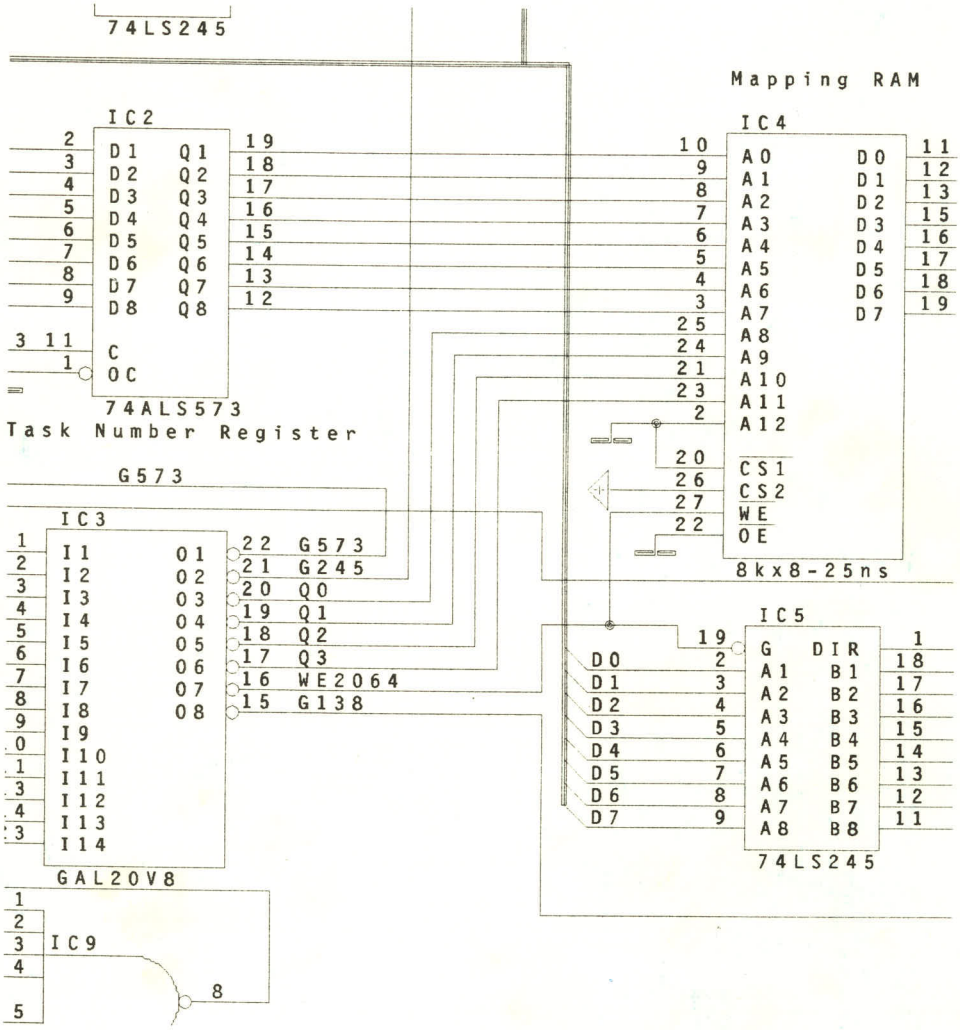
De µP Kenner

Zeventiende jaargang nr. 1
Februari 1993
80



In dit nummer o.a.:

Leer ook C
Spinneweb rond BBS
Bankswitched RAM voor DOS65
MacBarth is back and strikes again!
Slabakken en treuzelen in PostScript

Inhoudsopgave

De μP Kenner

Nummer 80, februari 1993
Verschijnt 5 maal per jaar
Oplage: 250 stuks
Druk: FEBO Offset, Enschede

De redactie:
Gert van Opbroek
Geert Stappers
Nico de Vries

Eindredactie:
Gert van Opbroek

Vormgeving:
Joost Voorhaar
Nico de Vries

Redactieadres:
p/a Gert van Opbroek
Bateweg 60
2481 AN Woubrugge

De μP Kenner nummer 81 verschijnt op 18 april 1993.

Kopijsluitingsdatum voor nummer 81 is vastgesteld op 4 april 1993.

Vereniging

Uitnodiging voor de clubbijeenkomst 5
Voortgang KGN68k 29
Van de bestuurstafel 37

Algemeen

Redactioneel 4
Netwerken op het BBS 6
Stoeien met PostScript, deel 2 9
Inleiding voor verantwoord computergebruik 19
Nieuwtjes en andere wetenswaardigheden 20

Software/Talen

VIRUS! 8
Programmeertalen 11

Hardware/DOS65

EP revisited 10
Een 1 Mbyte RAMkaart voor DOS65 21
Poorten in een PC 30

Systemen

De Motorola 68030; het hart van KGN68k (deel 4) 31

De μP Kenner is het huisorgaan van de KIM gebruikersclub Nederland en wordt bij verschijnen gratis toegezonden aan alle leden van de club. De μP Kenner verschijnt vijf maal per jaar, in principe op de derde zaterdag van de maanden februari, april, augustus, oktober en december.

Kopij voor het blad dient bij voorkeur van de leden afkomstig te zijn. Deze kopij kan op papier, maar liever in machine-leesbare vorm opgestuurd worden aan het redactieadres. Kopij kan ook op het Bulletin Board van de vereniging gepost worden in de redactie area.
Nadere informatie kan bij het redactieadres of via het bulletin board opgevraagd worden.

Redactioneel

Op het moment dat ik dit schrijf, proberen een aantal mensen met bloed zweet en tranen een blad bijeen te schrijven zodat we toch nog voor de bijeenkomst in maart met een μP Kenner uit kunnen komen. Dat dit gaat lukken is voor mij een gegeven; hoe en hoe dik, dat weet ik op dit moment nog niet.

Bij niet iedereen is bekend dat er weer een ingrijpende wijziging in bestuur en redactie is geweest. Nadat Jan Derksen per 1 januari zijn bestuurslidmaatschap had neergelegd, is Joost Voorhaar ook gestopt met zijn werk in de redactie en het bestuur. De reden die hij hiervoor aangeeft is dat de leden naar zijn mening te weinig interesse in de vereniging hebben. Hij leidt dit ondermeer af uit de geringe hoeveelheid kopij die spontaan door de leden ingestuurd wordt. Helemaal ongelijk kan ik hem daar niet in geven, maar ik denk er toch iets genuanceerder over. Ik ben er van overtuigd dat er nog veel interesse bij de leden is, dat de mensen alleen niet de moed hebben iets in het blad te schrijven of zitting te nemen in het bestuur. Denk daar toch eens over na. Ik zit zelf nu 7 jaar (!) in het bestuur en ik kan je verzekeren dat het heel leuk is. Verder is het ook zo dat ik van mijn geschrijf alleen maar beter ben geworden. Toen ik enkele maanden geleden een nieuwe baan nodig had, ben ik uitgebreid psychologisch getest. Daar kwam uit naar voren dat ik taalkundig behoorlijk sterk ben en ik kan je verzekeren dat dit voor een belangrijk deel komt door de ervaring die je opbouwt met het schrijven van artikelen. Verder is de technische kennis die je opdoet met het uitzoekwerk voor een artikel natuurlijk ook nooit weg.

Kortom: schrijf eens een bijdrage voor de μP Kenner.

Kortom schrijf eens een bijdrage voor de μP Kenner. Als je niet weet waarover, dan heb ik hier wat ideeën die ik graag uitgewerkt zou willen zien:

- MIDI, muziek-software etc.
- SoundBlaster, wat doe je daar eigenlijk mee?
- Werkt Windows nu echt zo lekker en hoe is het bijvoorbeeld in vergelijking met de Macintosh?
- Wie schrijft er in het vervolg de ShareWare bijdrage?
- Er is vraag naar een cursus programmeren in 'C', wie schrijft die?
- Fuzzy Logic
- Voor de rubriek "Toepassingen" zijn er vast wel artikelen te schrijven
- Er is al jaren vraag naar een beginnersrubriek; wie is net geen beginner meer zodat hij/zij anderen met zijn ervaringen kan helpen?
- Waarvoor gebruik je het computerpark dat je thuis hebt?

Deze lijst kan vast nog wel aangevuld worden. Kijk eens om je heen en schrijf een artikeltje. Anderen zijn over het algemeen zeer geïnteresseerd in wat je te schrijven hebt.

Blijft over het niveau van het blad. Uit de enquête komt naar voren dat een belangrijk deel van ons ledenbestand een opleiding heeft op het niveau MTS. In het verleden hebben we binnen bestuur en redactie besloten ons met name te richten op het niveau HTS en hoger. Het blijkt dus dat het niveau van het blad niet goed aansluit bij het niveau van de leden. Zodoende komen er ook geregeld opmerkingen dat men het blad "te moeilijk" vindt. Probleem is alleen dat mensen die de kopij aanleveren meestal ook op dat hogere niveau zitten en het is heel moeilijk een paar treden af te dalen. Kortom, alweer een reden voor het gemiddelde clublid eens een artikeltje op zijn eigen niveau te schrijven.

Mocht je met mij of één van de andere bestuursleden willen overleggen hoe je actiever binnen bestuur en of redactie kunt worden, dan kun je ons bereiken op de telefoonnummers die achterin het blad staan of uiteraard via The Ultimate.

Groetjes van je redacteur ad interim:

Gert van Opbroek

Uitnodiging voor de clubbijeenkomst

Datum: 20 maart 1993
Lokatie: gebouw 't Kruispunt
Slachthuisstraat 22
5664 EP Geldrop
Telefoon: 040-857527
Thema: Toekomst DOS-65
Entree: Leden gratis
Niet-leden fl. 10,--

Routebeschrijving

Trein:

Geldrop is ieder half uur bereikbaar per trein (stoptrein Eindhoven-Weert). Vanuit het station rechts afslaan, de Parallelweg, dan tweede straat links, de Laarstraat. Aan het einde daarvan rechts afslaan en direct daarna weer linksaf, de Laan der vier Heemskinderen. Op de hoek van de eerste straat links, de Slachthuisstraat, vindt u het gebouw 't Kruispunt.

Auto:

Vanaf 's Hertogenbosch of Breda naar autoweg Eindhoven-Venlo. De eerste afslag na Eindhoven is Geldrop. Ga richting Geldrop, dan komt u vanzelf op de Laan der vier Heemskinderen, dit is nl. een verplichte afslag naar rechts. Zie verder boven.

Vanaf Eindhoven door het centrum van Geldrop richting Heeze. Na winkelstraat en daarna het ziekenhuis aan de rechterzijde de eerste straat links bij de stoplichten. Dit is de Laan der vier Heemskinderen. Zie verder hierboven.

Programma:

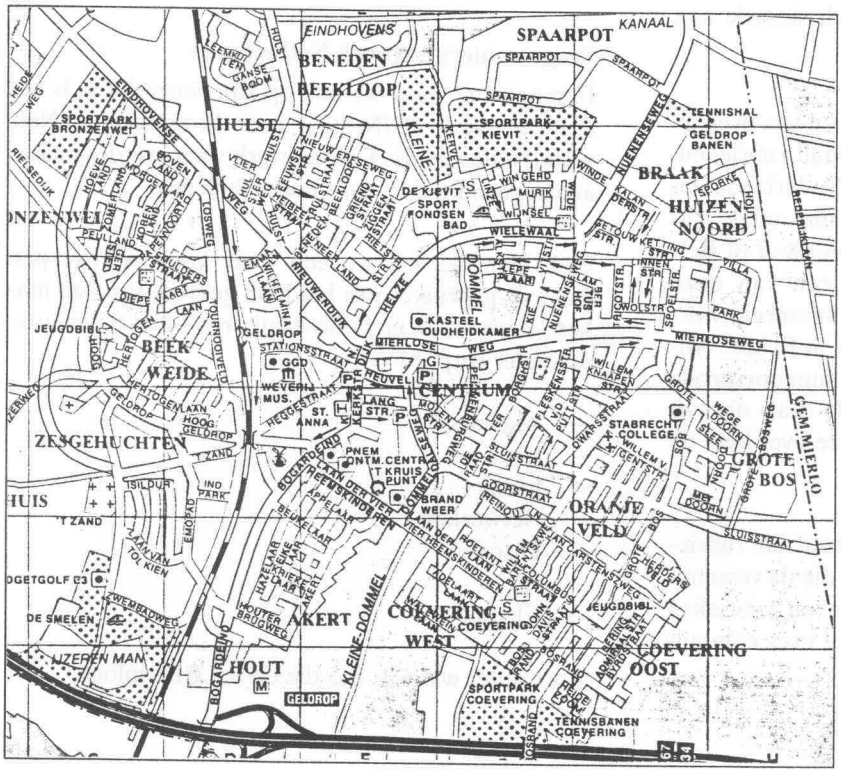
9:30 Zaal open met koffie
10:15 Opening
10:30 Plannen t.a.v. DOS-65; wat zijn de mogelijkheden, wat zijn de wensen.
11:45 Forum en markt.
12:00 Lunchpauze.

Aansluitend het informele gedeelte met de mogelijkheid om andermans systemen te bewonderen en Public Domain software uit te wisselen. U en uw systeem zijn uiteraard van harte welkom.

17:00 Sluiting

Let op:

Het is ten strengste verboden illegale kopieën van software te verspreiden. Aan personen die deze regel overtreden zal de verdere toegang tot de bijeenkomst ontzegd worden. Breng verder alleen software mee die u legaal in uw bezit heeft. Het bestuur aanvaardt geen enkele aansprakelijkheid voor de gevolgen van het in bezit hebben van illegale software.



Hiernaast een hulpje hoe U moet rijden...

Netwerken op het BBS

Inleiding

Een nadere uitleg van de verschillende netwerken waar het BBS op is aangesloten is waarschijnlijk wel gewenst. Vooral omdat het nu toch best wel een beetje uit begint te groeien zodat ik het zelf soms niet meer overzie... (foutje????)

In dit artikel komen twee soorten gebieden naar voren. Dat zijn in de eerste plaats de Echo-Mail gebieden en in de tweede plaats de file-area's. Echo-Mail is met name bedoeld voor het versturen van berichten terwijl de file-area's, de naam zegt het al, bedoeld zijn voor programma"s etc. Verder heeft de Echo-Mail nog als eigenschap dat een bericht dat in één van deze rubrieken wordt geschreven tijdens nachtelijke connecties wordt verstuurd naar de BBS-sen die ook op die rubriek zijn aangesloten.

FidoNet

Als eerste bestaat er het zogenaamde FidoNet. Het merendeel van de rubrieken op The Ultimate komt vanuit dit netwerk. Even ter verduidelijking, alle verdere netwerken maken eigenlijk ook gebruik van het FidoNet principe c.q. de structuur van FidoNet. Alleen hebben de andere netwerken niets met de policies (regels) te maken die in het FidoNet gelden. Verschillende netwerken zijn dan ook ontstaan doordat "men" zich stoorde aan de "regels" binnen het FidoNet-werk.

Vooral omdat het nu toch best wel een beetje uit begint te groeien zodat ik het zelf soms niet meer overzie...

Maar dat is weer iets anders. Eerst even iets over de rubrieken, met name over de Echo-Mail rubrieken. De rubrieken zijn onder te delen in Nederlands- en anders-talige rubrieken. De Nederlands-talige rubrieken vinden hun oorsprong in Nederland of België. Berichten die men schrijft in één van deze rubrieken zullen dus ook niet verder verspreid worden dan Nederland en misschien ook in België. Anders-talige rubrieken hebben meestal hun oorsprong in het land waar de voertaal hetzelfde is als die van de rubriek, of ze is in het Engels als ze over heel de wereld wordt verspreid.

Er staat een behoorlijke verscheidenheid aan rubrieken op het BBS. Zo zijn er rubrieken die de verschillende programmeertalen behandelen en rubrieken waar men het heeft over netwerken of over communicatie programmatuur.

DENKNET

We zijn ook aangesloten op het zogenaamde "DENKNET", een netwerk bedoeld voor mensen die als hobby het uitoefenen van een denksport zoals schaken of dammen op de computer hebben. Verschillende probleemstellingen komen daarbij aan de orde. Er zijn aan de berichtengebieden ook verschillende file-gebieden gekoppeld waarin men de verschillende probleemstellingen kan binnenhalen (downloaden) waarna men ze rustig thuis, offline, kan bestuderen.

Gernet

Een volgend netwerk waar we op aangesloten zijn is het Gernet. Dit is het netwerk waar het computerblad C't de drager van is. De sources die in het blad staan afgedrukt worden ook via dit netwerk verspreid, zoals we dat ook doen met de sources die in de μP Kenner staan afgedrukt. Ook vind je in de betreffende file-rubriek de inhoudsopgaven van de C't. Zo kan men ongeveer een halve maand voordat de nieuwe C't uitkomt al op het BBS bekijken welke items in de volgende C't ter sprake komen.

De voertaal van de berichten is vanzelfsprekend Duits.

Programmers Network Netherlands

Een netwerk waar we ook op zijn aangesloten is het PNN netwerk. PNN staat voor Programmers Network Netherlands. Een netwerk waar we door onze achtergrond wel in thuis horen leek ons.

Ook hier hebben we een keur aan onderwerpen waar de gebruiker van het BBS gebruik van kan maken. Zo bestaan er Echo-Mail area's met de volgende onderwerpen:

- Algemeen
- Unix
- Linux
- Netwerken
- Pascal
- C
- Electro

Voor ieder wat wils dus die op het BBS inlogt.

Aan file area's biedt ons dit netwerk de volgende rubrieken:

- Unix
- Linux
- Netwerk-software
- Pascal
- C

Dus ook de nodige rubrieken om in de gaten te houden indien men zich interesseert voor één of meer van deze items.

ShareWare Distribution Network

Een netwerk waar we al erg lang mee verbonden zijn, naast de aansluiting binnen FidoNet is het SDN-netwerk (SDN = ShareWare Distribution Network). Binnen dit netwerk bestaat er een berichtenrubriek waar nieuwe share-ware programmatuur besproken wordt of waar men vragen kan stellen over share-ware programmatuur die via het SDN netwerk is verspreid. De file-gebieden hebben ook een behoorlijke verscheidenheid aan keuzes. Een minpuntje is misschien dat er vaker programmatuur wordt verspreid die specifiek bedoeld is voor de amerikaanse gebruiker. Het netwerk heeft namelijk een amerikaanse oorsprong.

Zo kunnen we het volgende via SDN aanbieden, wel allemaal voor de PC reeks bedoeld:

- Programmatuur voor de zakelijke omgeving, helaas veelal afgestemd op de amerikaanse markt.
- Communicatie software
- DataBase en Spreadsheet programmatuur
- Grafische programmatuur, ook voor de allerkleinsten zit er leuk spul tussen.
- Diverse utilities, puur om het werk op de PC te vereenvoudigen, b.v. een makkelijk kopieer-programma of een harddisk menu-programma.
- Tekstverwerkers of utilities daarvoor.
- Miscellaneous files over van alles en nog wat
- Spelletjes, voor de allerkleinsten of voor de gevorderde PC gebruiker
- Teksten over de meest verschillende onderwerpen.

Al deze programmatuur heeft als extensie (achtervoegsel) .SDN. Dit staat voor de compressie methode die men gebruikt heeft. Het decomprimeer-programma dat de gebruiker dient te gebruiken is PAK251.EXE. Dit programma staat ter download in het hoofdmenu van het BBS.

Naast de meeste files in deze rubrieken staan er zogenaamde .SDA files. Deze files hebben veelal de zelfde naam als de .SDN file. Ze horen dan eigenlijk ook bij elkaar. De .SDA file bevat namelijk een uitleg over de .SDN file, eigenlijk een korte beschrijving van/over de .SDN file. Deze files staan niet gecomprimeerd op het BBS en kunnen dus bekeken worden voordat men een .SDN file gaat downloaden. De .SDN files zijn namelijk soms nogal groot en het is dus wel aardig als men eerst kan kijken wat men eigenlijk gaat binnenhalen (downloaden). De file is gewoon te bekijken via het BBS. Of men download eerst de .SDA file om dan offline eerst te lezen wat voor een file men wil gaat downloaden.

Diversen

Tenslotte zijn we nog aan diverse andere file-netwerken aangesloten zoals:

- DVN :: DesqView Network, programmatuur dus waar men utilities of patches aanbiedt voor onder een DesqView omgeving.
- WINNET :: Windows Network, een netwerk waar men utilities aanbiedt voor windows, zoals printer-drivers, Grafische drivers (b.v. drivers voor verschillende VGA kaarten/versies), spelletjes, programmatuur enz.
- SND :: Sound Network Distribution, programmatuur voor verschillende geluidskaarten.
- ANSI :: Verschillende programmatuur voor ANSI-omgevingen. Veel plaatjes en ook de programmatuur waarmee men zulke plaatjes kan maken. Met of zonder geluid.
- CLN :: Clipper Network, programmatuur en verschillende utilities voor een clipper omgeving.
- ADS en SKY netwerk :: Dit zijn beide Amiga netwerken via welke ook weer een verscheidenheid aan programmatuur binnenkomt. Per week ongeveer een 500kb tot 1Mb aan software.
- AST Netwerk :: Een Atari File netwerk waar helaas niet erg veel voor binnenkomt. Maar ook voor de Atari komen er dus geregeld nieuwe files binnen via dit netwerk op het BBS.
- VGA-Net :: Een netwerk waarin van allerlei VGA plaatjes en utilities worden verspreid. Van mooie vrouwen tot auto's of schilderijen die men gedigitaliseerd heeft.

Naast al die soorten van Echo-Mail en file netwerken hebben we ook nog een hele andere schare aan rubrieken. Zo hebben we in de loop van de tijd dat het BBS in de lucht is een hele verzameling aan software verzameld en we proberen die altijd te verdelen over de verschillende rubrieken zoals:

- Algemeen informatie
- Comprimeer tools
- Communicatie pakketten
- PC utilities weer onderverdeeld in:
- File/Disk utilities
- Tekstverwerkers en utilities hiervoor

- Spelletjes
- Keyboard -
- Printer -
- Display -
- Memory utilities

Eigenlijk teveel om op te noemen.

The Ultimate

Men prijst me wel eens voor de verscheidenheid aan programmatuur die op het BBS te vinden is, maar zegt er tegelijk bij dat het eigenlijk teveel is. Men kan de software niet meer vinden, zoiets van: Men ziet door het bos de bomen niet meer. Tja, een grote verscheidenheid aan programmatuur brengt dit waarschijnlijk met zich mee. Tenslotte zitten we op het moment met een 185 file-rubrieken verdeeld over zo'n 500Mb aan disk-ruimte en groeien gestaag verder. Zijn we zo'n 6 jaar geleden begonnen op een XT met 20Mb en één (1) telefoon lijn. Nu zitten we te werken met twee 386 DX systemen aan elkaar gekoppeld via netwerkkaarten en hebben we vier (4) telefoonlijnen en 1Gb (1.000Mb) aan diskruimte. Dat wil gelukkig niet zeggen dat we het zo slecht deden in het begin van het BBS. 6 Jaar geleden was het gewoon niet nodig (en niet te betalen!) om zoveel diskruimte aan je BBS te hangen. De programmatuur was waarschijnlijk ook beter geschreven waardoor je niet zo ontzettend veel ruimte nodig had.
Een praktisch programma nam niet meer dan een paar tot hooguit enkele 10-tallen kb's in beslag, terwijl tegenwoordig met de hele PC/AT reeks en Macintosh machines, de Amiga 4000 en vooral de grootte van het interne geheugen en daarmee samengaand de grootte van de harddisk de schrijvers er niet meer voor terug deinzen programmatuur te schrijven die snel 500Kb tot vele Megabytes diskruimte in beslag neemt. Vooral een grafische interface neemt nogal wat geheugen in beslag.

We worden als maar groter wat we natuurlijk perfect vinden. De illusie dat we de grootste van Nederland zouden kunnen worden hebben we echter al heel snel laten varen. Er bestaan andere BBS'en - echt professioneel opgezet- met 32 of 64 telefoonlijnen en meerdere Giga-bytes online dus dat kunnen we gevoeglijk vergeten. Maar daar gaat het tenslotte ook niet echt om. Bewijst het BBS in deze vorm z'n nut of niet, dat is het eerste waar we aan moeten denken. En ons inziens lukt dat aardig. Het BBS wijst niet alleen nieuwe mensen de weg naar onze vereniging, de donateurs die het BBS ook nog eens rijkelijk is steunen het BBS dat hun uitstekend bevalt, waardoor de uitgaven van de vereniging voor het BBS verantwoord worden besteed en toch in het redelijke gehouden kunnen worden.

Ook aan Echo-Mail rubrieken loopt het steeds verder op. Op dit moment is de keuze ongeveer verdeeld over zo'n goede 100 rubrieken met een verscheidenheid aan onderwerpen. Maar indien je iets zoekt, en je vindt het onderwerp er niet tussen. Laat het dan eens weten, misschien is de rubriek wel voor handen maar hebben we de rubriek gewoon niet aangesloten omdat er nog geen animo voor was. Een verzoek om de rubriek aan te sluiten wordt nooit afgewezen. Met die uitzondering natuurlijk dat ze ook ergens te krijgen is.

Uw Sysop ,Jacques Banser

VIRUS!

Bron: Automatiserings gids 27e jaargang Nr.4

Windows is geen applicatiepakket of een integraal deel van het besturingssysteem van een PC. Het is een virus, menen deskundigen in de VS en het waterdichte bewijs voor deze stelling circuleert over vele elektronische prikborden. Het bulletin-board van de Electronic Frontier Foundation (EFF) verwoordde het bewijs als volgt:

Windows is een virus, gezien de zeer grote verspreiding van deze software. Vrijwel iedereen die over een PC beschikt, heeft ook Windows. De software 'vreet' letterlijk opslagruimte op schijf en het neemt de besturing van de computer geheel over. Bovendien zorgt de programmatuur ervoor dat de computer in kwestie langzamer gaat lopen, terwijl op de meest onvoorspelbare momenten I/O-handelingen worden verricht. Wanneer Windows actief is, verschijnen er malle boodschappen op het scherm en als klap op de vuurpijl kan de PC door toedoen van Windows ook nog eens geheel 'crashen'. Met andere woorden: het pakket vertoont alle eigenschappen van een computervirus.

Stoeien met PostScript, deel 2

De vorige keer heb ik jullie verteld wat er minimaal nodig is om "Hello World" op papier te krijgen. Het bleek niet anders te zijn dan een printopdracht voorafgegaan door een initalisatie. Wat bij de ene taal **PRINT "String"** of **printf("string");** is, bleek in PostScript **(string) show** te zijn.

Programmeeromgeving

Wat ik de vorige niet verteld is wat U nodig heeft om met PostScript te kunnen stoeien:

- Een editor die gewoon botte ASCII, platte tekst of DOS-tekst kan afleveren. Scheidingstekens tussen de regels is **CR** en/of **LF**.
- Een utility waar je de PS-file mee kunt downloaden. In Novell netwerk bijvoorbeeld met **NPRINT psfile /Q=laser /NB**, waar **psfile** de PostScript file die met de editor hebt aangemaakt en en **laser** de naam van de printque is waar de PostScript print mee verbonden is. **/NB** zorgt er voor dat niet de zogenaamde 'banner', die geen PostScript code is, naar de printer gaat.
- En last but not least een PostScript device.

Lijntjes papier

Wat ik deze keer van PostScript wil weten is hoe een herhalingsstructuur er uitziet. Het Einddoel wat ik voor ogen heb is een blad gelinieerd papier. Na een bepaalde **afstand** moet er een lijn, met een bepaalde **dikte**, van **links** naar **rechts** getrokken worden. Dat moet herhaald worden totdat we aan het **eind** van de pagina zijn.

In den beginne

Dankzij een schepper hebben wij nu een uitgangspunt. Hoe het standaard in PostScript zit, zal ik voor de duidelijkheid toch maar even vertellen. De oorsprong ligt linksonderaan op de pagina. De positieve X-as gaat naar rechts en de positieve Y-as gaat naar boven. De eenheden (units) zijn in één twee en zeventigste (1/72) inch. Het punt **144 72** bevindt zich dus 2 inch rechts en 1 inch boven van de linkeronderhoek.

for

Op zoek naar herhalingstructuren kwam ik de operator **for** tegen. In BASIC heet het een "for next lus". De syntax is **initial increment limit proc FOR** . De procedure **proc** wordt herhaald in stappen van **increment** totdat **initial** de waarde **limit** bereikt heeft.

Forth

Echt in Forth heb ik nog nooit geprogrammeerd, maar wat me er nog van bij staat, heeft PS er wel wat van weg. In de listing kun je ook zien dat PostScript meer is als een printerdefinitie. Het kan aan mij liggen, maar ik ken geen andere uitvoerapparaten, die zo aangesproken moeten worden.

Geert Stappers

Literatuuroverzicht:

* PostScript Language Reference Manual, second edition; Adobe Systems Incorporated; Addison-Wesley; ISBN 0-201-18127-4
* Handboek voor PostScript Programmeurs; David A. Holzgang; Kluwer; ISBN 90-201-2360-2
* Text und Grafik seitenweise; Tilo Rossmanith, C'T November 1991, Heize Verlag


```
%Lijntjes papier
%Jawel een procent geeft commentaar aan
%
%Constanten en omrekeningen hiervan
%uitgegaan van A4 formaat
/cm {72 mul 2.54 div} def
/breedte 21 cm def
/hoogte 29.7 cm def

%Declaratie & Initalisatie van variabelen
/afstand 50 def
/dikte 0.5 def
/links 0 def
/rechts breedte def
/eind hoogte def
/regel 0 def

%lijndikte instellen
dikte setlinewidth

%voorbereiding voor 'for' operator
regel afstand eind

%wat er herhaald moet worden
{
/regel regel afstand add def
links regel moveto
rechts regel lineto
closepath
stroke
}

%zeggen dat het herhaald moet worden
for

%en het resultaat willen we zien
showpage

%End of File
```

Fig .1: lijntjespapier in PostScript

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.

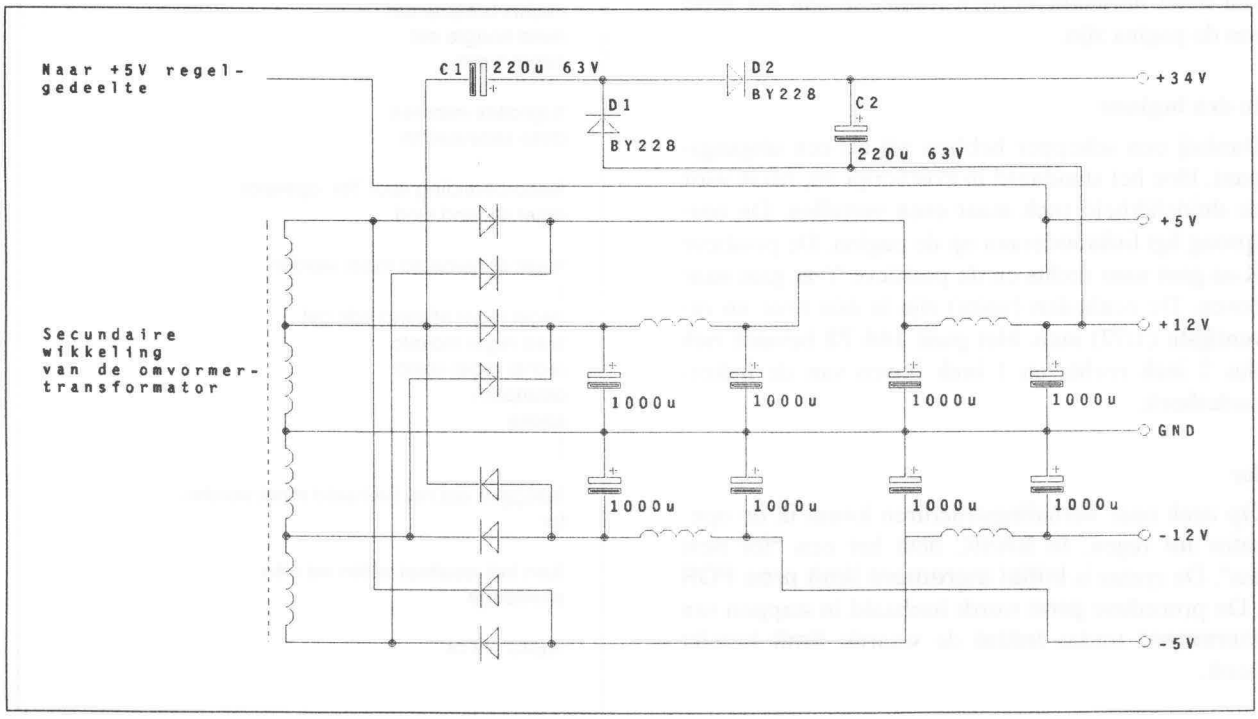
Tekenfoutje in het schema

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

Programmeertalen

Inleiding

Een deel van de KGN-leden zijn onlangs telefonisch door Tonny benaderd met enkele vragen. Deze enquête was voor het bestuur een manier om ook eens in contact te komen met leden die niet geregeld op bijeenkomsten komen. Nu is het niet de bedoeling onder de kop "programmeertalen" een samenvatting van deze enquête te geven maar wel om al vast een stapje te zetten in een door de leden gewenste richting.

In de enquête kwam naar voren dat een deel van de leden behoefte heeft aan een cursus in de programmeertaal 'C'. Dat is eigenlijk best begrijpelijk. Vrijwel de hele Unix wereld is 'C' en ook bij het KGN68k-project komt deze taal zeer veelvuldig naar voren. Een tweede reden waarom er in de μP Kenner ruimte is voor een dergelijke cursus is het feit dat het bestuur en de redactie toch een klein beetje een verkeerd beeld van het ledenbestand blijken te hebben. Wij gingen er namelijk vanuit dat het merendeel van de leden een opleiding op minimaal HBO-niveau zouden hebben en dat ze vaak ook in de automatisering werkzaam zijn. Dit blijkt echter niet helemaal juist te zijn. Zeker de helft van de leden heeft een opleiding op het niveau MBO. De klacht dat het niveau van de μP Kenner wat aan de hoge kant is werd dan ook veelvuldig geuit. Dat geeft helemaal niet, in tegendeel, de redactie vindt het juist prettig te horen op welke punten het blad verbeterd kan worden.

Een deel van de leden heeft behoefte aan een cursus in de programmeertaal 'C'.

Al met al betekent dit dat het waarschijnlijk heel nuttig is in een aantal afleveringen wat meer te gaan vertellen over een programmeertaal, bijvoorbeeld over de taal 'C'. Blijft alleen het feit dat ik waarschijnlijk niet de juiste persoon ben om deze cursus alleen te schrijven. Ten eerste omdat ik de taal 'C' ook niet vloeiend schrijf (spreek?), ten tweede omdat ik het belangrijk vindt dat er van tijd tot tijd ook eens wat nieuwe auteurs aan bod moeten komen. Kortom, zie dit verhaaltje als een inleiding op een cursus programmeren in 'C' en denk er eens over na of je niet zelf één of meer afleveringen zou willen schrijven. Ik wil je daar natuurlijk graag bij helpen. Mochten er mensen zijn die zich aangesproken voelen, laten die dan even contact met mij opnemen.

Generaties

De termen 4 GL, fourth generation language of vierde generatie taal zijn tegenwoordig echte modewoorden. Waar komen die termen vandaan en wat zijn de andere generaties?

Welnu, de term "derde generatie" is waarschijnlijk als eerste geïntroduceerd door IBM in de jaren 60. IBM bracht toen de 360-serie computers uit en de drie betekent in dit geval "Computersysteem van de derde generatie". Dit computersysteem had namelijk de beschikking over een echt operating systeem dat er voor kon zorgen dat de verschillende programma's achter elkaar automatisch opgestart werden en dat de resultaten op papier werden afgedrukt. Dit was het begin van de zogenaamde "Mainframes", grote computers, opgesteld in een computer- "Zaal" die werden bediend door "Operators" die in "Shifts" werkten. Mensen die is staat waren die machine te programmeren werden op een voetstuk geplaatst en hadden binnen een bedrijf vaak meer macht dan de directie.

Het is niet mijn bedoeling alle generaties computer- en operating systemen te behandelen, dit artikel zou tenslotte over programmeertalen gaan. Wel wil ik nog even kwijt wat de vijfde generatie (we hebben tegenwoordig de vierde generatie) moet worden. Welnu voor een deel van de lezers is dat eenvoudig uit te leggen: De vijfde generatie is vergelijkbaar met het HAL 9000 (Heuristically ALgorithmic computer) computersysteem uit de film 2001: A Space Odyssey van Stan Kubrick. Voor andere mensen nog even een echte uitleg: Het ontwikkelen van computers van de vijfde generatie is een aantal jaren geleden als project gestart in Japan. Dit zouden computers moeten zijn waar gewone mensen mee om moeten kunnen gaan en die, zonder dat ze daar speciale programma's voor hoeven te schrijven, de computer problemen op kunnen laten lossen. Je problemen moet je verder op eenvoudige manier, bijvoorbeeld via spraakherkenning, aan de computer voor kunnen leggen:

"MIK, zoek eens voor mij uit hoeveel geld ik het afgelopen jaar aan eten besteed heb en wat het mij extra kost als Oma met Kerst bij ons komt dineren". Waarna computer MIK ogenblikkelijk antwoord: "Drieëntwintig gulden twee en veertig als je de boodschappen bij BI koopt en daar op fiets heen gaat".

Zover zijn we nog niet, alhoewel in laboratoria al zeer interessante voorlopers van deze hulpjes staan opgesteld. Het lijkt allemaal Science Fiction maar het zal over niet al te lange tijd realiteit worden.

De generaties in computers en operating systemen vindt je ook terug in de bijbehorende programmeertalen. Laten we het daar eens over gaan hebben.

Eerste generatie: Machinetaal

De eerste generatie computertalen is wat de meeste mensen ook wel machinetaal noemen. Elke computer heeft binnen in zich een ding dat we de processor noemen en waar de verwerking plaats vindt. Deze processor leest de dingen die hij moet doen (de instructies) uit het geheugen en voert ze uit. Het geheugen van een computer is opgebouwd uit allemaal cellen die "iets" hebben of "iets" niet hebben. Vroeger was dat "iets" meestal een magneetje; het magnetisch veld wees in de ene richting of in de andere richting. Tegenwoordig is dat "iets" een spanning op een uitgang van een zogenaamde flipflop (statisch RAM) of een lading op een condensator (dynamisch RAM). Om het gemakkelijke te maken geven we een cel die het "iets" heeft de waarde 1 en een cel die het "iets" niet heeft de waarde nul en noemen we een dergelijk cel een "Bit" van Binairy Digit. Elke instructie wordt gecodeerd met een unieke reeks van enen en nullen. Deze reeks noemen we de machinetaal en bij computers die je in machinetaal moest programmeren werden die enen en nullen met bijvoorbeeld schakelaars in het geheugen gezet. Voorbeelden van dergelijke computers zijn de KIM (ja, die waar de club zijn naam aan te danken heeft) en de Elektuur Junior.

Vroeger was dat "iets" meestal een magneetje; het magnetisch veld wees in de ene richting of in de andere richting.

Om geen hele reeksen van nullen en enen te hoeven onthouden heeft men bedacht dat het wel handig is een groepje van drie of vier Bits bij elkaar te nemen. Zo krijg je de zogenaamde "Octale" en "Hexadecimale" getallen, die ook in de 'C' cursus terug zullen komen:

Octaal:

000	= 0	100	= 4
001	= 1	101	= 5
010	= 2	110	= 6
011	= 3	111	= 7

Hexadecimaal:

0000	= 0	1000	= 8
0001	= 1	1001	= 9
0010	= 2	1010	= A
0011	= 3	1011	= B
0100	= 4	1100	= C
0101	= 5	1101	= D
0110	= 6	1110	= E
0111	= 7	1111	= F

Op deze manier kun je de machinetaal instructies iets gemakkelijker onthouden: A9 = Neem de inhoud van de volgende geheugen-locatie over in de processor.

Tweede generatie: Assemblers

Omdat het wel lastig is precies de codes voor de instructies te onthouden, heeft men voor elke instructie een naam bedacht. De instructie die als code A9 heeft kreeg de naam LDA # of te wel Load Accumulator Immediate. Als je weet hoe de processor in elkaar zit en welke onderdelen hij heeft is het gemakkelijker deze naam (in het vakjargon Mnemonic) te onthouden dan de bijbehorende code. Om niet zelf iedere keer bij een Mnemonic de code te hoeven zoeken en om ook precies bij te houden waar wat in het geheugen van een computer staat heeft men programma's gemaakt die in staat waren de mnemonics in te lezen en ze vervolgens om te zetten in de nullen en enen die de computer begrijpt. Vervolgens konden die nullen en enen automatisch of met de hand in de computer gezet worden en kan het programma worden gedraaid. Een dergelijke omzetter van mnemonics naar machinetaal noemt men een "Assembler" en assemblers zijn programmeertalen van de tweede generatie.

In figuur 1 staat een stukje uitvoer van een assembler. Dit noemen we de listing. Aan de hand van dit stukje programma voor een Motorola 68000, wil ik nog een paar zaken uitleggen. De eerste kolom in de listing is het regelnummer in het programma, niets spannends dus. De tweede kolom geeft het adres in het geheugen van de computer aan waar de instructies die op die regel staan terecht moeten komen. Het startadres wordt aangegeven door de opdracht 'ORG' waarna de assembler zelf de adressen bijhoudt. De derde kolom geeft aan wat er in het geheugen op het betreffende adres komt te staan; dit is dus de machinetaal. Vroeger moest je dat met de hand in het geheugen zetten; nu zijn daar gelukkig hulpmiddelen voor. Op de rest van de regel staat de

```
KGN-68k assembler version 1.0a (c) KIM Gebruikersclub Nederland  page 1
26 Feb 1993 19:04:49  Demo second generation language -  (demo.asm)

 1   ffffffff                    NAME Demo second generation language
 2   ffffffff                    ;
 3   ffffffff                            ORG     $1000
 4   00001000                    Times   EQU     10
 5   00001000                    ;
 6   00001000  700A              Start   move.l  #Times,d0   ; Load register d0 with Times
 7   00001002  4281                      clr.l   d1          ; Clear register d1
 8   00001004  5281              Loop    addq.l  #1,d1       ; Add 1 to regsiter d1
 9   00001006  5380                      subq.l  #1,d0       ; Subtract 1 from register d0
10   00001008  66FA                      bne.s   Loop        ; If d0 not equal 0, goto Loop
11   0000100a                    ;
12   0000100a  60FE              Hang    bra.s   Hang        ; Loop here forever
13   0000100c                    ;
14   0000100c                            END

Symbol table :

Times $0000000a  Start           $00001000
```

Fig. 1: uitvoer van een assembler

eigenlijke instructie met o.a de mnemonic en uiteraard een hoeveelheid commentaar.

Wat je ook ziet in het stukje programma is het gebruik van zogenaamde labels. In de eerste plaats komt er een constante Times in voor die gelijk gemaakt wordt aan de waarde 10. Verder staat er op drie plaatsen een zogenaamd label in het programma. De assembler onthoudt nu op welke plaats in het geheugen de betreffende regel terechtkomt en zal er voor zorgen dat bijvoorbeeld sprongopdrachten precies naar die geheugenplaats uitgevoerd worden. Naar welke plaatsen in het geheugen die labels verwijzen, kun je ook nog aflezen uit de zogenaamde Symbol Table, die als laatste staat afgedrukt.

Wat je in figuur 1 ook nog ziet is dat een assembler (tweede generatietaal) een hulpmiddel is om machinetaal (eerste generatie taal) te genereren.

Derde generatie: "Hogere" programmeertalen

Assembler-talen hebben een paar hele grote nadelen. Het grootste nadeel is waarschijnlijk wel dat ze machine-afhankelijk zijn. Dit betekent dat iemand die goed uit de voeten kan met een assembler voor een DOS-65 (een 6502-systeem) en daar een programma voor ontwikkeld heeft, dit programma niet "zo maar" om kan zetten naar een programma voor de PC (die op 8088 etc. is gebaseerd). Voor elke nieuwe computer moet je een compleet nieuwe assembler leren. Een tweede nadeel van assemblertalen is dat ze meestal niet meer bieden dan de processor "in huis" heeft. Dit betekent dat je meestal alles zelf moet programmeren: heeft de processor geen floating point instructies, jammer dan, je zult ze zelf moeten programmeren. Tenslotte moet je bij een assembler vrijwel alles zelf regelen. Waar zet je wat neer in het geheugen, maak je gebruik van registers etc. Nogal omslachtig dus. Hiertegenover staat echter één heel groot voordeel: In assembler geschreven programma's zijn verreweg het snelst omdat je alles optimaal kunt regelen en er geen overbodige of inefficiënte instructies gebruikt worden.

Om het allemaal wat gemakkelijker te maken zijn er enkele tientallen (honderden?) programmeertalen ontwikkeld. Voorbeelden zijn o.a Ada, Algol, APL, Basic, C, Comal, Cobol, Forth, Fortran, Lisp, Logo, Modula II, Pascal, PL/1, Prolog, RPG, SmallTalk en Simula. Veel talen hebben hun eigen specifieke toepassing. Cobol en RPG worden veel gebruikt in administratieve omgevingen, Fortran vindt je voornamelijk bij mensen die veel (wetenschappelijke) berekeningen doen, Lisp en Prolog worden voornamelijk gebruikt voor zogenaamde Kunstmatige Intelligentie en Simula kom je nog al eens tegen bij programma's die iets moeten simuleren (bijvoorbeeld hoe een groep mensen in paniek een winkel verlaat). Talen die tegenwoordig voor "van alles en

nog wat" gebruikt worden zijn Basic (ja zeker!), C en Pascal.

Het belangrijkste nadeel van assembler talen is dat ze slechts op één type computer kunnen draaien. Dit nadeel heb je niet bij derde generatie talen. Van elke taal bestaat er wel één of andere standaard (meestal opgesteld door degene die de taal bedacht heeft). Elke leverancier van een computertaal voor een bepaalde machine zal er voor zorgen dat "zijn" versie van de taal minimaal aan deze standaard voldoet. Wel zal hij vaak extra zaken aan de taal toevoegen waardoor de taal een eigen dialect krijgt (het Turbo Pascal 6.0 dialect) maar als je die uitbreidingen nu maar niet gebruikt, dan zal je programma bijna altijd lopen op een andere machine die dezelfde taal heeft. Voor enkele talen zijn er standaards door officiële instanties vastgesteld. Deze standaards worden over het algemeen zeer nauwkeurig door de taal-fabrikanten opgevolgd en vaak zit daar ook een vorm van certificering (kwaliteits-merk) aan vast. Voorbeelden hiervan zijn: ADA, Cobol, (ANSI) C en Fortran. Dit betekent dat bijvoorbeeld een programma in ANSI C zal kunnen draaien op alle computers die een dergelijke taal hebben, waaronder in de toekomst ook KGN68k.

Elke leverancier van een computertaal voor een bepaalde machine zal er voor zorgen dat "zijn" versie van de taal minimaal aan de standaard voldoet.

Vierde generatie: SQL en aanverwante talen

De vijfde generatie is nog niet bereikt. Wel zijn er al heel wat zaken die men tegenwoordig "Vierde generatie" noemt. Hierbij hoort ook een groep computertalen. Eén van de gemeenschappelijke eigenschappen die ik ken van vierde generatie talen is het feit dat ze allemaal werken in combinatie met een database-pakket. Ze zijn dus echt bedoeld voor het opslaan, opvragen en manipuleren van gegevens. Dit noemt men over het algemeen de administratieve automatisering in tegenstelling tot de technische automatisering waarbij de computer een technisch proces bestuurt. (Een fabriek o.i.d.). Het pakket DBASE voor de PC kun je ook opvatten als een vierde generatie taal en zo zijn er nog tientallen andere pakketten. Voorbeelden: Oracle, Powerhouse, Progress, Ingress, SyBase etc.

Er bestaat ook een gestandaardiseerde vierde generatie taal: Structured Query Language of SQL. In figuur 2 staat van deze taal een klein voorbeeld.

Dit stukje programma (statement) zoekt uit twee bestanden (GROEPEN en SALARISSEN) een aantal gegevens op waarna er een lijst wordt afgedrukt waar per salarisgroep op staat wie er in die groep vallen en hoeveel de persoon verdient.

Met SQL kun je niet alleen dingen opvragen, je kunt uiteraard ook gegevens toevoegen, wissen en wijzigen. SQL werkt vrijwel altijd in combinatie met een zogenaamde Relationele Database. Dit is een programma dat gegevens op een zodanige manier vastlegt dat je ze weer op eenvoudige manier terug kunt vinden. Om niet altijd de "kale" SQL-statements in te hoeven brengen kennen de meeste pakketten ook nog zogenaamde "Workbenches" waarmee het werken met een vierde generatie omgeving nog gemakkelijker wordt.

Een ander voorbeeld van vierde-generatie omgevingen zijn de zogenaamde programma-generatoren. Dit zijn pakketten waarmee je op een eenvoudige manier, bijvoorbeeld grafisch, een programma inbrengt. De taal die hiervoor gebruikt wordt is veel meer probleemgericht dan een programmeertaal uit de derde generatie. Vervolgens wordt door een vertaalprogramma vanuit het ingevoerde programma een programma in bijvoorbeeld Cobol gegenereerd. Hier komt dan ook het echte verschil tussen de derde en vierde generatie naar voren. De derde generatie is nog steeds machine gericht. Je geeft als het ware opdracht bepaalde delen van de machine te gebruiken en geeft ook aan hoe een computer het probleem technisch op moet lossen. Vierde generatie

```
SELECT    NR, NAAM, BEDRAG
FROM      GROEPEN, SALARISSEN
WHERE     BEDRAG BETWEEN LAAG AND HOOG
ORDER BY  NR, NAAM
```

Fig. 2: voorbeeld van een SQL-programma

talen zijn probleemgericht. Je beschrijft wat je wilt weten en laat de computer het vervolgens zelf maar uitzoeken.

Interpreters <-> Compilers

Als we teruggaan naar de hogere programmeertalen, dan kunnen die nog in twee klassen onderverdeeld worden. Ik had er vroeger grote moeite mee dit onderscheid te begrijpen en ik zal daarom extra mijn best doen het verschil tussen compilers en interpreters duidelijk te maken.

Om een programma in een programmeertaal op een computer te laten draaien, zal er iets met het programma of de computer moeten gebeuren. Computers begrijpen uit zichzelf namelijk alleen bepaalde bitpatronen en met een regel Pascal zullen ze over het algemeen maar heel weinig kunnen doen. Daarom zijn er programma's nodig die op de computer draaien om een bijvoorbeeld een programma in Basic uit te laten voeren.

Die programma's heb je in twee soorten: een interpreter (in goed nederlands een tolk) en een compiler (een vertaler). Bij de taal Basic waren van oorsprong alleen interpreters en een taal als Pascal kent bij mijn weten alleen compilers. Tegenwoordig heb je ook voor Basic compilers omdat een programma dat vertaald is sneller loopt als een programma dat geïnterpreteerd wordt.

Computers begrijpen uit zichzelf namelijk alleen bepaalde bitpatronen .

De beste manier om het verschil uit te leggen is waarschijnlijk toch het verschil tussen een tolk en een vertaler in het dagelijks leven te beschrijven. Bij een interpreter leest het interpreter-programma een programmaregel in waarna deze regel meteen door de processor wordt uitgevoerd. Als er dus in het programma PRINT "HELLO WORLD" staat, dan wordt meteen na het inlezen van de regel de tekst afgedrukt, voordat de volgende regel wordt uitgevoerd. Het interpreter-programma leest een programma dus stukje bij beetje in en laat het stukje dat ingelezen is meteen door de processor uitvoeren. Moet een programma twee of meer keren gedraaid worden, dan wordt het oorspronkelijke programma iedere keer opnieuw ingelezen. Het is dus als het ware zo dat een russische tolk een in het nederlands gedrukt boek van W.F. Hermans voorleest aan een Rus. Iedere keer opnieuw moet de tolk het boek vertalen.

Uiteraard is het veel gemakkelijker een boek één keer te vertalen en daarna de russische versie in de boekenkast te leggen. Als de Rus dan voor de twaalfde keer dit stukje nederlandse literatuur wil lezen, dan kan hij dat zelf doen. Dat is precies het werk van een compiler. Een compiler leest een programma in bijvoorbeeld C compleet in en maakt daar een programma in machinetaal (de moedertaal van de computer) van. Dit vertaalde programma wordt bewaard en als je het wil draaien, dan kan dit zonder verdere tussenkomst van een ander programma gedraaid worden. Omdat een compiler verder kijkt dan één regel, kunnen er ook wat optimalisaties uitgevoerd worden waardoor het programma sneller wordt. Bovendien hoef je een programma slechts één keer te vertalen.

In de eerste alinea van deze paragraaf schreef ik dat er iets met de computer of het programma moet gebeuren. Bij een interpreter verander je door middel van het interpreter programma de computer zodat hij bijvoorbeeld Basic begrijpt, bij een compiler verander je het programma zodat het een programma in machinetaal wordt. Het is echter ook denkbaar dat je zowel een compiler als een interpreter gebruikt. Een voorbeeld hiervan is UCSD Pascal. Hierbij is een compiler die Pascal omzet naar een tussentaal, P-code, en voor deze P-code liep er op de computer een interpreter. Tegenwoordig wordt echter bij mijn weten door compilers altijd naar machinetaal gecompileerd.

In zijn algemeenheid kun je stellen dat bij ontwikkeling het werken met een interpreter wat gemakkelijker gaat omdat je sneller fouten kunt corrigeren en dat als een programma "af" is, je hem beter kunt compileren. Bij diverse Basic-versies kom je daarom ook zowel een interpreter als een bijbehorende compiler tegen. Een interpreter heeft bovendien als voordeel dat je een opdracht ook even uit kunt proberen. Je kunt een programma vrijwel regel voor regel intikken en meteen door de computer uit laten voeren.

Ook bij 4 GL heb je interpreters en compilers. Eén van de mooiste voorbeelden op dat gebied is DBASE in combinatie met Clipper. DBASE is een interpreter en Clipper bevat een compiler die dezelfde DBASE programma's compileert. Vervolgens lopen ze aanzienlijk sneller en kunnen bovendien door iemand die alleen het vertaalde programma heeft, niet meer gewijzigd worden.

Linkers en Debuggers

Als we voor een computersysteem een compiler hebben, dan zit hier bijna altijd een zogenaamde "Linker" bij. Dit programma leest de vertaalde programmadelen in en voegt ze samen. Verder zorgt hij er voor dat er nog een aantal programmastukken aan worden toegevoegd die er voor zorgen dat het operating systeem met het programma om kan gaan (opstarten, normaal beëindigen en afbreken op een foutsituatie). Bovendien kunnen er aan een programma de zogenaamde bibliotheek-routines worden toegevoegd. Dit zijn routines die door vrijwel alle programma's gebruikt worden zoals bijvoorbeeld het inlezen vanaf een toetsenbord of het geven van uitvoer op je scherm.

Een ander programma dat je vaak tegenkomt is een zogenaamde Debugger (onluisder). Dit is een stuk gereedschap waarmee je tijdens het draaien van een programma "mee kunt kijken". Je kunt het programma op aangegeven plaatsen onderbreken, het programma stap voor stap uitvoeren, gegevens opvragen die het programma gebruikt etc. Kortom je kunt precies zien wat het programma allemaal doet om zo op een comfortabele manier de laatste programmeerfouten (bugs of luizen) uit je programma te halen.

Een ander programma dat je vaak tegenkomt is een zogenaamde Debugger (onluisder).

Voorbeeld

Ik wil afsluiten met een klein voorbeeldje in de taal C. Het operating systeem OS/9-68k voor computers met een 68000 microprocessor kan heel mooi laten zien hoe je van een derde generatie taal naar machinetaal komt. De C compiler die bij dat systeem hoort levert namelijk niet rechtstreeks machinetaal af maar een programma in de 68000 assemblertaal. Deze uitvoer wordt vervolgens door de assembler omgezet naar machinetaal waarna de linker deze machinetaal aanvult met een aantal standaard-zaken waardoor het programma kan draaien.

Het programma dat gebruikt wordt staat afgedrukt in figuur 3. Dit programma lijkt heel veel op het programma dat in figuur 1 staat afgedrukt. Beide programma's hebben met elkaar gemeen dat ze "niets" doen. Het C programma drukt alleen aan het eind nog een mededeling af dat hij ook werkelijk gedraaid heeft. Om dit te kunnen doen hoor je volgens de officiële C standaard hiervoor de opdracht #include <stdio.h> ook in je programma op te nemen. De C compiler weet dan dat je functies voor input en output wilt gaan gebruiken. Voor de C compiler die ik gebruikt heb is dat in dit geval niet nodig. Dit heeft als voordeel dat de uitvoer iets korter is waardoor er minder "papiervervuiling" optreedt.

Nadat de source (dat is wat er in figuur 3 staat), door de C compiler heen is geweest, ontstaat er een programma in 68000 assembler taal. Hierin zijn, door middel van een extra opdracht aan de C compiler, de oorspronkelijke regels C source weer als commentaar afgedrukt. Deze source in assembler taal is vervolgens weer ingelezen door de assembler die hiervan de in figuur 4 afgedrukte listing geproduceerd heeft.

```
/* Demo program for article about languages    */
/*                                             */
/* G. van Opbroek  27-02-1992                  */
/*                                             */

#define Times 10

main()
{    int i,j;

     j = 0;
     for (i = Times ; i > 0 ; i--)
          j++;
     printf("End-of-program\n");
}
```

Fig. 3: demonstratieprogramma in C

```
Microware OS-9/68000 Resident Macro Assembler V1.6
00001                           psect   demo_c,0,0,0,0,0
00002                           nam     demo_c
00003  * /* Demo program for article about languages      */
00004  * /*                                                */
00005  * /* G. van Opbroek     27-02-1992                  */
00006  * /*                                                */
00007  *
00008  * #define Times 10
00009  *
00010  * main()
00011  *       { int i,j;
00012                           ttl     main
00013  0000   48e7    main:    movem.l #_1!1,-(sp)
00014  0004   203c             move.l  #_3,d0          :6
00015  000a=  6100             bsr     _stkcheck
00016
00017  *
00018  *       j = 0;
00019  000e   518f             subq.l  #8,sp           :2
00020  0010   4297             clr.l   (sp)            :2
00021  *       for (i = Times ; i > 0 ; i--)
00022  0012   700a             moveq.l #10,d0          :2
00023  0014   2f40             move.l  d0,4(sp)
00024  0018   6000             bra     _7
00025  *       j++;
00026                   _5
00027  001c   5297             addq.l  #1,(sp)         :2
00028  *       printf("End-of-program\n");
00029                   _8
00030  001e   53af             subq.l  #1,4(sp)
00031                   _7
00032  0022   4aaf             tst.l   4(sp)
00033  0026   6e00             bgt     _5
00034                   _6
00035  002a   41fa             lea     _9(pc),a0
00036  002e   2008             move.l  a0,d0           :2
00037  0030=  6100             bsr     printf
00038  *       }
00039  0034   508f             addq.l  #8,sp           :2
00040                   _4
00041  0036   588f             addq.l  #4,sp           :2
00042  0038   4cdf             movem.l (sp)+,#_1
00043  003c   4e75             rts                     :2
00044  ffffffb4        _3      equ     0xffffffb4      :0
00045  00000100        _1      equ     0x00000100      :0
00046  0000000c        _2      equ     0x0000000c      :0
00047  003e   456e     _9      dc.b    "End-of-program"
00048  004c   0d               dc.b    $d
00049  0000004d                dc.b    ""
00050  004d   00               dc.b    $0
00051
00052  0000004e                ends
```

Fig. 4: assemblerlisting van DEMO.C

In deze source staat op een aantal plaatsen de instructie "bsr", gevolgd door een naam. Dit zijn aanroepen van functies die niet in het stukje programma staan. De Linker zal er voor zorgen dat deze functies ook in het uiteindelijke programma worden opgenomen en dat de echte adressen worden ingevuld.

Wat je meteen ziet is dat het programma een stuk langer is dan het programma uit figuur 1. Verder kan ik je met zekerheid vertellen dat het programma ook minder snel is. Dat komt omdat er veel meer gecontroleerd wordt en omdat het programma niet optimaal gebruik maakt van de mogelijkheden van de processor. Nu is het wel zo dat de zogenaamde "Optimizer", een programma dat het naar assembler vertaalde programma nog eens verbetert, niet gelopen heeft maar dan nog wordt het programma nooit zo snel als een programma dat door een ervaren programmeur rechtstreeks in assembler geschreven is.

Afsluiting

Zo, dit was mijn inleiding op een cursus 'C' programmeren. Wie durft de handschoen op te pakken en schrijft het volgende deel? Reacties graag bij de auteur van dit artikel.

Literatuur:

Het SQL-voorbeeld is overgenomen uit:

"Het SQL Leerboek" van Rick F. van der Lans, uitgeven door Academic Service. Bij dit boek behoort "De SQL Standaard" van dezelfde auteur.

Gert van Opbroek

Inleiding voor verantwoord computergebruik

Veel woorden zijn bij computers in het Engels: "Disk", "Keyboard", "Monitor", "Hard-Disk", "Directory", "File". Nooit opgevallen? En weet je waarom dat zo is? Niet omdat we daarvoor geen Nederlandse woorden hebben, maar omdat de bedenkers vroeger al slecht in het spellen van Nederlandse woorden waren (onvoldoendes op school, nablijven en dergelijke laten diepe sporen achter op een onbesmet kinderzieltje!) En nu lopen ze minder snel tegen de lamp... (Er is in Nederland toch geen hond, die voldoende computer-Engels spreken, laat staan verstaan. Dus hun deskundologische gebrabbel valt dan minder op. Als het Nederlandse termen waren, zouden ze snel door de mand vallen... Maar ik zou het nu over programma's hebben.

Bij mijn vorige relaas had ik het al over Wordthetooitperfect, een tekstverwerkings-programma, dat in Nederland tot de populairste programma's geteld mag worden. Het kost dan ook niet meer dan die bewuste diskette-schijfjes, waarvan een veelvoud voor mij nog steeds onduidelijk is en een goed boek. De meeste mensen hebben dat programma op hun computer staan, waarmee ze dan hun PC (uitgesproken als PeeCee) bedoelen (De overeenkomsten met de afkorting APC, een populaire pijnstiller is naar mijn idee niet toevallig).

"Kun je WeePeëen?" is een interessant klinkende vraag.

Verder hebben ze naast de onvermijdelijke spelletjes, waarover straks meer, vaak ook een zogenaamde data-base. Dat is een lelijk woord voor bestands-beheer-programma. Soms hebben ze ook nog een Spreadsheet, dat geen overeenkomsten heeft met de broodnodige variatie van beleg, maar een ander vies woord is voor rekenprogramma. Nu, ik wil het nu hebben over deze drie programma's. Tekstverwerkers, bestandsbeheer-programma's en rekenprogramma's.

Tekstverwerkers

Tekstverwerkers zijn programma's waarmee je korte briefjes aan je partner kunt schrijven, om te vertellen, dat je wat later thuis bent voor het eten, omdat je een computer-cursus volgen moet. Of waarmee je de scriptie van je jongste zoon/dochter over mag typen, terwijl zij achter hun Nintendo zitten. Het is vreemd, dat een tekstverwerkings-programma zo populair is in Nederland, want het is eigenlijk alleen maar een zeer dure methode om eindelijk je naam eens op de televisie te krijgen. (Al weigert elke weldenkend TV-station dit van je over te nemen.) Elke computer in welke willekeurige winkel wordt ermee geconfronteerd. Mensen, die hun kinderen, als die mee mogen, of zelf, als de verkoopster/koper niet kijkt, hun naam intypen op zo'n tentoongestelde computer. Nu is het goedkoper om een oude TV te kopen op de rommelmarkt en je naam daarin te laten kalligraferen, maar zonder een tekstverwerker speel je niet meer mee voor je omgeving. "Kun je schrijven?" is in ons gealfabetiseerde Nederland bijna een overbodige vraag, "Kun je WeePeëen?" is een interessant klinkende vraag. Daarbij doelen de vraagstellers dan op een tekstverwerkings-programma. (Het leuke is trouwens, dat ze dan vaak ook vragen "Welke versie?" Waarbij een lager nummer een onderontwikkelde ziel moet zijn... (Ik werk trouwens met versie 2.0)

Een truc om dergelijke, zichzelf superieur voelende personen op hun nummer te zetten is deze vraag te beantwoorden met "Ach, ik schrijf sneller met een vulpen" of "Ik werk veel liever met een goed pakket als SM Woord 6.0a", waarbij de "a" zeer nadrukkelijk moet klinken, zodat de vraagsteller zich in een klein hokje terugtrekt en na begint te denken over haar/zijn slechte jeugd, de oorlog in Joegoslavië, de nimmer aflatende honger in Afrika en andere frustraties. Een vreemde zaak is, dat omdat men in Amerika (3000 kilometer aan de andere kant van de aarde) alleen maar één soort aanhalingsteken kent, moeten alle programma's die ook ontberen. Wat voor een investering iemand moet doen, om wel de Europese leestekens te krijgen grenst aan het ontoelaatbare. Kortom, het schrijven van kleine briefjes gaat sneller met de hand, het schrijven van enveloppen sneller met een ouderwetse Typemachine en brievenschrijvers worden door hun nu leesbare schrift aangekeken op het gebrek aan creativiteit met de mogelijkheden van de gebruikte tekstverwerker. Zo zijn de mensen nooit tevreden te krijgen...

Bestandsbeheersprogramma's

Bestands-beheers-programma's zijn een volgend excuus om een computer aan te schaffen, maar in de praktijk worden ze er zelden voor gebruikt. Hoe vaak ik niet gehoord heb, hoe makkelijk het wel is voor "de vereniging", waar ook zelfs een vereniging van suikerzakjes-spaarders in L. (10 leden) uit de toch al overbelaste pot een computer aanschaft. Dat een computeraar dergelijke programma's alleen maar gebruikt om gegevens van vrienden en kennissen makkelijker op een lijst te krijgen. Dat bijna niemand, die een dergelijk programma aanschaft het ook *blijft* gebruiken om hun platen/CeeDee- en (vi-

deo-)cassetteverzameling op bij te houden, geeft te denken. Tientallen mogelijkheden worden geprogrammeerd, maar (bijna) niemand gebruikt die programma-onderdelen, omdat deze de computer, de printer of de gebruiker niet ondersteund. Ik werd bijvoorbeeld een gebeld door een gebruiker, die klaagde, dat hij het programma niet meer kon verlaten door een bepaalde menu-keuze te kiezen. De menu-keuze werkte nu echt en hij kreeg een lijst van zijn video-banden (25 stuks) in hondervoud uit zijn printer... Nee, het gebruiken van een computer is een hobby en het bewaren van video-banden een andere... Een andere zieligerd had al de adressen van zijn kennissen en vrienden in de computer "gestopt" en wist niet, dat je die gegevens altijd ook op een aparte diskette moest bewaren. Hij was na het opzoeken van een kennis in Madagascar al de gegevens kwijt, die hij in drie maanden had ingevoerd. Gelukkig had hij zijn kaartenbakje nog.

Het onthouden van gegevens met behulp van een computer gaat perfect, zolang je maar elke dag de computer raadpleegt, wat je nu weer gaat beleven. Vergeet je een dag, kom je opeens tot de ontdekking, dat je vrouw je huwelijksdag *niet* vergeten is, de auto van de straat gehaald is, omdat de APK-keuring niet meer geldig was en meer van dat soort rampen.

Rekenprogramma's

De laatste groep programma's waarover ik het hebben wil, zijn de reken-programma's, de zogenaamde Spreadsheets. Spreadsheets zijn gecomputeriseerde rekenbladen. Nu is een computer al bij andere toepassingen al een ramp, bij reken-bladen is het helemaal erg. En ze zijn zo ouderwets. Dat het zo slecht met Wallstreet, de US dollar, endergelijke gaat is niet verwonderlijk, als je weet, dat ze alles op zo'n programma berekenen... Groetjes van

MacBart!

Nieuwtjes en andere wetenswaardigheden.

Een autorijdende vlieg

Op 18 januari jongstleden is Jos Nijhuis gepromoveerd op een geslaagde proef met een neuraal netwerk. Hij reed op de snelweg tussen Stuttgart en Neurenberg met een testauto zo'n tien minuten met zijn handen van het stuur met een snelheid van 130 kilometer per uur. De auto stuurde zelf zoals hij de omstandigheden op de weg had leren interpreteren. Naar volkomen natuurlijke menselijke gedragingen. Dit hoeft niet te betekenen dat autorijden nu niet leuk meer is. De vinding die het voorspelbare gedrag van een automobilist imiteert, moet vooral de veiligheid dienen. Wij worden nog weleens afgeleid door een mooie vrouw langs de kant van de weg. Een neuraal netwerk blijft echter onder alle omstandigheden alert reageren op basis van het aangeleerde gedrag. Wat nu als dat neurale netwerk die vrouw ook mooi vindt? Die mogelijkheid blijft volgens mij bestaan, temeer omdat Jos weet dat het werkt maar nog niet uit kan leggen hoe. Maar het werkt wel, dus we hebben niets te vrezen. Zo'n chip kan de menselijke hersenen bij lange na niet evenaren. De geheugencapaciteit van die chip kun je vergelijken met de omvang van de hersenen van een vlieg. Die hoeveelheid is genoeg om een auto te besturen. Als je binnenkort een vlieg een auto ziet besturen, dan weet je dat Jos weer aan het testen is.

Tonny Schäffer

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.

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.

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

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.

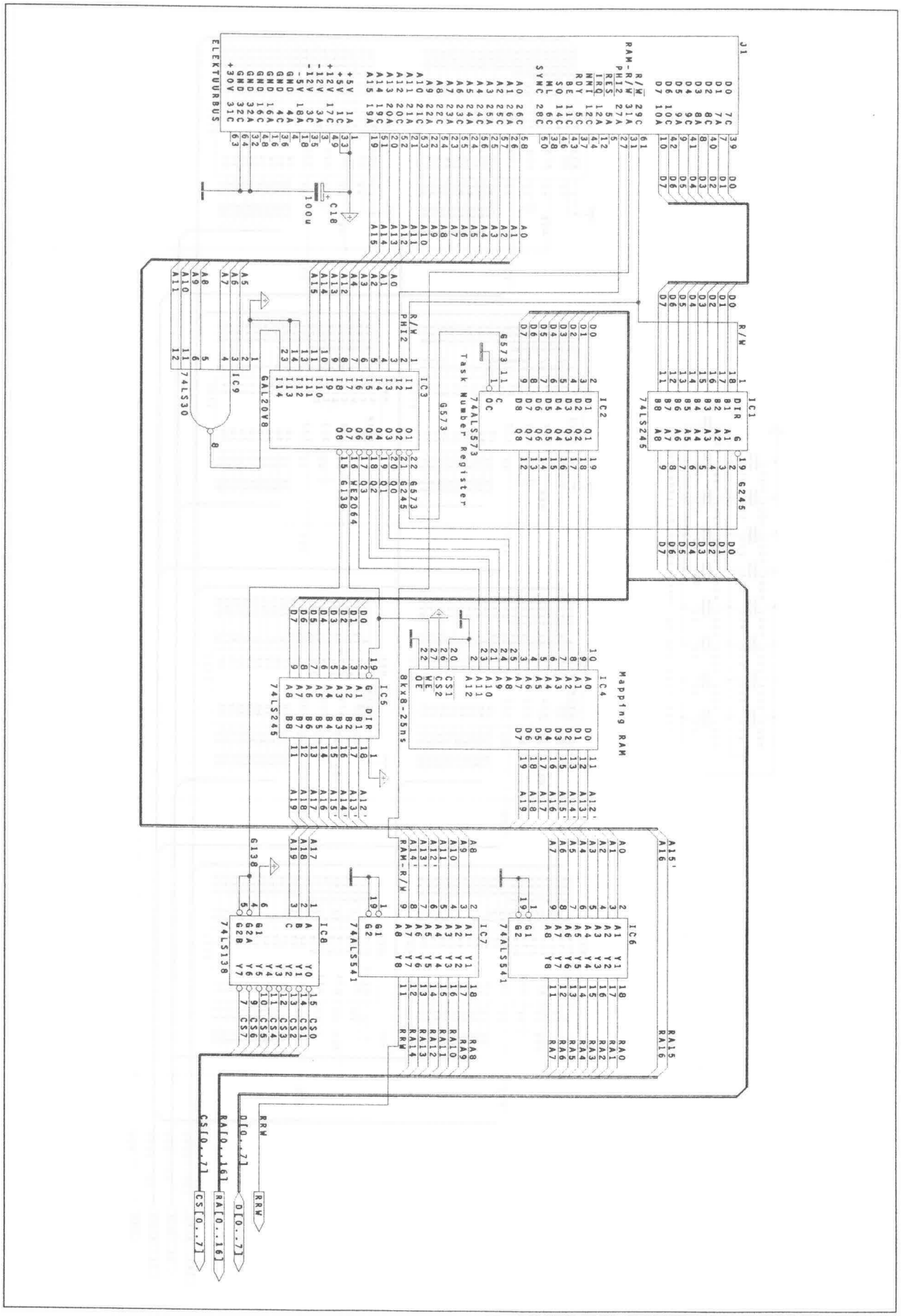
De meeste buslijnen worden belast met 1 TTL-load. PHI2 en A0 tot en met A11 zien twee TTL-loads.

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.

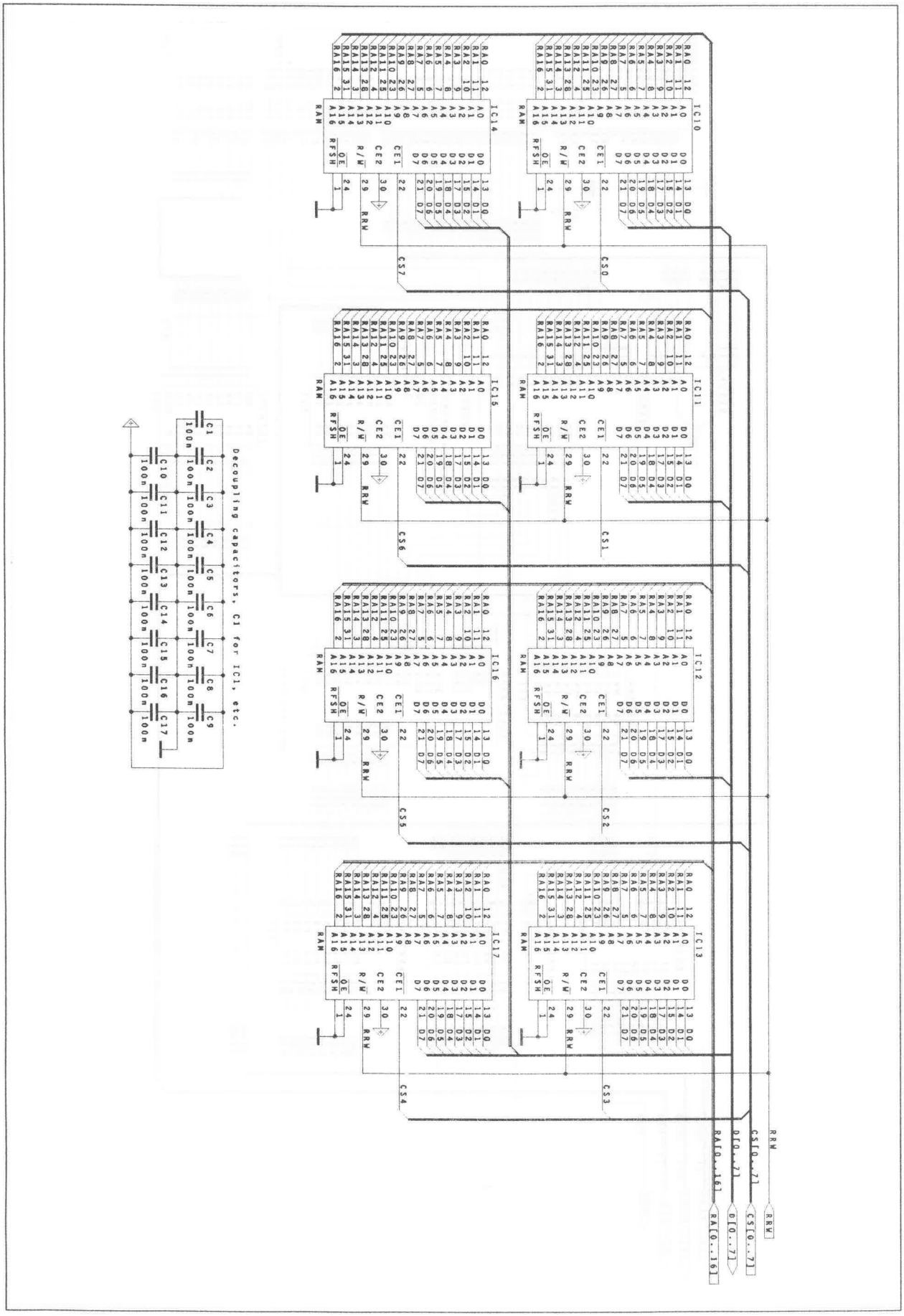
Nu komt het hevige stuk: de print! Gezien de ervaringen met EP doe ik hier nog geen uitspraken over...

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

```
GAL20V8                                                GAL DESIGN SPECIFICATION
VDISK3 4th VERSION                                     N. DE VRIES  15-02-1993
TC5588 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  G138  WE5588  Q3  Q2  Q1 Q0  G245  G573  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 $FF00-$FF0F
       +  /RW* A15* A14* A13* A12*/LS30* A4*/A3*/A2*/A1*/A0        ;write to $FF10
       +           /A13                                   ;access
       +        /A14                                      ;to
       +     /A15                                         ;RAM
       +      A15* A14*/A13                               ;array

/WE5588 =  PHI2*/RW* A15* A14* A13* A12*/LS30*/A4          ;write to $FF00-$FF0F

G573   =  PHI2*/RW* A15* A14* A13* A12*/LS30* A4*/A3*/A2*/A1*/A0        ;write to $FF10

/Q0    =  /A0* A15* A14* A13* A12*/LS30*/A4    ;write to $FF0X: A0 is TC5588 addressline
       +             /A12                     ;else let A12 through

/Q1    =  /A1* A15* A14* A13* A12*/LS30*/A4    ;write to $FF0X: A1 is TC5588 addressline
       +          /A13                        ;else let A13 through

/Q2    =  /A2* A15* A14* A13* A12*/LS30*/A4    ;write to $FF0X: A2 is TC5588 addressline
       +       /A14                           ;else let A14 through

/Q3    =  /A3* A15* A14* A13* A12*/LS30*/A4    ;write to $FF0X: A3 is TC5588 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

WE5588 is the write enable input for the TC5588 and the enable for the
       write data buffer to the TC5588

Q0..Q3 are the lower addresslines for the TC5588
```

Afb. 3: vergelijkingen voor de PAL op de geheugenkaart

Item	Aantal	Referentie	Omschrijving
1	17	C1,C2,C3,C4,C5,C6,C7,C8, C9,C10,C11,C12,C13,C14, C15,C16,C17	100nF
2	1	C18	100uF, 10V
3	2	IC1,IC5	74LS245
4	1	IC2	74ALS573
5	1	IC3	GAL20V8 of PAL22V10
6	1	IC4	SRAM, 8kx8-25ns (0.3 inch behuizing)
7	2	IC6,IC7	74ALS541
8	1	IC8	74LS138
9	1	IC9	74LS30
10	8	IC10,IC11,IC12,IC13,IC14, IC15,IC16,IC17	1 Mbit (128kx8) SRAM of Pseudo SRAM
11	1	J1	Connector DIN 41612, a/c, male haaks
12	1	-	Print

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.

Afb. 4: onderdelenlijst

Ik heb interesse in de KGN en wil

☐ Lid worden van de KGN ☐ Meer informatie over de KGN

Naam : _____

Adres : _____

Postcode en Woonplaats : _____

Datum : _____ Handtekening : _____

Voortgang KGN68k

Of er nu twee delen van voortgang KGN68k opgenomen zijn in deze mu-pee kenner weet ik niet. Wat ik wel weet is dat er de vorige keer geen deel in stond terwijl er nog steeds voortgang is in het KGN68k project. Dat het project op vierkante wielen rolt werd me in de werkgroep tegengesproken. Wel vond men dat de wielen ovaal zijn.

MC68040

Zaterdag 13 februari was de werkgroep bij elkaar. Ik was even weg geweest om een werkgroeplid van het station te halen. Bij terugkomst werd me verteld dat men het concept wilde wijzigen in een ontwerp met MC68040. Het bestaande concept moet dan wel helemaal over de kop, omdat een 040 synchroon de bus op gaat. Nadat ik de heren verteld had dat het een technisch interessant idee is en dat ook voor de nul veertig een OS nodig is en dat we dat eerst naar de bestaande nul dertig porteren, werd me verteld dat het een grapje was.

Wel vond men dat de wielen ovaal zijn.

Documentatie

De documentatie, die lange tijd geen hoge prioriteit heeft gehad, begint nu ook duidelijkere vormen aan te nemen. Het geraamte is er en kan nu verder aangekleed worden. Het materiaal hiervoor; schema's, PLD equations & listings, hebben we al. Het werk zit dus in het samenvoegen tot een net geheel.

Print

De printplaat waar nu de meeste aandacht aan besteed wordt is Dombo68k, de diskcontrollerkaart. We zouden er eigenlijk al aan het meten moeten zijn, maar helaas door een opzegging binnen de werkgroep van een goede weg naar een print fabrikant is er wat vertraging ontstaan. Het floppy-diskgedeelte is al beproefd in de AB8, maar het SCSI gedeelte is wel nieuw. Op korte termijn is ook de DRAM kaart te verwachten. Daar zal meer aan moeten gebeuren, het hele DRAM refresh gebeuren is namelijk nog niet getest.

Software

Na enkele verkenningstochten in de source van LINUX blijkt dat vrijwel alles in C geschreven is. Er zijn schijnbaar maar weinig specifiek 80386 stukken. Het opstarten gebeurt zo te zien in 8088 code. En dat zijn dan ook precies de stukken waar door ons het werk moet gebeuren.

Op andere plaatsen binnen de werkgroep wordt gewerkt aan de Power On Self Test, de monitor en het opstarten. Bij een zelftest is het altijd moeilijk om een uitgangspunt te vinden. Bij een IBM-compatible machine gaan ze er van uit dat de luidspreker altijd werkt en beginnen dan vrolijk met het testen van de processor. De ROM testen waar het zelftestprogramma in staat lijkt nutteloos. Tijdens het boot proces wordt er maar een klein stukje van een veel grotere ROM gebruikt. Bij een ROMtest kan men echter het hele ROM gebied controleren. De monitor heeft nu de essentiële commando's load, go, dump memory & change memory. Waar nu aan gewerkt wordt is het kunnen disassembleren van stukken geheugen. Zodat we straks kunnen zien wat de compiler er van gemaakt heeft, maar het is ook al handig bij het debuggen van de stukken software waar nu aan gewerkt wordt. De trace exception software moet ook kunnen disassembleren. Maar daar wordt nog aan gewerkt en zo hebben we last van een kip-ei probleem.

Bootfile

Nu de diskcontroller zo goed als af is, wordt er ook aan een stuk software gewerkt dat van een diskette, met een MS-DOS indeling, een bepaalde file kan binnenlezen. Dit als vervanging van of aanvulling op het serieel downloaden. Alleen de indeling (format) moet gelijk aan dat van een MS-DOS machine zijn. Op je huidige compile engine, wat voor een hardware platform het is maakt niet, kun je dan de file aanmaken en schrijven op floppy. Die wordt dan ingelezen op de KGN68k target-machine.

Geven/Nemen

Als afsluiting heb ik nogal een keer om iets gevraagd. Nu bied ik iets aan. Heeft U een 680xx machine met een seriële poort en een hardware floppy diskcontroller dan kunnen we die samen naar de huidige software status brengen van het KGN68k. Om U toe te zeggen dat naar uw machine ook het UNIX-like operating system geporteerd gaat me op dit moment te ver. Laat U echter niet afschrikken door het woord "samen", het grootste deel van het werk is al door de werkgroep gebeurd. U kunt zich gewoon bij mij melden. Uw projectleider KGN68k:

Geert Stappers (04781-41279)

Poorten in een PC

Geen angst, met poorten in een PC bedoel ik niet een nieuw soort gezelschapsspel noch een "vertaalde" Engelse uitdrukking maar gewoon de seriële en parallelle aansluitingen die achterop die kast met die floppies zitten. Het heeft wat tijd gekost voor ik de logica (?) inzag van de manier waarop die dingen moeten worden ingesteld maar hierbij dan een korte impressie van mijn bevindingen.

COM poorten

De PC kent er maximaal 4 (onder DOS, Unix is een ander verhaal). Deze COM poorten zijn genummerd 1 t/m 4. Dat zal niemand verbazen. Elke COM poort heeft een adres waarop de hardware te vinden is en een interruptadres. De hardware adressen zijn voor iedere poort uniek maar de IRQ oftewel interruptadressen niet. In de praktijk is dit nogal eens een probleem. Communicatie werkt bijvoorbeeld niet goed als een andere kaart ook gebruik maakt van dezelfde IRQ (zoeken maar!).

De volgende adressen worden gebruikt:

poort	adres	IRQ
COM1	3F8	4
COM2	2F8	3
COM3	3E8	4
COM4	2E8	3

Als je geen interruptadressen dubbel wilt gebruiken kun je maar twee poorten installeren. Een andere mogelijkheid is om een COM-kaart zo om te bouwen dat er een andere interruptvector wordt gebruikt (bijv. 10 of 12). Het probleem is dat op het XT gedeelte van de bus alleen de lage 8 IRQ's bereikbaar zijn (en daarvan wordt er al één gebruikt om de hoge 8 af te handelen). IRQ8 t/m 15 bevinden zich op de kleine AT connector en de meeste COM-kaarten gebruiken alleen maar de XT connector. Als je al een hogere interrupt kunt instellen dan moet het programma er ook gebruik van kunnen maken. Kortom, niet iets om 's avonds om tien uur nog even te proberen.

Ik heb eens mijn muis onder Windows 3.0 op COM-poort 3 gehad. Ik moest daarvoor alleen wel COM1 kiezen in de setup en dan in de mouse driver van windows even zoeken naar '3F8' en dit veranderen in '3E8' maar het werkte prima. Alleen jammer dat ik COM1 niet meer kon gebruiken...

Parallelle poorten

De meeste PC's hebben er één of twee. Er kunnen er toch meer worden aangesloten, namelijk maximaal 4.

Bij de parallelle poorten speelt er een andere eigenaardigheid. De adressen worden in een bepaalde volgorde afgetast tijdens de start van de machine. Het eerste gevonden adres wordt LPT1 het tweede LPT2 enz. Bij de adressen horen dan wel bepaalde interruptvectoren. Het volgende heb ik gevonden:

scanvolgorde	adres	IRQ
1e	3BC	7
2e	378	7
3e	278	5
4e	2BC	5

Communicatie werkt niet goed als een andere kaart gebruik maakt van dezelfde IRQ (zoeken maar!).

Als je nu nog weet dat er "BIOSsen" schijnen te zijn die maar twee poorten herkennen dan voel je alweer nattigheid. De meeste kaarten geven de twee parallelle poorten de adressen 378 en 278 mee, daarmee zijn eventuele IRQ problemen mooi omzeild. Ik heb trouwens nog geen problemen gemerkt met twee printers, de ene op 3BC en de andere op 378 en beide op IRQ7. Het kan blijkbaar wel goed gaan.

Unix

Eerder in dit artikel schreef ik dat het met Unix op een 486 heel anders gesteld is. Om een voorbeeld te noemen: Er zijn Unix machines, meestal uitgerust met een of meer 486's, EISA bus, 32Mbyte of meer intern geheugen en schijven van meer dan een gigabyte waar meer COM-poorten nodig zijn dan bij een gewone PC. Alle terminals en printers hebben een eigen poort nodig en er zijn gewoon niet genoeg interrupts in de normale PC structuur aanwezig (15) om alle terminals een eigen IRQ te geven. Er wordt dan gebruik gemaakt van speciale seriële kaarten met bijv 8 COM-poorten met een eigen processor en één enkele IRQ. Als er dan bijv. vier van deze kaarten worden gebruikt voor 32 terminals, dan worden er toch maar 4 interruptvectoren gebruikt (bijv 10 t/m 13).

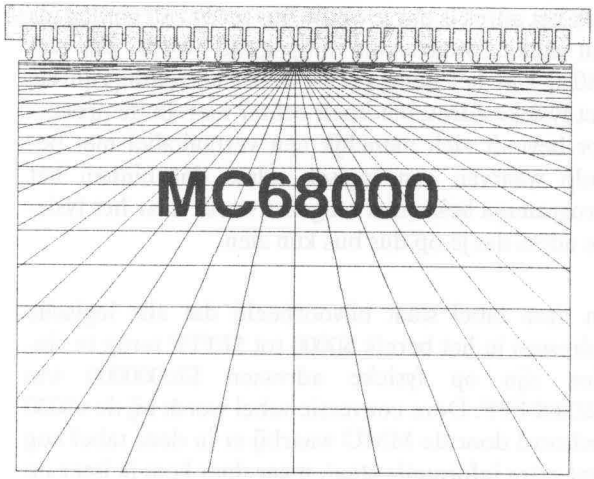
Mocht er extra informatie beschikbaar zijn dan zou ik daar graag eens wat over lezen in de μP Kenner.

Ernst Elderenbosch

De Motorola 68030; het hart van KGN68k (deel 4)

Inleiding

Op het moment dat u dit leest, zijn de meeste kaarten van de KGN68k computer in prototype gereed en wordt er hard gewerkt aan het ontluizen van de hardware en het vergaren van de benodigde systeemsoftware. Het wordt dus de hoogste tijd de speciale eigenschappen van de 68030 te gaan behandelen.



In deze aflevering van de serie gaan we beginnen met de opbouw van het geheugen van de 68030. Dat betekent dat we het onderscheid gaan behandelen tussen logische en fysieke adressen. Verder worden de caches in 68030 behandeld. In de volgende aflevering wordt de behandeling van de processor voorlopig afgerond met de beschrijving van de eigenschappen en het gebruik van de Memory Management Unit of te wel de MMU.

Fysieke en logische adressen

Toen (hobby-)computers nog klein en eenvoudig waren (KIM, Junior, C64), had een computer een vaste geheugen-indeling. Het programma werd altijd op een vaste plaats (= adres) in het geheugen geladen en je wist precies waar je aan toe was. Verder had bij een deel van de micro-computer wereld de periferie vaste adressen in het geheugenbereik van de computer zodat ook precies bekend was waar je het getal $30 weg moest schrijven om een '0' op papier te krijgen.

Het geheugen werd geadresseerd door middel van een adresbus van 16 bits zodat je een adresbereik had van 64 kilobyte en wonderwel bleek dit nog ruim voldoende voor een scala aan toepassingsprogramma's. De beperking van de adresbus tot een breedte van 16 bits werd enerzijds ingegeven door de prijs van geheugen (1 kB koste in 1982 ongeveer net zoveel als nu 1 MB) en anderzijds door het aantal pennen dat aan de processor mochten zitten. Zaten er namelijk meer dan 40 pennen aan, dan was het geen "normaal" IC meer en paste het niet in allerlei test- en ontwikkelings-apparatuur. De overstap naar IC's met meer dan 40 pennen vereiste een enorme investering zodat men die stap pas relatief laat genomen heeft.

Tegenwoordig heeft een PC (voor mij niet meer dan een veredelde typemachine) al minstens 640 kilobyte aan boord en zelfs dat blijkt voor heel veel toepassingen (lang) niet voldoende. Bovendien moet je in de PC nogal wat trucs uithalen om meer dan 640 kB te kunnen gebruiken.

Tegenwoordig heeft een PC (voor mij niet meer dan een veredelde typemachine) al minstens 640 kilobyte aan boord.

Bij de Motorola 68030 heeft men niet bezuinigd op een paar adresbitjes, nee, men heeft de processor uitgerust met een volledige 32 bits adresbus. Dit houdt in dat je ruim 4 miljard byte (4 Gigabyte of GB) kunt adresseren, hetgeen bij de huidige prijzen van RAM-chips nog steeds ruim voldoende is: 4 MB kost nu pakweg fl. 400,- dus 4 GB kost nog altijd de prijs van een fraaie sportwagen uit Italië (en die heb ik liever dan RAM-chips.....). Verder kun je met behulp van de functie-code bits dit geheugenbereik nog ver-achtvoudigen zodat je aan adresruimte zeker geen gebrek zult hebben. De periferie wordt, evenals bij de 6502 en aanverwante chips, 'ergens' in het enorme adresbereik ondergebracht.

Een computersysteem met een 68030 bevat een handvol megabytes aan RAM, een hoeveelheid ROM en periferie. Al die zaken hebben hardwaretechnisch een vast adres, op adres $00000000 begint eerst een hoeveelheid ROM, dan op bijvoorbeeld adres $10000000 het RAM en op adres $F0000000 de registers van de periferie-bouwstenen. Dit zijn de zogenaamde fysieke adressen. Het fysieke adres is het adres dat op de bus te meten is als de geheugenlokatie benaderd wordt.

De assembler-programmeurs onder ons weten dat in de assembler-listing een adres afgedrukt wordt. Bij een 6502 zijn dit meteen ook de fysieke adressen. Wat dus betekent dat het adres dat in de listing staat ook het adres is dat je op de bus terug zult vinden als dat stukje programma uitgevoerd wordt. Bij een 68030 kan het aangegeven adres hetzelfde zijn als het fysieke adres, het hoeft echter niet. In de processor bevindt zich namelijk een vertaaltabel met behulp waarvan een logisch adres dat binnen het programma bekend is omgezet wordt naar het fysieke adres dat je op dus bus kun zien.

In deze tabel staat bijvoorbeeld dat alle logische adressen in het bereik $0000 tot $FFFF terug te vinden zijn op fysieke adressen $10000000 t/m $1000FFFF. Deze conversie-tabel wordt bij de 68030 beheerd door de MMU waarbij er in deze tabel nog wat extra informatie staat, maar daar kom ik later op terug.

Functiecodes

In de eerste aflevering zijn de drie functie-code pennen FC0 t/m FC2 al even aan de orde geweest. Deze signalen zijn eigenlijk een uitbreiding van de adresbus met nog drie bits waardoor je als het ware 8 volledig onafhankelijke (orthogonale voor de theoretici) adresruimten krijgt. Elk van deze ruimten wordt gebruikt voor een bepaald type geheugen benadering:

FC2	FC1	FC0	Omschrijving:
0	0	0	Niet gebruikt bij 68030
0	0	1	User Data Space
0	1	0	User Program Space
0	1	1	Niet gebruikt bij 68030
1	0	0	Niet gebruikt bij 68030
1	0	1	Supervisor Data Space
1	1	0	Supervisor Program Space
1	1	1	CPU Space

Hiervan wordt de CPU-space gebruikt voor de interrupt acknoledge en de benadering van de coprocessor(s). In dit "geheugengebied" kan dus geen echt geheugen zitten; de adressen zijn alleen voor de CPU bedoeld.

Van de overige ruimten zijn er 3 niet benut in de 68030. Ze zijn echter wel aanwezig zodat toekomstige 68xxx processoren upwards compatibel kunnen blijven met de 68030. Ze zijn dus gereserveerd voor toekomstige uitbreidingen. De resterende vier spreken voor zichzelf; er zijn aparte adresruimten voor het benadering van programma en data in zowel de supervisor mode als in de user mode. Binnen de KGN68k zijn deze ruimten echter samengevoegd tot

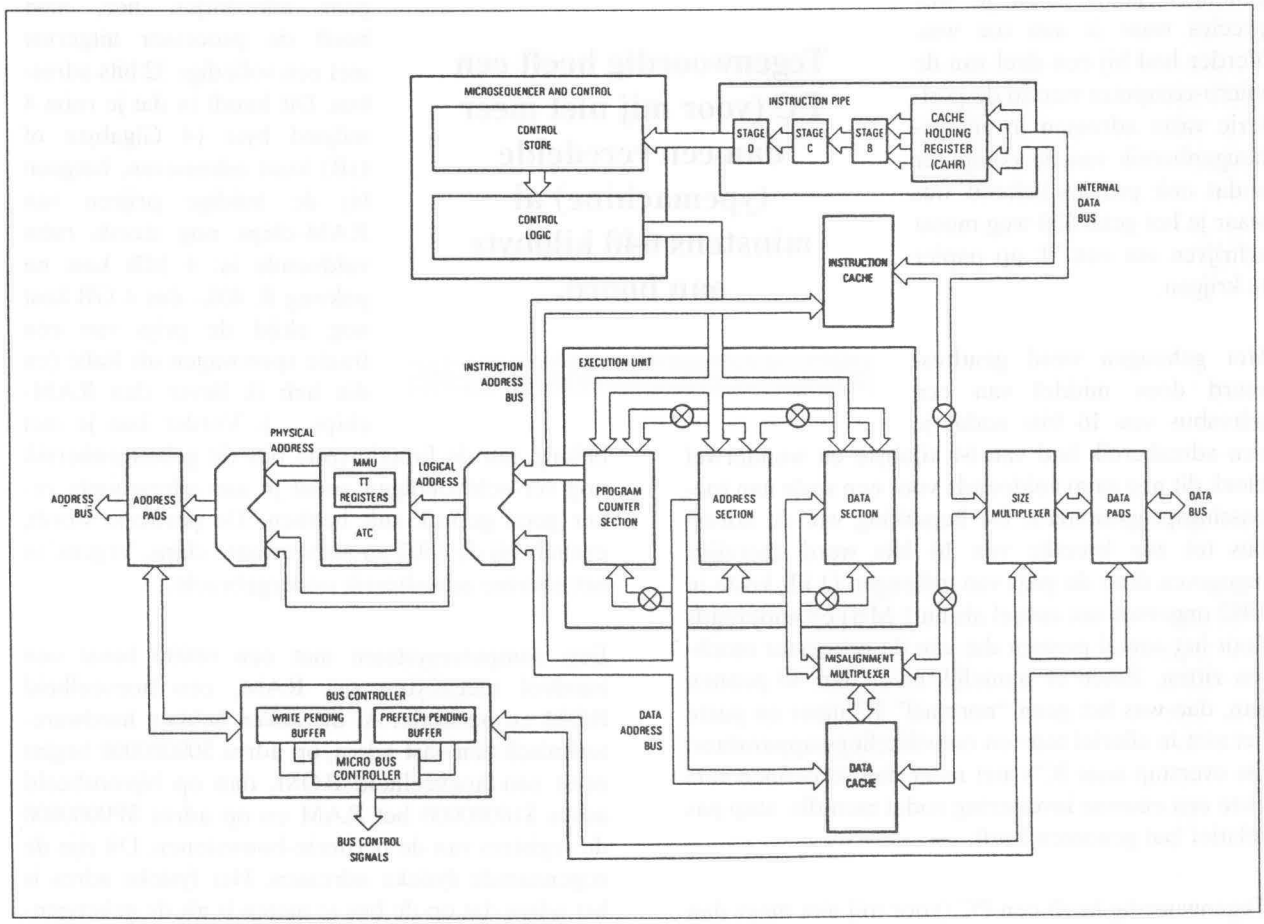


Fig. 1: interne caches van de 68030

één stuk fysiek geheugen met een maximale afmeting van 4 GB.

Uiteraard hebben de functie-codes ook invloed op de omrekening van logische naar fysieke adressen. Een logisch adres $1000 met functiecodes 001 is een heel ander adres als een logisch adres $1000 met functiecodes 110. Om deze reden worden de functie-codes in de Memory Management Unit en in de caches meegenomen als waren het extra adresbits.

Caches (algemeen)

In figuur 1 is de interne opbouw van de 68030 (nogmaals) weergegeven waarbij de nadruk met name ligt op de aanwezige caches. Hoe caches in het algemeen werken staat beschreven in deze paragraaf.

Zoals reeds in deel 1 van deze serie is opgemerkt, bezit de 68030 een instructie cache en een data cache, beide met een afmeting van 256 byte. Wat een cache is en hoe het werkt, wordt in deze paragraaf behandeld.

Caches zijn op te vatten als een stuk zeer snel geheugen. Over het algemeen is het namelijk zo dat hoe sneller een stukje geheugen toegankelijk is, hoe duurder het geheugen per bit is. Zo is de prijs van een kilobyte op een floppy disk slechts ongeveer een tiende cent terwijl dynamisch RAM nog altijd ongeveer een 10 cent per kilobyte kost. Een harddisk ligt daar ongeveer tussenin. Zo is het ook met de snelheid, RAM is sneller dan een harddisk en dat is weer sneller dan floppy's. Wat doe je dus in de praktijk? Je slaat informatie die je op dat moment veelvuldig nodig hebt op in het snelste geheugen (RAM) iets wat je geregeld nodig hebt, sla je op op je harde schijf en je archief van bijvoorbeeld gepubliceerde artikelen bewaar je op floppy's of eventueel op magneetband.

In grote lijnen is cache-geheugen de volgende etage van de piramide. Dit is geheugen dat nog een grootte-orde sneller is dan RAM maar waarvan de prijs ook een flink stuk hoger ligt. Hier tracht je de informatie in te bewaren die je zeer spoedig weer nodig denkt te hebben. Van een programma is dat bijvoorbeeld het lusje van 30 instructies die je enkele honderden malen doorloopt. De hoogste trede op de piramide zijn de interne processor-registers; die zijn nog weer veel sneller dan welk type RAM dan ook. Deze zijn echter zeer schaars.

Zo is de prijs van een kilobyte op een floppy disk slechts ongeveer een tiende cent terwijl dynamisch RAM nog altijd ongeveer een 10 cent per kilobyte kost.

Als je caches hebt, dan kan kunnen die nog op verschillende manieren in het computersysteem zitten. Zo kun je externe caches hebben wat betekent dat ze als afzonderlijke (geheugen-) chips in de computer zijn aan te wijzen. Dit zie je bij de 80386-systemen die statisch RAM gebruiken als extern cache-geheugen. Een 68030 heeft interne caches, dit betekent dat de caches in de processor zelf zijn opgenomen. Dit heeft behalve de snelheid nog een ander voordeel, als namelijk de processor de gezochte informatie in cache vindt, dan is er geen benadering van de externe bus nodig waardoor de processor parallel kan doordraaien met activiteiten die ook de bus gebruiken zoals bijvoorbeeld DMA. Bovendien is de snelheid waarmee interne caches benaderd kunnen worden over het algemeen nog wat groter dan voor externe caches. Bij de 68030 kun je, naast de interne caches, ook nog externe caches aansluiten waarmee weer eens duidelijk hoe krachtig deze processor eigenlijk is.

De opbouw van een cache is in het algemeen zo dat het cache-geheugen parallel aan de werkelijke geheugen-locaties staat. Zo kun je een stukje cache-geheugen opvatten als een spiekbriefje waar opstaat: op adres A staat nu het (hexadecimale) getal X, op adres B het getal Y etc. Wordt er nu een adres benaderd, dan gaat de processor op dit spiekbriefje kijken of de inhoud in cache staat, bovendien wordt er voor de zekerheid al vast een benadering van het externe geheugen gestart. Mocht het gezocht adres in cache aanwezig zijn, dan wordt de benadering van het externe geheugen afgebroken en kan de processor weer aan het werk.

Bij processors met een omrekening van logische naar fysieke adressen, kun je nog kiezen of het cache-geheugen werkt met de logische of fysieke adressen. Je spreekt in dat geval over logische resp. fysieke cache. Bij de 68030 zijn de adressen die in cache weggeschreven worden de logische adressen. Dit heeft als voordeel dat je niet eerst de adres-omrekening hoeft te doen voordat je het cache benadert maar als nadeel dat na een wijziging van de omreken-tabel de inhoud van de caches niet meer betrouwbaar is omdat de logische adressen dan naar heel andere fysieke adressen kunnen verwijzen.

Bij de 68030 hebben we tenslotte zowel een instructie als een data cache. De instructie cache wordt benaderd bij het ophalen van een instructie terwijl de

data cache benaderd wordt bij het ophalen van de operanden, ook als ze (met bijvoorbeeld program relative adressering) tussen de instructies in het programma staan. In figuur 1 is te zien dat de instructieen de data cache binnen de processor volledig gescheiden zijn en ook aparte datapaden naar de overige delen van de processor hebben. Deze opbouw noemt men de Harvard architectuur, in tegenstelling tot de Von Neuman architectuur waarbij er geen onderscheidt is tussen data en instructies. De Harvard architectuur komt de snelheid ten goede omdat, in combinatie met pipelining van instructies, de instructies en de data binnen de processor volledig onafhankelijk van elkaar en dus parallel benaderd kunnen worden.

Instructie cache bij de 68030

In figuur 2 is de instructie cache schematisch weergegeven. Het cache-geheugen is opgebouwd uit 16 "regels" met een lengte van 4 long words. Voor elke regel is er de zogenaamde "Tag". Deze bevat de 24 meest significante adresbits (van de 32), functiecode FC2 en vier zogenaamde "Valid" bits. FC2 is opgenomen om onderscheidt te kunnen maken tussen de adresruimte voor User State van de processor en voor Supervisor State. Adres-bits A4 t/m A7 bepalen altijd in welke regel (van de 16) de instructie terecht komt. Door vervolgens de bijbehorende "Tag" te vergelijken met adres-bits A8 t/m A32 en functiecode FC2 kan onderzocht worden of het gezochte adres op de betreffende regel staat. Is dit het geval, dan bepaalt het bijbehorende "Valid" bit of de inhoud in overeenstemming is met de inhoud van het RAM-geheugen op het betreffende logische adres.

Is het adres aanwezig (en valid), dan vindt er een zogenaamde "Hit" plaats. De benadering van het externe geheugen wordt afgebroken en het gevraagde word wordt aan de processor doorgegeven. Is het adres niet aanwezig (of niet valid), dan noemt men dit een "Miss". (N.B. Ik prefereer de engelse termen i.p.v. een Treffer en een Misser). In dat geval moet de informatie uit het externe RAM komen. Als het adres- en/of de functiecode niet klopt, dan wordt er een long word uit het geheugen gehaald en in de cache geschreven. Eventueel kan, afhankelijk van de inhoud van het Cache Control Register, in zogenaamde "Burst Mode" meteen een hele regel van 4 long words uit het geheugen gehaald worden. Klopt het adres wel, maar is het betreffende long word niet valid, dan wordt alleen het betreffende long word opgehaald, tenzij alle vier long words niet valid zijn, ook dan wordt er eventueel een hele regel in "Burst Mode" opgehaald.

Bij het instructie cache is een entry invalid geworden als de Tag van de regel gewijzigd als zonder dat het bijbehorende long word opgehaald is of als de instructie cache of de betreffende regel, expliciet door een instructie of een RESET gewist is

Data cache bij de 68030

Vergelijken we figuur 2 met figuur 3, dat valt op dat alleen de "Tag" van een regel anders is. Bij de data cache zijn behalve FC2 ook de twee andere functiecode bits opgenomen. Dit komt omdat we bij bijvoorbeeld Program Counter Relatieve adressering de data moeten benaderen in de Instructie-space van het geheugen. Voor leesoperaties is de benadering van de cache en het geheugen precies gelijk aan de afhandeling bij de instructie cache. Dit betekent dat, onafhankelijk van de lengte van de operand, er altijd één of meer long words worden ingelezen. Is de operand niet aligned, dan kan het zelfs voorko-

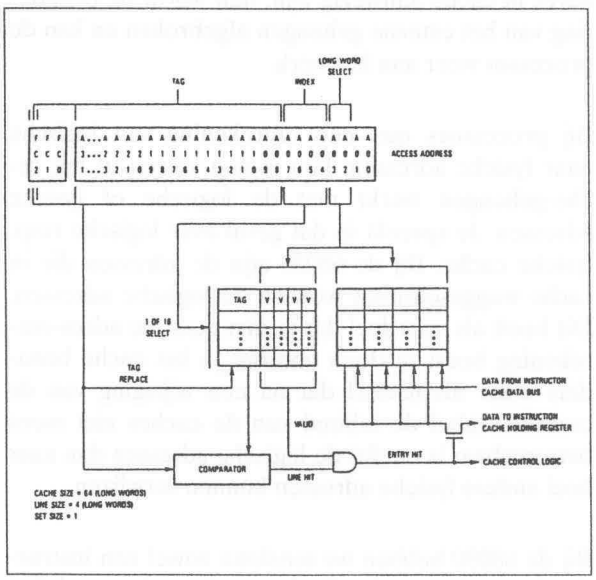


Fig. 2: opbouw van de instructiecache

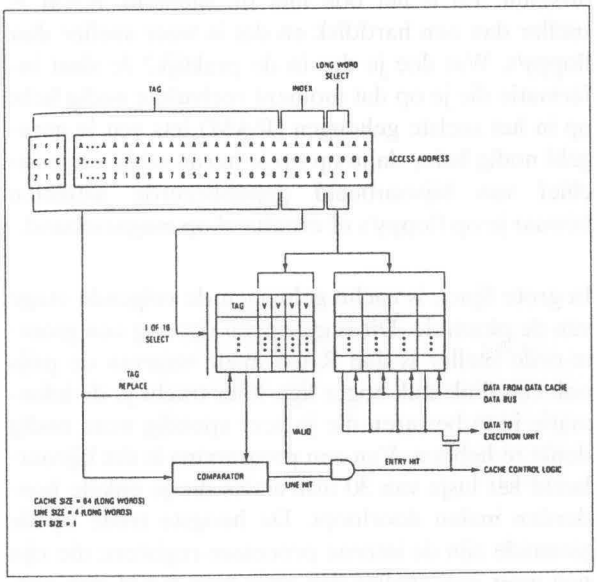


Fig. 3: opbouw van de datacache

men dat er twee long words na elkaar benaderd moeten worden. Deze twee long words worden onafhankelijk van elkaar bekeken, het eerste kan dus een Miss geven in de cache en de tweede een Hit waarna alleen het eerste long word opgehaald wordt.

Wat wel principieel verschilt is het feit dat we bij het benaderen van instructies alleen maar lees-operaties uitvoeren op het geheugen. Bij data doen we echter ook schrijf-operaties waarbij het uiteraard onvoldoende is de gegevens alleen maar in de data cache te schrijven, nee, ze zullen ook overgenomen moeten worden door het geheugen. Het mechanisme dat bij de 68030 in dat geval gebruikt wordt, is het zogenaamde "Write Through" mechanisme. Dit betekent dat bij een "Hit" op een schrijf-operatie altijd zowel de cache als het externe geheugen beschreven wordt; ongeacht de lengte van de operand en of de cache entry bevroren (frozen, zie volgende paragraaf) is.

Wat nog wel enige aandacht vraagt is het probleem dat twee logische adressen naar hetzelfde fysieke adres kunnen verwijzen. Dit moet dan als zodanig in de vertaaltabel tussen logische en fysieke adressen opgenomen zijn. Verder kan het uiteraard voorkomen dat een fysiek adres zowel in User State als in Supervisor State benaderd wordt. In deze gevallen is de Tag van het logische adres in de cache verschillend en kan het dus voorkomen dat de inhoud van een adres wel in de cache aanwezig is maar niet gezien wordt omdat bijvoorbeeld FC2 de verkeerde waarde heeft. Op dat moment hebben we dus een Miss bij de schrijfoperatie maar de gegevens staan wel in de cache. Afhankelijk van de programmering van het Cache Control Register wordt bepaald of in geval van een Miss de inhoud van de Cache wordt aangepast. Doet men dit niet, dan zou het voor kunnen komen dat je later met verouderde en dus verkeerde gegevens werkt. Het bit dat dit bestuurt is het zogenaamde Write Allocation bit in het Cache Control Register.

Besturing van de caches

In het supervisor programmeermodel vinden we twee registers die te maken hebben met de besturing van de caches. Dit zijn het zogenaamde Cache Control Register en het Cache Address Register. In het Cache Control Register vinden we in de eerste plaats het Write Allocate bit waarmee wordt aangegeven of bij een Miss op een schrijfoperatie de inhoud van de data cache moet worden aangepast.

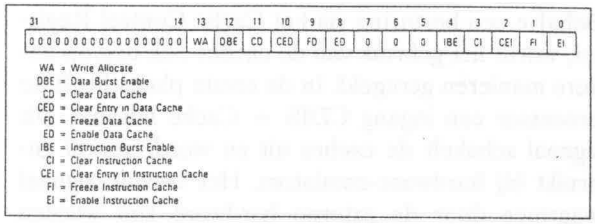


Fig. 4: het cache control register

De Data Burst Enable en Instruction Burst Enable bits geven aan of de caches gevuld mogen worden in de zogenaamde Burst Mode. Veel typen RAM-geheugen kunnen namelijk sneller meerdere long words achter elkaar benaderen dan stuk voor stuk. In die gevallen kan het dus voordelig zijn een hele regel cache in één klap in te lezen. Ook bij de KGN68k is dit het geval. Hier zijn in de hardware maatregelen genomen om optimaal gebruik van deze mogelijkheid te kunnen maken.

De bits Enable Data Cache en Enable Instruction Cache geven uiteraard aan of caching überhaupt wel toegestaan is.

Met behulp van de bits: Clear Data Cache resp. Clear Instruction Cache kunnen de caches compleet gewist worden. Het is van groot belang dit te doen nadat de vertaal-tabel tussen logische en fysieke adressen gewijzigd is. Met behulp van de bits: Clear Entry in Data Cache resp. Clear Entry in Instruction Cache kan een long word in de data cache resp. de instructie cache gewist worden. Welk long word wordt gewist staat in bits 2 t/m 7 van het Cache Address Register. De overige bits in dit register worden bij de 68030 niet gebruikt.

De bits Enable Data Cache en Enable Instruction Cache geven uiteraard aan of caching überhaupt wel toegestaan is.

Met behulp van de Freeze Data Cache en Freeze Instruction Cache kan men aangeven dat de caches tijdelijk bevroren (frozen) zijn. Dit betekent dat bij een Miss op een leesoperatie er geen gegevens naar de cache gehaald worden. Hiermee kun je dus forceren dat de aanwezige informatie in de cache bewaard blijft en niet wordt overschreven omdat je even wat gegevens ophaalt die je maar één maal nodig hebt.

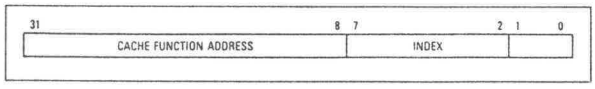


Fig. 5: het cache address register

Behalve een besturing via het Cache Control Register, wordt het gebruik van de caches ook nog op andere manieren geregeld. In de eerste plaats heeft de processor een ingang CDIS = Cache Disable. Dit signaal schakelt de caches uit en wordt vooral gebruikt bij hardware-emulators. Het tweede signaal waarmee door de externe hardware kan worden aangegeven dat een bepaald adres niet in de cache mag worden opgenomen is CIIN = Cache Inhibit In. Hiermee kan tijdens een benadering van het geheugen aangegeven worden dat de gegevens die gelezen worden niet in de cache mogen worden opgenomen. Dit kan bijvoorbeeld het geval zijn als de gegevens steeds veranderen omdat ze afkomstig zijn van een periferie-bouwsteen (timer o.i.d.). Het derde signaal waarmee aangegeven wordt dat de gegevens niet in de caches overgenomen (mogen) worden is het CIOUT = Cache Inhibit Out signaal. Dit signaal wordt afgegeven door de MMU waar in staat dat de aangegeven pagina niet "cacheable" is. Omdat een 68030 systeem naast zijn interne cache eventueel ook nog over externe caches zou kunnen beschikken, is het noodzakelijk dit signaal ook naar buiten de processor te voeren.

Samenvattend kan worden aangegeven dat informatie cacheable is als voldaan is aan de volgende voorwaarden:

- De cache is enabled via het Cache Control register
- Het CDIS signaal is laag
- Het CIIN signaal is laag bij de benadering van het geheugen (afkomstig van de hardware)
- Het CIOUT signaal is laag bij de benadering van het geheugen (afkomstig van de MMU)
- De MMU staat de benadering van het geadresseerde geheugen voor de betreffende benadering toe (lezen/schrijven, User State, Supervisor State).

Afsluiting

Resteert natuurlijk de vraag hoeveel het hebben van caches nu werkelijk uitmaakt. Welnu, een leesoperatie op de cache gaat altijd 30 % sneller dan een leesoperatie op het snelste externe geheugen. Al er voor het benaderen van het geheugen zogenaamde Wait States tussengevoegd moeten worden, dan is de benadering van de caches nog veel sneller. Dit betekent dat als gegevens meerdere malen opgehaald moeten worden het zinvol is ervoor te zorgen dat ze in cache staan. Bij kleine programma-lusjes (zoals er vaak zeer veel in een programma zitten), zal een instructie cache zeker voordelen hebben, mits de cache groot genoeg is om het hele lusje te bevatten. Bij data ligt het iets gecompliceerder. Tenslotte is 256 byte niet zo heel veel en of het echt voordeel geeft is sterk afhankelijk van de wijze waarop van de data cache gebruik wordt gemaakt.

In de volgende aflevering zal, soldeertin en tekstverwerker dienende, de Memory Management Unit of MMU behandeld gaan worden.

Caches zullen ook zeker een voordeel hebben als we gebruik kunnen maken van de "Burst Mode". Ook daar valt een behoorlijke snelheidswinst te behalen mits het RAM-geheugen daarvoor ingericht is.

Blijft over een vooruitblik op het volgende artikel. Toen ik aan dit artikel begon, had ik het plan zowel de caches als de MMU te gaan behandelen. De caches leveren echter al voldoende stof tot schrijven op en bovendien blijken er voor de behandeling van de MMU ook nog wel enkele pagina's μP Kenner nodig te zijn. In de volgende aflevering zal, soldeertin en tekstverwerker dienende, de Memory Management Unit of MMU behandeld gaan worden.

Tot de volgende keer dus.

Gert van Opbroek

Literatuur

1: Motorola: MC68030 Enhanced 32-bit microprocessor user's manual; Prentice Hall 1990. ISBN: 0-13-566423-3.

2: Werner Hilf: Der 68030; MC 4, 5 en 6 1988, Franzis Verlag GmbH.

Documentatie 1 is beschikbaar gesteld aan de KGN68k werkgroep door EBV Elektronik te Maarssenbroek.

Van de bestuurstafel

Het jaar is nog maar net begonnen of de eerste problemen doen zich al weer voor. Voor de algemene ledenvergadering was er een oproep voor nieuwe bestuursleden. Helaas was niemand bereid om het bestuur te komen versterken. Tot onze schrik nam Joost Voorhaar tijdens de laatste bestuursvergadering ook nog eens afscheid van het bestuur. Hij vindt dat de leden te weinig interesse tonen voor de vereniging. Er wordt nog steeds te weinig kopij ingestuurd voor ons lijfblad. Dat maakt het hem, volgens eigen zeggen, onmogelijk om zijn functie als redacteur nog langer verantwoord uit te voeren. Ondanks dat het ons spijt dat Joost weg gaat kunnen we zijn argumenten onderschrijven. Voorlopig zullen Gert van Opbroek en Tonny Schäffer zijn taak waarnemen.

Door het vertrek van Joost zijn er nu nog maar vier bestuursleden over. Dat betekent dat we er drie tekort komen. Voor de vereniging ontstaat er nu een instabiele situatie. Er zijn te weinig bestuursleden. Dat betekent dat het gevaar bestaat dat één van de overgebleven bestuursleden ook zijn taken neerlegt, omdat hij teveel werk moet verzetten. Als het bestuur dan ook nog eens het blad moet vullen omdat de leden te weinig kopij aanleveren, acht ik de kans levensgroot dat er een bestuurscrisis ontstaat. De goede samenwerking en sfeer binnen het bestuur zal daar niets aan kunnen veranderen.

Als we nu niets van jullie horen, moet je niet raar opkijken als wij ook in stilzwijgen veranderen.

De leden kunnen een crisis alleen nog verhinderen door zich beschikbaar te stellen voor een bestuursfunctie en/of door het leveren van kopij. Als we nu niets van jullie horen, moet je niet raar opkijken als wij ook in stilzwijgen veranderen.

Dat er ook een andere sfeer binnen onze vereniging kan heersen was te merken in Krommenie. Daar is de sfeer over het algemeen toch al goed. Maar deze keer was er een voordracht die letterlijk het licht deed zien. Antoine Megens en Jaap Prenger hadden een boel apparatuur en evenzoveel interessante informatie bij zich. Een theater-lichtinstallatie waaraan ze zijn begonnen vanuit de hobbysfeer is uitgegroeid tot een professioneel geheel. Alles is vanuit een monitorprogramma in te stellen en op te slaan, zodat het later weer opgeroepen kan worden. Het kan dan weer automatisch alle lampen bedienen en indien nodig eventueel met de hand bij geregeld worden. Zowel de hardware als de software is door Jaap en Antoine ontwikkeld. Eén en ander ziet er zeer goed uit en ze hebben er een mooie voordracht van gemaakt. De leden die niet aanwezig waren in Krommenie hebben echt iets gemist.

Het mag hier nog wel eens gezegd worden dat de ontvangst en het verblijf bij Forbo toch elk jaar weer een goede indruk achter laat bij de bezoekers. Een goede locatie en er wordt ook aan de inwendige mens gedacht op een manier die een dikke pluim verdiend. Langs deze weg wil ik Co Filmer en het bedrijf waarvoor hij werkt, Forbo-Krommenie, nog eens hartelijk bedanken voor hun gastvrijheid.

De volgende bijeenkomst is in Geldrop en zal geheel gewijd worden aan DOS65. Ik wil hierbij dan ook alle DOS65 bezitters oproepen om in Geldrop aanwezig te zijn. De werkgroep zal er zijn om eventuele vragen te beantwoorden en om samen met de aanwezigen kijken wat er in de toekomst met dit systeem moet gebeuren. Kan er nog wat verbeterd worden? Moeten er nog nieuwe dingen ontwikkeld worden? De werkgroep wil wel als er voldoende belangstelling voor bestaat. Met andere woorden laat je zien en horen zodat je je favoriete systeem kunt blijven gebruiken en verder ontwikkelen.

Tot ziens in Geldrop.

Tonny Schäffer

Informatie

De μP Kenner (De microprocessor Kenner) is een uitgave van de KIM gebruikersclub Nederland. Deze vereniging is volledig onafhankelijk, is statutair opgericht op 22 juni 1978 en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305. Het gironummer van de vereniging is 3757649.

De doelstellingen van de vereniging zijn sinds 1 januari 1989 als volgt geformuleerd:

- Het vergaren en verspreiden van kennis over componenten van microcomputers, de microcomputers zelf en de bijbehorende systeemsoftware.
- Het stimuleren en ondersteunen van het gebruik van micro-computers in de meer technische toepassingen.

Om deze doelstellingen zo goed mogelijk in te vullen, wordt onder andere 5 maal per jaar de μP Kenner uitgegeven. Verder worden er door het bestuur per jaar 5 landelijke bijeenkomsten georganiseerd, beheert het bestuur een Bulletin Board en wordt er een softwarebibliotheek en een technisch forum voor de diverse systemen in stand gehouden.

Landelijke bijeenkomsten:

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

Bulletin Board:

Voor het uitwisselen van mededelingen, het stellen en beantwoorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board beschikbaar gesteld.
De telefoonnummers zijn: 053-328506, 053-303902 of 053-327457.

Software Bibliotheek en Technisch Forum:

Voor het beheer van de Software Bibliotheek en technischeondersteuning streeft het bestuur ernaar zgn. systeemcoordinators te benoemen. Van tijd tot tijd zal in de μP Kenner een overzicht gepubliceerd worden. Dit overzicht staat ook op het Bulletin Board.

Correspondentie adres

Alle correspondentie betreffende verenigingszaken kan gestuurd worden aan:

KIM Gebruikersclub Nederland
Postbus 1336
7500 BH Enschede

Het Bestuur

Het bestuur van de vereniging wordt gevormd door een dagelijks bestuur bestaande uit een voorzitter, een secretaris en een penningmeester en een viertal gewone leden.

Tonny Schäffer (voorzitter)
Paul Krügerstraat 27
7532 PW Enschede
Telefoon 053-613678

Jacques H.G.M. Banser (penningmeester)
Haaksbergerstraat 199
7513 EM Enschede
Telefoon 053-324137

Gert van Opbroek (secretaris/redactie μP Kenner)
Bateweg 60
2481 AN Woubrugge
Telefoon 01729-8636

Geert Stappers (KGN/68k coordinator)
Engelseweg 7
5825 BT Overloon
Telefoon 04781-41279

Ereleden:

Naast het bestuur zijn er een aantal ereleden, die zich in het verleden bijzonder verdienstelijk voor de club hebben gemaakt:

Erevoorzitter:
Siep de Vries

Ereleden:
Mevr. H. de Vries-van der Winden
Anton Müller
Rinus Vleesch Dubois