ENTERface

Geerweg 2, 2611 VN Delft

DRUKWERK

PORT BETAALD
PORT PAYE
DELFT

SEPT–OKT–NOV.

ENTERface

INHOUDSOPGAVE

GEBRUIKERSDAG-HCC-DAGEN
25 EN 26 NOVEMBER JAARBUERSHALLEN UTRECHT

GEBRUIKERSDAG 28 januari 1989
H.F. Witte dorpshuis, Henri Dunantplein 4, DE BILT
bus 57 vanaf CS, halte Nobellaan, Nobellaan in, 3e weg links

COLOFON

De *ENTERFACE* is het officiele orgaan van de *Dutch Enterprise User Group* en verschijnd eens per twee maanden.

REDAKTIE ..... Robin Ketelaars *voor toesturen kopij* .....
Geerweg 2, 2611 VN DELFT, 015-126422

MEDEWERKENDEN .... Kees, Stan, Arjan, Spock, Jonny, Hennus, Martin .....
.....waar blijven jullie nou? .....

VOORPLAAT ..... Een PLAATJE uit de ENTERPRISE

KOPIJ ..... Inzenden op DISK: of TAPE:
ZIE VOORWAARDEN IN HET APRIL/MEI NUMMER
72 tekens * 50 regels TAB's onder L(oad) en E(xit)
1e regel = titel die bovenaan bladzijde komt
<ALT L> voor een zin geeft nieuwe titel & nieuwe bladzij
<ALT \> voor een zin geeft alleen een titel
PRINT met F3 <naam>.KPY voor automatische verwerking

OVERNEMEN ..... van de gehele of gedeeltelijke inhoud toegestaan

HAMER + PENNINGEN Stan Tuinder .... *voor geldkwesties en adviezen* .....
Willemstr 170, 2713 AJ ZOETERMEER, 079-169523

SECRETARIAAT ..... Fons Kraakman ... *voor aanmelden en opzeggen* .....
Bleekveld 8, 1852 JH HEILOO, 072-335131

SOFTWARE ..... Dik Fennema ..... *voor aanvragen/toesturen software*
v Duivenvoordeln 30, 2241 ST WASSENAAR, 01751-17249

KONTAKTEN ..... NOORD: Kees 05945-15626; Peter 050-775850
MIDDEN: Peter 08885-02186; BRABANT: Rob 01651-1245
ZEELAND: Ben 01185-2525; ZUID: Rene&Jos 043-617623
WEST: Hans 02510-46630; Erik 01803-17051

OPBELLEN ..... BEL ALLEEN BOVENSTAANDE NUMMERS TUSSEN 20 EN 22 UUR

DRUKWERK ..... DRUK.TAN HECK, Pluympot 1a, 2611 LX DELFT. 015-142981

REDAKTIONEEL

Het is de redaktie toch weer gelukt een ENTERFACE klaar te krijgen. Het heeft even geduurd, (vakantie, ziekte, geen tijd) uiteindelijk ligt ie er nu eindelijk voor je neus.

Een extra dik nummer met een heleboel machinetaal. Je kunt dadelijk zelf je DEVICE-driver schrijven. Stan heeft er zijn best op gedaan om het een en ander zo duidelijk mogelijk uit te leggen. Ikzelf heb het nog niet goed gelezen, maar het ziet er wel 'gebruikers-aantrekkelijk' uit.

Er zijn ook wat bijdragen in Pascal-taal. Er wordt melding gemaakt hoe je een kleur kunt samenstellen op de ENTERPRISE en er wordt recursief gedacht.

Verder een stukje over een TURBOPASCAL-uitbreiding die het mogelijk maakt MS-DOS turboprogrammaas direct te gebruiken op de enterprise, kleur en grafisch en al. Tevens een uitbreiding om alle EXOS goed te benutten.

Hennus legt in dit blad uit hoe er problemen kunnen worden opgelost in de 512k kaarten (niet iedereen heeft hier last van).

Reclame voor de muis is er ook. In het volgende nummer een test van de muis en de bijbehorende software door Arjan.

HARDDISK

Van het duitse front iets nieuws: Harddisk voor de Enterprise

Er is in ontwikkeling: een uitbreidingskaartje voor harddisk-controllers van het IBM-(kloon)-compatible type voor 2 X 30Mb harddisk.
Dit kaartje komt in een uitbreidingsbuskaart die tussen ENTERPRISE en diskcontroller zit. Op deze buskaart voldoende buffers en extra voedingseenheid om de ENTERPRISE te ontlasten.

Let wel! een normale WD of SEAGATE controller voor IBM-(kloon)-PC's kan hiervoor worden gebruikt. dus verkoop je XT, maar behoud je harddisk voor de ENTERPRISE.

De uitbreiding komt compleet met 20-zijdig handboek met alle pinouts, geheugentabellen, i/o-poort-adressen, en elektrise beschrijving.

Kosten:	MINIBUS-buskaart incl netdeel	DM 150,-	met 4 slots
	Harddisk-interface	DM 180,-	voor 1 slot
	1Mb uitbreiding zonder chips	DM 150,-	voor 1 slot
	Exdos-controller	DM 180,-	voor 1 slot

verkrijgbaar bij:
Werner Lindner H&S Ideeen, Landbergerstr. 49, D-8913 SCHONDORF a.A.

De ENTERPRISE kan onder IS-DOS ieder CP/M80 of CP/M 2.2 programma draaien, wanneer dit maar op een MS-DOS fileformatted disk staat. Hieronder behoort ook TURBO-PASCAL.

Helaas zijn de meeste programmaas voor MS-DOS machines. Dit betekend dat wanneer de grafische mogelijkheden van deze machine gebruikt worden, de ENTERPRISE ze niet kan verwerken.
Echter de CP/M-versies van turbo-programmas dunnen wel gebruikt worden. Een nadeel is dat deze programmaas nooit grafisch te werk kunnen gaan, omdat er geen grafische mogelijkheden zijn onder CP/M.
Alleen blokgrafieken of veranderde tekensets kunnen gebruikt worden.

Voor de ENTERPRISE is deze grafische beperking er niet. Niets staat haar in de weg gelijksoortige veelkleurige VEGA/EGA/CGA/ETC. plaatjes te maken in turbo pascal, maar dan moeten de turbo-grafische commandoos wel in de CP/M versie ondersteund worden.

Het is nu zover. In Duitsland heeft Martin Schillinger een pakket met procedures geschreven dat met het INLUDE van TURBO-PASCAL in je eigen programma opgenomen kan worden.

Dit grafiek-pakket heeft alle grafische procedures die ook op de IBM-(klonen)-peecees voorhanden zijn.
Het is nu dus mogelijk grafische turbo-pakketten op de ENTERPRISE te laten draaien.

Op de diskette die deze uitbreiding bevat staat ook een handleiding in het gebruik van de uitbreidingen en geeft de verschillen cq uitbreidingen op MS-DOS-TURBO. En enkele kleurrijke demo programmaas.
Zo is het op de ENTERPISE mogelijk grafische plaatjes te laden en op te slaan met een kommando. In het MS-DOS pakket moeten hier extra routines voor gemaakt worden.

Het enige nadeel is dat MS-programmaas die gebruik maken van windows of turtle grafics niet gedraaid kunnen worden. Ook moet je er op letten dat er niet veel stack-ruimte op de ENTERPRISE is. Arrays moeten binnen een bepaalde maat blijven om moeilijkheden te voorkomen.

Een tweede bijgeleverd pakket dient om alle ENTERPRISE mogelijkheden nog meer uit te buiten. Alle EXOS-commandoos kunnen met een PASAL bevel aangeroepen worden.
Openen, sluiten van kanalen, setten van systeemvariabelen zijn nu onder turbo geen probleem meer.

Met dit pakket kan niemand meer een probleem hebben alle mogelijkheden van de ENTERPRISE te benutten voor de TURBO-PASCAL compiler

Het is te koop voor DM 39,- bij:
Martin Schillinger, Hieronymusstrasse 16, D-8000 MUNCHEN 60

(vrije vertaling uit ENTER-NEWS 2/88)

SORCERY EN DEVIL'S LAIR NU OP DISK

Wanneer Sorcery of Devil's Lair van disk werd ingeladen werkte de game niet naar behoren. Tijdens de vergadering op de gebruikersdag van 11 juni j.l. werd gevraagd of iemand deze games werkend kon maken. Ik ben aan de slag gegaan en heb met de MON.XR de application programma's van beide games naar de printer gedisassembleerd, dit zijn twee behoorlijke stapels papier geworden.

SORCERY

Het probleem bij Sorcery was dat de sprite die door de joystick bestuurd moest worden altijd naar rechts onder ging en daar bleef hangen totdat het energy-level 0 was. Philip Homburg dacht dat er iets fout ging met de keyboard-buffer zodat de joysticks niet meer gelezen konden worden. Dit bleek inderdaad het geval te zijn.

In het programma worden enkele video kanalen, een sound kanaal en een keyboard kanaal geopend. Na talloze pogingen (video pagina's verkleinen, segmenten verwisselen, enz.) is het toch gelukt om de game werkend te krijgen.

Voordat het keyboard kanaal geopend wordt, wordt het sound kanaal geopend. Het enige wat ik nu gedaan heb is eerst het keyboard kanaal openen en daarna pas het sound kanaal. Hierdoor komt de channel-RAM die voor het keyboard vrijgemaakt moet worden op een andere plaats in het geheugen dan in het oude geval, en dit blijkt te werken.

DEVIL'S LAIR

Bij Devil's Lair lag het probleem iets moeilijker. Er waren eigenlijk twee problemen. Er werden files ingelezen zonder filenaam en na dit verbeterd te hebben startte de game niet op.

De eerste file wordt ingelezen door het BASIC-programma. Omdat geen filenaam werd gegeven werd naar programma "start" gezocht op disk. Deze file moest dus een naam hebben, ik heb hem daarom DL.APP genoemd (de naam wordt gedefinieerd vanaf adres 1000 (hex.) in procedure FILENAME). In DL.APP komt hetzelfde probleem driemaal voor, voor deze drie files heb ik de volgende namen gekozen: DEVIL1.DAT, DEVIL2.DAT en DEVIL3.DAT.

Het volgende probleem bleek te maken te hebben met een bepaald adres van de video pagina. Wanneer een video kanaal wordt geopend kan met een EXOS-CALL het eerste adres van deze pagina verkregen worden. In het BASIC-programma dat moet worden uitgevoerd om dit spel in te laden en uit te voeren worden enkele POKE commando's gebruikt. Op adres 4810 en 4811 (12CA hexadecimaal) wordt een waarde gezet. Het adres 12CA (hex.) wordt tientalle malen uitgelezen in het machinetaal programma (DL.APP), het blijkt dat op dit adres een vaste plaats op de video pagina is opgeslagen.

Om achter het eerste adres van de video pagina te komen heb ik een stukje source geschreven en toegevoegd aan DL.APP. Dit stukje source zorgde ervoor dat het eerste adres van de video pagina naar een file op tape

geschreven werd. Dit programma heb ik uitgevoerd op de ENTERPRISE met en zonder de EXDOS-controller. Op deze manier kon ik het nieuwe adres van de video pagina achterhalen en wijzigen in het BASIC-programma.

De ENTERPRISE 64, 128 en 576K blijken ieder een ander adres nodig te hebben. Daarom heb ik een menu gemaakt waarin aangegeven moet worden welke ENTERPRISE je hebt. Daarnaast heb ik nog twee andere menu's toegevoegd waarbij de interne joystick of een van de externe gekozen kan worden en waarbij je een keuze kan maken of je de interne speaker aan of uit wilt zetten.

De totale tijd die ik erin gestopt heb is ongeveer 30 uur.

Veel plezier bij het spelen van deze games.

Kees Bevaart

RGB FUNCTIE IN PASCAL

In IS-BASIC is het mogelijk een van de 256 kleuren van de Enterprise te kiezen m.b.v. de RGB-functie. Het leek mij wel handig om deze functie ook in PASCAL beschikbaar te hebben.

In de Technical Manual staat beschreven hoe een byte (geheel getal van tussen 0 en 255) geinterpreteerd wordt door de NICK-chip (video chip). Een byte is opgebouwd uit 8 bits (bit 0 tot en met 7), ieder bit kan hoog (=1) of laag (=0) zijn. Op deze manier is het dus mogelijk om de getallen 0 tot 255 te krijgen.

```
bit 0 : 2^0 =   1        4/7 * ROOD
bit 1 : 2^1 =   2        4/7 * GROEN
bit 2 : 2^2 =   4        2/3 * BLAUW
bit 3 : 2^3 =   8        2/7 * ROOD
bit 4 : 2^4 =  16        2/7 * GROEN
bit 5 : 2^5 =  32        1/3 * BLAUW
bit 6 : 2^6 =  64        1/7 * ROOD
bit 7 : 2^7 = 128        1/7 *       GROEN
```

Om bijvoorbeeld de kleur magenta te krijgen moet dus volledig rood en volledig blauw gekozen worden:

```
byte = 4/7*ROOD + 2/7*ROOD + 1/7*ROOD + 2/3*BLAUW + 1/3*BLAUW

byte =     1    +     8    +    64    +     4     +     32

byte = 109
```

Probeer maar eens in IS-BASIC "SET BORDER 109" of "SET BORDER MAGENTA" of "SET BORDER RGB(1,0,1)", dit zal dezelfde border-kleur opleveren.

In IS-BASIC kan per kleur een waarde opgegeven worden van 0 tot 1. Wanneer je echter de waarde van deze kleur 1/10 verhoogt of verlaagt zal de kleurtint niet altijd veranderen. Daarom heb ik gekozen voor gehele waarden van 0 tot en met 7 voor rood en groen en van 0 tot 3 voor blauw. Verhoging of verlaging van een van de kleuren met een geheel getal levert altijd een andere kleurtint op. Wanneer de waarden niet tussen de gegeven grenzen liggen zal een foutmelding worden gegeven en de functie zal de waarde -1 retourneren. De functie wordt op dezelfde manier aangeroepen als in IS-BASIC. Magenta is in dit geval dus: RGB(7,0,3).

De listing van de functie is als volgt:

```
FUNCTION rgb(red, green, blue: real): integer;
VAR r, g                        : ARRAY[1..3] OF integer;
    b                           : ARRAY[1..2] OF integer;
    kleur, red1, green1, blue1: integer;
    waardeok                    : boolean;

BEGIN
  kleur:=-1;
  red1:=TRUNC(red); green1:=TRUNC(green); blue1:=TRUNC(blue);
```

```
  waardeok:=true;     { gebruik alleen de gehele delen van de getallen }
  IF NOT (red1 IN [1..7]) THEN
    BEGIN
      writeln('*** Foute waarde voor rood');
      waardeok:=false;
    END;
  IF NOT (green1 IN [1..7]) THEN
    BEGIN
      writeln('*** Foute waarde voor groen');
      waardeok:=false;
    END;
  IF NOT (blue1 IN [1..3]) THEN
    BEGIN
      writeln('*** Foute waarde voor blauw');
      waardeok:=false;
    END;
  IF waardeok THEN
    BEGIN
      kleur:=0;
      r[1]:=red1 DIV 4;
      IF r[1]=1 THEN kleur:=kleur+1;     { 4/7 * ROOD }
      r[2]:=red1 MOD 4;
      r[1]:=r[2] DIV 2;
      IF r[1]=1 THEN kleur:=kleur+8;     { 2/7 * ROOD }
      r[3]:=r[2] MOD 2;
      r[1]:=r[3] DIV 1;
      IF r[1]=1 THEN kleur:=kleur+64;    { 1/7 * ROOD }
      g[1]:=green1 DIV 4;
      IF g[1]=1 THEN kleur:=kleur+2;     { 4/7 * GROEN }
      g[2]:=green1 MOD 4;
      g[1]:=g[2] DIV 2;
      IF g[1]=1 THEN kleur:=kleur+16;    { 2/7 * GROEN }
      g[3]:=g[2] MOD 2;
      g[1]:=g[3] DIV 1;
      IF g[1]=1 THEN kleur:=kleur+128;   { 1/7 * GROEN }
      b[1]:=blue1 DIV 2;
      IF b[1]=1 THEN kleur:=kleur+4;     { 2/3 * BLAUW }
      b[2]:=blue1 MOD 2;
      b[1]:=b[2] DIV 1;
      IF b[1]=1 THEN kleur:=kleur+32;    { 1/3 * BLAUW }
    END;
  rgb:=kleur;
END;
```

Kees Bevaart

Uit onderstaand programma blijkt, dat recursie in Turbo Pascal niet goed werkt!

```
//BEGIN

PROGRAM test(input,output);
  FUNCTION fac(n:integer):integer;
    BEGIN
      IF n=0
      THEN fac:=1
      ELSE fac:=fac(n-1)*n
    END;
  BEGIN
    writeln(fac(5):5)
  END.

//EINDE
```

De kern van het programma is:

```
  fac := fac(n-1) * n;
```

Als uit de recursie wordt teruggekeerd, is de waarde van n overschreven, met als gevolg dat de waarde 0 wordt afgedrukt!
Draai je de volgorde van de expressie om, dan gaat het wel goed (waarde = 120):

```
  fac := n * fac(n-1)
```

Mijn vraag is nu: is er een manier (bijv. m.b.v. compiler optie {A+}) om recursie goed te laten functioneren ?

Jonny Uiterweerd
Gentiaanstraat 58
7322 BM Apeldoorn
tel. 055-663266

MODIFICATIE 512KB KAART

Voor alle mensen die problemen hebben met hun 256/512kB RAM geheugen uitbreiding van Roelof Horst, volgen hier een aantal tips en wat uitleg om toe te lichten hoe het kan dat er problemen mee ontstaan en hoe je dat kunt verhelpen.

De RAM uitbreiding maakt gebruik van Dynamische RAMs (DRAMs) die gegevens opslaan door middel van een matrix van zeer kleine condensatortjes. Deze matrix is ingedeeld in rijen en kolommen. Om het aantal pennetjes om een geheugen chip wat te verlagen, worden de adres bits voor de rijen en kolommen gemultiplexed over de helft van de anders benodigde adres lijnen (9 ipv 18 bij 256k bit DRAMs).

Als je een geheugencel wilt lezen of schrijven, dan moet je eerst het rij-adres op de adres-bus van de DRAM chip zetten, een Row Address Strobe (RAS) puls geven, vervolgens het kolom-adres, een Column Address Strobe (CAS) geven en een poosje wachten. Al deze dingen moeten binnen een bepaald tijdschema afgewerkt worden, anders gaat het niet goed en neemt de kans dat de gelezen of geschreven waarde verloren gaat sterk toe. De speelruimte die je hebt waarbinnen het wel goed gaat wordt door de fabrikant vrij nauwkeurig omschreven. De waarden verschillen echter wel van fabrikant tot fabrikant en elke fabrikant heeft natuurlijk ook weer een aantal verschillende versies die een wat snellere of tragere toegangstijd hebben.

Als je er verder niets aan zou doen, dan zou de informatie in de DRAM-chips binnen 4 milliseconden verdwenen zijn, omdat de condensatortjes hun lading `langzaam` verliezen. Om dit probleem te verhelpen moet je steeds binnen 4 milliseconden alle rij adressen een voor een op de adres bus van de DRAM zetten, de Write enable lijn hoog houden en een RAS puls geven. De DRAM ververst (refreshed) dan een hele rij geheugencellen in een keer door ze uit te lezen en snel weer terug te schrijven. Dit wordt mogelijk gemaakt door de interne opbouw van van de DRAM. De Z80 processor in de Enterprise helpt hierbij een handje door regelmatig als de processor even geen toegang tot het geheugen nodig heeft een signaal, notRFSH, actief te maken en op de laagste 7 adreslijnen van de Z80 komt dan de inhoud van het register R te staan. De Z80 verhoogt dit register elke keer met 1.
(Het achtste bit kun je trouwens zelf programmeren, maar je kunt beter van dit register afblijven, want je loopt dan het risico de refresh te verstoren als je niet weet waar je mee bezig bent.)

De DRAM kan het refresh signaal van de Z80 gebruiken om een rij te verversen en dat heet dan een refresh-cycle. Een klein probleempje is dat de DRAMs voor hun refresh bijna altijd nog een adres bitje meer nodig hebben dan de 7 die de Z80 tijdens een refresh cycle aanlevert. Dit wordt meestal met een tellertje op de geheugenkaart opgelost. De geheugencel matrix in de meeste 256k bit DRAMs zijn 256*1024 of 512*512 bits. Die van 256*1024 hebben intern een 8 bits rij-adres nodig bij de refresh (8 bits refresh) en die van 512*512 9 bits rij-adres (9 bits refresh). Bij 8 bits refresh kun je tijdens een refresh cyclus het hoogste adresbit van de DRAM een willekeurige waarde geven omdat die dan niet nodig is. Bij 9 bits refresh echter, moet dat bit wel bijgehouden worden.

Dit verversen gebeurt ook als je gewoon een geheugencel leest of schrijft, waardoor het soms heel moelijk kan zijn om een fout die onstaat door een verkeerde refresh op te sporen. Het handigste is een programma te schrijven dat de interrupts uitzet, een paar seconden in een heel korte lus blijft hangen en vervolgens kijkt of de rest van het geheugen nog dezelfde waardes bevat als ervoor. Het uitzetten van de interrupts heeft als doel te voorkomen dat de processor in het midden van de korte lus even ergens naar toe springt om een interrupt af te handelen en zodoende tevens een groot gedeelte van het geheugen ververst, want dat zou het nog veel moeilijker maken het probleem om te sporen. Er zijn namelijk ook DRAMs die veel langer dan 4ms zonder refresh kunnen.

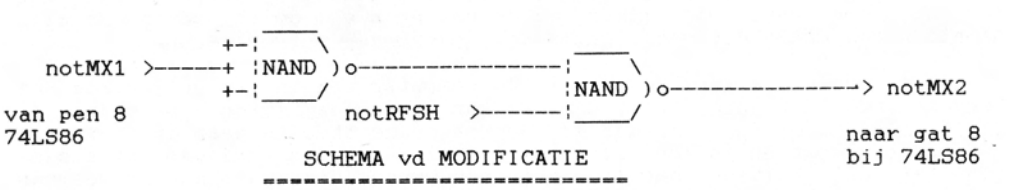
Door op Hold te drukken komt de computer ook in een vrij korte lus terecht met de interrupts uit. Zo kun je dus een (groot) programma in BASIC inladen, op Hold drukken een uurtje of zo thee te gaan drinken, als je dan terugkomt en je kunt niet dezelfde listing meer krijgen, of als de computer dan tijdens het listen ineens vreemd doet, dan kun er wel van uitgaan dat het aan de refresh ligt. Het verraderlijke van refresh-problemen is dat als je niet weet waar je naar moet zoeken, je nooit helemaal zeker weet waar de problemen van komen en vaak merk je niet eens dat er problemen mee zijn, zeker niet als je Hold weinig gebruikt.

Van de geheugenkaart van Roelof zijn er 2 series gemaakt, in de eerste werden er volgens mij DRAMs van NEC gebruikt die kennelijk perfect werken met de geheugenkaart, want ik heb nog niemand horen klagen die een uitbreidingskaart van de eerste serie had. Bij de tweede serie werden er andere DRAMs gebruikt (Mitshubishi) en tevens werden een paar onderdelen die voor de timing van de RAS, CAS en het multiplexen zorgen veranderd.

Ik heb een oplossing gevonden om deze problemen te verhelpen. Het is nog niet helemaal duidelijk waarom het werkt maar de reden is waarschijnlijk dat Mitshubishi DRAMS een langere periode nodig hebben voordat CAS gegeven mag worden dan de NEC DRAMs. Ik heb het met success uitgetest op drie computers en iemand anders heeft het ook nog met succes uitgeprobeerd op een of twee computers. Het vergt wat precisie-soldeerwerk dus als je weinig soldeer-ervaring hebt: Blijf er dan van af want je kunt je hele computer onbruikbaar maken!

Het steeds in en uit-solderen van de geheugenkaart (om uit te vissen waar de fout zat) heeft als resultaat een werkende geheugen uitbreiding en een niet meer uitbreidbare Enterprise 64 opgeleverd; de interne uitbreidings-connector in die computer is nu niet meer als zodanig bruikbaar. Dus als je de uitbreiding er nu nog in moet zetten, zoek dan een paar connectoren zoals die in de 64k uitbreiding voor de Enterprise 64 van Enterprise Ltd. zelf gebruikt werden; dan kun je je geheugen-uitbreiding er altijd weer uithalen als er iets mee is. Neem liever geen cardedge-connectoren want dat zijn (zoals bekend van de diskcontrollers) weinig betrouwbare connectoren.

Om te beginnen: Zet de Enterprise voor je neer, zonder behuizing, en metde geheugenuitbreiding rechtsboven. Zoek nu pin 8 van het IC 74LS86 op en soldeer dat pennetje heel voorzichtig los en haal het omhoog zodat het soldeergat waar het in zat vrij komt te liggen. Dit gaat het handigst met een soldeerbout van 15-25 Watt en een heel dun en klein schroevendraaiertje. Mischien kun je het beste eerste even oefenen met een ander printje want als dat pennetje afbreekt staat je een hoop soldeerwerk te wachten.

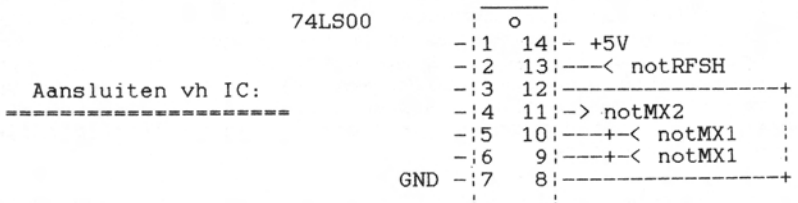


SCHEMA vd MODIFICATIE

Wees heel voorzichtig dat het pennetje niet afbreekt want dat hebben we zometeen nog nodig. Soldeer nu een draadje in het vrijgekomen soldeergat en noem dat signaal notMX2. Soldeer vervolgens een draadje aan het pennetje zelf en noem dat notMX1.

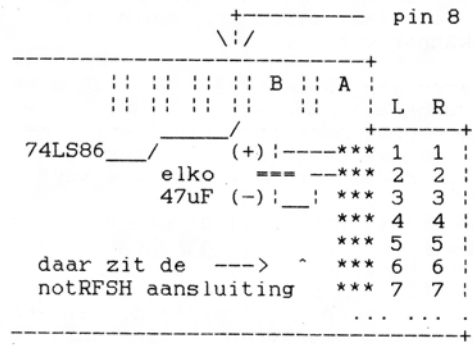
Van een soldeereilandje helemaal rechts bij de connector moeten we nu notRFSH vandaan halen. Begin helemaal rechtsbovenaan, bij connector pen 1 te tellen. Direct links van de connector aansluitingen zitten onder en bovenop de print vlakjes om de kaart na wat zaagwerk in een cardedge connector te kunnnen doen. Direct links van het gebied tussen het 5de en het 6de cardedge vlakje zit een gaatje in de print om het refresh signaal (notRFSH) van onder de print naar bovenop de print te leiden. Soldeer daar een draadje in.

De 2 NAND gates kun je het makkelijkst uit een 74LS00 IC halen. Het IC kun je of op een stukje veroboard (wel goed isoleren!) zetten of helemaal linksboven open de geheugenkaart waar nog ruimte is overgelaten voor dit soort dingen; er is daar ruimte voor een 20 pins IC en de voedingsspanning is al aangesloten, dus je kunt een ervan (het handigste is de ground aan pen 7 te nemen en zelf de +5 Volt van de connector te halen) gewoon gebruiken. Verder moeten natuurlijk notRFSH, notMX1 en notMX2 aangesloten worden.



Verder zou het heel verstandig zijn om nog een condensator van 47 micro Farad op de cardedge eilandjes te solderen om zo de storing die de DRAMs genereren wat te ontkoppelen van de rest van het systeem. Het eerste cardedge vlakje aan de bovenkant is +5V en de tweede aan de bovenkant GND.

Er is eerder in het blad voorgesteld om de twee voedingsspannings stabilisatoren, 7805 te vervangen door 78S05 omdat die 2 Ampere ipv 1 Ampere kunnen leveren. Als je dat doet is het wel raadzaam je transformator ook te vervangen door een die twee keer zoveel aankan. De gewone Enterprise transformator kan maar 2 Ampere leveren en met 2 maal 75S05 loop je een groot risico dat er 4 Ampere of meer aan je transformator ontrokken wordt.



RC netwerkjes:

	versie1 A	versie1 B	versie2 A	versie2 B
R in ohm	470	470	220	470
C in pico Farad	33	68	56	56

Als het bovenstaande niet afdoende helpt kun je ook nog de timing signalen wat veranderen door de RC netwerkjes op de kaart wat te veranderen. Deze timingsignalen regelen de tijd die tussen RAS, multiplexer omschakelen en CAS moet zitten.

Bij A en B bevinden zich 2 RC netwerkjes. De waarden voor de weerstanden en de condensatortjes staan hierboven-rechts. Gelukkig zijn de waarden niet al te kritisch bij de Enterprise om dat het geheugen in verhouding tot de processor vrij snel is.

Veel success!

P.S.

Er zijn nog printjes over om zelf in te bouwen, zonder DRAMS maar met de andere chips er wel op gesoldeerd voor ongeveer f80,-.

Hennus Bergman, Barmaheerd 90, 9737 MK Groningen.

Een device is een extern "zintuig" van de computer waardoor de computer kan communiceren met de "buitenwereld". Voorbeelden zijn o.a. keyboard, printer, video, editor ed. Maar ook licht_pen, modem, terminal, plotter, harddisk, muis, pio, robot_arm, midi, eprom_prog, bio_rithm, scratch_file en wie weet er nog een op te noemen. Het device is gebouwd rond een aantal functies die onder bepaalde voorwaarden worden uitgevoerd. Er bestaan eenvoudige zintuigen die alleen kunnen praten zoals Printer en Sound. Andere zijn ontworpen om uitsluitend te luisteren zoals het Keyboard naar de toets aanslagen. Er zijn echter devices waarbij het lijkt alsof deze het gemiddeld menselijke niveau zijn gepasseerd; ze kunnen praten EN luisteren zoals Editor. Laat U geen rad voor de ogen draaien, het is te gemakkelijk om met andermans veren te pronken. De Editor doet niets anders dan de opdrachten ondershands doorgeven aan video en keyboard en tegenover U hangt hij de alles-kunner uit.

Wil een zintuig goed functioneren dan is het nodig dat signalen worden doorgegeven naar een "centraal zenuwstelsel". Voor een device geldt min of meer hetzelfde, het moet in contact staan met het operating system (EXOS in dit geval). De actie waarbij dit contact wordt gelegd heet "linken". We zullen dit nog tegenkomen.

We zouden ook graag willen dat het device een blijvend onderdeel van het systeem gaat worden maw. dat het als een system-extension kan functioneren.

Zie hier de kluif die voor ons ligt en die, verdeeld over een paar Enterfaces, zo ver mogelijk voor U zal worden uitgebeend. Ik wil niet de indruk wekken hier een nuttig device naar ieders wensen te presenteren maar kan U wel wijzen op mogelijkheden die U zou kunnen benutten.

Wilt U Uw fantasie er op los laten en zelf een device maken dan moet U er zich van bewust zijn dat er wel een paar eisen aan gesteld worden! Een Enterprise laat zich niet zomaar een nieuw oor aan naaien. Aan de andere kant, welke echte Enterprise gebruiker deinst daar voor terug?? Iets anders is, of U de beschikking heeft over een goede assembler. Heeft U die niet, dan kunt U op een van de gebruikers dagen kijken of iemand U assistentie kan verlenen.

Ik wil achtereenvolgens bespreken: de extension header, een device descriptor met table en als laatste wat wandelen door de routine-tuin/jungle. De in te typen assembler-code staat tussen de [-> <-] haken

De Extension Header.
Aan het begin van een assembler listing staan meestal een aantal definitie en macro statements. Met deze statements kunt U, net als in basic, aan een variabele een waarde toe kennen. Door goede namen te kiezen kunt U hierdoor bijzonder makkelijk werken. C.

Rappard uit Almere heeft voor de Enterprise een zeer volledige set van files gemaakt waarin alle systeem constanten bij naam worden genoemd. Deze files, EXOS*.EQU en EXOS*.MAC, kunt U naar believen met een INCLUDE in de assembler listing opnemen.
[->

```
        ;include *.EQU files
        *I EXOS-VAR.EQU
        *I EXDOSVAR.EQU
        *I EXOSACTI.EQU
        *I EXOSSPEC.EQU
        *I EXOSFUNC.EQU
        ;include *.MAC files
        *I EXOS.MAC
        *I EXOSFUNC.MAC

        CR EQU £0D
        LF EQU £0A
```

<-]
Afhankelijk van de complexiteit van het device, zult U later ook de bijzonder goede "standaard" system extension headers van Richard en Patric met een include binnen halen. De heren hebben niet de zelfde opvatting over afkortingen zodat op een paar plaatsen een letter of streepje moet worden veranderd.

Nu staat U voor de voorlopige beslissing om het device in een .XA of een .XR module te plaatsen. In de ontwikkelings fase van een device doet U er het beste aan om een .XA te kiezen. Zo'n module komt altijd op pag 3 en heeft ook daar zijn entry-point op £C00A. Dit legt U vast met
[->

```
          ORG £C00A
```

<-]
Besluit U om later toch een .XR module te maken, bijvoorbeeld omdat het device zeker geen volledig segment van 16K nodig heeft, dan haalt U eenvoudig het ORG-statement weg en assembleert met R 7 in plaats van R 6.

De eerste echte code die U laat wegschrijven komt op dit zo geheten ENTRY_POINT. Zet daar een JR instructie naar de ENTRY.
[->

```
          JR ENTRY
```

<-]
Kom nu met program naam en versie nummer zodat je niet de halve source hoeft te listen om te weten waar je mee bezig bent.
[->

```
        VERSION:
          DEFB _VERSION-$-£01              ;PROG NAME £18 bytes
          DEFM "LAZY_EXTENSION ver 0.0"    ;    AND
          DEFB CR,LF                       ; VERSION
        _VERSION:
```

<-]
{Sommige EXPERTS onder ons laten de JR ENTRY weg en starten

meteen met de version. Deze grap mag alleen als de version_text maar 24 bytes lang is. De hoofdletter L geeft dan tussen _VERSION en ENTRY een buffer van 76-24=52 bytes}

Vervolgens nu de secundaire entry met de "standaard" extension header.

```
[->
        ENTRY:
          PUSH DE         ;POINTER TO TEXT
          PUSH BC         ;LEN(FIRST_WORD) IN B
          LD A,C          ;ACTION-CODE IN C
          DEC A
          JP Z,COLD_RES   ;COLD RESET
          DEC A
          JP Z,COMMAND    ;COMMAND IS GIVEN
          DEC A
          JP Z,HELP       ;HELP STRING
          DEC A
          JP Z,EX_VAR     ;EXOS VARIABLE
          DEC A
          JP Z,EXPL_ERR   ;EXPLAIN ERROR-CODE
          DEC A
          JP Z,LOAD_MOD   ;LOAD MODULE
          DEC A
          JP Z,ALL_RAM    ;RAM ALLOCATION
          DEC A
          JP Z,INIT       ;EXTENSION INITIALISATION
<-]
```

Komt U hier doorheen dan bent U vanzelf bij de Exit_action_code_ active.

```
[->
        EXIT_ACA:
          POP BC          ;RESTORE BC
          POP DE          ;RESTORE DE
          RET
<-]
```

Heeft deze extension een opdracht volledig uitgevoerd dan wordt meestal de action-code ingetrokken. De klus verlaat de extension bij Exit_action_code_clear.

```
[->
        EXIT_ACC:
          POP BC          ;CLEAR STACK
          POP DE          ;CLEAR STACK
          XOR A           ;ZERO RETURN
          LD C,A          ;NO ACTION-CODE
          RET
<-]
```

Deze "standaard" EXOS-header behandelt alle action-codes en geeft een veilige terugkeer naar het systeem. Belangrijk is dat de registers DE en BC op de stack worden bewaard omdat deze registers informatie bevatten die mogelijk ook voor andere extensions zijn bedoeld.

Tenzij U iets heel speciaals wilt, kunnen, zoals gebruikelijk, de action-codes 1, 4, 5, 6 en 7 meteen worden doorverwezen naar de EXIT with Action Code Active.

```
[->
        NOT_USED:
        COLD_RES:
        EX_VAR:
        EXPL_ERR:
        ALL_RAM
         JR EXIT_ACA
<-]
```

De codes 2, 3 en 8 kunnen naar behoefte verder worden ontwikkeld.

De "extended" EXOS-header die zo hier en daar met succes in gebruik is, heeft na het EXIT_ACC stukje een extra hoofdstuk " General Routines". Hierin zit code die U geregeld nodig hebt.

Denk bijvoorbeeld aan video-output.

```
[->     WR_A_OUT:
          PUSH BC                 ;SAFETY FIRST
          LD B,A
          LD A,(OUT_CHAN)         ;OUT_CHAN DECLARED IN DATASECTION
          EXOS _WR_CHR
          POP BC
          RET                                                    <-]
en
[->     WR_HL_OUT:      of      WR_DE_OUT:
          PUSH BC                                 ;SAFETY FIRST
          LD A,(HL)       of        LD A,(DE)     ;LEN(TEXT)
          INC HL
          OR A
          JR Z,EXIT_WR_OUT                        ;NO TEXT
          LD B,A
        WR_LOOP:
          LD A,(HL)       of        LD A,(DE)     ;GET CHAR
          CALL WR_A_OUT                           ;DO OUTPUT
          INC HL          of        INC DE        ;NEXT CHAR
          DJNZ WR_LOOP
        EXIT_WR_OUT:
          POP BC                                  ;RESTORE BC
          RET                                     ;HL or DE HAS CHANGED
<-]
en
[->     WR_NEXT:
          EX (SP),HL
          CALL WR_HL_OUT
          EX (SP),HL
          RET
<-]
```

Voor het geval dat channels worden geopend,

```
[->     GET_FREE_CHAN_A:
```

```
          PUSH DE          ;SAFETY FIRST
          PUSH BC          ;AND PUSH ON STACK
          LD B,£FE         ;£FF IS DEFAULT
        LOOP_FREE_CHAN:
          PUSH BC          ;SAVE ASKED CHAN_NUMBER
          LD A,B
          LD DE,DEV_VIDEO        ;DATA SOMEWHERE IN DATASECTION
          EXOS _OPEN
          POP BC           ;RESTORE CHAN_MUNBER
<-]
```

als dit channel vrij is dan is register A=£00 anders bevat A de code voor .CHANX channel exists.

```
[->       JR Z,FREE_CHAN
          DJNZ LOOP_FREE_CHAN                                    <-]
```

wanneer U hier door heen zakt, bent U een bijzonder drukke Enterprise gebruiker met maar liefst 255 channels tegelijkertijd open!!! Meldt dit ons! De Enterface zal Uw naam vermelden.

```
[->
        FREE_CHAN:                                               <-]
```

het register B bevat het nummer van het nieuwe channel of £00 als U er al 255 open hebt staan.

```
[->       LD A,B           ;CHAN_NUMBER                          <-]
( voor de record jagers
          OR A
          JR Z,MELDT_ENTERFACE  )
[->       PUSH BC          ;SAVE CHAN_NUMBER
          EXOS _CLOSE
          POP BC
          LD A,B           ;THIS CHAN_NUMBER IS FREE NOW
          POP BC           ;RESTORE
          POP DE           ;RESTORE
          RET
<-]
```

Heeft U meer "algemene" routines? Zet ze dan bij deze sectie.

We kunnen nu verder met de overgebleven action-codes. Een volgorde is niet verplicht. Heeft Uw device slechts 1 commando dan is de "standaard" header toereikend. Zijn er meer noten op Uw zang neem dan de "extended" action-code routines.

De HELP-routine is altijd nuttig als geheugensteuntje. De " extended" header maakt vanzelfsperkend gebruik van de boven beschreven routines. In de COMMAND_TABLE staan routine pointers en help_info direct na elkaar zodat:

```
[->     HELP:
          LD A,B
          CP £00
          JR Z,GEN_HELP          ;VERSION ONLY
          CALL CHK_COMMAND
          JP Z,EXIT_ACA          ;NOT FOUND
          CALL DISP_VERSION
          CALL WR_HL_OUT         ;COMMAND NAME
[ heeft U geen help_info dan NIET de volgende 3 regels]
          INC HL                 ;SKIP RTN_POINTER
          INC HL
          CALL WR_HL_OUT         ;HLP_INFO
          JP EXIT_ACC

        GEN_HELP:
          CALL DISP_VERSION
          JP EXIT_ACA

        DISP_VERSION:
          PUSH HL                ;SAFE HL ON STACK
          LD HL,VERSION
          CALL WR_HL_OUT
          POP HL
          RET
<-]
```

De COMMAND_routine heeft tot taak om uit te zoeken of het gegeven commando voor deze extension bedoeld is.

```
[->
        COMMAND:
          CALL CHK_COMMAND
          JP Z,EXIT_ACA ;NOT FOUND
          LD C,(HL)     ;GET LEN(COMMAND)
          LD B,£00
          INC HL        ;TO START COMMAND
          ADD HL,BC     ;SKIP COMMAND
          PUSH DE       ;SAVE TAIL OF TEXT
          LD E,(HL)
          INC HL
          LD D,(HL)
          EX DE,HL      ;HL POINTS TO RTN_<COMMAND>
          POP DE
          JP (HL)       ;TO RTN_<COMMAND>
<-]
```

Wordt er in BASIC een EXT-commando, in EXDOS een :-commando of in ISDOS een commando-line gegeven dan leurt EXOS met dit commando langs alle extensions. Het is dus zaak om zo snel mogelijk uit te zoeken of het commando wel voor deze extension is bedoeld. Hiertoe testen we de lengte van het eerste woord uit het commando met de lengte van de commands in deze extension. Voor de overzichtelijkheid zetten we de commands gesorteerd op lengte in een "command_table". Er zijn minimaal 3 items per command opgenomen. Deze zijn:

- 1 byte voor naamlengte
- n bytes voor de command_naam
- 2 bytes voor het adres van de routine

Wilt U het nog mooier maken voeg er dan help-info aan toe

- 1 byte voor lengte (help-info + cr,lf)
- n bytes voor (help-info + cr,lf)

Sluit de command_table met een "terminator" { DEFB £00 }

Het adres met de text van het commando staat in de registers DE en de lengte van het eerste woord staat in register B. Wees daar zuinig op!!

[->

```
        CHK_COMMAND:
          PUSH DE          ;SAFETY FIRST
          PUSH BC          ;AND SAVE ON STACK
          LD HL,COMMAND_TABLE
        NEXT_COMM:
          LD A,B           ;LEN(FIRST_WORD)
          CP (HL)          ;LEN(COMMAND)
          JR C,FAIL_COMM;LEN(COMMAND) TO LONG
          PUSH HL          ;SAVE ADRES_LEN(COMMAND)
          JR Z,LEN_FITS ;IF ok LET'S TEST NAME
        SORRY_DONT_FIT:
          POP HL
          LD C,(HL)        ;LEN(COMMAND)
          LD B,£00
          INC HL
          ADD HL,BC        ;SKIP OVER NAME
          INC HL           ;SKIP OVER RTN_ADRES
          INC HL
[ heeft U geen help_info dan 3 regels overslaan]
          LD C,(HL)        ;LEN(HLP_INFO)
          INC HL
          ADD HL,BC        ;SKIP OVER HLP_INFO
[ en hier verder ]
          POP BC           ;RESTORE LEN(FIRST_WORD)
          POP DE           ;RESTORE POINTER TO TEXT
          PUSH DE
          PUSH BC
          LD A,(HL)        ;LOOK FOR TERMINATOR
          OR A
          JR NZ,NEXT_COMM
        FAIL_COMM:
```

```
          XOR A
          JR EXIT_CHK_COMM      ;FLAG ZERO IS SET!
<-]
```

Indien de lengte van het eerste woord gelijk in aan het command dan test U verder.

```
[->
        LEN_FITS:
          INC HL          ;TO FIRST CHAR OF COMM
          INC DE          ;TO FIRST CHAR OF TEXT
[maak een loopje om text te vergelijken ]
        NEXT_CHAR_COMM:
          LD A,(DE)       ;CHAR OF TEXT
          CP (HL)         ;COMPAIR TO CHAR OF COMM
          JR Z,SORRY_DONT_FIT   ;NOT ok, NEXT TIME BETTER
          INC HL
          INC DE
          DJNZ NEXT_CHAR_COMM                                    <-]
```

alles klopt tot nu toe, het commando is voor deze extension.

```
[->       OR A            ;GET FLAGS FOR NONZERO
          POP HL          ;RESTORE ADRES_LEN(COMMAND)

        EXIT_CHK_COMM:
          POP BC          ;RESTORE
          POP DE          ;RESTORE
          RET
<-]
```

Als flags=nonzero stopt deze routine met HL op het len_byte van het <commamd>. In het andere geval is het command niet gevonden.

De command_table past hier mooi achter.

```
[->
        COMMAND_TABLE:
        LAZY:
          DEFB _LAZY-$-£03
          DEFM "LAZY"
          DEFW RTN_LAZY
        _LAZY:
          DEFB _LAZY1-$-£01
          DEFM "  Humf, geen zin"
          DEFB CR,LF
        _LAZY1:

        WERKEN:
          DEFB _WERKEN-$-£03
          DEFM "WERKEN"
          DEFW RTN_WERKEN
        _WERKEN
          DEFB _WERKEN1-$-£01
          DEFM "??  laat dat maar aan hout over!"
          DEFB CR,LF
        _WERKEN1:
```

```
        LINK_LAZY:
          DEFB _LINK_LAZY-$-£03
          DEFM "LINK_LAZY"
          DEFW RTN_LINK
        _LINK_LAZY:
          DEFB _LINK_LAZY1-$-£01
          DEFM "  inlinken van LAZY"
          DEFB CR,LF
        _LINK_LAZY1:
```

<-]
Mischien heeft U zelf nog wat andere suggesties.
[->

```
          DEFB £00      ;TERMINATOR
```

<-]

De INIT-routine.
Direct nadat U deze extensie met het LOAD ????.XA commando hebt ingeladen doet EXOS een "EXOS reset" call en stuurt de action-code 8 naar het entry_point. Vandaar kan worden doorgesprongen naar deze routine. Op deze plaats kunt U het doel van deze extensie uitvoeren nl. het linken van een nieuw device.
[->

```
        INIT:
         CALL TRY_TO_LINK
         JP EXIT_ACA
```

<-]
De routine TRY_TO_LINK zal ook door de RTN_LINK worden gebruikt en heeft daar een andere uitgang.

Vervolgens handelt U de commands af.
[->

```
        RTN_LAZY:
        RTN_WERKEN:
         CALL WR_NEXT
         DEFB END_WHO-$-£01
         DEFM "Who, ME???    "
        END_WHO:
         CALL WA_BLIEF
         JP EXIT_ACC

        RTN_LINK:
         CALL TRY_TO_LINK
         JP EXIT_ACC
```

<-]
De routine TRY_TO_LINK test of het device LAZY al aanwezig is want aan een luiwammes hebben we meer dan genoeg!
[->

```
        TRY_TO_LINK:
         CALL GET_FREE_CHAN_A    ;LOOK FOR AFREE CHANNEL A
         PUSH AF                 ;SAFE CHANNEL NUMBER
         LD DE,DEV_LAZY          ;DATA SOMEWHERE IN DATA-SECTION
         EXOS _OPEN
```

```
        CP £FA                  ;ERROR NODEV
        JR Z,GO_ON              ;SO NOT YET LINKED IN!
        POP AF
        EXOS _CLOSE             ;NO USELESS CHANNELS OPEN
        RET
<-]
Nu de echte link-routine die eerst een pop af moet wegwerken.
[->
      GO_ON:
        POP AF
<-]
. breng het DD_TAB adres (3 bytes) voor page 1 in orde
[->       LD A,(DD_TAB+£01)     ;HI_BYTE
          SET 6,A               ;SET TO PAGE 1 ADRES
          RES 7,A
          LD (DD_TAB+£01),A
          IN A,(£B3)            ;THIS SEGMENT
          LD (DD_TAB_SEG),A                                    <-]
. wijs met DE naar het DD_TYPE field
[->       LD DE,DD_TYPE                                        <-]
. vraag nul of 17 bytes device_ram aan met BC (zie interrupts)
[->       LD BC,£0000   of      LD BC,£0011
          EXOS _LINK
          RET
<-]
```

Dan nu het device. Het simpelste is het NULL-device uit ISDOS en het moeilijkste zal deze of gene uit onze vereniging wel maken. In principe hebben ze allemaal de zelfde bouw.

1. de DESCRIPTOR
2. de TABLE
3. de ROUTINES

de DESCRIPTOR.
Een device descriptor bestaat uit 8 vaste onderdelen die ook in een vaste volgorde staan. Het zijn:
. 3 bytes voor verwijzing naar het volgende device in de chain. U mag hier niets invullen omdat het volgende device nog onbekend is. EXOS zorgt voor een goede schakeling

```
[->     DD_NEXT_ADR:
         DEFW £0000
        DD_NEXT_SEG:
         DEFB £00                                              <-]
```

. 1 byte voor het dd_type field. Dit moet ALTIJD nul zijn.

```
[->     DD_TYPE:
         DEFB £00                                              <-]
```

. 1 byte voor de interrupt flag. Bij elk interrupt worden alle devices nagelopen of ze zouden willen reageren. De 4 soorten interrupts hebben alle een eigen bit in de flag. Het meest interessant zijn het 50 Hz video-interrupt op bit 5 en het external-interrupt op bit 7. Bedenk evenwel dat een paar devices met lange interrupt-routines erg snel die 1/50 seconde nodig hebben zodat niet U, maar de door U gebouwde devices met de Enterprise aan het spelen zijn!!

```
[->     DD_IRQFLAG:
         DEFB %?0?0?0?0          ;VRAAGTEKEN INVULLEN NAAR
                                  BEHOEFTE                     <-]
```

. 1 byte voor dd_flags. Dit byte bepaalt of het device de privileges van een video-device krijgt en de te openen channels binnen het video-gebied liggen (en dus te "displayen" zijn). Normale channels kunnen maximaal 16 K groot zijn, video channels mogen daar ver overheen 50K ? Privileges scheppen verplichtingen dus:

```
[->     DD_FLAG:
         DEFB £00                                              <-]
```

. 3 bytes voor het verwijzen naar de device_entry_table.

```
[->     DD_TAB:
         DEFW DEV_ENTRY_TAB
        DD_TAB_SEG:
         DEFB £00        ;NOG ONBEKEND                         <-]
```

De eerste twee bevatten het adres van de tabel en de laatste bewaart het segment waarin dit device zit. Bij het inlinken van het device worden deze adressen aangepast (zie aldaar).

. 1 byte voor dd_unit_count. Het is binnen EXOS mogelijk meerdere devices met de zelfde naam te definieren. Maakt U bijvoorbeeld het device "AGENDA:" en U wilt schoolzaken niet mengen met sport of Enterprise activiteiten, dan zet U dit dd_unit_count op non_

zero.

```
[->     DD_UNIT_COUNT:
         DEFB £??        ;£00 is goed
. 1 byte voor len(device_name)
[->     DD_NAME:
         DEFB £nn        ,£nn=£04
. n bytes voor de device naam
[->      DEFM "LAZY"
```

U bent nu klaar met de descriptor.

De TABLE.
De device_entry_table bevat 14 pointer (16-bit) naar 14 functie routines. Soms zijn enkele routines gelijk of bijna gelijk. Vele malen wijzen ze naar een RET. Bij een open_channel call komt U er niet zo gemakkelijk af, daar moet iets meer gebeuren. Maar nu de table. Zoals boven al is aangegeven draagt deze het label:

```
[->     DEV_ENTRY_TAB:                                          <-]
```

Gebruik meteen de goede namen en type in:

```
[->      DEFW INTERRUPT
         DEFW OPEN
         DEFW CREATE
         DEFW CLOSE
         DEFW DESTROY
         DEFW READ_CHAR
         DEFW READ_BLOCK
         DEFW WRITE_CHAR
         DEFW WRITE_BLOCK
         DEFW READ_CHAN_STAT
         DEFW SET_CHAN_STAT
         DEFW SPEC_FUNC
         DEFW INITIALISATION
         DEFW BUFFER_MOVE
<-]
```

De ROUTINES
Tot nu toe kon ik in het algemeen makkelijk wat over een device zeggen. Vanaf hier komen de GROTE verschillen tussen de doelstellingen van de devices tot uitdrukking. Neem nu "LAZY:", alle routines komen op het zelfde neer; HL wijst naar een string en een output-routine doet de rest. Alleen de Open en Create routines zijn "verplicht" een allocate buffer call te doen. Zie hier LAZY.

```
[->
        OPEN:
        CREATE:
         LD DE,£0000    ;ZERO BYTES CHANNEL BUFFER
         RST £30        ;EXOS
         DEFB £1B       ;_ALLOC_BUF
        INTERRUPT:
        CLOSE:
```

```
        DESTROY:
        READ_CHAR:
        READ_BLOCK:
        WRITE_CHAR:
        WRITE_BLOCK:
        READ_CHAN_STAT:
        SET_CHAN_STAT:
        SPEC_FUNC:
        INITIALISATION:
        BUFFER_MOVE:
        WA_BLIEF
         LD HL,SORRY_TEXT
         CALL WR_HL_OUT     ;general output routine
         RET

       ;DATA SECTION *********************************

       OUT_CHAN:
        DEFB £FF

       DEV_VIDEO:
        DEFB END_DEV_VIDEO-$-£01
        DEFM "VIDEO:"
       END_DEV_VIDEO:

       DEV_LAZY:
        DEFB END_DEV_LAZY-$-£01
        DEFM "LAZY:"
       END_DEV_LAZY

       SORRY_TEXT:
        DEFB END_SORRY_TEXT-$-£01
        DEFM "Sorry BOSS"
        DEFB CR,LF
       END_SORRY_TEXT:
```

<-]

Alhoewel U met "LAZY:" niet zo bar veel zult kunnen berijken geeft :XDEVS uit de toolkit wel degelijk een nieuw device in de lijst.

In het volgende onderdeel zullen we de routine-tuin eens door wandelen.

tot over twee maanden

Stan

LIDMAATSCHAP

Willen de mensen die voor 1988 nog geen lidmaatschapsgeld betaald hebben dit zo spoedig mogelijk doen door middel van het gireren van 45,- op gironummer 5989669 tnv penningmeester DEUG te zoetermeer.

Het is voor de DEUG natuurlijk niet mogelijk ENTERFACEs te blijven sturen aan mensen die geen lidmaatschap hebben betaald. Het volgende nummer zal daarom niet gaan naar mensen die niet betaald hebben.

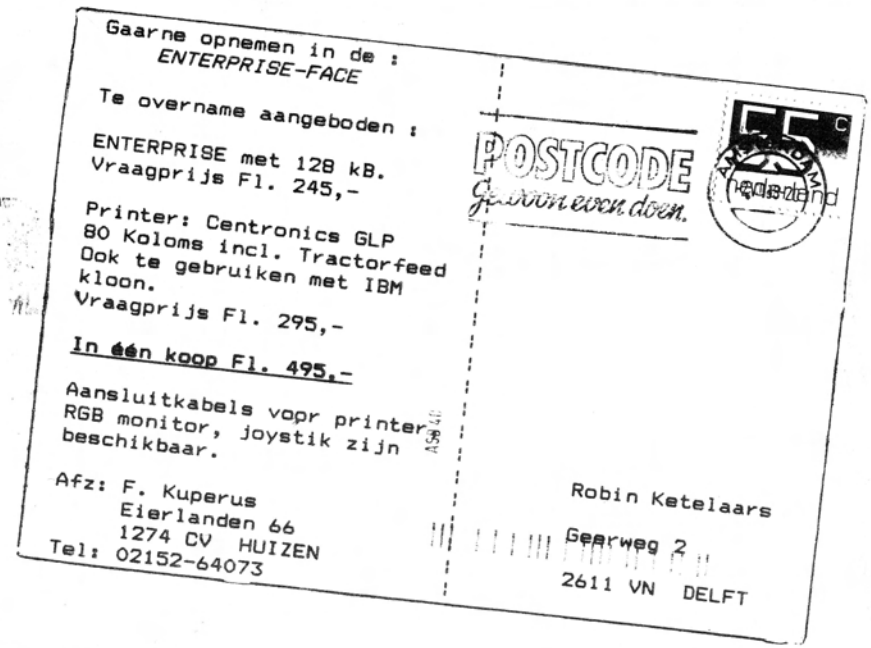
Je kunt zien of je betaald hebt (wanneer je dit niet meer weet) doordat de eerste twee cijfers van je lidnummer (zie adresetiket) overeenkomen met het jaar dat je het laatst contributie hebt betaald.

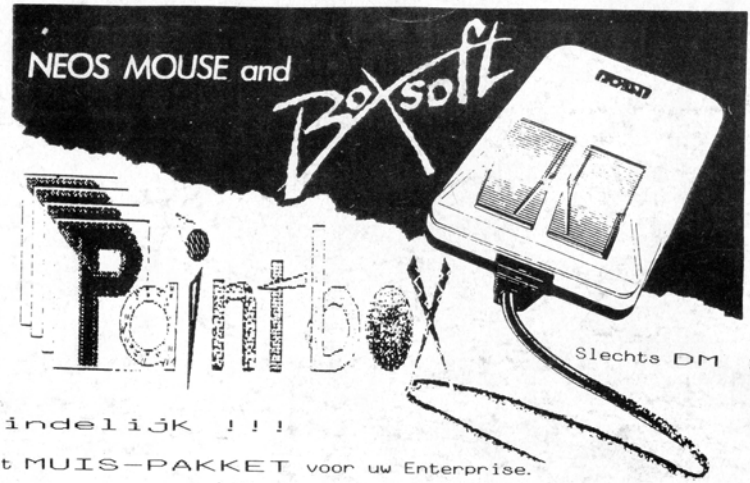
Dus wil je de vogende ENTERFACE ontvangen dan dient je lidnummer tenminste groter te zijn dan 88000.

Voor volgend jaar staan er 6 nummers op stapel + 6 gebruikersdagen (de eerste in 1989 op 28 januari). Wil je ze allemaal ontvangen dan dien je je lidmaatschapsgeld a 45,- voor 1989 zo spoedig mogelijk te voldoen op giro 5689669 tnv penningmeester DEUG te zoetermeer.

Is je lidnummer groter dan 89000 dan zit je het hele jaar 1989 gebeiteld.

ADVERTENTIE





Eindelijk !!!

Het MUIS-PAKKET voor uw Enterprise.

Met dit fantastische pakket haalt U niet alleen een kwalitatief zeer hoogwaardige muis in huis, maar ook een muisinterface (kan ook als intelligente joystickinterface voor Commodore of Atari joysticks gebruikt worden), een muis device-driver (wat U in staat stelt de muis vanuit al uw zelfgeschreven programma's te gebruiken) en een universeel printer-device (door instelbare stuurcodes in combinatie met praktisch iedere printer met graphic-mogelijkheden te gebruiken).

Bovendien krijgt U ook nog het PAINTBOX teken-programma, dat alle boven beschreven hard- en software uitbreidingen benut. Enkele features zijn:

* volledig menugestuurde software (ICONS)
* 12 grafische modi
* maximaal 256 kleuren
* variabele letterbreedte
* groot aantal vrij definieerbare tekenstiften
* airbrush (spuitbus) met vrij definieerbaar vulpatroon
* tekenstiften en vulpatronen kunnen op disk of cassette bewaard worden
* vergrootglasfunctie voor detailbewerkingen
* ...

Bestel de Paintbox vandaag nog bij:

Wernér Lindner
Hard- und Softwareideen
Landsberger Straße 49
D-8913 Schondorf a. A.

* Bij verzending naar Nederland wordt de Duitse BTW (14%) afgetrokken. DM 20,-- verzendkosten. Bestellingen alleen onder rembours (betalen bij ontvangst). Geen geld of cheques opsturen ! Alle programma's alleen op cassette verkrijgbaar, echter ten alle tijde op diskette kopieerbaar.