts

initvdu         jsr     inivdu          ; set CRTC registers, cursor on
                jmp     iniscr          ; draw screen

                end     start
```

Fig. 3: demo_main.mac

```
;---------------------------------------------------------------------------
; filename    : demo_misc.mac
; last edit   : 09-oct-1992
; author      : Antoine Megens
; description : Miscellaneous support routines
;---------------------------------------------------------------------------

XLBUT   equ     12          X position of left mouse button on screen
;
; This routine shows how to use the buttons register
; This routine shows the current status of each button in the fake mouse on
; the screen.
; The BBS instruction can only be used for the 65C02. If you want to use
; the routines for a standard 6502, use something like:
;
;     65C02                       6502
;     bbs     #fclick,buttons,1.f lda     buttons
;                                 and     #fclick
;                                 bne     1.f

showbut lda     buttons
                cmp     oldbut
                beq     4.f
                sta     oldbut
                jsr     curoff
                ldx     #XLBUT              position of fake left button on screen
                ldy     #15
                jsr     setcur
                bbs     #fclick,buttons,1.f
                jsr     erabut
                bra     10.f
1               jsr     drwbut
10              ldx     #XLBUT+5            position of fake middle button on screen
                ldy     #15
                jsr     setcur
```

Fig. 4: demo_misc.mac

```
                bbs     #mclick,buttons,2.f
                js      erabut
                bra     20.f
2               jsr     drwbut
20              ldx     #XLBUT+10           position of fake right button on screen
                ldy     #15
                jsr     setcur
                bbs     #eclick,buttons,3.f
                jsr     erabut
                bra     30.f
3               jsr     drwbut
30              jsr     curon
                jsr     gomouse
4               rts
;
; Draw depressed button on screen
;
drwbut          jsr     print
                fcc     $1b,'i',$1b,'n',0
                rts
;
; Draw released button on screen
;
erabut jsr    print
                fcc     '  ',0
                rts
;
; This routine draws the screen, do not change anything because
; else the QUIT box and/or mouse buttons do not coincide with the
; fixed X,Y locations in the rest of the program
;
iniscr          jsr     print
                fcc     '\f\r\r'
                fcc     '   ',$1b,'FF F F F F F F F F F F F F F F',$1b,'G\r'
                fcc     '   ',$1b,'FF',$1b,'G'
                fcc     ' Demo DOS-65 Mouse Driver  '
                fcc     $1b,'FF',$1b,'G\r'
                fcc     '   ',$1b,'FF',$1b,'G'
                fcc     '   Rev 1.0 by A.Megens    '
                fcc     $1b,'FF',$1b,'G\r'
                fcc     '   ',$1b,'FF',$1b,'G'
                fcc     '     (C) KGN 1993       '
                fcc     $1b,'FF',$1b,'G\r'
                fcc     '   ',$1b,'FF F F F F F F F F F F F F F F',$1b,'G\r'
                fcc     '\t\t',$1b,'FPXXXXV',$1b,'G\r'
                fcc     '\t\t',$1b,'FY',$1b,'G'
                fcc     'QUIT'
                fcc     $1b,'FY',$1b,'G\r'
                fcc     '\t\t',$1b,'FRXXXXT',$1b,'G\r'
                fcc     '   Click in QUIT box with left button\r'
                fcc     '        to return to DOS65\r'
                fcc     '\t',$1b,'FPXXXXXXXXXXXXXXXXXV',$1b,'G\r'
                fcc     '\t',$1b,'FY PXXXVPXXXVPXXXV Y',$1b,'G\r'
                fcc     '\t',$1b,'FY Y   YY   YY   Y Y',$1b,'G\r'
```

Fig. 4: demo_misc.mac

```
                fcc     '\t',$1b,'FY RXXXTRXXXTRXXXT Y',$1b,'G\r'
                fcc     '\t',$1b,'FY        Y',$1b,'G\r'
                fcc     '\t',$1b,'FY        Y',$1b,'G\r'
                fcc     '\t',$1b,'FY        Y',$1b,'G\r'
                fcc     '\t',$1b,'FY        Y',$1b,'G\r'
                fcc     '\t',$1b,'FY        Y',$1b,'G\r'
                fcc     '\t',$1b,'FRXXXXXXXXXXXXXXXXXT',$1b,'G\r\r'
                fcc     $1c,0
                rts
;
; This routine shows the current mouse X,Y location in the fake mouse
; on the screen. Too avoid unneccesary updates (resulting in flicker
; at the mouse cursor) only rewrite if X,Y changed from previous call
; (oldx,oldy)
;
showxy          ldx     mousex          get mouse X
                cpx     oldx            changed?
                beq     1.f             no, check Y
                jsr     curoff
                ldx     #XLBUT
                ldy     #20
                jsr     setcur
                jsr     print
                fcc     'X: ',0
                lda     mousex
                sta     oldx
                jsr     hexdec
                jsr     hexout
1               ldx     mousey
                cpx     oldy
                beq     2.f
                jsr     curoff
                ldx     #XLBUT+7
                ldy     #20
                jsr     setcur
                jsr     print
                fcc     'Y: ',0
                lda     mousey
                sta     oldy
                jsr     hexdec
                jsr     hexout
2               jsr     curon
                jmp     gomouse
;
; Set cursor on/off routines
; always switch cursor off when writing stuff to screen, this reduces
; flicker on the mouse cursor
;
curon           pha     save A
                phx     save X
                lda     #$00            steady block cursor
                bra     1.f             write to CRTC
curoff          pha
                phx
```

Fig. 4: demo_misc.mac

```
                lda     #$20            cursor off
1               ldx     #10             write to CRTC register 10
                stx     crtcar          set CRTC address
                sta     crtcrf          write data
                plx                     restore X
                pla                     and A
                rts
```

Fig. 4: demo_misc.mac

```
                pag
;---------------------------------------------------------------------
; filename    : demo_map.mac
; version     : 1.00
; last edit   : 09-oct-1993
; author      : Antoine Megens
; description : symbol and hardware definitions
; copyright   : KGN, 1993
;---------------------------------------------------------------------
; edit history:
;   date            name        ver. description
; 09-oct-1993       am          1.00 initial release for uP article
;---------------------------------------------------------------------

; addresses for program development on DOS65

zeropag         equ     $0020           zeropage
syswram         equ     $0200           start of system work RAM
dataram         equ     $2000           start of data RAM
prgrom          equ     $8000           start of program EPROM
iospace         equ     $e080           start of I/O space
aciasr          equ     $e131           DOS65 6551 ACIA status register
aciacmd         equ     aciasr+1        DOS65 6551 ACIA command register
aciactl         equ     aciasr+2        DOS65 6551 ACIA control register
crtcar          equ     $e140           DOS65 6845/6545 CRTC address register
crtcrf          equ     crtcar+1        DOS65 6845/6545 CRTC register file
datrame         equ     prgrom          end of data RAM
;
; zeropage definitions
;
mouszpg         equ     zeropag         zeropage RAM loc for mouse routines
oldbut          equ     mouszpg+16      old button mask, for demo only
oldx            equ     oldbut+1        old X, for demo only
oldy            equ     oldbut+2        old Y, for demo only
qflag           equ     oldbut+3        quit flag, for demo only
freezp          equ     qflag+1         first free zero page location
;
; system ram definitions
;
```

Fig. 5: demo_map.mac

```
mousram         equ     syswram
mouslen         equ     16                      ;reserve space in RAM for mouse routines
freesr          equ     mousram + mouslen       ;first free system ram location
;
; IO-65 / DOS-65 entries
;
setcur          equ     $f024                   ;place cursor at X,Y
inivdu          equ     $fd1c                   ;init VDU
print           equ     $c03b                   ;print null terminated string
hexdec          equ     $c044                   ;convert A to decimal
hexout          equ     $c038                   ;print A in hex
```

Fig. 5: demo_map.mac

```
;-------------------------------------------------------------------------
; filename    : do_demo.mac
; version     : 1.00
; last edit   : 17-oct-1993
; author      : Antoine Megens
; description : demo routine for mouse driver
; copyright   : KGN, 1993
;-------------------------------------------------------------------------
; edit history:
;   date            name        ver. description
; 17-oct-1993       am          1.00 initial release for uP article
;-------------------------------------------------------------------------

;
; This demo function allows to move the cursor over the entire active
; mouse area. It also checks all three buttons. In this demo function
; the movement is 'freezed' if a button is depressed. After the button
; is released it's actual attached function routine is executed.
; (In this demo just write a text on screen)
; There is also one 'hot' area on the screen, namely the text QUIT
; in the quit box. If the cursor is in that area the text QUIT is
; drawn in inverse video. If the left button is clicked in this area
; the program return to DOS65
;

XQUIT           equ     17                      mouse locations of QUIT box
YQUIT           equ     8

do_demo         lda     buttons                 check buttons
                bmi     wfrelft                 function activated, wait for left release
                and     #leftbut                check left button
                beq     lbutpr
                jmp     chkrght                 not pressed, check right
;-------------------------------------------------------------------------
; Left button pressed, set bit and exit
;-------------------------------------------------------------------------
```

Fig. 6: do_demo.mac

```
lbutpr          smb     #fclick,buttons     set function_clicked_bit in buttons
                bra     clrmov              and return, no movement allowed anymore!
;-------------------------------------------------------------------------
; function was 'clicked' now wait for button to release before actual function
; subroutine is executed.
;-------------------------------------------------------------------------
wfrelft         and     #leftbut            check left button
                beq     clrmov              not released!
                rmb     #fclick,buttons     clear function clicked bit
;
; left button function here, for demo just draw text
;
                jsr     curoff
                ldx     #XLBUT
                ldy     #18
                jsr     setcur
                jsr     print
                fcc     'Left Function  ',0
                jsr     curon
                lda     qflag               clicked in QUIT box?
                bne     1.f
                jmp     gomouse
1               pla                         adjust stack pointer
                pla                         now pointing to DOS65 again
                jmp     exit
;-------------------------------------------------------------------------
; Any button depressed, then freeze mouse movement
;-------------------------------------------------------------------------
clrmov          stz     deltax
                stz     deltay              keep clearing mouse movement
;
; Clear function text in mouse, this part is just for the demo
;
                jsr     curoff
                ldx     #XLBUT
                ldy     #18
                jsr     setcur
                jsr     print
                fcc     '   Wait...   ',0
                jsr     curon
                jmp     gomouse
;-------------------------------------------------------------------------
; check right mouse button (escape)
;-------------------------------------------------------------------------
chkrght         lda     buttons
                bbs     #eclick,buttons,wfrrel
                and     #rghtbut
                bne     chkmid              not pressed, check middle button
;-------------------------------------------------------------------------
; right button is pressed, set escape bit and exit
;-------------------------------------------------------------------------
                smb     #eclick,buttons
200             jmp     clrmov
;-------------------------------------------------------------------------
```

Fig. 6: do_demo.mac

```
; right is 'clicked' wait for button release to execute function
;-------------------------------------------------------------------------
wfrrel          and     #rghtbut
                beq     200.b           still depressed, keep mouse quiet
                rmb     #eclick,buttons
;
; right button function here, for demo just draw text
;
                jsr     curoff
                ldx     #XLBUT
                ldy     #18
                jsr     setcur
                jsr     print
                fcc     'Right Function ',0
                jsr     curon
                jmp     gomouse
;-------------------------------------------------------------------------
; check middle mouse button
;-------------------------------------------------------------------------
chkmid          lda     buttons
                bbs     #mclick,buttons,wfmrel
                and     #midlbut
                bne     mousexy         not pressed, check for cursor movement
;-------------------------------------------------------------------------
; middle button is pressed, set status bit and exit
;-------------------------------------------------------------------------
                smb     #mclick,buttons
200             jmp     clrmov
;-------------------------------------------------------------------------
; middle is 'clicked' wait for button release to execute function
;-------------------------------------------------------------------------
wfmrel          and     #midlbut
                beq     200.b           still depressed, keep mouse quiet
                rmb     #mclick,buttons
;
; middle button function here, for demo just draw text
;
                jsr     curoff
                ldx     #XLBUT
                ldy     #18
                jsr     setcur
                jsr     print
                fcc     'Middle Function',0
                jsr     curon
                jmp     gomouse
;-------------------------------------------------------------------------
; mouse movement to cursor
;-------------------------------------------------------------------------
mousexy         jsr     updmous         update mouse X,Y location
                ldy     mousey
                cpy     #YQUIT          on QUIT line?
                bne     1.f
                jsr     chkquit
                bcs     gomouse
```

Fig. 6: do_demo.mac

```
1               jsr     eraqbut
gomouse         ldy     mousey
                ldx     mousex
                inx                     DOS-65 Home position is at 1,1
                iny                     so add +1 to X,Y
                jmp     setcur          this should be last cursor setting in
;                                       main loop

;
; QUIT box routines
;
chkquit         ldx     mousex
                cpx     #XQUIT          in QUIT button area?
                bcc     1.f
                cpx     #XQUIT+4        in QUIT button area?
                bcc     drqbut
1               clc                     C=0, means left QUIT area
                rts
drqbut          lda     qflag           QUIT flag set?
                bne     10.f            yes, no need to redraw text
                jsr     curoff
                ldx     #XQUIT+1
                ldy     #YQUIT+1
                jsr     setcur
                jsr     print
                fcc     $1b,'iQUIT',$1b,'n',0
                jsr     curon
                lda     #1
                sta     qflag           set QUIT flag
10              sec                     C=1, means in QUIT area
                rts
eraqbut         lda     qflag           QUIT flag cleared?
                beq     1.b             yes, no need to redraw text
                jsr     curoff
                ldx     #XQUIT+1
                ldy     #YQUIT+1
                jsr     setcur
                jsr     print
                fcc     'QUIT',0
                stz     qflag
                jmp     curon
```

Fig. 6: do_demo.mac

```
;--------------------------------------------------------------------------
; filename    : mouse.mac
; version     : 1.00
; last edit   : 09-oct-1993
; author      : Antoine Megens
; description : mouse driver
; copyright   : KGN, 1993
;--------------------------------------------------------------------------
; edit history:
;   date            name        ver. description
; 09-oct-1993      am          1.00 initial release for uP article
;--------------------------------------------------------------------------
;
; This subroutine reads mouse data and jumps to the selected mouse function
;
mouse           jsr     mouspak         sample mouse package
                bcc     1.f             package not complete yet, so exit
                jsr     xmove           compute delta X
                jsr     ymove           and delta Y
                jmp     [mousfun]       and do mouse function
1               rts
;
; Initialize mouse parameters
;
inimous         stz     mousex          clear X,Y loc. and button register
                stz     mousey
                stz     buttons
resmous         stz     mxmin           default mouse area is entire screen
                stz     mymin
                lda     #79
                sta     mxmax
                lda     #23             (may not enter status line)
                sta     mymax
clrmous         stz     deltax
                stz     deltay
                lda     #do_demo&255    default mouse function is demo
                ldy     #do_demo>8
                sta     mousfun
                sty     mousfun+1
                rts
;
; default mouse movement routine
; use deltax,deltay to compute new mouse X,Y location
;
updmous         lda     deltax          positive or negative movement?
                bmi     8.f
                cmp     #16             15?
                bcc     9.f             no, use it
                lda     #15             clip to 15
9               tax
                lda     posx,x          get offset from sensitivity table
                beq     doy             no need to update, check Y movement
                bra     1.f             else update X position
8               and     #$0f            clip to 15
```

Fig. 7: mouse_demo.mac

```
                tax
                lda     negx,x          get offset from sensitivity table
                beq     doy             no need to update
1               sta     deltax          save offset
                lda     mousex          add to mouse X location
                clc
                adc     deltax          now check result
                ldy     mousey          at top line?
                beq     5.f             then allow for full screen width
                cmp     mxmin
                bpl     3.f
                lda     mxmin           X out of (minimum) range
                bra     2.f
5               cmp     #0
                bpl     3.f
                lda     #0
3               ldy     mousey          at top line?
                beq     5.f             then allow for full screen width
                cmp     mxmax           reached max. mouse X limit?
                bcc     2.f
                lda     mxmax
                bra     2.f
5               cmp     #79             for top line check full screen width
                bcc     2.f
                lda     #79
2               sta     mousex          save new value in mouse X location
                stz     deltax          and reset delta value
;
; Mouse Y is positive in up direction, video Y however moves down when
; increased, so delta Y is subtracted from mousey counter
;
doy             lda     deltay          see comment at X movement
                bmi     8.f
                cmp     #16
                bcc     9.f
                lda     #15
9               tax
                lda     posy,x
                beq     updex
                bra     1.f
8               and     #$0f
                tax
                lda     negy,x
                beq     updex
1               sta     deltay
                lda     mousey
                sec
                sbc     deltay
                cmp     mymin
                bpl     4.f
                lda     mymin           no negative Y allowed
4               cmp     mymax           reached max. mouse Y limit?
                bcc     3.f
                lda     mymax
```

Fig. 7: mouse_demo.mac

```
3               sta     mousey
                stz     deltay
updex           rts
;
; Get X movement from mouse serial data buffer
;
xmove           lda     deltax          get current delta X value
                clc
                adc     mpack           add data from buffer
                adc     mpack+2
                sta     deltax          save new value
                rts
;
; Get Y movement from mouse serial data buffer
;
ymove           lda     deltay
                clc
                adc     mpack+1
                adc     mpack+3
                sta     deltay
                rts
;
; Read serial port and check for mouse sync byte and X,Y data
;
;               8X XX YY XX YY          mouse data at 1200 Bd
;               1 2 3 4 5
;
mouspak         jsr     rdmouse         get byte from serial port buffer
                pha                     save original data
                and     #%11111000
                cmp     #%10000000      sync byte starts with 10000xxx
                beq     sav_but         in sync byte also button info
                pla
                ldx     mbptr           X or Y data?
                beq     1.f             no, sync byte not received yet, ignore byte
                sta     mpack-1,x       save X,Y data
                inx
                stx     mbptr
                cpx     #5              package complete?
                bne     1.f             not yet...
                stz     mbptr           reset mbptr for next run
                sec                     and signal caller we've got something!
                rts
sav_but         pla
                and     #%00000111      save mouse button info
                tax
                lda     buttons
                and     #%11111000      clear old button info
                stx     buttons         put in new button info
                ora     buttons         restore button status bits
                sta     buttons         and save it all
                lda     #1
                sta     mbptr           set sync condition
1               clc                     package not complete yet, keep carry cleared
```

Fig. 7: mouse_demo.mac

```
                rts
;
; Mouse movement sensitivity tables
; a NULL value means no offset, but delta value will not reset
; any other value is used as offset and delta value is cleared
;
;                       0 1 2 3 4 5 6 7 8 9 A B C D E F
posx            fcc     0,0,0,1,1,1,2,3,3,3,4,4,4,5,5,6
posy            fcc     0,0,0,0,0,0,0,0,1,1,1,1,2,2,2,3
;                        F E D C B A 9 8 7 6 5 4 3 2 1 0
negx            fcc     -6,-5,-5,-4,-4,-4,-3,-3,-3,-2,-1,-1,-1,0,0,0
negy            fcc     -3,-2,-2,-2,-1,-1,-1,-1,0,0,0,0,0,0,0,0

;
; low level I/O65 routine to read serial port
;
rdmouse         ldx     #3              ; get byte through serial channel
                jmp     inpx
```

Fig. 7: mouse_demo.mac

```
;-------------------------------------------------------------------------
; filename    : mouse_def.mac
; version     : 1.00
; last edit   : 27-feb-1992
; author      : Antoine Megens
; description : mouse symbols and definitons
; copyright   : KGN, 1992
;-------------------------------------------------------------------------
; edit history:
;   date            name    ver. description
; 22-apr-1991       am      0.00 started
; 29-apr-1991       am      0.01 added mouse range mxmin,mymin/mxmax,mymax
; 13-may-1991       am      0.02 added mousfun pointer
; 01-jul-1991       am      0.03 prepared for actual host card
; 08-jul-1991       am      0.04 added mbptr and mpack
; 02-nov-1991       am      0.05 added mclick definition
; 27-feb-1992       am      1.00 initial release for demo disk
;-------------------------------------------------------------------------
;
; Constants
;
leftbut         equ     %00000100       left button mask
midlbut         equ     %00000010       middle button mask
rghtbut equ     %00000001 right button mask
fclick          equ     7               function clicked bit
eclick          equ     6               escape button clicked bit
mclick          equ     5               middle button clicked bit
inpx            equ     $f009           input through device X
;
```

Fig. 8: mouse_def.mac

Voortgang DOS65

Het is zover. Van de bestelde vijf protoprinten zijn er inmiddels twee bestukt. De bestukkers hadden de euvele moed om de print in hun DOS65-wonder te steken. En wat bleek: de kaart werkte. De batterij-backup is nog niet getest, omdat de DS1210 battery controller moeilijk te leveren blijkt te zijn. Voor de rest blijkt alles naar verwachting te werken.

Daarom dit artikeltje.

Veranderingen

Er zijn ten opzichte van de reeds gepubliceerde schema's een paar veranderingen doorgevoerd. De simpelste verandering is dat de nummers van de verschillende IC's op de kop zijn gezet. Dat is het gevolg van het feit dat er een een keer een auto-annotate is gedaan in het schema-tekenpakket. De IC's worden dat in volgorde van plaatsing genummerd.

Ook al gemeld is de toevoeging van de DS1210 battery controller en een lithium batterijtje. De bedoeling hiervan zal iedereen duidelijk zijn: de RAMmetjes worden daar aanzienlijk minder vergeetachtig van.

De derde veranderinging is de pinout van de PAL op de print. We hebben gepoogd het printtekenpakket het leven zo min mogelijk zuur te maken door een paar pootjes te verplaatsen.

Aanwijzingen voor de bouw

Hieronder staat een boodschappenlijstje afgedrukt. Alles staat erop, ook de de IC-voetjes. U kent het verhaal wel: gebruik altijd voetjes in uw hobby-project want dat is gemakkelijker met foutzoeken en veranderingen aanbrengen. Gebruik ook fatsoenlijke voetjes,dus die wonders met gedraaide contacten.

Het opbouwen van de print is niet moeilijk. De print is dubbelzijdig uitgevoerd en het verschijnsel draadbrug wordt u dan ook bespaard. Breng eerst de IC-voetjes aan, gevolgd door de elco en de 100 nF condensatortjes. Deze laatste hebben een steek van 7.5 mm. De stuklijst gaat ervan uit, dat het voetje voor de 8kx8 SRAM wordt gemaakt door twee 14-pens voetjes te gebruiken. Mocht ook het smalle 24-pens voetje een probleem zijn, dan kan hiervoor een 8-pens en een 16-pens voetje worden benut. Let er verder even op dat de geheugen IC's pin 1 aan de andere kant hebben ten opzichte van de andere IC's. Monteer daarna de busconnector. Vergeet de twee boutjes niet. Laat de IC's en de batterij nog even weg.

Meet nu eerst tussen de pennen 32a/c en 1a/c van de busconnector of er misschien een sluiting in de voeding zit. Zo ja, deze eerst opsporen. Meet u niets of een zich opladende elco, dat is alles OK.

Wel of geen batterij

Nu komt de keuze of er wel of geen batterij op de geheugenkaart moet komen. Als u pseudo-statische RAMs gaat toepassen heeft de batterij geen zin: hij zou in korte tijd leeg zijn, en de RAMs krijgen toch geen refresh (activiteit op de adreslijnen) als de computer uit staat. Een batterijtje en een DS1210 hebben dus alleen zijn bij toepassing van echte CMOS statische RAMs.

Als het batterijtje niet wordt gebruikt vervalt dus ook de DS1210 (IC10) wen mag de 74HCT138 (IC8) ook wel een 74LS138 worden. Verbindt dan de pinnen 1 en 8 (VccI en VccO), en 5 en 6 (CEI en CEO) met twee draadjes door.

Verder opbouwen

Steek nu alle IC's, behalve IC1 (de databusbuffer) in hun voetjes. Sluit de batterij provisorisch aan en meet de stroomopname. Deze mag niet meer bedragen dan 50 uA: de battrij voetd de DS1210 (minder dan 100 nA), de 74HCT138 (4 uA) en 8 SRAMs (totaal maximaal 40 uA). Is het meer, verwijder dan een voor een de genoemde IC's tot de stroomvreter is gevonden.

Plaats de print, zonder IC1 in uw systeem. Laat de bestaande geheugenkaart op zijn plaats. Na inschakelen moet het sysateem normaal opstarten. Indien niet, eerst de oorzaak (sluiting van buslijnen) proberen te vinden.

Start het systeem wel normaal op, dan komt nu het grote moment: beide geheugenkaartenverwijderen, IC1 plaatsen de nieuwe RAMkaart plaatsen. Als het systeem nu nog steeds start, is de kaart zeer waarschijnlijk OK.

Virtual disk

Tijdens de evalutatie van de oude virtual diskkaart met DRAMs en de nieuwe die hier besproken wordt is er nog een verschilletje boven water gekomen: in de oude kaart wordt de mappinginformatie door 74LS189's geinverteerd opgeslagen, maar in de nieuwe normaal. Dit levert geen problemen op, als de kaart helemaal volbestukt is met 8 SRAMs.

Is de kaart echter gedeeltelijk leeg, dan is er een kunstje. Vervang IC2 dan door een 74LS640, dit is

een inverterende 74LS245. De kaart is dan echt compatibel met de oude VDISK-kaart.

Moeilijke onderdelen

Een aantal onderdelen is wat moeilijker verkrijgbaar. Zo blijkt bijna niemand de DS1210 te leveren, en een geprogrammeerde PAL of GAL is ook niet even eenvoudig. Verder blijken ook 74ALS541's niet zo een, twee, drie te verkrijgen te zijn.

Voor de 74ALS541 kan eventueel ook een gewone 74(A)LS245 gebruikt worden, mits pin 1 niet in de voet wordt gestoken, maar over de bovenzijde van het IC wordt verbonden met pin 20 (Vcc).

Voor de overige onderdelen wordt op dit moment onderhandeld om deze via de club te gaan leveren. De volgende prijzen lijken hierbij realiseerbaar:

DS1210	*f*	10,00
GAL22V10 (geprogr.)	*f*	15,00
8kx8 SRAM, 25 ns	*f*	8,00
128kx8 CMOS SRAM	*f*	30,00 (dagprijs!)

In het volgende nummer zullen de definitieve prijzen, en de prijs voor de print (dubbelzijdig, doorgemetalliseerd, met soldeermasker en opdruk) bekend gemaakt worden, en hoe u dit allemaal kunt bestellen.

Tot de volgende keer.

Nico de Vries

Item	Aantal	Referentie	Omschrijving
1	1	C1	100uF, 10V
2	17	C2 t/m C18	100nF
3	2	IC1,IC2	74LS245
4	1	IC3	GAL20V8 of PAL22V10
5	1	IC4	SRAM, 8kx8-25ns (0.3 inch behuizing)
6	1	IC5	74ALS573
7	2	IC6,IC7	74ALS541
8	1	IC8	74LS138
9	1	IC9	74LS30
10	1	IC10	Dallas DS1210
11	8	IC11 t/m IC18	1 Mbit (128kx8) SRAM of Pseudo SRAM
12	1	J1	Connector DIN 41612, a/c, male haaks
13	1	BT1	Batterij, lithium, 3.6V
14	1	-	Print
15	1	-	IC voetje, 8 pins
16	3	-	IC voetje, 14 pins
17	1	-	IC voetje, 16 pins
18	5	-	IC voetje, 20 pins
19	1	-	IC voetje, 24 pins, 0.3 inch
20	8	-	IC voetje, 32 pins

Mogelijkheden voor SRAM, 8kx8:
UMC: UM6164AK-25
Cypress: CY7C185-25
Toshiba: TC5588P-25
Fujitsu: MB81C78A-25, of equivalenten.

Mogelijkheden voor 1Mbit SRAM of Pseudo SRAM:
Toshiba: TC551001P (CMOS SRAM) of TC518129P (Pseudo Static)
Hitachi: HM628128 (CMOS SRAM) of HM658128 (Pseudo Static)
NEC: μPD431000 (CMOS SRAM), of equivalenten.
Snelheid 200ns (bij 1 MHz) of 150 ns (bij 2 MHz) of sneller.
LET OP: bij toepassing van de DS1210 en de batterij alleen CMOS SRAM gebruiken!

Onderdelenlijst

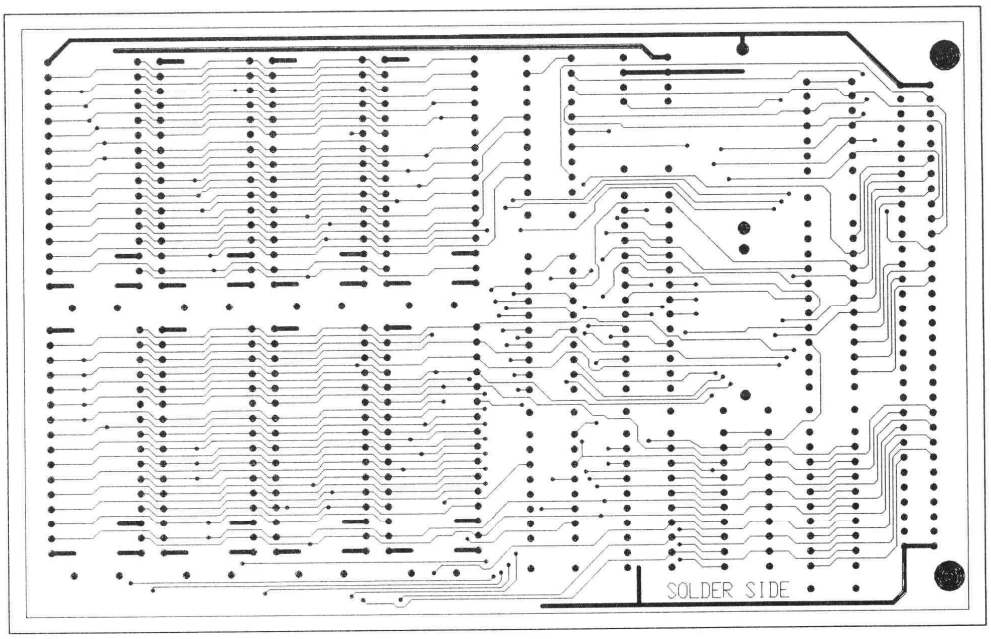


Fig. 1: print, sporen aan soldeerzijde

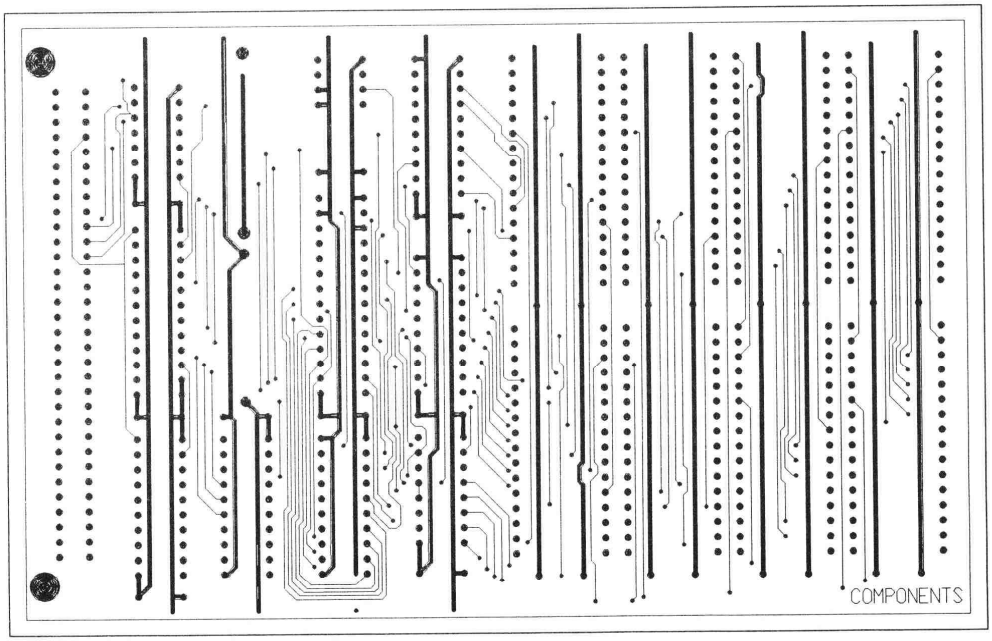


Fig 2: print, sporen aan componentenzijde

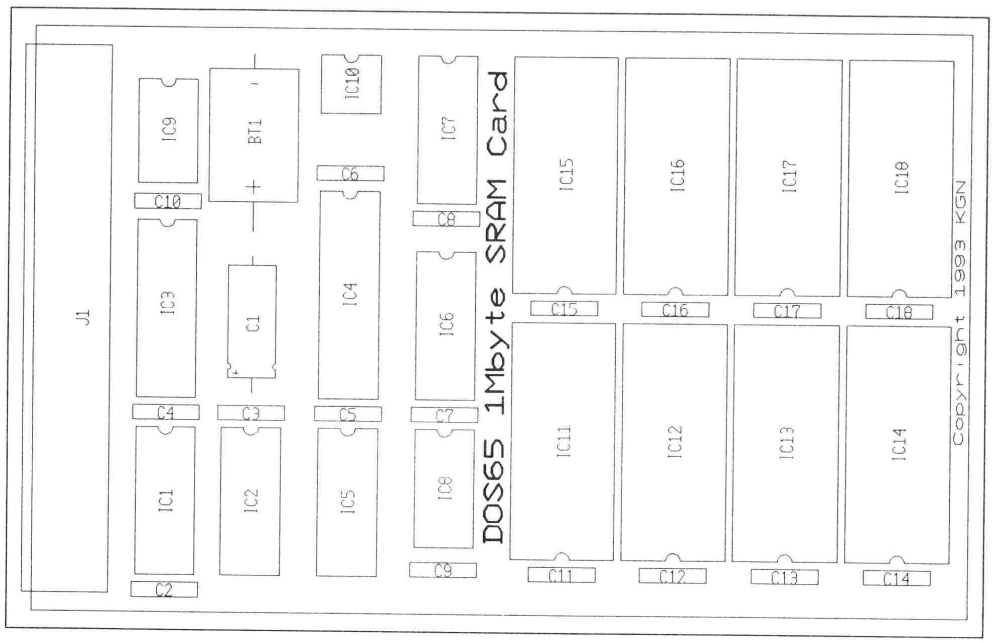


Fig 3: print, componentenopstelling

Gezocht: Penners en Typers!!!

De KGN zoekt leden voor de redactie en auteurs van artikelen.

Vaardigheden: Een vlotte pen.

Wie kan incidenteel of op regelmatige basis (eens per twee maanden) een stukje schrijven over bijvoorbeeld de volgende onderwerpen:

- **ShareWare**
- **Gelezen in andere computertijdschiften**
- **Een column over computerzaken**
- **Nieuwe ontwikkelingen op het gebied van Hard- en Software**
- **Toepassingen van micoprocessors of computers**
- **Een strip met ons logo in de hoofdrol**
- **Etc.**

Bellangstelling of een bijdrage? Geef het bestuur een berichtje of upload het naar het BBS!

```
GAL20V8                                                    GAL DESIGN SPECIFICATION
VDISK3 5th VERSION                                         N. DE VRIES  07-02-1994
8kx8 SRAM CONTROLLER PAL FOR DOS65 VDISK CARD WITH SRAMS, 1M VERSION
NDV FIRMWARE
RW  PHI2 A0    A1      A2    A3  A4  A12 A13 A14  A15  GND
NC1 NC2  G245  WE2064 G573  Q3  Q2  Q1  Q0  G138 LS30 VCC

/G138  = PHI2*/A15                          ;access
       + PHI2*   /A14                       ;to
       + PHI2*      /A13                    ;RAM
       + PHI2* A15* A14*/A13                ;array

/G245  = /RW* A15* A14* A13* A12*/LS30*/A4                ;write to $FFE0-$FFEF
       + /RW* A15* A14* A13* A12*/LS30* A4*/A3*/A2*/A1*/A0        ;write to $FFF0
       +          /A13                                  ;access
       +       /A14                                     ;to
       +    /A15                                        ;RAM
       +     A15* A14*/A13                              ;array

/WE2064 =  PHI2*/RW* A15* A14* A13* A12*/LS30*/A4        ;write to $FFE0-$FFEF

G573   =  PHI2*/RW* A15* A14* A13* A12*/LS30* A4*/A3*/A2*/A1*/A0        ;write to $FFF0

/Q0    = /A0* A15* A14* A13* A12*/LS30*/A4    ;write to $FFEX: A0 is 8kx8 SRAM addressline
       +             /A12                     ;else let A12 through

/Q1    = /A1* A15* A14* A13* A12*/LS30*/A4    ;write to $FFEX: A1 is 8kx8 SRAM addressline
       +          /A13                        ;else let A13 through

/Q2    = /A2* A15* A14* A13* A12*/LS30*/A4    ;write to $FFEX: A2 is 8kx8 SRAM addressline
       +       /A14                           ;else let A14 through

/Q3    = /A3* A15* A14* A13* A12*/LS30*/A4    ;write to $FFEX: A3 is 8kx8 SRAM addressline
       +    /A15                              ;else let A15 through


DESCRIPTION

G245 is the enable for the databusbuffer

G573 is the enable for the task number register (must be active high)

G138 is the enable for the RAM array address decoder

WE2064 is the write enable input for the 8kx8 SRAM and the enable for the
       write data buffer to the 8kx8 SRAM

Q0..Q3 are the lower addresslines for the 8kx8 SRAM
```

Vergelijkingen voor de GAL/PAL

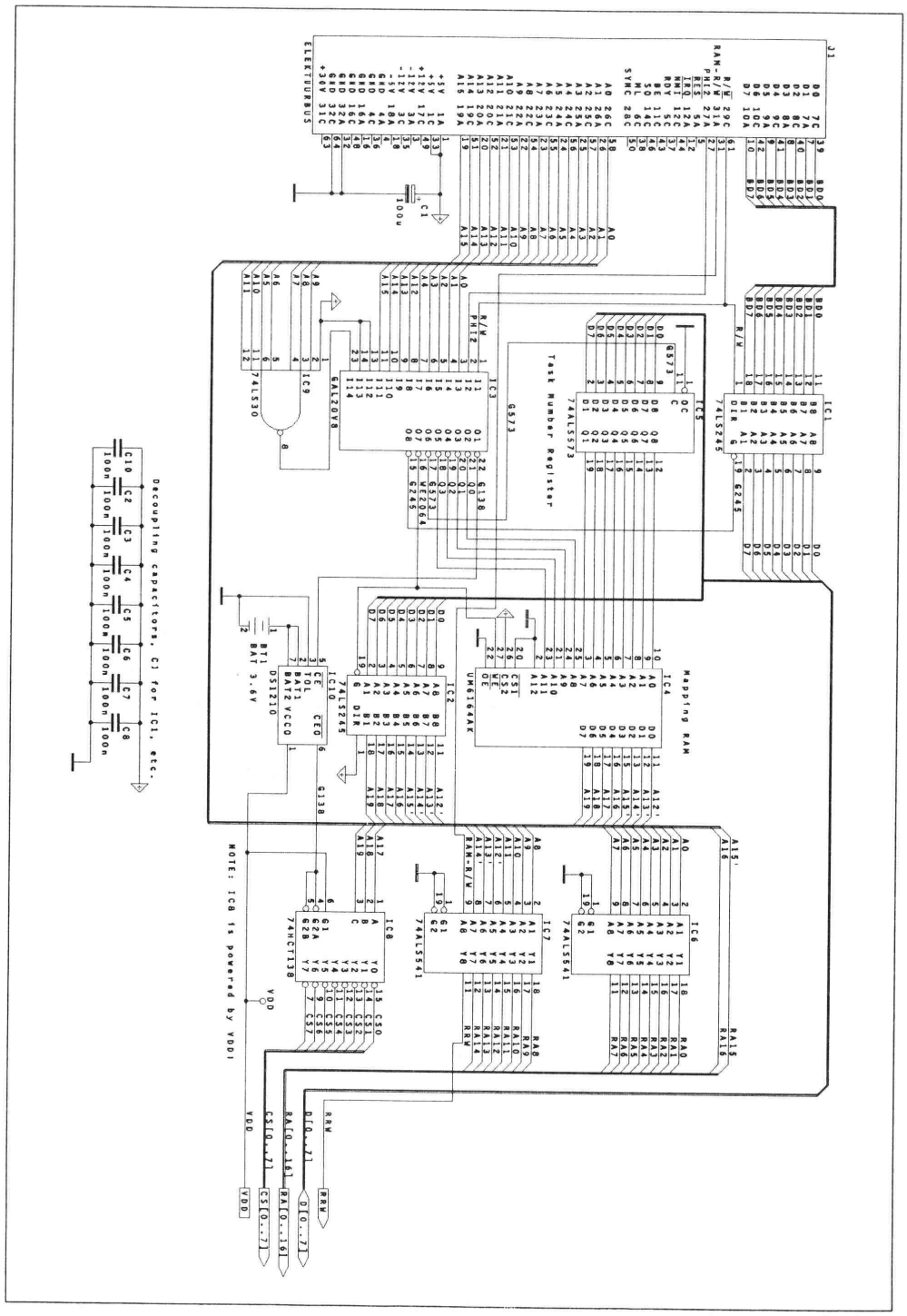


Fig. 5: Schema van het stuurgedeelte

Als men gebruik maakt van gewone hulpmiddelen, dan komt dit vooral doordat DPMS het gros van het werk overlaat aan de programmeur. Zo is een programma dat gebruik maakt van DPMS een gewoon programma dat geheel of gedeeltelijk onder gebracht wordt in XMS geheugen. De verplaatsing binnen het geheugen gebeurt in feite niet door DPMS, maar door de programmacode zelf. Vandaar de complexiteit van het probleem.

De programmeur zal moeten bepalen of DPMS wordt geïnstalleerd. Hij zal DPMS moeten oproepen om in de protected modus segment-descriptoren toe te kennen, XMS geheugen toewijzen, nog eens DPMS oproepen om de descriptoren in te stellen (basisadres, grenswaarden en types). Pas dan kan het programma de real modus-segmenten onderbrengen in het daartoe ingestelde geheugen. DPMS maakt wel een kopie, maar het is het programma zelf dat de vertaling van real modus naar protected modus en omgekeerd zal moeten voorzien. Wanneer alles klaar is in protected modus, hoeft de ontwikkelaar nog slechts het in real modus vrijgemaakte geheugen vrij te maken en de uitvoering van het programma verder te zetten.

Een ontwikkelaar die een programma met DPMS ondersteuning wil ontwerpen, staat er zo'n beetje alleen voor. Het momenteel enige beschikbare hulpmiddel is WDEB386, dat deel uitmaakt van de Windows-development kit. Deze debugger is beter dan niks, maar kan hoegenaamd niet tippen aan producten als CodeView of Turbo-Debugger.

De oplossing voor de toekomst?

DPMS is compatibel met MS-DOS, DR-DOS, Novell-DOS en Windows; tenminste met de huidige versies van die programma's. DPMS is een interessante techniek die nog maar eens het onvermijdelijke uitstelt, namelijk de overgang naar een 32 bits bestuurs-

systeem zoals OS/2, Windows NT of het langverwachte Windows 4 (codenaam Chicago).

De 32 bits modus is de enige afdoende manier om uit het keurslijf te breken en om schoonschip te maken met het verleden.

Voor u gelezen in CM Corporate: een artikel van Francois Piette: Novell DPMS beta doorgelicht

DOS65 1 MByte RAMkaart: printen en prijzen

Of: voortgang werkgroep DOS65

Elders in dit nummer staat een testprogramma voor de 1 Mbyte RAMkaart voor DOS65. Er is dus vooruitgang geboekt. Er is nog meer vooruitgang: de printen voor de kaart zijn in bestelling. Daarbij heeft de werkgroep in goed onderling overleg een veer gelaten.

De bedoeling is altijd geweest om de leden een dubbelzijdige, doorgemetalliseerde, van soldeermasker en opdruk voorziene print aan te bieden. Een tweede bedoeling was ook om de print niet veel duurder te maken dan ongeveer *f* 80,-. Deze twee wensen kunnen we helaas niet realiseren, zonder een onverantwoorde belasting te leggen op de gezonde financiële situatie binnen de club.

Daarom is besloten de printen aan te bieden zonder soldeermasker en componentenopdruk. De werkgroep was het hier unaniem over eens, en een aantal leden die we hierover hebben aangesproken vond het ook prima: de reeds gepubliceerde componentenopdruk is duidelijk genoeg en een beetje soldeerkunst is ook voldoende voorhanden. Dus: dezelfde printen als de prototypen, want veranderingen zijn niet nodig gebleken.

De prijzen voor de diverse spullen bevatten ook een bijdrage aan de clubkas, om volgende projectjes op te kunnen zetten, en voorraad printen te kunnen financieren. Het lijstje is nu:

Print	*f*	80,00
Dallas DS1210	*f*	10,00
GAL22V10 (geprogr.)	*f*	15,00
8kx8 SRAM, 20 of 25 ns	*f*	8,00
128kx8 CMOS SRAM	*f*	30,00

De laatste prijs is een dagprijs. Afwijkingen groter dan 5 procent zullen worden doorberekend, zowel positief als negatief. Bij iedere print wordt een 5.25" diskette geleverd met daarop de source van VTEST en de JEDEC-file voor de GAL/PAL.

Bestellen

U kunt de spullen als volgt bestellen: maak een lijstje van wat u hebben wilt en tel de bedragen op. Vermeld op uw lijst ook uw giro- of bankrekeningnummer, zodat we een eventueel overschot (bij een prijsdaling van de SRAMs) kunnen terug storten. Stuur het lijstje naar Nico de Vries en maak het benodigde bedrag over op de girorekening van de club: 3757649. Binnen zes weken zullen de bestelde spullen worden geleverd.

Nico de Vries

Bijtjes pesten of bytjes testen?

Hieronder staat de source voor het programma VTEST. Het programma is oorspronkelijk gemaakt om het prototype van de nieuwe 1 Mbyte RAMkaart te testen. Het is niet "mooi" geschreven: quick and dirty is een betere benadering, want het programma doet wat het moet doen. Het kan dus vast wel mooier, slimmer en korter. Ook is er geen aandacht besteed aan snelheid en efficiëntie.

Dit doet het

Het programma doet het volgende. Als eerste wordt er gekeken of er wel een 1 Mbyte RAMkaart in het systeem zit. Is dat niet het geval dan wordt er gestopt met een foutmelding. Daarna wordt al het mogelijk aanwezige RAM gevuld met nul-bytes om een beginsituatie te maken. Vervolgens wordt gekeken hoeveel RAM er nu werkelijk op de kaart zit. Dit gebeurt relatief eenvoudig: op iedere kilobytegrens wordt het nummer van die kilobyte weggeschreven. Dit proces begint bij kilobyte nummer 1023 en telt naar beneden tot kilobyte nummer 57. 56 en lager is immers het werkgeheugen van het programma en DOS65 zelf, en dat moet heel blijven. Als alle nummertjes zijn weggeschreven wordt er gezocht naar het hoogste kilobytenummer dat te vinden is: dat is de hoeveelheid RAM die werkelijk aanwezig is.

Pas dan begint het werkelijke testen. Dat gebeurt voor iedere kilobyte apart door $55, $AA, $FF, $01, $02, $04, $08, $10, $20, $40, $80, $FE, $FD, $FB, $F7, $EF, $DF, $BF, $7F en tenslotte $00 weg te schrijven en weer terug te lezen. Daarna volgt nog een test op kortgesloten adreslijnen, door slechts 1 bit weg te schrijven en te controleren of dit bit nergens anders opduikt.

Het programma sluit af met een test of het taaknummerregister werkt.

VTEST.MAC en VTEST.LIB staan op het BBS, en worden ook meegeleverd op diskette bij iedere print voor de 1 Mbyte RAMkaart.

Nico de Vries

```
ttl             'ftpFile: %sDate: %s Page %2d'
;***********************************************
;*                                             *
;* VTEST                                       *
;*                                             *
;* A test program for the VDISK RAM card.      *
;*                                             *
;* The program will determine the amount of    *
;* RAM present on the VDISK card and thou-     *
;* roughly test all RAM on the card.           *
;*                                             *
;***********************************************
;
; Name: VTEST.MAC
;
; Written by: Nico de Vries
;Mari Andriessenrade 49
;2907 MA  Capelle aan den IJssel
;The Netherlands
;
; History:
;
; 30-05-1993 Start of work
; 08-05-1994 V0.02: address of taskregister/datram changed to $FFE0
; 09-05-1994 V0.03: fixed bug in sizing, preventing finding size over 256k
; 09-05-1994 V0.04: added rudimentary stuck address line test
```

Fig. 1: VTEST.MAC, de RAM(p)tester

```
                ; 09-05-1994 V0.05: testing now starts at 56k, not at 64k
                ; 09-05-1994 V1.00: release version, equal to 0.05
                ; 09-05-1994 V1.01: bug fixed in card recognition part
                ;
                ; Description of DATRAM and TASKREG use:
                ;
                ; TASKREG:
                ; Initialized by IO65 at $FF
                ; All testing is done with tasknumber $FE loaded.
                ; The functioning of the taskregister is also tested.
                ;
                ; DATRAM:
                ; The 16 locations of DATRAM control the memory mapping as follows:
                ; DATRAM+0: Addresslines A12 through A19 for $0000-$0FFF
                ; DATRAM+1: Addresslines A12 through A19 for $1000-$1FFF
                ; DATRAM+2: Addresslines A12 through A19 for $2000-$2FFF
                ; DATRAM+3: Addresslines A12 through A19 for $3000-$3FFF
                ; DATRAM+4: Addresslines A12 through A19 for $4000-$4FFF
                ; DATRAM+5: Addresslines A12 through A19 for $5000-$5FFF
                ; DATRAM+6: Addresslines A12 through A19 for $6000-$6FFF
                ; DATRAM+7: Addresslines A12 through A19 for $7000-$7FFF
                ; DATRAM+8: Addresslines A12 through A19 for $8000-$8FFF
                ; DATRAM+9: Addresslines A12 through A19 for $9000-$9FFF
                ; DATRAM+A: Addresslines A12 through A19 for $A000-$AFFF
                ; DATRAM+B: Addresslines A12 through A19 for $B000-$BFFF
                ; DATRAM+C: Addresslines A12 through A19 for $C000-$CFFF
                ; DATRAM+D: Addresslines A12 through A19 for $D000-$DFFF
                ; DATRAM+E: Addresslines A12 through A19 for $E000-$EFFF: normally IO/video
                ; DATRAM+F: Addresslines A12 through A19 for $F000-$FFFF: normally IO65 ROM
                ;
                ; The data written into DATRAM+? is as follows:
                ; D7 D6 D5 D4 D3 D2 D1 D0
                ; |  |  |  |  |  |  |  |
                ; |  |  |  |  |  |  |  \------- A12 for $?000-$?FFF
                ; |  |  |  |  |  |  \---------- A13 for $?000-$?FFF
                ; |  |  |  |  |  \------------- A14 for $?000-$?FFF
                ; |  |  |  |  \---------------- A15 for $?000-$?FFF
                ; |  |  |  \------------------- A16 for $?000-$?FFF
                ; |  |  \---------------------- A17 for $?000-$?FFF
                ; |  \------------------------- A18 for $?000-$?FFF
                ; \---------------------------- A19 for $?000-$?FFF
                ;
                ; It is therefore possible to access the entire amount of RAM on the
                ; VDISK card through one window of 4 kbyte.
                ;
                ; VTEST uses the window at $1000 through $1FFF for testing.
                ;
                pag
                ;
                ;*** External Labels
                ;
rev             equ     "1V"                    ;note: backwards!
release         equ     "10"                    ;note: backwards!
                ;
                ;
                lib     vtest.lib
```

```
                ;
                pag
                ;
                org     $a000                   ;standard utility address
                ;
                ;*** Print sign-on message
                ;
vtest           jmp     vtest1                  ;skip info part and variables
                fcc     $c8,$c5,$cc,$d0         ;flag for help text
                fcc     $4c,$4c,$4c             ;dummy bytes
                fcc     'VTEST is a test program to verify that the'
                fcc     ' VDISK RAM card is operating correctly.\r'
                fcc     0                       ;end of help text
vtest1          jsr     prtext                  ;print sign-on message
                fcc     $0c,' VTEST '
                fdb     rev
                fcc     '.'
                fdb     release
                fcc     $1b,'i\r  by  NdV  ',$1b,'n\r'
                fcc     0
                jsr     remap                   ;set original mapping
                lda     $1000                   ;get 1 address of testwindow
                ldx     $1800                   ;get middle address
                ldy     $1fff                   ;get last too
                sta     deci1                   ;save
                stx     deci2                   ;values
                sty     deci3                   ;for later
                ldx     #$11                    ;select 17th 4k block on card
                stx     datram+1                ;to map at $1000-$1fff
                inc     $1000                   ;change
                lda     $1800                   ;all
                eor     #$ff                    ;three
                sta     $1800                   ;memory
                dec     $1fff                   ;locations
                ldx     #$01                    ;select 2nd 4k block on card
                stx     datram+1                ;to map at $1000-$1fff
                lda     $1000                   ;get data
                cmp     deci1                   ;if different
                bne     nocard                  ;no card is there
                lda     $1800                   ;get data
                cmp     deci2                   ;if different
                bne     nocard                  ;no card is there
                lda     $1fff                   ;if last data also different
                cmp     deci3                   ;report error
                beq     cardthr                 ;else test card
nocard          dec     $1000                   ;get
                lda     $1800                   ;original
                eor     #$ff                    ;data back
                sta     $1800                   ;in memory
                inc     $1fff                   ;locations
                jsr     prtext                  ;report error
                fcc     '\rThis system has no VDISK card!\r'
                fcc     0
                jmp     exit                    ;and exit
                ;
                ;*** Fill entire 1 Mbyte with zeroes to initialize the RAM
```

```
                ;*** The bottom 56 kbyte is skipped, because DOS65 and this
                ;*** program live there.
                ;
cardthr         jsr     prtext
                fcc     '\rFilling 1 Mbyte of RAM with zeroes... '
                fcc     0
                ldy     #$FE                    ;get tasknumber
                sty     taskreg                 ;set tasknumber
                ldy     #$ff                    ;get inner loop count
                lda     #0                      ;get data
1               sty     datram+1                ;use $1000-$1fff
                tax                             ;start at beginning
2               sta     $1000,x                 ;zero 1st kilobyte
                sta     $1100,x                 ;zero 1st kilobyte
                sta     $1200,x                 ;zero 1st kilobyte
                sta     $1300,x                 ;zero 1st kilobyte
                sta     $1400,x                 ;zero 2nd kilobyte
                sta     $1500,x                 ;zero 2nd kilobyte
                sta     $1600,x                 ;zero 2nd kilobyte
                sta     $1700,x                 ;zero 2nd kilobyte
                sta     $1800,x                 ;zero 3rd kilobyte
                sta     $1900,x                 ;zero 3rd kilobyte
                sta     $1A00,x                 ;zero 3rd kilobyte
                sta     $1B00,x                 ;zero 3rd kilobyte
                sta     $1C00,x                 ;zero 4th kilobyte
                sta     $1D00,x                 ;zero 4th kilobyte
                sta     $1E00,x                 ;zero 4th kilobyte
                sta     $1F00,x                 ;zero 4th kilobyte
                inx                             ;count bytes
                bne     2.b                     ;and repeat
                dey                             ;count loops
                cpy     #$0d                    ;if not entering 1st 56k
                bne     1.b                     ;loop
                jsr     prtext
                fcc     'Done.\r'
                fcc     0
                jsr     remap
                ;
                ;*** Size RAM by writing size pattern on start of each
                ;*** kilobyte.
                ;*** Then search for the highest size number.
                ;
                jsr     prtext
                fcc     '\rSizing RAM.... '
                fcc     0
                ldx     #3                      ;1 Mbyte is 1024 kbyte, is $3ff plus one
                lda     #$ff                    ;
                tay
1               sta     datram+1                ;use $1000-$1fff
                sty     $1c00                   ;store count low
                stx     $1c01                   ;store count high
                dey
                cpy     #$ff                    ;
                bne     2.f                     ;if Y went from 0 to FF
                dex                             ;adapt X
2               sty     $1800                   ;store count low
```

```
        stx     $1801           ;store count high
        dey
        cpy     #$ff
        bne     3.f             ;if Y went from 0 to FF
        dex                     ;adapt X
3       sty     $1400           ;store count low
        stx     $1401           ;store count high
        dey
        cpy     #$ff
        bne     4.f             ;if Y went from 0 to FF
        dex                     ;adapt X
4       sty     $1000           ;store count low
        stx     $1001           ;store count high
        dey
        cpy     #$ff
        bne     5.f             ;if Y went from 0 to FF
        dex                     ;adapt X
5       sec     ;prepare for subtract
        sbc     #1              ;decrement A
        cmp     #$0d            ;if entering lowest 56k
        beq     6.f             ;exit loop
        jmp     1.b             ;and loop
6       lda     #$0e            ;start at 56k
        ldy     #56             ;get expected size low
        ldx     #0              ;get expected size high
7       sta     datram+1        ;set mapping
        cpy     $1000           ;compare size low
        bne     sizefnd         ;if different, go print size
        cpx     $1001           ;compare size high
        bne     sizefnd         ;if different, go print size
        iny                     ;adapt size
        bne     8.f             ;if Y=0
        inx                     ;adapt X
8       cpy     $1400           ;compare size low
        bne     sizefnd         ;if different, go print size
        cpx     $1401           ;compare size high
        bne     sizefnd         ;if different, go print size
        iny                     ;adapt size
        bne     9.f             ;if Y=0
        inx                     ;adapt X
9       cpy     $1800           ;compare size low
        bne     sizefnd         ;if different, go print size
        cpx     $1801           ;compare size high
        bne     sizefnd         ;if different, go print size
        iny                     ;adapt size
        bne     10.f            ;if Y=0
        inx                     ;adapt X
10      cpy     $1c00           ;compare size low
        bne     sizefnd         ;if different, go print size
        cpx     $1c01           ;compare size high
        bne     sizefnd         ;if different, go print size
        iny                     ;adapt size
        bne     11.f            ;if Y=0
        inx                     ;adapt X
11      clc                     ;prepare for add
        adc     #1              ;adapt mapping
```

```
                bne     7.b             ;if not all done, loop
sizefnd         sty     ramsize         ;store size low
                stx     ramsize + 1     ;and hi
                txa                     ;get high
                pha                     ;save high
                tya                     ;get low in A
                tax                     ;and in X
                pla                     ;get high in A
                jsr     remap
                jsr     praxdec         ;print size in decimal
                jsr     prtext
                fcc     ' kbyte found.\r\r'
                fcc     0
                ;
                ;*** Test RAM.
                ;
                ldx     #56             ;get low
                lda     #0              ;get high
                stx     curkilo         ;set low
                sta     curkilo + 1     ;set high
1               lda     curkilo + 1     ;get current kilobyte high
                cmp     ramsize + 1     ;if not same as end
                bne     2.f             ;adapt and test
                lda     curkilo         ;else get low
                cmp     ramsize         ;if same as last
                beq     testOK          ;finish test
2               ldx     #1              ;column#
                ldy     #8              ;line#
                jsr     crsadr          ;position cursor
                jsr     prtext
                fcc     'Testing RAM... '
                fcc     0
                ldx     curkilo         ;get low
                lda     curkilo + 1     ;get high
                jsr     testram         ;test 1 kilobyte of RAM
                php                     ;save the flags
                inc     curkilo         ;else adapt low
                bne     3.f             ;if not zero, restore flags
                inc     curkilo + 1     ;else adapt high
3               plp                     ;restore flags
                bcs     endtest         ;if error, stop now
                ldx     curkilo         ;get low
                lda     curkilo + 1     ;get high
                jsr     praxdec         ;print in decimal
                jsr     prtext
                fcc     ' kbyte OK.'
                fcc     0
                jmp     1.b             ;and test
endtest         jsr     crlf            ;go to new line
                jsr     prtext
                fcc     $1b,'iRAM error at '
                fcc     0
                ldx     curkilo         ;get low
                lda     curkilo + 1     ;get high
                jsr     praxdec         ;print in decimal
                jsr     prtext
```

```
                fcc     ' kbyte.',$1b,'n'
                fcc     0
testOK          jsr     crlf                    ;go to new line
                ;
                ;*** Test tasknumberregister by changing the mapping of
                ;*** $1000-$1FFF 256 times and verifying that the window
                ;*** can be changed by switching tasks only.
                ;
                jsr     remap
                lda     $1000                   ;get original data
                sta     deci1                   ;save it
                lda     $1fff                   ;get original data
                sta     deci2                   ;save it
                lda     ramsize+1               ;get ramsize high
                sta     work                    ;copy it
                lda     ramsize                 ;get size low
                lsr     work                    ;get maximum
                rora    ;task
                lsr     work                    ;count
                rora                            ;in A
                sta     work                    ;and in work
                jsr     prtext                  ;
                fcc     '\rTesting taskregister.... '
                fcc     0
                ldy     #0
                ldx     #$10                    ;start with 65k
1               cpx     work                    ;if RAM mapped
                beq     testtsk                 ;check storage
                sty     taskreg                 ;set task register
                stx     datram+1                ;map $1000-$1fff
                tya                             ;get data
                sta     $1000                   ;move to begin
                sta     $1fff                   ;and end of window
                iny                             ;else get next task
                inx                             ;and next map
                bne     1.b                     ;if not zero, loop
testtsk         ldy     #0                      ;else get task zero back
                ldx     #$10                    ;start with 65k
2               cpx     work                    ;if entire RAM mapped
                beq     taskOK                  ;report no errors
                sty     taskreg                 ;set task register
                stx     datram+1                ;map $1000-$1fff
                tya                             ;get data
                cmp     $1000                   ;check data
                bne     taskerr                 ;if error, report it
                cmp     $1fff                   ;check data at end of window
                bne     taskerr                 ;if error, report it
                iny                             ;else get next task
                inx                             ;and next map
                bne     2.b                     ;then loop
taskerr         jsr     prtext                  ;
                fcc     $1b,'iError.',$1b,'n\r'
                fcc     0
                jmp     exit                    ;and proceed
taskOK          jsr     prtext                  ;
                fcc     'OK.\r'
```

```
                fcc     0
                ;
                ;*** Exit program.
                ;
exit            lda     deci1           ;get original data
                sta     $1000           ;save it
                lda     deci2           ;in original
                sta     $1fff           ;location
                jsr     remap           ;get original mapping for $1000-$1fff
                lda     #0              ;get tasknumber
                sta     taskreg         ;set task
                lda     #$01            ;get mapping for 2nd 4 kbyte
                sta     datram+1        ;set mapping RAM
                jmp     crlf            ;start on next line and exit
                ;
                ;*** REMAP: set original mapping for $1000-$1FFF
                ;
remap           pha                     ;save A
                lda     #$FE            ;get tasknumber
                sta     taskreg         ;set task
                lda     #$01            ;get mapping for 2nd 4 kbyte
                sta     datram+1        ;set mapping RAM
                pla                     ;restore A
                rts                     ;and exit
                ;
                ;*** PRAXDEC: print AX in decimal
                ;NOTE: Stolen from IO65!
praxdec         jsr     conver          ;convert to decimals
                pha                     ;save A
                txa                     ;save X
                pha                     ;too
                jsr     pridec          ;print result
                pla                     ;get old X
                tax                     ;in X
                pla                     ;get old A
                rts                     ;and exit
                ;
                ;*** Convert from 2 hex bytes to 3 decimal bytes ***
                ;NOTE: Stolren from IO65!
conver          stx     deci1           ;store low
                sta     deci2           ;and high bytes
                pha                     ;save A
                txa                     ;save X
                pha                     ;too
                lda     #0              ;zero result
                sta     deci3           ;area
                tax                     ;get a zero byte
                sed                     ;set decimal mode
                beq     hdf             ;and start (branch always)
hda             dec     deci2           ;adapt hi
                txa                     ;get units
                adc     #$56            ;add 56
                tax                     ;to units
                pla                     ;get hundreds
                adc     #$02            ;add 200
                bcc     hdf             ;if no carry, loop
```

```
                inc     deci3           ;else adapt tenthousands
                bne     hdf             ;and loop (branch always)
hdb             dec     deci1           ;decrement low
                txa                     ;get units
                adc     #1              ;add one
                tax                     ;save units
                pla                     ;restore carry
                adc     #0              ;add carry
hdf             pha                     ;save carry byte
                clc                     ;prepare for add
                lda     deci1           ;get low
                bne     hdb             ;if not zero, add one to units
                lda     deci2           ;if hi zero, end
                bne     hda             ;else convert hi byte
                cld                     ;set hex mode
                pla                     ;restore hundreds
                sta     deci2           ;store them as results
                stx     deci1           ;store units too
                pla                     ;get old X
                tax                     ;in X
                pla                     ;get old A
                rts                     ;and exit
                ;
                ;*** Print decimal numbers ***
                ;NOTE: Stolen from IO65!
pridec          lda     deci3           ;get tenthousands
                bne     conoe           ;if any, print them
                lda     deci2           ;get hundreds
                bne     conof           ;if any, print them
                lda     deci1           ;else get units
                jmp     nhxout          ;and print least sign. byte
conoe           jsr     nhxout          ;print tenthousands
                lda     deci2           ;get hundreds
                jsr     prbyt           ;print them
conog           lda     deci1           ;get units
                jmp     prbyt           ;print them
conof           jsr     nhxout          ;print hundreds
                jmp     conog           ;and print units
                ;
                ;*** Print a byte (in decimal mode)
                ;NOTE: Stolen from IO65!
nhxout          cmp     #$10            ;if smaller than 10 (decimal mode)
                bcc     prnibl          ;print lo nibble only
                ;
                ;*** Print a byte (in hex mode)
                ;
prbyt           pha                     ;else save the byte
                lsra                    ;get high
                lsra                    ;nibble
                lsra                    ;into
                lsra                    ;low nibble
                jsr     prnibl          ;print it
                pla                     ;restore the byte and:
                ;
                ;*** Print low nibble as hex digit
                ;
```

```
prnibl      and     #%00001111          ;get low nibble only
            cmp     #$0a                ;Number or A...F
            bcc     na                  ;if below A, go convert
            adc     #$06                ;else add offset for letter
na          adc     #$30                ;Calculate ASCII value
            jmp     output              ;print digit
            ;
            ;*** TESTRAM: test 1 kilobyte of RAM starting at curkilo
            ;
testram     pha                         ;save
            tya                         ;pricipal
            pha                         ;registers
            txa                         ;on
            pha                         ;stack
            lda     curkilo+1           ;get curent kilobyte, high
            sta     work                ;in work area
            lda     curkilo             ;get current kilobyte, low
            lsr     work                ;divide
            rora                        ;current
            lsr     work                ;kilobyte
            rora                        ;by four
            sta     datram+1            ;map that into $1000-$1fff
            lda     curkilo             ;get current kilobyte, low
            and     #%00000011          ;get kilobyte number within frame
            asla                        ;multiply by
            asla                        ;four
            clc                         ;prepare for add
            adc     #$10                ;add frame address high
            sta     rcurr+1             ;set zero page address
            ldy     #0                  ;get low
            sty     rcurr               ;set that too
            lda     #$55                ;get odd checkerboard
            jsr     test1k              ;test 1 kilobyte
            bcs     ramp2               ;if error, exit
            lda     #$AA                ;get even checkerboard
            jsr     test1k              ;test 1 kilobyte
            bcs     ramp2               ;if error, exit
            lda     #$FF                ;get all ones
            jsr     test1k              ;test 1 kilobyte
            bcs     ramp2               ;if error, exit
            lda     #%00000001          ;single bit
bitloop     jsr     test1k              ;test 1 kilobyte
            bcs     ramp2               ;if error, exit
            asla                        ;shift single bit
            bcc     bitloop             ;and repeat
            lda     #%11111110          ;single bit
bitlop2     jsr     test1k              ;test 1 kilobyte
            bcs     ramp2               ;if error, exit
            sec                         ;shift in a  1
            rola                        ;shift single bit
            bcs     bitlop2             ;and repeat
            clc                         ;reset error flag
            jsr     testadr             ;test for stuck address lines
            bcs     ramp2               ;if error, set carry
            lda     #0                  ;
            jsr     test1k              ;test 1 kilobyte and leave zeroes
```

```
ramp2   pla                     ;restore
        tax                     ;saved
        pla                     ;registers
        tay                     ;from
        pla                     ;stack
        rts                     ;and exit
        ;
        ;*** TEST1K: test 1 kilobyte of RAM with pattern in A
        ;
test1k  ldy     #0              ;get start offset
1       sta     [rcurr],y       ;store data
        iny                     ;do next byte
        bne     1.b             ;if not whole page done, repeat
        inc     rcurr+1         ;else adapt high byte of address
2       sta     [rcurr],y       ;store data
        iny                     ;do next byte
        bne     2.b             ;if not whole page done, repeat
        inc     rcurr+1         ;else adapt high byte of address
3       sta     [rcurr],y       ;store data
        iny                     ;do next byte
        bne     3.b             ;if not whole page done, repeat
        inc     rcurr+1         ;else adapt high byte of address
4       sta     [rcurr],y       ;store data
        iny                     ;do next byte
        bne     4.b             ;if not whole page done, repeat
5       cmp     [rcurr],y       ;compare data
        bne     ramp            ;if error, exit
        iny                     ;do next byte
        bne     5.b             ;if not whole page done, repeat
        dec     rcurr+1         ;else adapt high byte of address
6       cmp     [rcurr],y       ;compare data
        bne     ramp            ;if error, exit
        iny                     ;do next byte
        bne     6.b             ;if not whole page done, repeat
        dec     rcurr+1         ;else adapt high byte of address
7       cmp     [rcurr],y       ;compare data
        bne     ramp            ;if error, exit
        iny                     ;do next byte
        bne     7.b             ;if not whole page done, repeat
        dec     rcurr+1         ;else adapt high byte of address
8       cmp     [rcurr],y       ;compare data
        bne     ramp            ;if error, exit
        iny                     ;do next byte
        bne     8.b             ;if not whole page done, repeat
        clc
        fcc     $24             ;skip next instruction
ramp    sec
        rts
        ;
        ;*** TESTADR: Test 1 kilobyte for stuck address lines
        ;
testadr ldy     #0              ;get start offset
        tya                     ;A=0
1       sta     [rcurr],y       ;store data
        iny                     ;do next byte
        bne     1.b             ;if not whole page done, repeat
```

```
                inc     rcurr+1         ;else adapt high byte of address
2               sta     [rcurr],y       ;store data
                iny                     ;do next byte
                bne     2.b             ;if not whole page done, repeat
                inc     rcurr+1         ;else adapt high byte of address
3               sta     [rcurr],y       ;store data
                iny                     ;do next byte
                bne     3.b             ;if not whole page done, repeat
                inc     rcurr+1         ;else adapt high byte of address
4               sta     [rcurr],y       ;store data
                iny                     ;do next byte
                bne     4.b             ;if not whole page done, repeat
                lda     #%00000001      ;get single bit
                sta     deci1           ;save pattern
                lda     #%00000000      ;get default data
                sta     deci2           ;save too
adrloop         jsr     adrtest         ;test for stuck address lines
                bcs     erradr          ;if error, exit
                asl     deci1           ;then shift bit
                bcc     adrloop         ;if not all bits tested, loop
                ;repeat test with inverted data
                ldy     #0              ;get start offset
                lda     #$ff            ;
1               sta     [rcurr],y       ;store data
                iny                     ;do next byte
                bne     1.b             ;if not whole page done, repeat
                inc     rcurr+1         ;else adapt high byte of address
2               sta     [rcurr],y       ;store data
                iny                     ;do next byte
                bne     2.b             ;if not whole page done, repeat
                inc     rcurr+1         ;else adapt high byte of address
3               sta     [rcurr],y       ;store data
                iny                     ;do next byte
                bne     3.b             ;if not whole page done, repeat
                inc     rcurr+1         ;else adapt high byte of address
4               sta     [rcurr],y       ;store data
                iny                     ;do next byte
                bne     4.b             ;if not whole page done, repeat
                lda     #%11111110      ;get single bit
                sta     deci1           ;save pattern
                lda     #%11111111      ;get default data
                sta     deci2           ;save too
adrlop2         jsr     adrtest         ;test for stuck address lines
                bcs     erradr          ;if error, exit
                sec                     ;shift in a 1
                rol     deci1           ;shift bits
                bcs     adrlop2         ;if not all bits tested, loop
                clc
                rts
erradr          sec
                rts
                ;
                ;*** ADRTEST: store and verify single bit pattern
                ;
adrtest         ldy     deci1           ;get offset
                lda     deci1           ;get data
```

```
                sta     [rcurr],y       ;store in RAM
                ldy     #0              ;get start offset
                lda     deci2           ;get default data
5               cmp     [rcurr],y       ;if RAM location is zero
                beq     51.f            ;do next byte
                lda     [rcurr],y       ;else fetch data
                cmp     deci1           ;if not test pattern
                bne     adrerr          ;then address error
                cpy     deci1           ;if test pattern, but not
                bne     adrerr          ;on correct address, then address error
                lda     deci2           ;else get default data back
51              iny                     ;else do next byte
                bne     5.b             ;if not whole page done, repeat
                dec     rcurr+1         ;else adapt high byte of address
6               cmp     [rcurr],y       ;if RAM location is not zero
                bne     adrerr          ;then address error
                iny                     ;else do next byte
                bne     6.b             ;if not whole page done, repeat
                dec     rcurr+1         ;else adapt high byte of address
7               cmp     [rcurr],y       ;if RAM location is not zero
                bne     adrerr          ;then address error
                iny                     ;else do next byte
                bne     7.b             ;if not whole page done, repeat
                dec     rcurr+1         ;else adapt high byte of address
8               cmp     [rcurr],y       ;if RAM location is not zero
                bne     adrerr          ;then address error
                iny                     ;else do next byte
                bne     8.b             ;if not whole page done, repeat
                inc     rcurr+1         ;else adapt
                inc     rcurr+1         ;high byte
                inc     rcurr+1         ;of address
                ldy     deci1           ;get offset
                lda     deci2           ;get default data
                sta     [rcurr],y       ;overwrite test byte in RAM
                clc
                rts                     ;and exit

adrerr          ;pha                    ;save data
;               tya                     ;get offset
;               clc                     ;prepare for add
;               adc     rcurr           ;add to addres low
;               sta     rcurr           ;store new low
;               bcc     nocary          ;if a carry remained
;               inc     rcurr+1         ;adapt high byte
;nocary         jsr     prtext          ;report error
;               fcc     '\rData: '
;               fcc     0
;               pla                     ;get data back
;               jsr     prbyte          ;print data
;               jsr     prtext          ;report error
;               fcc     ' Physical address: '
;               fcc     0
;               lda     rcurr+1         ;get address high
;               jsr     prbyte          ;print address high
;               lda     rcurr           ;get address low
;               jsr     prbyte          ;print address low
```

Virtual disk op de nieuwe RAMkaart

In dit artikeltje wordt een nieuwe virtual FORMAT voorsgeteld, en wordt uit de doeken gedaan hoe de nieuwe virtual disk aan de praat te krijgen is. Het blijkt namelijk dat er ook an DOS65 zelf nog wat veranderd moet worden om de virtual disk op de nieuwe RAMkaart aan de praat te krijgen.

Wijzigingen in DOS65

Om de VRAM kaart te laten werken moet het onderstaande in DOS65 worden gewijzigd. Maak hiervoor eerst een nieuwe systeemschijf aan. Dit moet als volgt:

- Doe uw originele systeemschijf in drive 0:
- Doe een blanco diskette is drive 1:
- Formatteer de blanco de diskette;
- Kopieer alle files van de systeemdiskette naar drive 1:

In drive 1: zit nu uw nieuwe systeemdiskette.Start vervolgens de monitor op en verander de volgende zaken:

Er was:				Dat wordt:		
$CE4A	wvirt	SEC		SEC		
	rvirt	PHP		PHP		
		SEI		SEI		
		STA	$FFFD	STA	$FFED	
		LDA	$FF	LDA	$FF	
$CE6E		PLA		PLA		
		STA	$FF	STA	$FF	
		LDA	#$02	LDA	#$0D	
		STA	$FFFD	STA	$FFED	
		PLP		PLP		
$CEF9		TXA		TXA		
		ADC	$CC03	ADC	$CC03	
		EOR	#$0F	EOR	#$00	
		BCC	$CF17	BCC	$CF17	
		RTS		RTS		

Kort samengevat, u moet met de monitor de volgende bytes wijzigen:

adres	oude en nieuwe waarde	
$CE4E	FD	ED
$CE72	02	0D
$CE74	FD	ED
$CEFE	0F	00

Als dit gedaan is, de nieuwe BOOT wegschrijven met:

SAVE BOOT BF00,DFFF en vervolgens:BOOT-LINK BOOT

Doe de nieuwe systeemdiskette in drive 0: en reset het systeem. Het systeem moet nu vanaf de nieuwe drive starten. Nu kunt u de nieuwe virtual format gebruiken als 640 kByte virtual disk. Let op: Uw virtual disk wordt drive 2:. Veel plezier ermee.

Voor de rest...

Enkele opmerkingen: We hopen binnenkort de virtual format zo te verbeteren dat er 1 MByte virtual disk mogelijk is. Nuttig gebruik van VF: De meeste mensen hebben 2 drives, dan is het kopieren van een schijf ietwat onhandig, omdat de systeemschijf een drive in beslag neemt. Nu kunt u echter veel handiger kopieren: Als drive 0: de systeem schijf is, dan

- copy 1:* 2:
- floppy verwisselen
- copy 2:* 1:

Ook kunt u bijvoorbeeld uw hele systeemschijf kopieren op virtual disk, dus:

- vf
- copy 0:* 2:
- asn s2

Daarna gebruikt uw computer deze virtual disk als systeemschijf en dus starten allerlei commando's dan wel heel erg snel, en zonder allerlei herrie van uw floppen machines.

Henk Speksnijder

```
; file vformat.mac
; Format the 1 MByte RAM card as 640 kByte virtual disk.
; 5 aug. 1994 H.Speksnijder

;DOS65 i/o and special adresses
inch        equ     $C020       get character from keyboard
crlf        equ     $C02F       print CR + LF
prch        equ     $C023       print character
prhx        equ     $C038       print byte hexadecimal
prtx        equ     $C03b       print text
loupch      equ     $C041       lower to uppercase
sopt        equ     $C068       scan options
voffset     equ     $CC03       voffset for DOS65
vtracks     equ     $CC04       vtracks for DOS65
date        equ     $ABFC       4 bytes date & time

;RAM-card adresses
offset      equ     $0E         15 blocks = 60 kB ram is standard work memo
vmem        equ     $2000       memory adress used by vdisk
vdat        equ     $FFE2       adress to switch sectors of vdisk
vmap        equ     $02         original datablock at vdat

;REMARK: Changes in Dos65 BOOT:
;adres      data    explanation
;$CEFE      $00     was eor #$0f becomes eor #00
;$CE4E      $ED     was sta $FFFD becomes sta $FFED
;$CE72      $0D     was lda #02 becomes lda #$0D
;$CE74      $ED     was sta $FFFD becomes sta $FFED

            org     $200
vf          jsr     sopt        scan options
            fcc     'Y',0       only option is -Y
            bcs     90.f        if illegal option(s), report
            stx     opt         else store option
            jsr     testrom     test if you have the new EPROM
            bcs     9.f         if error, exit
            jsr     testboo     test if you hace the right BOOT
            bcs     9.f         if error, exit
            jsr     ramcard     test for card, accu=0= NO card
            beq     91.f        if no card, report that
1           bit     opt         if '-Y' then skip askinit
            bmi     2.f
            jsr     askinit     ask if you want to initialize
            bcs     9.f         if not, exit now
2           jsr     search      else count memory size
            jsr     vdinit      initalize Vdisk
            bcs     9.f         if error, exit
            jsr     seevd       else show Vdisk size to user
9           rts                 and exit to DOS
90          jmp     vderr       incorrect options
91          jmp     vdserr      no card found
```

Fig. 1: VF.MAC

```
;TESTROM test if you have the old eprom or the new eprom.
;This routine compares a table with the eprom.
testrom     ldx     #22             get byte count
1           lda     $F033,x         get EPROM byte
            cmp     tabold,x        compare data according to old eprom
            bne     2.f             must be different
            dex                     if not, proceed
            bne     1.b             if old EPROM
            jmp     11.f            detected, report error and exit
2           ldx     #22             else get byte count
3           lda     $F033,x         get EPROM byte
            cmp     tabnew,x        compare data according to new eprom
            bne     10.f            if same
            dex                     do next byte
            bne     3.b             if done
9           clc                     clc = EPROM is OK
            rts                     and exit
10          jsr     prtx            else print string
            fcc     '\rUnknown EPROM detected in your CPU-card.'
            fcc     '\rContinue with format anyway ? Y/N: ',0
            jmp     getyn
11          jsr     prtx
            fcc     '\rOld EPROM detected in your CPU-card.'
            fcc     '\rThis should be replaced.\r',0
            sec                     sec = not ok
            rts

;TABLE with ram initialisation of old eprom
tabold      fcc     $D8               cld
            fcc     $78               sei
            fcc     $A2,$0F           ldx #$0f
            fcc     $8A             1 txa
            fcc     $49,$0F           eor #$0f
            fcc     $9D,$F0,$FF       sta $fff0
            fcc     $CA               dex
            fcc     $10,$F7           bpl 1.b
            fcc     $A9,$00           lda #$00
            fcc     $A2,$51           ldx #$51
            fcc     $9D,$FF,$E6     2 sta $ffe6
            fcc     $CA               dex
            fcc     $D0,$FA           bne 2.b

;TABLE with ram initialisation of new eprom
tabnew      fcc     $78               sei
            fcc     $D8               cld
            fcc     $A0,$00           ldy $00
            fcc     $8C,$F0,$FF     1 sty $fff0
            fcc     $A2,$0F           ldx #$0f
            fcc     $8A             2 txa
            fcc     $9D,$E0,$FF       sta $ffe0,x
            fcc     $CA               dex
            fcc     $10,$F9           bpl 2.b
            fcc     $C8               iny
```

```
                fcc     $D0,$F1             bne 1.b
                fcc     $8C,$F0,$FF         sty $fff0
                fcc     $9A                 txs

;TESTBOO check if you used the correct BOOT
testboo         lda     $CEFE               get location
                cmp     #$00                if not modified
                bne     7.f                 report error
                lda     $CE4E               get next location
                cmp     #$ED                if not modified
                bne     7.f                 report error
                lda     $CE72               get location
                cmp     #$0D                if not modified
                bne     7.f                 report error
                lda     $CE74               get location
                cmp     #$ED                if not modified
                bne     7.f                 report error
9               clc                         CLC=OK
                rts                         exit to caller
7               jsr     prtx                print string
                fcc     '\rThis is NOT the expected version of BOOT.\r'
                fcc     'Is this boot suitable for the new RAM-card ? Y/N: ',0
                jmp     getyn

;RAMCARD  test if a ramcard (new model) is in your system
; remark: this routine will not destroy any data in
; your memory or in the virtual disk.
ramcard         php                         save flags
                sei                         no interrupts, please
                lda     vmem                get data
                sta     temp                save it
                lda     #0                  write $00 in this block
                sta     vmem
                lda     #offset             switch other block in place
                sta     vdat
                lda     vmem                save data from block
                sta     temp+1
                lda     #$FF                write $FF in this block
                sta     vmem
                lda     #vmap               switch to standard block
                sta     vdat
                lda     vmem                see data in standard block
                bne     1.f                 if data=0
                lda     #offset             then card present
                sta     vdat                switch other block
                lda     temp+1              restore data there
                sta     vmem
                lda     #vmap               switch standard block
                sta     vdat
                ldx     #1
                bne     2.f
1               ldx     #0                  else NO card present
2               lda     temp                restore data
                sta     vmem
```

```
                plp                     restore flags (IRQ back on)
                txa                     accu=0=no card, accu=1=card
                rts

;SEARCH how much memory you have
; by counting the number of blocks, each block=4kB
; remark: return carry=0=ok, carry=1=no RAM card
; remark: if more than 160 tracks, use only 160 tracks.
search          lda     #offset
                sta     voffset
                sta     mut             mut = memory under test
1               jsr     is4k            test 4k, accu=0=not found
                beq     2.f
                inc     mut
                bne     1.b
2               sec
                lda     mut
                sbc     #offset
                cmp     #161
                bcc     3.f
                lda     #160            accept only < 160 tracks
3               sta     vtracks
                rts

is4k            php
                sei
                lda     mut
                sta     vdat            segment under test
                ldx     #0
                lda     #$55
                sta     vmem            test with $55
                eor     vmem
                bne     7.f
                lda     #$AA
                sta     vmem            test with $AA
                eor     vmem
                bne     7.f
                inx
7               lda     #vmap           restore old segment
                sta     vdat
                plp
                txa
                rts
61              jmp     vdserr          illegal size

vdinit          ldy     voffset
                tya
                clc
                adc     vtracks
                bcc     1.f
                bne     61.b
1               cmp     voffset         check if number of tracks is correct
                beq     61.b
                php
```

```
                sei                             no interrupts
                lda     voffset
                sta     vdat
                ldx     #0
                txa
2               sta     vmem,x                  clear sis
                sta     vmem+$100,x
                sta     vmem+$200,x             clear 1st dir
                sta     vmem+$300,x             clear 2nd dir
                dex
                bne     2.b
                ldx     #16
3               txa
                sta     vmem-1,x                sector lookup table
                dex
                bne     3.b
                lda     vtracks                 set tracks and sectors
                sta     vmem+$21                tracks
                lda     #16
                sta     vmem+$22                sectors/cilinder
                sta     vmem+$23                sectors/cilinder/side
                lda     #$FF
                sta     vmem+$24                free sector map
                sta     vmem+$25
                sta     vmem+$26
                sta     vmem+$27

                ldx     #$FF                    name
4               inx
                lda     vdname,x
                sta     vmem+$40,x
                bne     4.b
                ldx     #3                      copy date & time
5               lda     date,x
                sta     vmem+$58,x              create
                sta     vmem+$5C,x              modified
                dex
                bpl     5.b

                lda     #1
                sta     vmem+$28                allocate count=1    (sectors each bit)
                sta     vmem+$29                track shift=1
                lda     #2
                sta     vmem+$2C                data allocation track
                lda     #$F8
                ldx     vtracks                 set free bitmap
                cpx     #80+1                   check lower equal 80
                bcc     6.f                     branch if vtracks Q
                txa
                lsra
                tax
                inc     vmem+$28                allocate 2 sectors
                dec     vmem+$29                no track shift
```

```
                lda     #$04
                sta     vmem+$2F1       pointer in directory

                lda     #$FC
6               ldy     #$60
7               sta     vmem,y          set free
                lda     #$FF            ( make 16 bits/track )
                iny
                sta     vmem,y
                iny
                dex
                bne     7.b
                lda     #vmap           original segment
                sta     vdat
                plp
                clc
                rts

vdname          fcc     "Virtual disk 640k HS",0

vderr           jmp     vderr0
                fcc     $C8,$C5,$CC,$D0
vderr0          jsr     prtx
                fcc     "\rInitializes virtual disk"
                fcc     "\rSyntax: VFormat [-Y]"
                fcc     "\rOption: -Y : don't ask for permission."
                fcc     "\rAbbreviation: VF\r",0
                sec
                rts

vdserr          jsr     prtx
                fcc     "No or illegal virtual disk.\r",0
                sec
                rts

askinit         jsr     prtx
                fcc     "Are you sure to initialize the virtual disk ? Y/N: ",0

getyn           jsr     inch
                jsr     loupch
                cmp     #'Y
                beq     9.f
                jsr     crlf
                sec                     when user entered not Y then carry=1
                rts
9               jsr     crlf
                clc                     when user entered Y then carry=0
                rts

seevd           jsr     prtx
                fcc     "Virtual disk: ",0
                lda     voffset         print adress of first block
                jsr     prhx
                jsr     prtx
```

DOS65: RAM(P)-kaart?

Huidige status

Er nu verschillende kaarten gebouwd. Ze werken allemaal goed. Sommigen hebben moeite met het verkrijgen van het batterijtje, dat gespecificeerd is als een Lithium cel van 3.6 Volt. Die blijken schaars. De standaardwaarde van 3 Volt is ook goed. De Dallas DS1210 battery controller vindt namelijk dat de batterij pas leeg is als de spanning is gezakt tot 2.0 Volt. Marge genoeg dus.

Een NiCd accu is niet echt aanbevolen, ofschoon het wel kan. Als eerste bezwaar geldt dat er op de print geen laadcircuit is voorzien, zodat de accu tijdens bedrijf niet geladen wordt. Een tweede bezwaar is, dat de DS1210 geen hogere spanningen dan 4 Volt op de batterij-ingang mag hebben. Dus als u een NiCd met laadcircuit wilt gebruiken, neem dan een NiCd van 3.6 Volt (dus 3 cellen) en overbrug de accu met een zenerdiode van 3.9 Volt. Dan blijft in ieder geval de DS1210 heel. Het laadcircuit bestaat uit een 1N4148 en een weerstand die in serie tussen de +5V en de plus van de accu worden geschakeld. De waarde van de weerstand wordt bepaald door de laadstroom. Nominaal valt er 5 - 0.6 - 3.6 = 0.8 V over de weerstand. Bij een accu van 60 mAh is de laadstroom maximaal 6 mA, zodat de weerstand in dat geval minimaal 150 ohm moet zijn.

Vroeger is keurig beweerd dat de nieuwe RAM-kaart opwaards compatibel zou zijn.

Een ander probleem blijkt de real-time clock te zijn. Sommigen rapporteerden een hangend systeem. Het vermoeden gaat op dit moment in twee richtingen:

- De interrupt van de real-time clock wordt in IO65 V2.14 niet goed afgehandeld;
- Een residente routine doet iets heel smerigs door direct een routine in IO65 aan te roepen.

Het onderzoek loopt nog. U hoort er nog van.

Adressering

In een van de vorige artikelen is keurig beweerd dat de nieuwe RAM-kaart opwaards compatibel zou zijn met de virtual diskkaart met DRAMs. De oplettende lezer en bouwer heeft aan het vorige nummer kunnen zien dat dat eigenlijk niet klopt. Wat is het geval?

Op de versie met DRAMs ontbreekt het taaknummerregister. Het mapping-RAM bestaat uit twee 74S189's die geadresseerd worden op $FFF0 tot en met $FFFF. De SRAM-versie heeft op $FFF0 het taaknummerregister, terwijl het mapping-RAM op $FFE0 tot en met $FFEF te vinden is. Conclusie: niet zo heel erg compatibel. Aan het artikel in het vorige nummer over de virtual disk op de nieuwe kaart was dat dan ook te zien: alle adressen tussen $FFF0 en $FFFF moesten $10 teruggefloten worden.

Nieuwe wegen?

Op de laatste bestuursvergadering is dit onderwerp kort ter sprake geweest. Het bestuur vond dat de ontwerper teruggefloten moest worden, en dat de adressering op de kaart terug moest naar de oude situatie, dus taaknummerregister op $FFE0 en het mapping RAM op $FFF0 tot en met $FFFF. Dat levert de volgende zaken op:

- Nieuwe PAL. Helaas zijn de geleverde PALs niet herprogrammeerbaar.
- Nieuwe IO65. Deze zit in EPROM, kan dus wel worden overgeprogrammeerd.
- Nieuwe utilities. De bij de print gelverde diskette bevat het testprogramma VTEST en ook de in het vorige nummer gepresenteerde gewijzigde virtual disk. De meeste wijzigingen voor de virtual disk moeten dan weer ongedaan worden.

Omdat deze wijzigingen eigenlijk niet zoveel schade opleveren (ik hoop niemand de PAL direct in de print gesoldeerd heeft) is min of meer besloten om de adressering om te draaien. De bestellers van de printer krijgen daarom binnkort een pakketje thuis met de volgende inhoud:

- Nieuwe PAL
- Nieuwe utility diskette
- Nieuwe IO65 in EPROM

Het pakketje zal kosteloos zijn.

Voor hen die niet kunnen wachten is in de figuur de nieuwe inhoud voor de PAL gegeven. De daar getoonde file is voor het pakket ABEL van Data I/O. De enige wijziging is dat waar !A4 stond nu A4 staat, en andersom. Samen met de gegevens uit het artikel uit het vorige nummer is de klus dan eventueel ook zonder externe steun te klaren.

De DOS65 werkgroep biedt alvast de excuses aan voor het ongemak. Maar we staan nu nog aan het begin, zodat de verandering met relatief weinig moeite doorgevoerd kan worden. Namens de werkgroep,

Nico de Vries

```
module _VDISK3B;                    flag '-r0';

title
'VDISK3B 5th VERSION                N. DE VRIES  01-11-1994
 TMM2064 CONTROLLER PAL FOR DOS65 VDISK CARD WITH SRAMS, 1M VERSION
 NDV FIRMWARE';

"VDISK3B device 'P20V8S';
VDISK3B device 'P22V10';

"declarations

        H,L = 1,0;
        X,Z,C = .X.,.Z.,.C.;

"power pins
        GND     pin 12;
        VCC     pin 24;

"inputs
        RW      pin  1;     A12     pin  8;
        PHI2    pin  2;     A13     pin  9;
        A0      pin  3;     A14     pin 10;
        A1      pin  4;     A15     pin 11;
        A2      pin  5;     NC1     pin 13;
        A3      pin  6;     NC2     pin 14;
        A4      pin  7;     LS30    pin 23;

"outputs
        G245    pin 15;     Q2      pin 19;
        WE2064  pin 16;     Q1      pin 20;
        G573    pin 17;     Q0      pin 21;
        Q3      pin 18;     G138    pin 22;

equations

!G138   = (PHI2 & !A15                              "access
        # PHI2 & !A14                               "to
        # PHI2 & !A13                               "RAM
        # PHI2 & A15 & A14 & !A13 );                 "array

!G245   = (!RW & A15 & A14 & A13 & A12 & !LS30 & A4  "write to $FFF0-$FFFF
        # !RW & A15 & A14 & A13 & A12 & !LS30 & !A4  "write to $FFE0
```

Fig. 1: ABEL source file voor de nieuwe PAL

```
                    & !A3 & !A2 & !A1 & !A0
          # !A13                                          "access
          # !A14                                          "to
          # !A15                                          "RAM
          # A15 & A14 & !A13 );                            "array

!WE2064   = (PHI2 & !RW & A15 & A14 & A13 & A12 & !LS30
                    & A4 );                              "write to $FFF0-$FFFF


G573      = (PHI2 & !RW & A15 & A14 & A13 & A12 & !LS30
                    & !A4 & !A3 & !A2 & !A1 & !A0 );     "write to $FFE0

!Q0       = (!A0 & A15 & A14 & A13 & A12 & !LS30 & A4    "write $FFFX: A0 is TMM2064 address
          # !A12 );                                      "else let A12 through

!Q1       = (!A1 & A15 & A14 & A13 & A12 & !LS30 & A4    "write $FFFX: A1 is TMM2064 address
          # !A13 );                                      "else let A13 through

!Q2       = (!A2 & A15 & A14 & A13 & A12 & !LS30 & A4    "write $FFFX: A2 is TMM2064 address
          # !A14 );                                      "else let A14 through

!Q3       = (!A3 & A15 & A14 & A13 & A12 & !LS30 & A4    "write $FFFX: A3 is TMM2064 address
          # !A15 );                                      "else let A15 through


test_vectors ([PHI2,RW,A0,A1,A2,A3,A4,LS30,A12,A13,A14,A15]
                    -> [WE2064,G138,G573,G245,Q0,Q1,Q2,Q3])

          "                          W
          "                          E
          "P              L          2 G G G
          "H              S A A A A  0 1 5 2
          "I  R A A A A A A3 1 1 1 1 6 3 7 4 Q Q Q Q
          "2  W 0 1 2 3 4 0 2 3 4 5  4 8 3 5 0 1 2 3
          "-----------------------------------------
          [H,H,H,H,H,H,H,L,H,H,H,H]-> [H,H,L,H,H,H,H,H];  " 1 $FFFX, reading TMM2064
          [L,H,H,H,H,H,H,L,H,H,H,H]-> [H,H,L,H,H,H,H,H];  " 2 $FFFX, reading TMM2064
          [H,H,L,L,L,L,H,L,H,H,H,H]-> [H,H,L,H,L,L,L,L];  " 3 $FFFX, reading TMM2064
          [L,H,L,L,L,L,H,L,H,H,H,H]-> [H,H,L,H,L,L,L,L];  " 4 $FFFX, reading TMM2064
          [H,L,L,H,L,H,H,L,H,H,H,H]-> [L,H,L,L,L,H,L,H];  " 5 $FFFX, writing TMM2064
          [L,L,L,H,L,H,H,L,H,H,H,H]-> [H,H,L,L,L,H,L,H];  " 6 $FFFX, writing TMM2064
          [H,L,H,L,H,L,H,L,H,H,H,H]-> [L,H,L,L,H,L,H,L];  " 7 $FFFX, writing TMM2064
          [L,L,H,L,H,L,H,L,H,H,H,H]-> [H,H,L,L,H,L,H,L];  " 8 $FFFX, writing TMM2064
          "
```

```
[H,H,L,L,L,L,L,L,H,H,H,H]->[H,H,L,H,H,H,H,H];   " 9 $FFE0, reading tasknumberreg.
[L,H,L,L,L,L,L,L,H,H,H,H]->[H,H,L,H,H,H,H,H];   "10 $FFE0, reading tasknumberreg.
[H,L,L,L,L,L,L,L,H,H,H,H]->[H,H,H,L,H,H,H,H];   "11 $FFE0, writing tasknumberreg.
[L,L,L,L,L,L,L,L,H,H,H,H]->[H,H,L,L,H,H,H,H];   "12 $FFE0, writing tasknumberreg.
"
[H,H,L,H,L,H,L,L,H,H,H,H]->[H,H,L,H,H,H,H,H];   "13 $FFEX, reading
[L,H,H,L,H,L,L,L,H,H,H,H]->[H,H,L,H,H,H,H,H];   "14 $FFEX, reading
[H,L,L,L,H,H,L,L,H,H,H,H]->[H,H,L,H,H,H,H,H];   "15 $FFEX, writing
[L,L,H,H,H,H,L,L,H,H,H,H]->[H,H,L,H,H,H,H,H];   "16 $FFEX, writing
"
[H,H,X,X,X,X,X,X,L,H,H,H]->[H,H,L,H,L,H,H,H];   "17 $EXXX, reading IO/video
[H,L,X,X,X,X,X,X,L,H,H,H]->[H,H,L,H,L,H,H,H];   "17 $EXXX, writing IO/video
[H,H,X,X,X,X,X,X,H,H,H,H]->[H,H,L,H,X,X,X,X];   "19 $FXXX, reading EPROM
[H,L,X,X,X,X,X,H,H,H,H,H]->[H,H,L,H,H,H,H,H];   "20 $FXXX, writing EPROM
"
[L,X,X,X,X,X,X,X,L,H,L,H]->[H,H,L,L,L,H,L,H];   "21 RAM: PHI2 low
[H,X,X,X,X,X,X,X,L,L,L,L]->[H,L,L,L,L,L,L,L];   "22 RAM: CPU address $0XXX
[H,X,X,X,X,X,X,X,H,L,L,L]->[H,L,L,L,H,L,L,L];   "23 RAM: CPU address $1XXX
[H,X,X,X,X,X,X,X,L,H,L,L]->[H,L,L,L,L,H,L,L];   "24 RAM: CPU address $2XXX
[H,X,X,X,X,X,X,X,H,H,L,L]->[H,L,L,L,H,H,L,L];   "25 RAM: CPU address $3XXX
[H,X,X,X,X,X,X,X,L,L,H,L]->[H,L,L,L,L,L,H,L];   "26 RAM: CPU address $4XXX
[H,X,X,X,X,X,X,X,H,L,H,L]->[H,L,L,L,H,L,H,L];   "27 RAM: CPU address $5XXX
[H,X,X,X,X,X,X,X,L,H,H,L]->[H,L,L,L,L,H,H,L];   "28 RAM: CPU address $6XXX
[H,X,X,X,X,X,X,X,H,H,H,L]->[H,L,L,L,H,H,H,L];   "29 RAM: CPU address $7XXX
[H,X,X,X,X,X,X,X,L,L,L,H]->[H,L,L,L,L,L,L,H];   "30 RAM: CPU address $8XXX
[H,X,X,X,X,X,X,X,H,L,L,H]->[H,L,L,L,H,L,L,H];   "31 RAM: CPU address $9XXX
[H,X,X,X,X,X,X,X,L,H,L,H]->[H,L,L,L,L,H,L,H];   "32 RAM: CPU address $AXXX
[H,X,X,X,X,X,X,X,H,H,L,H]->[H,L,L,L,H,H,L,H];   "33 RAM: CPU address $BXXX
[H,X,X,X,X,X,X,X,L,L,H,H]->[H,L,L,L,L,L,H,H];   "34 RAM: CPU address $CXXX
[H,X,X,X,X,X,X,X,H,L,H,H]->[H,L,L,L,H,L,H,H];   "35 RAM: CPU address $DXXX

"G245 is the enable for the databusbuffer
"
"G573 is the enable for the task number register
"
"G138 is the enable for the RAM array address decoder
"
"WE2064 is the write enable input for the TMM2064 and the enable for the
"     write data buffer to the TMM2064
"
"Q0..Q3 are the lower addresslines for the TMM2064

end _VDISK3B;
```