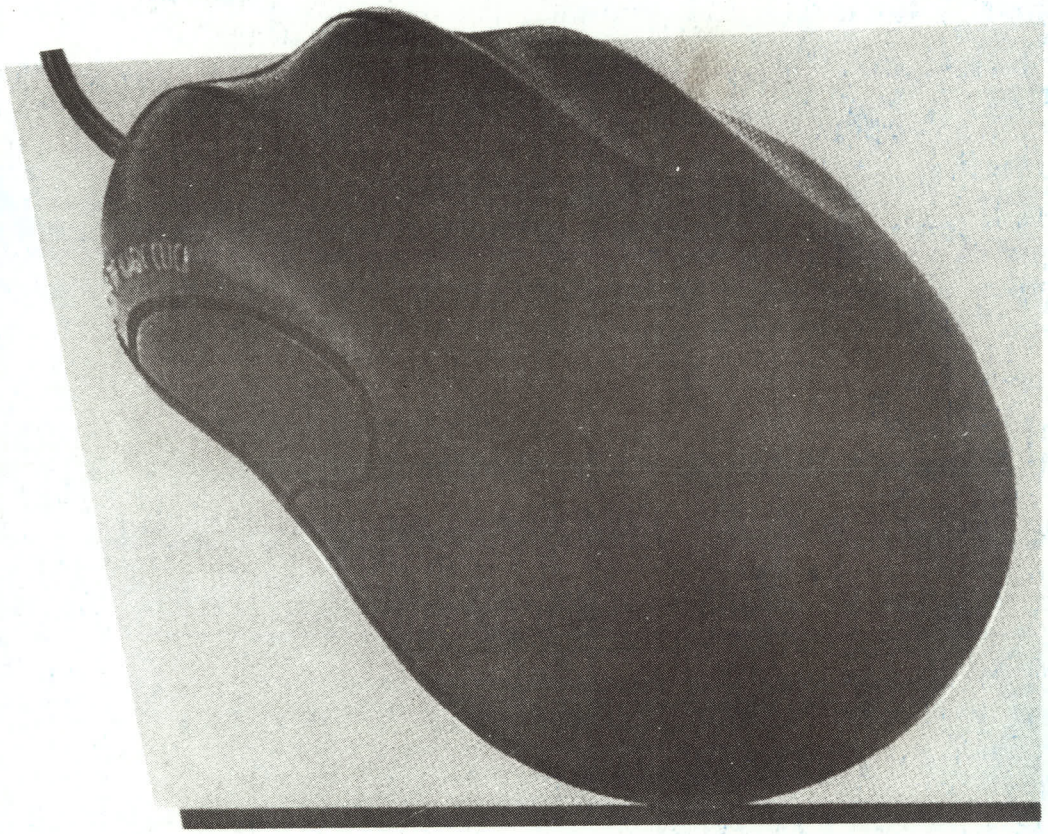
De µP Kenner

Zeventiende jaargang nr. 4
Oktober 1993
83



In dit nummer o.a.:

Inhoudsopgave

De μP Kenner

Nummer 83, oktober 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:

Nico de Vries
Joost Voorhaar

Redactieadres:

p/a Gert van Opbroek
Den Del 16
5071 TT Udenhout

v.a. 25-12-1993:
Nico de Vries
Van der Waalsstraat 46
2984 EP Ridderkerk

De μP Kenner nummer 84 verschijnt op 18 december 1993.

Kopijsluitingsdatum voor nummer 84 is vastgesteld op 4 december 1993.

Vereniging

Algemeen

Talen/Software

Hardware/DOS65

Systemen

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

Het zou wel eens zo kunnen zijn dat dit de laatste keer is dat ik het redactioneel schrijf. We hebben namelijk iemand gevonden die (tijdelijk) de redactie over wil nemen en dat persoon mag dan uiteraard ook deze column vullen. Een oude bekende, Nico de Vries, gaat zich de komende tijd wat meer met de redactie bezighouden zodat redacteur en layouter nu in één persoon zijn verenigd en de secretaris zich weer wat meer met onder andere KGN68k bezig kan gaan houden.

Zelf ben ik wel van plan voorlopig nog een behoorlijke bijdrage aan het blad te geven in de vorm van artikelen. Tenslotte kunnen we als leden van de KGN de redacteur er niet alleen voor op laten draaien vijf maal per jaar een blad te vullen. Ik hoop dat de lezers nog steeds niet genoeg van mijn schrijfsels hebben want het schrijven is zo langzamerhand ook een hobby van me geworden. Als ik iets heb uitgezocht of iets leuks tegen gekomen ben, dan sta ik gewoon te popelen om daarover iets in de μP Kenner te gaan schrijven. Zo ook deze keer. Ik heb de laatste maanden eens uitgeplozen hoe een object file voor de GNU linker in elkaar zit. Daarover heb ik toen meteen maar een artikel geschreven en dat staat in deze uitgave van de μP Kenner afgedrukt. Een tweede artikel van mijn hand gaat over de enige zinvolle toepassing van computers in het huishouden die ik ben tegengekomen namelijk het voeren van een soort elektronisch kasboek.

Als ik iets heb uitgezocht of iets leuks tegenkom, dan sta ik gewoon te popelen om daarover in de μP Kenner te gaan schrijven.

Naast mijn bijdragen staan er uiteraard ook nog een aantal andere artikelen in dit blad. Dat is in de eerste plaats al weer een deel van de cursus C van Hans van Boheemen en verder een artikel over de software voor een muis aan DOS 65 van Antoine Megens. Misschien zijn er nog wel meer interessante artikelen maar dat is voor mij op dit moment ook nog een verrassing. Het kan namelijk zijn dat er voordat het blad naar de drukker gaat nog wat binnenkomt en verder heeft Nico misschien ook nog wel wat gekregen of geschreven.
Je zult wel begrijpen dat we nog altijd kopij nodig hebben. We zijn ook nu weer helemaal tot het gaatje gegaan, kortom voor het decembernummer hebben we weer zo'n veertig pagina's kopij nodig. Wie werkt daar aan mee? Wat we zeker graag willen hebben is iemand die de ShareWare rubriek van Joost Voorhaar voortzet. Er is tegenwoordig zo ontzettend veel software in de Public Domain en ShareWare hoek dat een toonaangevend blad als de μP Kenner gewoon niet zonder een dergelijke rubriek uit zou mogen komen. Wie voelt zich geroepen elke twee maanden eens enkele programma's van een BBS te plukken en hierover een artikeltje te schrijven? Aanmeldingen bij Nico of bij mij.

Waarschijnlijk zal de KGN dit jaar niet op de HCC-dagen vertegenwoordigd zijn. Door allerlei omstandigheden waaronder chronische onderbezetting van het bestuur waren we helaas veel te laat met het indienen van een verzoek om tegen geringe kosten op de beurs te mogen staan. Zouden we de gebruikelijke prijs moeten betalen, dan moest de KGN fl. 1200,-- excl. b.t.w. dus zo'n fl. 1400,-- betalen. Welnu, dat kan Bruin echt niet trekken. Wel hebben we nog geprobeerd, evenals vorig jaar, gratis op de beurs te staan maar het lijkt er op dat dat dit keer niet gaat lukken. Jammer.....

Waar wij als KGN een traditie van willen maken is het houden van een jaarlijkse Linux dag. Zoals je allemaal wel weet, is Linux een afgeleide van een Unix operating system dat draait op een grote PC. Dit is ook het operating system dat we naar de KGN68k hardware willen gaan porteren. Nu zijn we vorig jaar begonnen met een speciale Linux dag waarvoor we ook een aantal mensen uit de Unix/Linux hoek hadden uitgenodigd. Zelf ben ik helaas niet op deze bijeenkomst geweest, maar ik heb begrepen dat het een redelijk succes geworden is.

Dit jaar willen we weer een dergelijke bijeenkomst houden en we zijn eigenlijk van plan dit tot een jaarlijks terugkerend evenement te maken. Het is nog niet precies bekend wat we allemaal gaan doen, maar zeker is in ieder geval dat er enkele Linux systemen aanwezig zullen zijn die gedemonstreerd kunnen worden en voor de lezing hebben we iemand op het oog die iets kan vertellen over netwerken onder Unix/Linux. Deze bijeenkomst zal gehouden worden op zaterdag 27 november, 1 week na de HCC-dagen op een nieuwe lokatie in Amersfoort. Uiteraard kun je de details weer in de uitnodiging lezen.
Rest mij nog iedereen veel leesplezier aan dit blad te wensen en tot ziens op één van de volgende bijeenkomsten.

Gert van Opbroek

Uitnodiging voor de clubbijeenkomst

Datum: 27 november 1993
Lokatie: Buurthuis "Het Klokhuis"
Haydenstraat 57
Amersfoort
Tel: 033-753163
Thema: Linux/Netwerken

Routebeschrijving

Vanuit Utrecht:

Afslag Woudenberg / Leusden
Na de eerste twee stoplichten linksaf, Bij het volgende stoplicht wederom linksaf en dan onder het viaduct door. Je bevindt je nu op de Arnhemseweg. Zie verder punt *

Vanuit Amsterdam / Apeldoorn:

Na verkeersplein Amersfoort, derde afslag nemen (Leusden zuid). Onderaan de afslag ga je rechts af. Je zit dan op de Arnhemseweg. Zie verder punt *

Vanuit Arnhem:

Autoweg A2 richting Den Haag. Afslag Maarsbergen nemen, hier richting Woudenberg gaan. Alsmaar op deze weg blijven. Door Leusden heen, in Leusden is dit reeds de Arnhemse weg. Vervolg de weg naar Amersfoort. Zie verder punt *

*** In Amersfoort:**

Na het bord Amersfoort (stadsbord) komt u eerst een stoplicht tegen (oversteekplaats voor voetgangers/fietsen). De weg vervolgen. De weg komt dan op een soort van "T" splitsing uit. De Arnhemse weg verloopt naar rechts over het Spoor. Direct na de spoorovergang komt u op een 5 sprong. Hier moet je rechts aanhouden (of eigenlijk de tweede straat van rechts nemen). Dit is de Gasthuislaan. Volg daar eventueel het bord ziekenhuis "Eemland". Vervolg deze weg totaan de eerste stoplichten. Ga hier rechtsaf. Dit is de Heiligenbergerweg. Vervolgens de eerste straat links. Dit is de Haydenstraat. Tot aan de "T" splitsing rijden. Rechts ligt dan het buurthuis "Het Klokhuis".

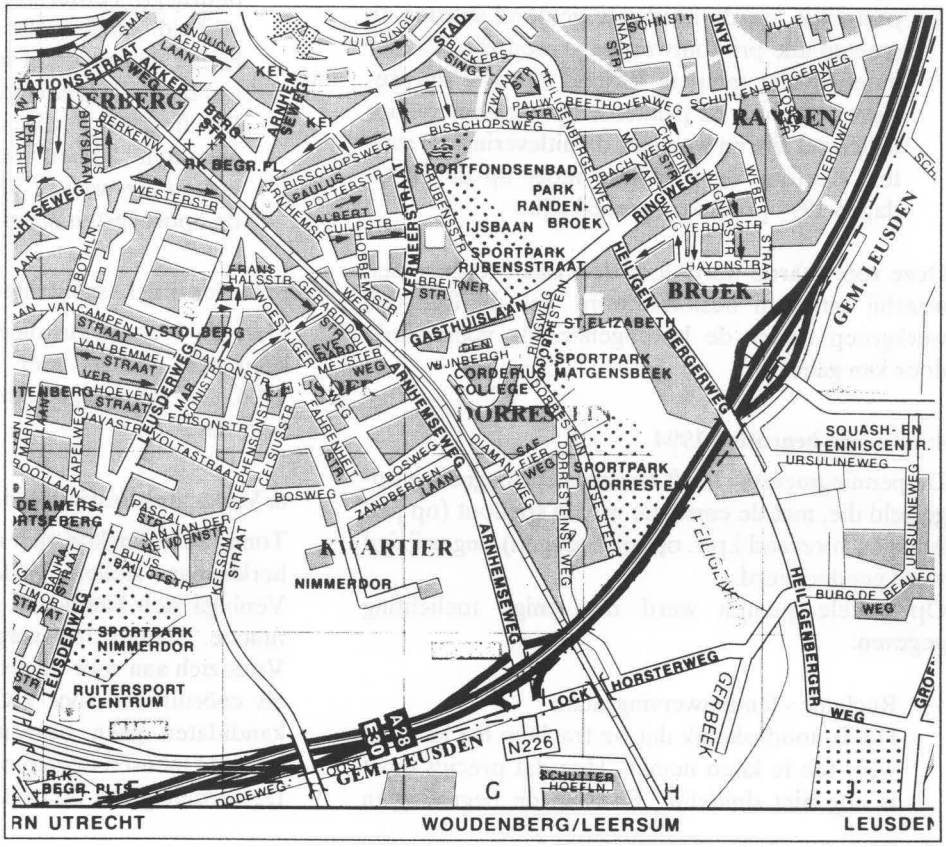
Openbaar vervoer

Vanaf station Amersfoort: met stadsdienst 7 of 8. Uitstappen op de Verdiweg. in de buurt van de kerk. Van daaruit kun je het klokhuis zien staan aan de andere kant van het grasveld.

Met het streekvervoer lijn 115,117 of 118: uitstappen op de Heiligenbergerweg bij het ziekenhuis.

Programma:

9:30	Zaal open met koffie
10:15	Opening
10:16	Voordracht over netwerken door Hugo van der Kooij
11:00	Demonstratie van de aanwezige Linux systemen, netwerken en ontwikkeltools t.b.v. KGN68k
12:30	Lunch, consumpties tegen betaling
13:30	Forum en Markt 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.



Verslag van de algemene ledenvergadering

Deze vergadering werd gehouden op 25 september 1993 in 't Veurbrook te Almelo.

1: Opening en vaststelling agenda

Na een voordracht over de stand van zaken in de werkgroep KGN68k werd de vergadering geopend. Het voltallige bestuur was aanwezig en +/- 15 leden. De voorgestelde agenda, zoals afgedrukt in μP Kenner 82, werd goedgekeurd.

2: Verslag ledenvergadering 15 mei 1993 te Utrecht

Het verslag van bovengenoemde vergadering, zoals afgedrukt in μP Kenner 82, werd zonder op- of aanmerkingen goedgekeurd.

3: Het project KGN68k

Nadat het bestuur, bij monde van Tonny Schäffer, uitgelegd heeft dat door de opgetreden vertragingen het bestuur niet langer alleen de verantwoordelijkheid voor het project kan dragen, is de vraag aan de leden voorgelegd of de KGN met het project moet stoppen of dat de werkgroep door mag gaan en, zo ja, onder welke voorwaarden.
De aanwezige leden waren van mening dat de werkgroep door kan gaan met het ontwikkelen van de KGN68k waarbij op voorstel van Nico de Vries wel een duidelijke randvoorwaarde gesteld werd. De voorwaarde die hij stelde was:

- Op de HCC-dagen 1994 moet er een werkend prototype van de hardware getoond worden en moeten alle printontwerpen gereed zijn. Verder moet op dat moment bekend zijn wat de precieze prijs voor de printen en programmeerbare logica zal zijn en wanneer de uitlevering zal starten. Kortom, de hardware moet op de HCC-dagen 1994 produktie-gereed zijn.

Deze voorwaarde werd door de leden overgenomen waarbij unaniem besloten werd dat de KGN68k werkgroep onder de bovengenoemde voorwaarde door kan gaan.

4: Concept-begroting 1994

De penningmeester heeft een concept-begroting opgesteld die, met de correctie van de spelfout (op peil brengen voorraad i.p.v. op pijl brengen) ongewijzigd werd goedgekeurd.
Op enkele punten werd nog enige toelichting gegeven:

- Reclame - lendenwervingsactie
  Het is noodzakelijk dat we trachten het ledental weer toe te laten nemen. Hoe dat precies moet is nog niet duidelijk. Er is in de begroting in ieder geval geld gereserveerd om deel te kunnen nemen aan de HCC dagen (mits tegen een gereduceerd tarief). Verder bestaat het plan scholen die leerlingen hebben in onze doelgroep enkele gratis μP Kenners toe te sturen. Uit de enquête blijkt dat ongeveer de helft van de leden een opleiding hebben op HBO niveau of hoger en ongeveer de helft op MBO niveau. Het bestuur vraagt de leden om adressen van mogelijk interessante scholen en/of opleidingsinstituten door te geven zodat die μP Kenners toegestuurd kunnen krijgen.
- Sponsoring Bulletin Board The Ultimate
  The Ultimate is in het verleden opgezet als BBS in eigendom van de KGN. Momenteel is het al lang niet meer zo dat The Ultimate uitsluitend ons eigendom is, sterker nog, slechts een zeer klein deel van de hardware is door de KGN betaald en ook van de totale kosten betaalt de KGN maar een klein deel.
  Nu is het de bedoeling dat The Ultimate in principe onafhankelijk van de KGN wordt en dat de KGN een contract met de eigenaar (nu nog Jacques Banser, straks misschien een stichting o.i.d.) afsluit waarin afgesproken wordt dat het BBS bepaalde diensten aan de KGN levert tegen een vaste vergoeding. Het verschil met de huidige constructie is dan dat The Ultimate de sponsor-bijdragen (die nu in de clubkas vloeien) mag behouden en dat de KGN een vast bedrag bijdraagt. Verder krijgt The Ultimate wat meer vrijheid omdat alleen over club-aangelegenheden de toe- en/of instemming van het bestuur van de KGN benodigd is.
  De fl. 2000,-- die in de begroting zijn opgenomen komen ongeveer overeen met de kosten die we aan The Ultimate hebben minus de opbrengsten uit sponsoring.

5: Verkiezing kascontrole-commissie 1994

Herman Hek en Antoine Megens hebben zich beschikbaar gesteld voor de kascontrole-commissie 1994. De aanwezige leden hadden hiertegen geen bezwaar.

6: Verkiezing bestuursleden

Tonny Schäffer kon zich om persoonlijke reden niet herkiesbaar stellen. Voor de vergadering had Jan Veninga zich kandidaat gesteld voor een bestuursfunctie. Staande de vergadering meldde Nico de Vries zich aan voor de redactie en Henk Speksnijder als coördinator voor DOS65. Aangezien tegen de kandidaten geen bezwaren waren, zijn bovengenoemde mensen opgenomen in het bestuur.
De verdeling van de bestuurstaken voor 1994 is nog

niet bekend. Wel is reeds bekend dat Nico de redactie gaat doen en Henk zich inzet voor DOS65.

7: Overige agendapunten

Geen

8: Rondvraag

Herman Hek stelde een vraag over het feit dat hij een bestand naar The Ultimate had geupload t.b.v. de redactie en dat hij die file niet in het overzicht had zien staan. Deze file is ook niet bij de redactie aangekomen. Jacques zegde toe dit uit te zoeken. Bovendien werd er gevraagd naar de inhoud van de area "KIM Club Specials". Ook dit wordt in orde gemaakt.

De uitslag van de enquête

Als eerste willen we alle leden die hun medewerking aan de enquête hebben gegeven hartelijk bedanken.

Het doel van de enquête was te kijken naar het profiel van de leden en of de doelstelling van de vereniging zou moeten worden bijgesteld. Tevens wilden we achterhalen waarom leden hun lidmaatschap opzegden.

Voor de enquête zijn ongeveer een kwart van de leden en ex-leden ondervraagd. Helaas waren van de ex-leden de adressen niet meer actueel zodat we niet hebben kunnen ervaren wat de redenen van de opzeggingen zijn geweest. Wel hebben de vragen een duidelijk beeld gegeven wie onze leden zijn en wat ze van de vereniging verwachten.

Wel hebben de vragen een duidelijk beeld gegeven.

Van de ondervraagden is ruim 80 % dagelijks via werk of studie bezig met microcomputers. Daarvan is weer ruim 80 % technisch bezig met microcomputers. Meer dan 90 % van de ondervraagden heeft een technische opleiding gevolgd of is daar mee bezig.

Het opleidingsnivo ligt voor 50 % op MBO en voor 50 % op HBO.

Bijna alle ondervraagden hebben thuis de beschikking over een IBM-compatible, uiteenlopend van een XT tot een AT-486. Bijna de helft heeft nog eens de beschikking over een DOS65 machine, een kwart beschikt ook nog over een KIM-computer. Verder werden er nog diverse andere systemen genoemd. Hieruit blijkt dat onze leden soms over diverse systemen beschikken. Als reden wordt dan aangegeven dat men daar zo lekker met de soldeerbout in kan roeren.

Bij meer dan de helft van de ondervraagden zijn de volgende programmeertalen bekend: BASIC, Pascal, assembler en 'C'.

Van de ondervraagden is meer als 60 % ook nog eens lid van een andere computervereniging. Vaak is dat een vereniging van het werk maar ook de HCC scoort hier niet slecht.

Er worden door de ondervraagden diverse vakbladen gelezen. Hier scoort Elektuur bijna 50 %.

De waardering over de clubbijeenkomsten wordt door meer als de helft van de bezoekers met goed beoordeeld. Alhoewel de DOS65 bezitters het op prijs zouden stellen als er meer aandacht voor hun machine zou komen. Ze beseffen echter dat ze daar zelf ook het één en ander aan moeten doen.

De μP Kenner wordt door alle leden gelezen. Bijna 65 % vindt het blad goed. Meer dan 90 % van de leden bewaart de bladen allemaal. Toch waren er wel wat aanmerkingen en suggesties enkele daarvan zijn:

- meer kopij minder listings
- goedkope en kleine projecten
- meer nieuws publiceren
- meer DOS65
- meer korte artikelen en variatie
- bespreken van programma's
- meer zelfbouw
- meer IBM-PC gericht
- cursus 'C'

De leden beseffen dat ze daar ook zelf meer aan zouden kunnen bijdragen.

Van de ondervraagden beschikte 67 % over een modem. Daarvan nam 79 % regelmatig contact op met ons BBS The Ultimate. Ruim 65 % heeft een

goede indruk van het BBS. Daar waar al een ontevreden opmerking werd gemaakt kon worden verteld dat er inmiddels verbeteringen waren aangebracht.

Driekwart van de ondervraagden bezoeken regelmatig de HCC-dagen. Ruim driekwart daarvan doen dat niet alleen voor de commerciële stands. Alle bezoekers hebben ook de KGN-stand bezocht.

Van de ondervraagden vindt 74 % dat de KGN op de HCC-dagen vertegenwoordigd moet zijn. Ze vragen wel van het bestuur er op te letten dat het financieel verantwoord blijft.

Bij bijna 1/3 van de ondervraagden is het besturingssysteem UNIX bekend. Terwijl driekwart wel eens van MINIX heeft gehoord. Dat laatste vooral via de μP Kenner.

De belangstelling om aan een projectgroep mee te doen valt tegen. Toch heeft bijna de helft van de ondervraagden wel belangstelling voor een zelfbouwsysteem. Het systeem mag alleen niet meer kosten dan zo'n fl. 2000, = =.

Samenvattend kun je zeggen dat een groot deel van de leden dagelijks bezig is met micro-computers. Een even groot deel doet dat technisch. Het opleidingsnivo ligt op MBO of HBO. De meeste mensen werken thuis met een IBM compatible machine en in mindere mate met een van de club computers, KIM of DOS65. De meest bekende programmeertalen zijn: Assembler, Basic, 'C' en Pascal. Een ruime meerderheid is ook nog lid van een andere computervereniging. Uit de bekende vakbladen blijkt dat er een belangrijke interesse ligt op het technische vlak. De leden weten goed wat er op de clubbijeenkomsten gebeurd. Ze zijn daar niet allemaal tevreden over en daar ligt dus een belangrijke taak voor het bestuur. Het clubblad en het BBS scoren niet slecht zij het dat aan het clubblad kleine verbeteringen kunnen worden toegevoegd die niet altijd door de redactie of het bestuur kunnen worden ingevuld. De KGN-stand blijkt erg populair te zijn op de HCC-dagen maar het mag toch niet te veel kosten. De vraag naar DOS65 liegt er niet om daar ligt dan ook iets voor de toekomst. Het mee willen werken aan een werkgroep valt tegen daar ligt voor het bestuur waarschijnlijk een taak. De vraag naar een zelfbouw-computer is hoog maar dan mag het niet te veel kosten. De kostprijs zal dan ook wel eens de bottle-neck kunnen zijn.

Tonny Schäffer

Begroting 1994

BATEN :

Contributie: 140 * ƒ 75,00	ƒ	10500,00
Reclame	-	250,00
Projecten	-	2500,00
Overige inkomsten	-	500,00
Totaal	ƒ	13750,00

LASTEN :

µP Kenner drukken/verzenden: 5 * ƒ 1000,00	ƒ	5000,00
Afschrijving inventaris	-	1000,00
Sponsoring Bulletin Board The Ultimate	-	2000,00
Bestuurs kosten	-	500,00
Reclame - ledenwervingsactie	-	1000,00
Projecten	-	2500,00
Bijeenkomsten	-	1000,00
Op peil brengen voorraad	-	2500,00
Onvoorzien	-	1350,00
Totaal	ƒ	16850,00

De begroting is dit jaar gemaakt met de gedachte dat we volgend jaar waarschijnlijk iets in zullen teren wat ons leden aantal betreft. Gezien het verloop van het leden aantal vermoeden we dat dit in 1994 helaas ook nog wel enigzins door zal zetten.

De verhoging van de contributie van afgelopen jaar was bedoeld om de bijeenkomsten vrij van entree maken voor onze leden. Die kosten moesten natuurlijk wel opgebracht worden. Dit besluit werd door meerdere leden positief ontvangen zoals we op de bijeenkomsten merkten.

Dit jaar menen wij dit te mogen doen, na eerst ruggespraak te hebben gehad met meerdere leden tijdens de bijeenkomsten in dit jaar. Niet alleen lopen we te ver in op onze vrije reserves, ook de PTT heeft afgelopen januari haar tarieven weer verder verhoogd, hier was tijdens de begroting van 1993 helaas geen rekening gehouden.

Het drukken van het blad gaat zoals wij dat graag zien, dus zullen wij ook volgend jaar op deze manier verder gaan. Echter met een opmerking, we proberen de bladen, en daar hebben we jullie hulp bij nodig, meer op tijd uit te laten komen. Waarschijnlijk geldt dit voor 1994 nog meer dan voor 1993.

- Ook hebben we, zoals te zien is, de hoop op een advertentie nog steeds niet opgegeven.

- Vorig jaar vertelden we het ook al, maar de project leider verzekerde ons dat het volgende boekjaar een echt KGN68k jaar wordt. Het KGN68k project is dus duidelijk in een stadium gekomen waar we met z'n allen (denk ik) lang op gewacht hebben. Het is dan ook zeer waarschijnlijk dat er volgend jaar al meerdere gebruikers met het systeem aan de slag zijn.

- De donatie/Point opbrengsten hebben we dit jaar laten vervallen daar het BBS gebeuren op een nieuwe manier benaderd zal gaan worden. De KGN zal met het BBS The Ultimate een sponsor verdrag aangaan. De eigenaar van het BBS zal dus per jaar een eenmalig bedrag van de KGN ontvangen waarmee hij dan een deel van de kosten die hij maakt kan bekostigen. De KGN 'koopt' als het ware dus voor haar leden 'rechten' op het BBS.

Uw penningmeester, *Jacques Banser.*

PostScript deel 3

Na wat onnozele stoeipartijtjes met Post Script, nu een nuttiger toepassing. De versie van Ultipost, de plot generator van Ultiboard, waar ik mee werk ondersteunt niet de mogelijk om zogenaamde 'centerhole' in de Post Script file te zetten. De centerholes geven je een beginpunt bij het boren, waar je anders een centerpons voor gebruikt. Bij HPGL en Gerber output is het wel mogelijk om centerholes te laten generen, bij Post Script helaas niet. Maar daar gaan we wat aan doen.
In de file staat nogal een keer **newpath X-positie, Y-positie, Radius 0 360 arc fill stroke**.

Op zulke plaatsen wordt een massief zwart rondje neergezet. En binnen dat rondje willen wij weer een wit rondje zetten. Bekijk het fragment van de Post-Script file maar eens.
Eerst is er met een edit een find "0 360 arc fill" en replace with "round_pad" gedaan. De definitie van "round_pad" moet ergens vooraan in de source staan, daar waar het andere definities niet beinvloedt.

"Round_pad" maakt eerst een copie van de plaats en radius, tekent dan de originele ronde in het zwart, berekent de straal voor het witte rondje en tekent die dan op dezelfe plaats.

Geert Stappers

```
%before the patch:

} def %end of newcommand
/othernewcommand
%other definitions
% and Post Script code

 newpath
7613 3020 54 0 360 arc fill
 stroke
 newpath
7343 3020 54 0 360 arc fill
 stroke
 newpath
6263 3020 54 0 360 arc fill
 stroke
%----------------------
```

```
%after the patch:

} def %end of newcommand

/holesize 25 def % in procent

/round_pad {
3 copy %  X,Y and Radius
0 360 arc fill % the original pad
stroke % paint and preforms im-
plicitly a NEWPATH
holesize 100 div mul % calculate
radius of hole
1 setgray % white
0 360 arc fill % the hole
0 setgray % back to black
} def % round_pad

/othernewcommand
%other definitions
% and Post Script code

 newpath
7613 3020 54 round_pad
 stroke
 newpath
7343 3020 54 round_pad
 stroke
 newpath
6263 3020 54 round_pad
 stroke
```

Fig. 1: Postscript source voorbeelden

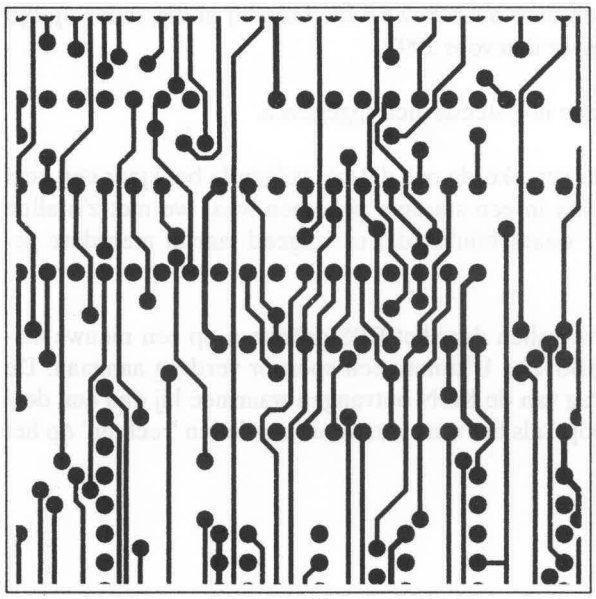


Fig. 2: het origineel

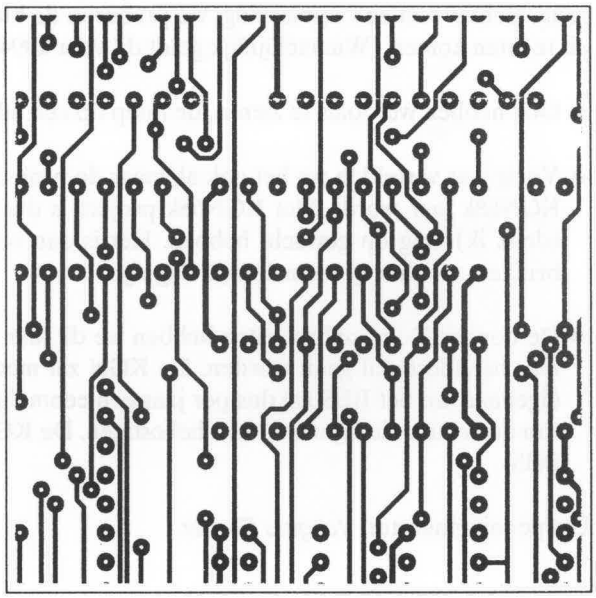


Fig. 3: na patch

Uw (financiële) huishouding in de computer

Inleiding

Als je zo om je heen vraagt of men een nuttige toepassing van de Personal Computer in het huishouden kan noemen, dan moet men meestal het antwoord schuldig blijven. Uiteraard is de computer zeer geschikt als een veredelde typemachine, mits je natuurlijk een printer hebt die het typewerk ook netjes af kan drukken. Verder kan de computer helpen bij invullen van het belasting-formulier en het is mogelijk de adressen van familie en kennissen er in op te slaan.

Vooral dat laatste zie ik niet echt zitten. Stel je voor dat je gebeld wordt door een clublid met de vraag of jij het telefoonnummer hebt van clublid X. Je moet dan eerst naar boven lopen, je computer opstarten, het programma waarmee je het adressenbestand kunt benaderen opstarten en pas daarna kun je het antwoord geven. Nee, geef mij maar een eenvoudig kaartenbakje naast de telefoon. Dat werkt in ieder geval veel sneller.

Hoewel ik nooit het nut van een computer in het huishouden in wilde zien, denk ik toch dat we iets gevonden hebben waarin de computer een rol kan spelen. We (mijn vrouw en ik) houden namelijk sinds twee jaar onze financiële reilen en zeilen bij in een spreadsheet programma en daarmee kun je goed zien waar je zuurverdiende geld elke maand weer blijft. Over deze toepassing gaat dit artikeltje.

Voordat ik verder ga met de beschrijving van hoe het allemaal werkt, denk ik dat het belangrijk is op te merken dat mijn vrouw Betty als boekhouder gewerkt heeft en dat ze het van nature in zich heeft om alle inkomsten en uitgaven zo goed mogelijk te registreren. Verder komen alle inkomsten in één pot waaruit alle gezins-uitgaven betaald worden. Voor persoonlijke uitgaven (cadeautjes, hobby's etc.) hebben we een bepaald bedrag per maand aan zakgeld. Wat we hier mee doen loopt niet via de administratie.

Spreadsheet-programma's

Voor onze toepassing wordt gebruik gemaakt van het spreadsheet deel van Microsoft Works. De keuze voor Works is eigenlijk een hele eenvoudige: dit pakket heb ik cadeau gekregen bij de aanschaf van een computer. Je kunt de toepassing ook maken met andere spreadsheet pakketten zoals Lotus 123, Excel, Symphony of zelfs een eenvoudig spreadsheet dat bijvoorbeeld bij een programmeertaal als voorbeeld geleverd wordt.

Voor die enkeling die nog niet weet wat een spreadsheet is, volgt hier nog even een hele korte uitleg. Een spreadsheet is een soort elektronisch rekenblad. Dit blad is opgebouwd uit cellen zoals in het voorbeeld in figuur 1. Die cellen worden aangeduid met hun coördinaten A1, A2, B1, B2 etc. In een dergelijke cel kun je tekst schrijven, getallen of een formule. De cellen met tekst kun je gebruiken om aan te geven wat er in een bepaalde kolom of op een bepaalde rij staat (bijvoorbeeld kolom C is de maand Mei en rij 13 is de post Hypotheek). Getallen kun je op meerdere manieren opslaan waaronder ook als valuta waarna er netjes een gulden-teken voor wordt gezet.

	1	2	3	4	5
A					som(A1:A4)
B					som(B1:B4)
C					som(C1:C4)
D					som(D1:D4)
E					som(E1:E4)
F	som(A1:E1)	som(A2:E2)	som(A3:E3)	som(A4:E4)	som(A5:E5)

Afb. 1: voorbeeld van een elektronisch rekenblad

Het leukste van een spreadsheet zijn de formules. Door in een cel een formule te zetten waarin verwijzingen naar andere cellen voorkomen, krijg je in een cel een getal dat afhankelijk is van de waarde van die andere cellen. Zo kun je bijvoorbeeld in cel C20 de formule SOM(C2:C19) zetten waarna in deze cel het totaal van de maandelijkse uitgaven voor de maand mei komt te staan. Verandert één van de cellen die in de formule voorkomt, dan verandert automatisch het getal in cel C20.

Nog een aardigheid van een spreadsheet is dat hij allerlei mogelijkheden heeft zaken te kopiëren. Hierbij worden automatisch de gebruikte referenties aangepast zodat in D20 automatisch komt te staan SOM(D2:D19) voor een kolom voor de maand Juni. Je kunt overigens ook aangeven dat hij toch dezelfde cel moet gebruiken, voor als je bijvoorbeeld een vast bedrag ergens hebt staan. Bij het spreadsheet dat ik gebruik, geef je dat aan door een dollar ($) teken voor het deel de coördinaat te zetten dat niet wijzigt. Een verwijzing naar een vaste cel C13 krijgt dat de coördinaat C13. Bedoel je wel een vaste rij, maar niet een vaste kolom, dan wordt het C$17

Op deze manier kun je met behulp van een spreadsheet heel snel verschillende alternatieven naast elkaar zetten en uit laten rekenen. Zo kun je bijvoorbeeld snel uit laten rekenen of je de auto beter in kunt ruilen tegen een Toyota Starlet bij Garage Kuipers of tegen een Opel Corsa bij Garage De Groot of dat het toch beter de Nissan Micra van Garage Peeters kan worden. Je zet alle kosten (open dakje, radio, afleveringskosten etc.) en opbrengsten (inruil) in een aantal rijen en de drie varianten in de kolommen. Vervolgens vul je de bedragen in waarna je precies kunt zien welke auto het best bij je portemonnee past. Hetzelfde kun ja natuurlijk ook nog eens doen met rente, afschrijving, onderhoud en benzineverbruik waarna je tot je schrik merkt dat een nieuwe auto je al snel fl. 1000,-- per maand kost.

De financiële registratie

In onze toepassing hebben we in de rijen een opsomming gemaakt van alle soorten uitgaven die we willen registreren. In de kolommen staan dan de maanden Januari t/m December en uiteraard een totaal-kolom voor het hele jaar. Per maand worden alle inkomsten en uitgaven ingevuld waarna aan het eind een bedrag komt te staan dat overeen moet komen met het saldo van de giro-rekening.

Nu kun je uiteraard alle afzonderlijke uitgaven gaan registreren maar dat bleek bij ons niet te werken. Daarom zijn er een aantal zaken bij elkaar geveegd. De vaste bedragen (abonnementen, hypotheek etc.) worden allemaal per giro betaald en daarvan doet de Postbank dus al een goede registratie. Deze bedragen worden gewoon van het afschrift over genomen. Huishoudgeld (eten drinken etc.) wordt voor een belangrijk deel contant betaald en verder met girobetaalkaart; alle andere uitgaven (kleding etc.) worden voornamelijk met een girobetaalkaart of met PIN-code betaald. Door het consequent invullen van het overzicht dat je bij je betaalkaarten krijgt en het bewaren van kassa-bonnetjes, kun je ook hier een goede registratie krijgen. Blijft eigenlijk alleen alles wat contant betaald wordt over. Uiteraard wordt weer door de Postbank vastgelegd wat er contant wordt opgenomen (uit de flappentap gepind). In principe is dit allemaal huishoudgeld. Worden er ook andere zaken contant betaald (cadeautjes, kleding, kapper etc.) dan wordt dit even apart genoteerd.

Waarna je dus een nauwkeurige vastlegging hebt van waar je geld gebleven is en hoeveel je hebt overgehouden (??!!).

Aan het eind van de maand worden alle gegevens verzameld en ingevoerd in de computer waarna je dus een nauwkeurige vastlegging hebt van waar je geld gebleven is en hoeveel je hebt overgehouden (??!!).

Planning

Nu is vastlegging één ding, maar je kunt nog een stap verder gaan. Van de vaste bedragen kun je een vrij goede schatting maken wanneer de rekening komt en hoe hoog hij ongeveer zal zijn. Het is nuttig die gegevens al vast in te vullen. Verder kun je, als je al een poosje een registratie hebt gevoerd, ook wel een goede schatting maken van de overige posten zoals kleding, schoeisel en zelfs huishoudgeld. Op die manier kun je al een soort financiële planning maken voor de komende maanden. Doe je dit goed (of zorg je toch voor een flinke post "onvoorzien"), dan kun je op elk moment vrij goed inschatten hoe je saldo er de komende tijd uit zal zien en of een bepaalde extra uitgave (nieuwe PC of iets dergelijks) verantwoord is. Je krijgt dan iets dat men in het bedrijfsleven een liquiditeitsplanning noemt.

Uiteraard is de betrouwbaarheid van de planning sterk afhankelijk van de betrouwbaarheid van de geschatte uitgaven. Het eerste jaar dat mijn vrouw (op papier toen nog) een dergelijk overzicht voor ons gezin gemaakt had, wist ze me te vertellen dat ik

zo'n fl. 500,-- netto per maand te weinig verdiende. Nu was het salaris voor dat jaar al vastgesteld zodat daar helaas niets aan te doen viel. Gelukkig bleek dat ze een aantal uitgaven veel te hoog had ingeschat zodat alles toch nog op z'n pootjes terecht kwam.

Vorig jaar waren de werkelijke uitgaven elke maand zo'n fl. 100,-- hoger dan de geschatte uitgaven zodat uiteindelijk de ruim duizend gulden die we aan het eind van het jaar over zouden houden ook op was.

Dit jaar loopt het allemaal veel beter. We hebben nu letterlijk voor alle uitgaven een schatting ingevuld die gebaseerd is op de uitgaven die we vorig jaar voor de betreffende post hadden. De afwijkingen tussen de werkelijke uitgaven en de geschatte uitgaven zijn iedere keer slechts een paar tientjes zodat we nu een echt goed inzicht hebben in hoe we financieel draaien.

Inkomsten
- Salaris
- Vakantiegeld
- Onkostenvergoeding
- Kinderbijslag
- Diversen (gratificaties etc.)

Uitgaven

Maandelijks:
- Hypotheek
- Leningen
- Verzekering Leven
- Verzekering Ongevallen
- Verzekering Ziektekosten
- Verzekering Auto
- Kerkelijke bijdrage
- Gas/Elektra
- Zakgeld Gert
- Zakgeld Betty
- Zakgeld kinderen
- Personeelsvereniging
- Huishoudgeld
- Totaal maandelijkse uitgaven

Per 2 maand:
- Telefoon
- Totaal per 2 maand

Per kwartaal:
- Abonnementen tijdschriften
- Abonnement Krant
- Motorrijtuigenbelasting
- Sport vereniging kinderen
- Centraal Antenne Systeem
- Betaalde rente "Rood" staan
- Totaal kwartaal uitgaven

Per half jaar:
- Abonnementen tijdschriften
- Omroepbijdrage
- Tandarts/Arts
- Totaal half jaarlijkse uitgaven

Jaarlijks:
- Lidmaatschappen
- Abonnementen
- Verzekering Opstal
- Verzekering Inboedel
- Bibliotheek
- Verontreinigingsheffing
- Riool en Reiniging
- Waterschap
- Hoogheemraadschap
- Onroerende Zaak Belasting
- Schoolfonds/Schoolvereniging
- Terugbetaling Studiebeurs
- Onderhoud CV ketel
- Kenteken deel 3
- Totaal jaarlijkse kosten

Diversen:
- Sparen
- Kleding en schoeisel
- Kapper
- Cadeautjes t.l.v. gezinsbudget
- Dierenarts
- Onderhoud auto
- Boeken, platen etc.
- Onderhoud Huis en Tuin
- Vakantie en uitstapjes
- Studiekosten
- Benzinekosten
- Totaal diverse uitgaven

Totalen
- Totaal inkomsten
- Totaal uitgaven
- Over/Tekort
- Cumulatieve inkomsten
- Cumulatieve uitgaven
- Saldo giro-rekening

Tabel 1: overzicht van inkomsten en uitgaven

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

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

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

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

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

Gert van Opbroek

DOS-65 uit de ijskast !

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

Een dosser die nog geen 65 is ...

Frank Vandekerkhove

Hang eens een muis aan je DOS65 (deel twee)

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

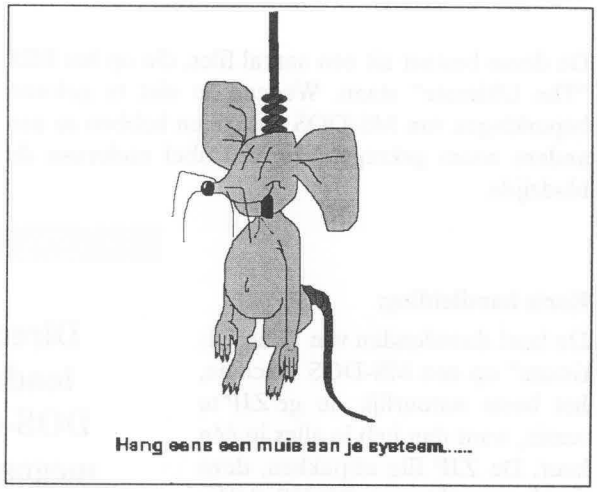
In dit deel een meer praktische toepassing van de muis data. De meest gebruikte toepassing van een muis, is het om de cursor op een gemakkelijke manier naar een willekeurige plaats op het scherm te bewegen. Zoals uit het vorige artikel al bleek stuurt de muis bij iedere beweging een pakketje data met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

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

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

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

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


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

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

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

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

Korte handleiding:

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

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

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

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

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

Antoine Megens

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

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

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

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

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

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

                opt     rc02

                lib     demo_map            ; memory map
                lib     mouse_def           ; mouse definitions

                org     prgrom

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

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

Fig. 3: demo_main.mac

```
                rts

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

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

                end     start
```

Fig. 3: demo_main.mac

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

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

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

Fig. 4: demo_misc.mac

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

Fig. 4: demo_misc.mac

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

Fig. 4: demo_misc.mac

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

Fig. 4: demo_misc.mac

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

; addresses for program development on DOS65

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

Fig. 5: demo_map.mac

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

Fig. 5: demo_map.mac

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

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

XQUIT           equ     17                      mouse locations of QUIT box
YQUIT           equ     8

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

Fig. 6: do_demo.mac

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

Fig. 6: do_demo.mac

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

Fig. 6: do_demo.mac

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

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

Fig. 6: do_demo.mac

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

Fig. 7: mouse_demo.mac

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

Fig. 7: mouse_demo.mac

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

Fig. 7: mouse_demo.mac

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

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

Fig. 7: mouse_demo.mac

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

Fig. 8: mouse_def.mac

```
; Zero page
;
mousfun         equ     mouszpg         pointer to mouse function
buttons         equ     mouszpg+2       mouse button data & status reg.
;
;                       b7 = function clicked bit
;                       b6 = escape clicked bit
;                       b5 = middle button clicked bit (special functions)
;                       b2 = actual left button
;                       b1 = actual middle button
;                       b0 = actual right button
;
; System RAM
;
mousex          equ     mousram         X location of mouse pointer
mousey          equ     mousex+1        Y location of mouse pointer
accu            equ     mousex+2        additional accu
deltax          equ     mousex+3        8 bit signed offset in X direction
deltay          equ     mousex+4        8 bit signed offset in Y direction
mxmin           equ     mousex+5        mouse range X minimum
mymin           equ     mousex+6        mouse range Y minimum
mxmax           equ     mousex+7        mouse range X maximum
mymax           equ     mousex+8        mouse range Y maximum
mbptr           equ     mousex+9        mouse package pointer
mpack           equ     mousex+10       mouse package buffer, 4 bytes!
```

Fig. 8: mouse_def.mac

Derde les C

Oei! Het zetduiveltje heeft de vorige les hard toegeslagen! Overal waar je in de vorige les ziet staan: &lbrk. (ook de punt aan het eind) moet je een linker - rechte haak denken en overal waar &rbrk. staat een rechter - rechte haak. Dus respectievelijk een [en een].

Daar blijft het niet bij! De #include statements zijn helaas ook in de vernieling gegaan; natuurlijk moet daar staan: **#include <stdio.h>**

Nog iets - maar dat is geen fout van de zetduivel - in het vierde voorbeeld wordt gebruik gemaakt van getchar() en ik vertel daarbij dat deze functie wacht totdat ENTER is gegeven. Bij het uitvoeren van het voorbeeld zal je zien dat het programma wel direkt doorgaat. Dat wordt veroorzaakt door het putchar(letter) statement. Het euvel is te verhelpen door het putchar() statement te vervangen door het printf() statement.

Vandaag behandelen wij loops. Je maakt kennis met FOR, WHILE en SWITCH.

FOR loop

De **FOR** loop bestaat uit 3 onderdelen, altijd door een punt-komma gescheiden. Het eerste onderdeel stelt een beginwaarde van een of meer tellers en/of variabelen in, het tweede onderdeel beschrijft de konditie en het laatste onderdeel verhoogt een teller. Deze laatste teller is dezelfde teller als de teller in de konditie.

Het FOR statement kan er als volgt uit zien:

```
for ( teller = 0; teller < 10; teller++ )
```

De drie expressions staan totaal tussen haakjes.

Uit les 2 weet je dat 'teller++' hetzelfde is als teller = teller + 1

Een voorbeeld:

```
main()
{
    int tel, totaal;
    for ( tel = 0, totaal = 0; tel < 10; tel++ )
    {
      totaal += tel;
      printf("tel = %d, totaal = %d\n", tel, totaal);
    }
}
```

Opmerkingen:

- totaal += tel is hetzelfde als: totaal = totaal + tel
- De eerste expressie bevat twee intialisaties, gescheiden door een komma.
- Assignen van de variabelen vindt nu plaats in de loop.

Oefening

Maak een programma dat de ASCII tabel toont met het bijbehorende volgnummer.

Oplossing:

```
main()
{
  int n;
  for ( n = 1; n < 256; n++ )
  {
    printf("%3d = %c\t", n, n);
  }
}
```

Een ASCII tabel is soms erg handig! Maar terzake; de \t kennen wij nog uit de eerste les als een tab-karakter.

Het printf() statement toont eerst het cyfer (%3d) en daarna het bijbehorende karakter (%c). 'C' toont de inhoud van de variabele n en vertaalt blijkbaar dit getal door het karakter met %c....

Het cijfer dat getoond wordt correspondeert met het volgnummer in de ASCII tabel. Dat betekent dus dat 'C' altijd eerst de ASCII tabel raadpleegt om het bijbehorende karakter te kunnen vinden. Door de ene keer d.m.v. de format specifier (zie les 1) aan te geven dat het volgnummer moet worden getoond en de tweede keer het karakter dat bij dit nummer hoort, krijg je de ASCII tabel op het scherm te zien (printf("%3d = %c\t", n, n).

De FOR loop mag natuurlijk ook worden genest. Bekijk het volgende voorbeeld maar:

```
main()
{
  int kol, rij;
  for ( rij = 1; rij < = 22; rij++ )
  {
    for ( kol = 1; kol < = 40; kol++ )
    printf("%\xDB");
  printf("\n");
  }
}
```

Dit programma vult het hele scherm met blokjes en wordt daardoor helemaal wit.

Duidelijk is te zien dat ten gevolge van de eerste loop, alles wordt doorlopen vanaf de { onder de eerste loop tot aan de } onder de tweede printf(). De nested loop doorloopt slechts 1 statement; de eerste printf(). De tweede printf() hoort bij de hoofdloop!

Denk daar eens goed over na... Het betekent dat de nested loop (ook wel 'inner loop' genoemd), telkens 40x wordt uitgevoerd waarbij telkens een blokje op het scherm wordt gezet. Aangezien er geen \n in de printf() is opgenomen, komen deze blokjes naast elkaar te staan en vormen zo een regel.
Elke nieuwe doorloop van de hoofd- of outer loop veroorzaakt 40x het uitvoeren van de inner loop.

Oefening

Schrijf een programma dat een matrix op het scherm toont. De horizontale as (bovenste regel) loopt van 1 t/m 12; de vertikale as (links) doet hetzelfde. Elk veld van de matrix moet de som opleveren van de vertikale velden samen.

```
 1  2  3  4  5  6  7  8  9 10 11 12
 2  4  6  8 10 12 14 16 18 20 22 24
 3  6  9 12 15 18 21 24 27 30 ... ...
 6 12 18 24 ... ...
12 ...
...
```

WHILE loop

WHILE kan vaak **FOR** vervangen. Eigenlijk mag je zeggen dat **FOR** betekent: TOTDAT en dat **WHILE** betekent: NET ZO LANG ALS.

Een voorbeeld:

```
main()
{
  int tel = 0;
  int totaal = 0;
  while ( tel < 10 )
  {
    totaal + = tel;
    printf("tel = %d, totaal = %d\n", tel + +, totaal);      }
}
```

Dit voorbeeld ben je al tegengekomen bij de behandeling van FOR. Nu echter is in het voorbeeld gebruik gemaakt van de WHILE. Aangezien het WHILE statement geen voorzieningen heeft voor het assignen van variabelen zoals bij de FOR moeten de variabelen vooraf worden assigned.

Nog enkele bezienswaardigheden:

- het verhogen van de teller is opgenomen in printf()
- het verhogen van 'tel' gebeurt NA printf(). Dat komt omdat er staat: 'tel + +'. Als er had gestaan: + + **tel** dan wordt de teller eerst verhoogd en dan op het scherm getoond.
- alles tussen de { } van de WHILE loop wordt de **Structure van de loop** genoemd. Deze benaming geldt natuurlijk ook bij andere soorten loops.

Oefening

Tik bovenstaand voorbeeld in en maak daarvan 2 versies. Een versie met tel + + en een andere versie met + + tel. Bekijk het effekt m.b.v. codeview waarbij je de inhoud van tel op de voet volgt.

Eeuwige loop

Vrijwel elk statement in 'C' geeft een Result Code terug vergelijkbaar met de '1' of '0' bij een 'IF' statement.

Zolang de konditie welke getest wordt in de 'WHILE', waar is, wordt er een '1' teruggeven. Dus in bovenstaand voorbeeld wordt een Result Code afgegeven zolang tel kleiner is dan 10.

Hiervan kunt je gebruik maken door de konstruktie:

```
while(1)
......
......
```

en maak je hiermee een eeuwige loop. Een dergelijke loop wordt afgebroken door het intikken van:

CTRL + BREAK toetsen

Return toets

In les 1 heb ik een overzicht gegeven van de 'escape' characters.

Wij maken nu een programma dat het aantal ingetikte letters en spaties telt:

```
main()
{
 int tel = 0;
 printf("Type een woord of zin in: \n");
 while ( getche() != '\r' )
  tel++;
  printf("\nAantal letters en spaties is: %d", tel);    }
```

- let op de konstruktie binnen while() van getche(); de inhoud van getche() wordt tegelijk vergeleken met '\r'
- != betekent: ongelijk aan (les 2)
- '\r' is het escape character voor de return toets (les 1)
- de 'structure voor de loop' bestaat uit 1 statement en daarom worden de { } weggelaten.
- in dit voorbeeld worden spaties ook geteld.

Oefening

Schrijf een programma dat een ingetikte letter test. Alleen de letters 'a' t/m 'd' worden beschouwd als goed; alle andere letters als fout.
Indien fout moet het programma blijven loopen. Indien goed, stoppen met een toepasselijke boodschap.

DO - WHILE loop

Normaal gesproken staat WHILE() aan het begin van een loop. Het mag echter ook staan aan het eind van een loop:

```
main()
{
 int tel = 0;
 int totaal = 0;
 do
  {
  totaal + = tel;
  printf("tel = %d, totaal = %d\n", tel++, totaal);        }
 while ( tel < 10 );
}
```

Om 'C' te laten weten dat er een loop moet worden uitgevoerd zonder verder nog te weten waarom het gaat, moet de loop begonnen worden met **DO**.
Daarna volgt de 'structure' van de loop, opgesloten tussen { } omdat er meerdere statements moeten worden uitgevoerd en dan pas komt de WHILE().

Wat denk je? Zal dit programma dezelfde output geven als het programma waarbij de WHILE() aan het begin staat?

Samenvoegen / nesten

In het laatste voorbeeld wordt in while de inhoud van 'tel' vergeleken met het getal '10'. Stel dat u een ingetikt karakter wilt testen. Dat kan worden geschreven op verschillende manieren:

```
do
ch = getche();
.......
while ( ch != 'a' )
 ..........
```

op deze wijze geschreven kan alleen met een DO - WHILE loop (waarom? beredeneer!). Met while aan het begin van de loop kan ook deze konstruktie worden toegepast:

```
.....
while ( ( ch = getche() ) != 'a' )
 .....
```

of vereenvoudigd:

```
.....
while ( getche() != 'a' )
 .....
```

Je mag in een while() loop met while() aan het begin NOOIT schrijven:

```
ch = getche();
while ( ch != 'a' )
 .....
```

waarom werkt dit niet?

Else - if

In de tweede les heb je kennis gemaakt met de 'IF - ELSE' konstruktie. Nu maken wij kennis met de omgekeerde schrijfwijze:

```
.....
if ( op == '*' )
  .....
else
  if ( op == '/' )
   .....
```

Dit mag ook zo worden geschreven:

```
.....
if ( op == '*' )
  .....
else if ( op == '/' )
 .....
```

ELSE - IF is niets anders dan een samenvoeging van twee statements. Het mag niet worden toegepast bij andere statements b.v: **else printf(.....)**.

Break

Een loop kan worden afgebroken, afhankelijk van kondities. Daarvoor gebruik je de break.

Een voorbeeld:

```
main()
{
 int tel = 0;
 char ch;
 while ( tel < 10 )
  {
  printf("Raad de juiste letter:\n");
  if ( ( ch = getche() ) == 'h' )
    break;
  else if ........
```

```
            tel++;
          }
        }
```

- de speler krijgt 10 kansen om de juiste letter te raden
- als de juiste letter is geraden ('h'), wordt de loop meteen verlaten. Is het een andere letter, dan wordt de teller verhoogd en krijgt speler een nieuwe kans.

Let bij het gebruik van break op het feit dat **ALLEEN** de loop wordt verlaten waarvan de break onderdeel uitmaakt. Bij nested loops, waarbij break is opgenomen in de inner loop, wordt dus alleen de inner loop verlaten.

Switch

Het mooiste heb ik voor het laatst bewaard! **SWITCH** is het mooiste statement om een aantal kondities te testen en vervangt daarmee een aantal 'IF - ELSE' statements:

```
main()
{
  float num1, num2;
  char op;
  while ( 1 )
   {
   printf("Type getal, operator, getal" );
   scanf("%f %c %f", &num1, &op, &num2):
   switch ( op )
    {
    case '+':
      printf(" = %f", num1 + num2);
      break;
    case '-':
      printf(" = %f", num1 - num2);
      break;
    case '*':
      printf(" = %f", num1 * num2);
      break;
    case '/':
      printf(" = %f", num1 / num2);
      break;
    default:
      printf("Verkeerde berekening ");
    }
   }
}
```

(dit is een mogelijk begin voor een duur zak-japannertje)

De syntax is altijd:

```
switch()
{
case ... :
...
break;
case ... :
...
break;
default :
}
```

- switch bevat de te testen variabele
- na switch openen met {
- case test de inhoud van de variabele uit switch
- default is VERPLICHT en dient voor afsluiten van cases
- je mag net zo veel cases benoemen als nodig maar ALTIJD EEN (1) default
- de break voorkomt dat de resterende statements ook worden uitgevoerd nadat aan een case is voldaan. De rest van de case wordt niet onderzocht. Nu je dit weet begrijp je ook dat de situatie(s) die het meest voorkomen, als eerste 'cases' in de switch worden geplaatst. Dat gaat sneller!
- case en default afsluiten met :
- het geheel afsluiten met }

Nog enkele zaken om te onthouden:

- tussen CASE en BREAK { } gebruiken als er meerdere statements moeten worden uitgevoerd die beginnen met IF, FOR of SWITCH.
  De structure welke dan ontstaat behoort niet tot de (hoofd) switch maar behoort tot de IF, FOR of nieuwe SWITCH!
- als er geen uitvoerend statement is voor DEFAULT, dan afsluiten met een ;

```
default:
;       < = = = wordt NOP genoemd
```

Aanvulling op het switch() statement.

'C' bouwt jumps vanuit de switch() naar de verschillende cases. Elke case vormt een label. De default bevat bevat een verwijzing naar het adres waarop het eerstvolgende 'normale' statement weer begint, dus het eerste statement na de afsluitende } van de switch(). De default moet altijd staan na de switch() maar mag geplaatst worden voor de eerste case.
De break's bevatten een verwijzing naar de plaats waar de default staat. Als een break wordt weggelaten, gaat 'C' verder met de rest van de cases. (behalve natuurlijk bij de laatste).

Volg dit m.b.v. CODEVIEW in assembler mode, vooropgesteld dat de kennis van assembler aanwezig is.

Oefeningen

Maak eerst uw keuze uit enkele oefeningen en/of voorbeelden uit dit hoofdstuk. Schrijf daarna het volgende programma:

Maak een spelletje waarbij de speler binnen een vastgesteld aantal pogingen een getal moet raden tussen 1 en 100. Sommige getallen echter zijn besmet verklaard.
Raadt iemand een besmet getal, dan verschijnt er een toepasselijke boodschap (wie bedenkt de leukste?) op het scherm en wordt het programma afgebroken.

Na elke raadpoging moet een boodschap verschijnen die verteld of het getal te hoog of te laag is.

Aan de slag! En pas alles toe wat je tot nu toe hebt geleerd.

Hans van Boheemen

Regelen met behulp van wazige logica

Inleiding

Na de uitvinding van de micro-processor werd deze bouwsteen op diverse plaatsen ingezet in de procesbesturing. Nu bleek al snel dat een apparaat die alleen 0 en 1 of "waar" en "niet waar" of "aan" en "uit" kent ook zijn beperkingen heeft. Volgens de uitdrukking kan iemand niet een beetje zwanger zijn; je bent zwanger of niet dus "waar" of "niet waar". Iets anders is het met het begrip temperatuur. Als je de temperatuur in een woonkamer bedoelt, dan heb je te maken met koud, warm, een beetje koud, niet al te warm etc. kortom tussen koud en warm ligt een heel gebied waar het niet echt duidelijk is of iets warm of koud is.

Hetzelfde vindt je bij het klassieke voorbeeld van de leeftijd van een persoon. Waar ligt de grens tussen "jong" en "oud"? Stel dat we zeggen op 60 jarige leeftijd, wordt iemand dan op zijn zestigste verjaardag meteen van jong oud? Natuurlijk niet. Dat zie je bijvoorbeeld aan de invoering van de begrippen "middelbare leeftijd" en "oudere jongeren".

Volgens de uitdrukking kan iemand niet een beetje zwanger zijn; je bent zwanger of niet .

Om deze problemen aan te kunnen pakken is er rond 1965 door de wiskundige professor Lotfi A. Zadeh van de Californische universiteit in Berkeley de zogenaamde "Fuzzy Logic" of zige logica ontwikkeld. In Japan heeft men vervolgens deze ideeën verder ontwikkeld om er allerlei processen met te gaan sturen. Daarvoor zijn tegenwoordig ook zogenaamde "Fuzzy processors" beschikbaar die je onder andere terugvindt in videocamera's, magnetrons en zelfs al hier in daar in voertuigen zoals metro's etc. Reden genoeg om er eens wat uitgebreider naar te gaan kijken.

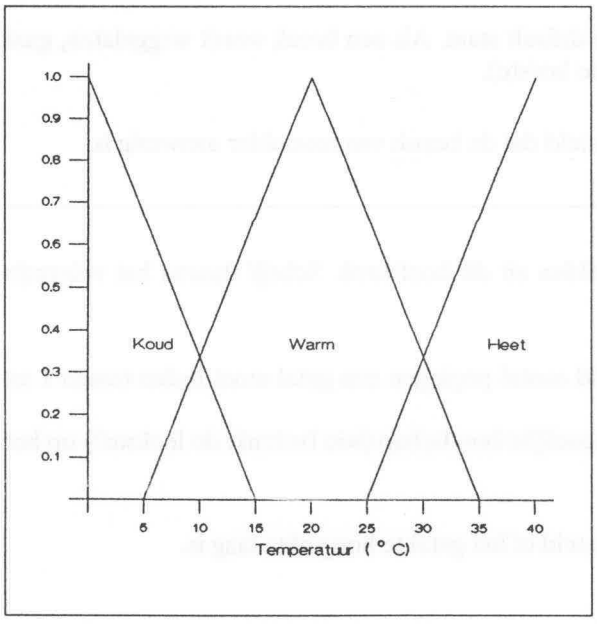


Fig. 1: temperatuur opgesplitst in "koud", "warm" en "heet"

Uitgangspunten van de wazige logica

De belangrijkste eigenschap van wazige logica is het feit dat iets niet helemaal tot een bepaalde verzameling (groep, klasse) hoeft te behoren. Zo kun je zeggen dat iemand voor 95% tot de groep ouderen en voor 5% tot de groep mensen op middelbare leeftijd behoort. In figuur 1 is dat voor het begrip "Temperatuur" schematisch weergegeven. We praten over drie klassen, "Koud", "Warm" en "Heet". Is de temperatuur 0 graden of lager, dan is het voor 100% koud. Bij 20 graden spreken we over 100% warm en bij 40 graden of hoger, dan is het 100% heet. In figuur 1 loopt het gebied "Warm" van 5 graden tot 35 graden. Bij 7,5 graden hebben we nu te maken met 50% "Koud" en 18% "Warm" (half-koud of een beetje warm dus). Juist met deze onscherpe of wazige grenzen kan de Fuzzy Logic goed omgaan.

Een ander voorbeeld is een auto die een file nadert of in een file rijdt. Hier neem je drie grootheden die een verband met elkaar hebben. In de eerste plaats is dat de snelheid. Hiervoor kiezen we de klassen "Langzaam", "Gemiddeld" en "Snel". In de tweede plaats de afstand tot de voorligger: "Klein", "Gemiddeld" en "Groot". En tenslotte de remkracht: "Zacht", "Gemiddeld" en "Hard". Nu kun je regels opstellen die als volgt luiden:

1: Als de afstand "Klein" is en de snelheid "Groot", dan moet je "Hard" remmen.
2: Is de afstand "Gemiddeld" en de snelheid "Groot", dan moet je "Gemiddeld" remmen.

Als je nu aan een regelsysteem die met Fuzzy Logic werkt vertelt hoe de verdeling van de afstand, de snelheid en de remkracht over de drie klassen is, dan kan dit systeem je vertellen hoe hard er ongeveer geremd moet worden. Rijdt je bijvoorbeeld voor 60% "Gemiddeld" en voor 40% "Hard" en is de afstand voor 20% "Klein" en voor 80% "Gemiddeld", dan zou de uitkomst voor de remkracht, voor

w
a

40% "Gemiddeld" en voor 20% "Hard" kunnen zijn met als uitkomst een remkracht van 51% (Zie het voorbeeld in mc 11,1993). Als alles goed ingesteld is, dan zou je de remkracht van een auto op deze manier over kunnen laten aan een Fuzzy processor. Het resultaat is nog wel iets afhankelijk van hoe je de twee bovengenoemde regels met elkaar mengt. In ons voorbeeld zeggen we dat regel 1 voor 20% waar is en regel 2 voor 40%; hierbij is het waarheidsgetal van een regel het minimum van de getallen voor de ingangs-grootheden "afstand" en "snelheid". Kiezen we als waarheidsgetal het maximum van de grootheden, dan is de uitkomst in ons voorbeeld 53% remkracht.

Op deze manier kun je op eenvoudige wijze aangeven hoe uit de ingangs-grootheden (snelheid en afstand) de uitgangs-grootheid berekend moet worden. Je geeft aan je Fuzzy systeem op hoe de indeling in klassen is voor de diverse grootheden waarbij je ook de vorm (in ons voorbeeld driehoekig, maar er zijn ook andere vormen mogelijk) opgeeft. Hierbij moet je er altijd voor zorgen dat de klassen (Fuzzy verzamelingen of sets) op elkaar aansluiten of elkaar overlappen. Vervolgens geef je de Fuzzy regels op hoe de uitgangs-grootheid afhangt van de ingangs-grootheid en de processor doet de rest.

Tijdens het optrekken en remmen van de trein wordt er geen water uit het aquarium gemorst.

Enkele toepassingen

De meest opvallende toepassing van Fuzzy Logic vind ik de nieuwste generatie video-camera's. Omdat er meestal vanuit de hand gewerkt wordt, moet je bij het filmen (?) een vaste hand hebben, anders krijg je een schokkend of bibberig beeld. Met behulp van Fuzzy Logic zijn de moderne camera's in staat het schokkerige of bibberige weg te halen. Hoe dat allemaal precies in zijn werk gaat, voert me nu veel te ver, maar de opgave aan de Fuzzy processor is niet gering. De processor moet namelijk het onderscheid kunnen maken tussen een beeld dat uit zich zelf beweegt, een camera die moedwillig met min of meer constante snelheid wordt bewogen en een camera die ongewild een beetje wordt heen- en weer bewogen.

Een tweede toepassing is in 1987 in de metro van Sendai (Japan) in bedrijf genomen. Hier regelt een Fuzzy systeem de snelheid van de trein. Hiermee worden tijdens het optrekken en afremmen schokken voorkomen. Om dit te demonstreren wordt er in een reclame-spotje een man getoond die zittend in de trein een met water gevuld aquarium op zijn bovenbenen heeft staan. Tijdens het optrekken en remmen van de trein wordt er geen water uit het aquarium gemorst.

Een derde toepassing van Fuzzy besturing staat in mc 3 van 1991 (Das Fuzzy-Mobil). Dit is een simulatie in TurboPascal 5.5 waarbij een autootje op een helling staat. Op dit autootje werkt de zwaartekracht, een variabele storing ("Ruis") en uiteraard de kracht van de motor. Nu is het de bedoeling dat de auto stil blijft staan. Dit kan de bestuurder (degene achter de PC) zelf doen, of hij kan de Fuzzy besturing inschakelen. Ik heb het nog niet geprobeerd, maar het lijkt me een leuk experiment.

Verder ben ik er zeker van dat er binnenkort nog heel veel toepassing van Fuzzy Logic bij zullen komen. Met name in de meet- en regeltechniek, procesbesturing, simulatie, patroonherkenning en zelfs kunstmatige intelligentie is "Wazige Logica" goed in te zetten. Je zult er dus nog heel veel over horen en/of lezen.

Literatuur:

Er is onderhand al heel veel over Fuzzy geschreven. Informatie die ik onder handbereik heb staat in:

1) Dr. Thomas Wolf: Fuzzy, die Revolution aus japanische High-Tech Tempeln; mc 3, 1991 (Overzichts-verhaal).
2) Dr. Thomas Wolf: Das Fuzzy-Mobil, Steuern met Fuzzy; mc 3, 1991 (simulatie van Fuzzy in TP 5.5).
3) Rolf-Dieter Klein/Alexander Schinner: Das mc-Fuzzy-Lab; mc 9, 1991 t/m mc 1, 1992 (serie over een Fuzzy insteekkaart voor PC).
4) Thomas A.W. Tilli: Fuzzy-die Lehre vom Unscharfen; mc 11, 1993 (begin van een nieuwe serie; theoretisch verhaal).

Gert van Opbroek

Voortgang KGN68k

Een ander?

Inderdaad, een ander. Deze voortgang wordt niet door Geert, maar door de (ex-) redacteur van de μP Kenner geschreven. Zo gek is dat op zich niet, want officieel is één van de vele petten die ik namens de KGN bezit die van secretaris van de werkgroep KGN68k. Nu wil het geval dat Geert het momenteel zeer druk op zijn werk heeft en mede daardoor geen tijd heeft om op een fatsoenlijke manier een rapportage naar de leden te doen. Vandaar dat ik die taak maar even overneem.

Samenstelling

Er is (al weer) een mutatie in de samenstelling van de werkgroep. Hugo kreeg de gelegenheid op een verkiesbare plaats op de lijst voor de gemeenteraad te komen. Hij vreest nu dat hij de komende tijd veel tijd kwijt is voor de verkiezing en heeft daarom aangegeven dat hij de werkzaamheden voor KGN68k een poosje op een laag pitje moet zetten.

Verder heeft Wim van Asperen zich aangemeld als lid van de werkgroep met als speciaal interessegebied de software. Dat is een goede zaak, want we kunnen zeker nog wel wat versterking in de software-hoek gebruiken. Zelf denk ik ook wat meer tijd gekregen te hebben doordat ik nu 's avonds om 17:45 al thuis ben en vrijdags om 15:45 en doordat Nico de redactie overneemt.

Ik denk dat we binnen de werkgroep de taken opnieuw gaan verdelen, maar daarover zullen jullie de volgende keer ingelicht worden.

Ledenvergadering

Zoals bekend, is het voortbestaan van de werkgroep agendapunt geweest op de afgelopen ledenvergadering. Ik vind dat een goede zaak. Het is waar dat de werkgroep zich stelselmatig niet aan afgesproken planningen heeft kunnen houden. Uiteraard zijn daarvoor wel redenen aan te geven, maar het heeft nu geen zin daar verder op in te gaan. Wel vinden we het als werkgroep zeer positief dat de leden van mening zijn dat de werkgroep door mag gaan met het project. Ook de randvoorwaarde dat het project op de HCC-dagen "af" moet zijn kun je positief zien. We hebben nu een hele duidelijke richtlijn en weten wat er gebeurt als we volgend jaar geen werkend systeem hebben. We denken dat de afgesproken termijn haalbaar is en zijn van plan er nu alles aan te doen de leden niet teleur te stellen.

Toch problemen

De voortgang van het project is al geruime tijd afhankelijk van het prototype van de print voor de Disk I/O (Dombo). Het probleem met deze print is dat ze eigenlijk ontworpen is voor een professionele print-produktie. Dat betekent dat de soldeer-eilandjes een vrij kleine afmeting hebben. We hebben nu enkele keren een prototype laten maken die we zelf moesten boren en juist daar ontstaat het probleem: bij het boren wordt de print dermate zwaar beschadigd dat het doormetalliseren en bestukken grote problemen heeft. Ga je dan de eilanden groter maken, dan wordt het wel beter, maar nog steeds blijkt het boren een hachelijke onderneming, ook omdat er geen gaatjes in de eilandjes zitten om de boor niet uit te laten glijden. Al met al blijkt het maar steeds niet te lukken een prototype van Dombo op te bouwen en die hebben we echt hard nodig om met de hardware verder te komen.

Nog steeds blijkt het boren een hachelijke onderneming,

Software

Met de software gaat het beter. De KGN68k assembler kan nu ook GNU-objects genereren vanuit source in Motorola notatie. Deze objects kunnen vervolgens meegelinkt worden met delen die in GNU C geschreven worden. Verder hebben we de benodigde sources van Linux voor een 68020 (Amiga) omgeving binnen en we zijn bezig te onderzoeken waar we wijzigingen aan moeten brengen om het systeem draaiende te krijgen op KGN68k.

Voortgang

Op vrijdag 5 oktober komt de werkgroep weer bij elkaar. Dan zullen we de status opnemen en ook een plan maken om zo snel mogelijk de problemen die we hebben op te lossen. Verder zullen we ook een nieuwe planning maken op basis waarvan we de komende tijd met het project verder zullen gaan. We zijn van plan met z'n allen de schouders er onder te zetten om uiterlijk op de HCC-dagen 1994 het systeem produktie-gereed te hebben. Verder staan wij uiteraard altijd open voor (opbouwende) kritiek; dus als je denkt een positieve bijdrage aan het project te kunnen leveren, neem dan contact met Geert of mij op. Namens de werkgroep KGN68k,

Gert van Opbroek

Een GNU-adventure voor KGN68k

Inleiding

Ik denk niet dat de lezers uit de titel af kunnen leiden waar dit artikel over gaat. Welnu, ik wil het in dit artikeltje gaan hebben over een stukje software-ontwikkeling dat ik voor KGN68k heb uitgevoerd. Gedurende deze ontwikkeling had ik bijna voortdurend het gevoel dat ik bezig was met het spelen van een adventure. Soms maakte ik snelle vorderingen, maar geregeld was ik meerdere avonden bezig en kwam maar heel langzaam verder.

Waar gaat het allemaal om? Welnu, zoals bekend, ben ik bezig met het ontwikkelen van een assembler voor KGN68k. Deze assembler moet source-code in Motorola 68030 assembler omzetten in iets dat op een KGN68k kan lopen. De eerste versies van deze assembler produceren een output in zogenaamde S-records, een schrijfwijze die onder andere gebruikt wordt voor EPROM-programmers. Omdat andere leden van de werkgroep al bezig zijn met het ontwikkelen van software in GNU C en GNU 68020 assembler, leek het ons binnen de werkgroep een goed idee als uitvoer uit mijn assembler samen kan werken met bijvoorbeeld uitvoer uit de GNU C compiler. Dit houdt in dat de KGN68k assembler dan een zogenaamde object-file moet kunnen genereren die vervolgens door de GNU linker verwerkt kan worden.

Hoe dat allemaal in zijn werk gaat, wordt in dit artikel beschreven.

Compiler, Assembler, Linker

Een aantal leden van de club kunnen programmeren in een hogere programmeertaal zoals Pascal, Fortran of C. Voor KGN68k komt zeker een zogenaamde C compiler beschikbaar met behulp waarvan een C programma (C source) omgezet kan worden in een stuk code dat door de 68030 processor begrepen wordt. Dit zal de GNU C compiler zijn omdat dit gereedschap een aantal voordelen heeft die hem uitermate geschikt maken voor KGN68k.

Het eerste voordeel is dat de compiler zeer compleet is. Hij ondersteunt de standaard ANSI C en bij mijn weten ook de uitbreiding op ANSI C die in de wandelgangen C++ heet. Verder genereert de compiler zeer snelle code die helemaal is toegesneden op een machine met een breedte van 32 bits (dus 68020, 80386 of hoger). De compiler vraagt wel een omgeving die een 32 bits machine waardig is zoals veel geheugen (meerdere Megabytes) en een redelijk forse harddisk. Ook deze zaken zijn op een "standaard" KGN68k voorhanden.

Het derde een waarschijnlijk meest belangrijke voordeel is dat de GNU compiler Public Domain software is. Ja, in Public Domain zijn óók goede produkten te vinden.... Dit betekent dat we zonder licentie-rechten te betalen van GNU gebruik kunnen maken. Verder is ook de source van de compiler en alles wat daarbij komt beschikbaar.

Hoe gaat de GNU compiler nu te werk?

Hoe gaat de GNU compiler nu te werk? Als voorbeeld gebruik ik hèt standaard C programma dat ieder C cursist wel eens is tegen gekomen. Dit is het Hello World programma dat in het oerboek Kerningan en Ritchie: The C Programming Language staat. Het programma staat afgedrukt in figuur 1.

Deze versie van het programma was één van de eerste test-programma's van KGN68k. De oorspronkelijke versie bevat de regel #include <stdio.h> en roept in plaats van "serial_put" de functie "printf" aan. Omdat printf nog niet werkt op KGN68k, heeft Pieter de Visser een routine gemaakt waarmee we via een seriële bouwsteen een teken uit kunnen geven. Dit is de routine "serial_put".

Nadat de compiler zijn werk gedaan heeft, ontstaat er een file met daarin de invoer voor de GNU assembler. Helaas maakt de GNU assembler gebruik van een ander formaat dan de Motorola standaard waardoor de assembler source voor 68000-mensen wat moeilijk te lezen valt, zoals te zien is in figuur 2.

```
#include "serial.h"

int main (void)
    {
        const char  *str = "Hello, world\n",
        *p;
        for (p = str; *p; p++)
        serial_put (*p);
        return 0;
    }
```

Fig. 1: KGN68k schrijft Hello World

```
#NO_APP
gcc2_compiled.:
.text
LC0:
        .ascii "Hello, world\12\0"
        .even
.globl _main
_main:
        link a6,#-8
        jbsr ___main
        movel #LC0,a6@(-4)
        movel a6@(-4),a6@(-8)
L2:
        movel a6@(-8),a0
        tstb a0@
        jeq L3
        movel a6@(-8),a0
        moveb a0@,d0
        extbl d0
        movel d0,sp@-
        jbsr _serial_put
        addqw #4,sp
L4:
        addql #1,a6@(-8)
        jra L2
L3:
        clrl d0
        jra L1
L1:
        unlk a6
        rts
```

Fig. 2: Hello World in GNU assembler voor 68030

IIn figuur 3 (zie volgende bladzijde) is de code uit figuur 2 omgezet in een iets leesbaarder formaat. Het betreft hier een source-file voor de KGN68k assembler.

Nadat de assembler zijn werk heeft gedaan, ontstaat er een file die we "object" zullen noemen. Dit is een beschrijving van de geassembleerde source. Hierin zitten dus onder andere de instructies in machinetaal. Omdat er in de assembler source enkele zogenaamde externe referenties staan, is deze file niet compleet. De code die bijvoorbeeld in de functie _serial_put staat, zit niet in de file. Daarom bevat de object ook een beschrijving van alles wat niet in de source-file voorkomt.

De object-file wordt, eventueel tezamen met andere source-files, aangeboden aan de Linker. Deze zal trachten uit de aangeboden object-files een werkend programma samen te stellen. Hiervoor moet uiteraard alles dat niet in de object voorkomt (alle externe referenties) aan het programma worden toegevoegd. De linker zoekt deze code in de andere object-files die worden aangeboden, en, als ze daar niet worden gevonden, in de functie bibliotheek.

Als alles goed gaat en alle externe referenties kunnen worden opgelost, dan krijgen we een programma dat alles bevat om op een 68030-machine te kunnen draaien, een zogenaamde executable of uitvoerbaar programma. In het geval van KGN68k, wordt de ontwikkeling van programmatuur gedaan op een LINUX-machine (een 368 PC met daarop het LINUX operating systeem). Om de code op KGN68k te laten draaien, wordt de executable eerst door een apart programma omgezet in S-records en vervolgens ingelezen in de KGN68k. Hier loopt het programma vervolgens naar wens (of niet).

Mijn klus gedurende de afgelopen maanden was er voor te zorgen dat de GNU assembler uit de source in figuur 2 dezelfde object file aanmaakt als de KGN68k assembler van de source in figuur 3 brouwt. Om dit te kunnen doen, was het noodzakelijk precies te begrijpen hoe een dergelijke object opgebouwd is. Deze kennis wordt in de rest van dit artikel gepresenteerd.

Gebieden binnen de object

In het verleden heb ik al eens uitgebreid beschreven dat het nuttig is in een programma de machine-instructies (code) te scheiden van de gegevens (data). Hiervoor zijn diverse redenen aan te geven. Het sterkste argument is wel het feit dat als er meerdere gebruikers zijn die hetzelfde programma gelijktijdig gebruiken, de code slechts één maal in het geheugen van de machine hoeft te staan. Elke gebruiker kan dan gebruik maken van dezelfde code maar krijgt uiteraard wel zijn eigen data. Om dit te kunnen doen, moet het operating system in staat zijn deze scheiding te maken en om deze reden wordt al in de object (eigenlijk zelfs in de assembler source) een onderscheid gemaakt tussen code (= machine-instructies) en data (= gegevens).

Een object uit de GNU assembler kent de volgende gebieden:

1: Een text segment met daarin de code
2: Een segment met daarin de data-velden die bij het starten van het programma geïnitialiseerd zijn (de DC -opdrachten in de KGN68k assembler). Dit segment heet "Initialized Data" segment.
3: Een segment met daarin de data-velden die niet afzondelijk geïnitialiseerd worden. Deze velden worden bij het starten van het programma met nul gevuld. Dit segment heet in het dagelijks leven "Uninitialized Data" segment.

Bij het laden van het programma kunnen de drie bovenstaande segmenten op heel verschillende plaatsen terecht komen. Het operating system moet

```
                NAME    Hello world  KGN68k
;
                BIN     GNU             ; GNU output format
;               ORG     $0000           ; Assemble from address 0
                OPT     -oe             ; Do not optimize
                CPU     $08             ; 68030 CPU
;
gcc2_compiled.:         ; Just a label
;               Text segment
LC0:            ascii   "Hello, world\n\0"
                align.w
;
;               Declaration of global labels
                local   gcc2_compiled.
                global  __main
                global  main
                global  serial_put
;
;               Code segment
main:
                link    a6,#-8          ; Get some space
                bsr.l   __main          ; Do ___main routine
                move.l  #LC0,-4(a6)     ; str = address text
                move.l  -4(a6),-8(a6)   ; for (p=str,
L2:
                movea.l -8(a6),a0
                tst.b   (a0)            ; *p,
                beq.s   L3              ; end of the loop
                movea.l -8(a6),a0       ;
                move.b  (a0),d0         ; get *p
                extb.l  d0              ; extend byte to long
                move.l  d0,-(sp)        ; copy *p to the stack
                bsr.l   _serial_put     ; _serial_put(*p);
                addq.w  #4,sp           ; drop *p from the stack
L4:
                addq.l  #1,-8(a6)       ; p++);
                bra.s   L2              ; close the loop
L3:
                clr.l   d0              ; clear return value
                bra     L1
L1:
                unlk    a6              ; free allocated space
                rts                     ; return
;
                END
```

Fig. 3: Hello World in source voor de KGN68k assembler

er dan (bijvoorbeeld m.b.v. de MMU) voor zorgen dat alles weer bij elkaar komt.

In figuur 4 is een dump van de object van ons voorbeeld weergegeven. Het betreft in dit geval de uitvoer uit de KGN68k-assembler.

Zo op het eerste gezicht zit er niet veel structuur in de informatie. Gelukkig wilde het geval dat ik, behalve een paar voorbeelden, ook een klein stukje documentatie had over de object. Laten we die er eens bijpakken.

```
Addr  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    0123456789ABCDEF
-------------------------------------------------------------------------
0000  00 00 01 07 00 00 00 50 00 00 00 00 00 00 00 00    .......P........
0010  00 00 00 30 00 00 00 00 00 00 00 00 18 00 00 00    ...0............
0020  48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 0A 00 4E 56    Hello,world..NV
0030  FF F8 61 FF FF FF FF EC 2D 7C 00 00 00 00 FF FC    ..a.....-|......
0040  2D 6E FF FC FF F8 20 6E FF F8 4A 10 67 18 20 6E    -n.....n..J.g.n
0050  FF F8 10 10 49 C0 2F 00 61 FF FF FF FF C6 58 4F    ....I./.a.....XO
0060  52 AE FF F8 60 E0 42 80 60 00 00 02 4E 5E 4E 75    R...'.B.'...N^Nu
0070  00 00 00 3A 00 00 03 D0 00 00 00 1A 00 00 04 40    ...:...........@
0080  00 00 00 14 00 00 02 D0 00 00 00 04 04 00 00 00    ................
0090  00 00 00 00 00 00 00 13 05 00 00 00 00 00 00 0E    ................
00A0  00 00 00 19 01 00 00 00 00 00 00 00 00 00 00 21    ...............!
00B0  01 00 00 00 00 00 00 00 00 00 00 00 2D 67 63 63 32  ............-gcc2
00C0  5F 63 6F 6D 70 69 6C 65 64 2E 00 5F 6D 61 69 6E    _compiled.._main
00D0  00 5F 5F 5F 6D 61 69 6E 00 5F 73 65 72 69 61 6C    .___main._serial
00E0  5F 70 75 74 00 00 00 00 00                         _put.....
```

Fig. 4: hexdump van een GNU object

De object begint met een zogenaamde header waarvan de structuur in figuur 5 is afgedrukt. Laten we deze eerst maar eens uit gaan pluizen.

De header begint met een byte waarvan het hoogste bit aangeeft of de object dynamisch gelinkt wordt of als zogenaamde shared object gebruikt wordt. Is dit niet het geval, zoals in alle "normale" situaties, dan is dit bit 0. Bij dynamisch linken, wordt de object niet gedurende het linken bij de executable gevoegd maar pas op het moment waarop het programma geladen wordt. Dit zie je wel eens bij bibliotheken met systeemfuncties of bijvoorbeeld een bibliotheek met rekenkundige bewerkingen. In dat geval staat de bibliotheek constant in het (virtuele) geheugen. Als nu een programma geladen wordt, dan worden pas op dat moment de verwijzingen naar deze bibliotheek ingevuld; iets wat normaal gesproken de linker doet. Precies hetzelfde heb je ook met zogenaamde Shared Objects. Dit zijn stukken programma (code) of data waarvan meerdere programma's gebruik maken. Ook deze objecten staan reeds in het virtuele geheugen als het programma geladen wordt.

De laagste 7 bits van het eerste byte kunnen het versienummer van de assembler bevatten. Eventueel kan de linker testen of zijn eigen versienummer correspondeert met het versienummer van de assembler. In de GNU-omgeving die wij gebruiken bevat dit veld de waarde 0 en wordt dus kennelijk niet gebruikt.

Het tweede byte bevat een code die aangeeft voor welke cpu de object bedoeld is. In ons geval heeft dit veld de waarde 2 = 68020. Aan de hand van de constantes kun je zien dat het GNU-formaat kennelijk afgeleid is van het formaat dat voor SUN werkstations gebruikt wordt. Dat klopt inderdaad; het verhaal gaat zelfs nog verder: ook de assembler syntax is precies gelijk aan de syntax van de assembler voor SUN.

Het derde en vierde byte bevatten een zogenaamd magisch nummer. In ons geval is dit de hexadecimale waarde 107.

Na de bovengenoemde constanten komen de velden waar wij in geïnteresseerd zijn. Het eerste veld (a_text) is de lengte van het zogenaamde text segment. In dit segment staat het geassembleerde programma (de code) en de zogenaamde ASCII-constantes, datavelden die niet gemodificeerd kunnen worden (teksten van meldingen etc.). Als we in de hexdump kijken, dan zien we op adres $20 de string Hello, world staan. Dit is het begin van het text segment dat een lengte heeft van 50 byte en dus eindigt op adres 6E,6F met de code voor de RTS instructie (4E75).

Het volgende veld geeft de lengte aan van het segment waar de initialized data komt te staan. In de object komt deze data meteen achter het text segment terecht. In ons voorbeeld hebben we geen initialized data. In figuur 6 is een klein voorbeeld opgenomen dat wel initialized data bezit.

De code die bij het voorbeeld uit figuur 6 hoort staat in figuur 7. We zien in dit geval dat in de header staat dat het initialized data segment 4 byte groot is. Dit segment begint op adres 3C. Aangezien segmenten altijd een lengte hebben die een veelvoud is van 4, wordt de initialisatie-waarde van b (3) gevolgd door twee vulbytes met de waarde 0.

```
/* a_dynamic macros    */

#define D_not_dynamic0
#define D_dynamic           1

/* a_toolversion macros */

#define T_version       0
/* a_machtype macros  */

#define M_ZERO          0
#define M_68010         1
#define M_68020         2
#define M_SPARC         3

/* a_magic macros */

#define OMAGIC          0x107
#define NMAGIC          0x108
#define ZMAGIC          0x10B

typedef struct exec {
    unsigned char a_dynamic:1;  /* has a__DYNAMIC      */
    unsigned char a_toolversion:7;    /* version of toolset      */
    unsigned char a_machtype;  /* machine type     */
    unsigned short     a_magic;       /* magic number          */
    unsigned long a_text;          /* size of text segment    */
    unsigned long a_data;          /* size of initialized data*/
    unsigned long a_bss;           /* size of uninitialized data */
    unsigned long a_syms;          /* size of symbol table    */
    unsigned long a_entry;         /* entry point         */
    unsigned long a_trsize;        /* size of text relocation */
    unsigned long a_drsize;        /* size of data relocation*/
    } t_gnu_header;
```

Fig. 5: definitie van de header in C

Na de lengte van het initialized data segment komt de lengte van het unitialized data segment te staan. In dit segment komt data terecht dat niet geïnitialiseerd is. In assembler wordt dit aangegeven met de pseudo instructie "DS". In C zijn dit globale of static variabelen die geen initialisatie hebben. In de object wordt alleen aangegeven hoeveel ruimte voor deze data gereserveerd moet worden. De data zelf wordt niet in de object opgenomen. Hoewel er geen initialisatie staat opgenomen, wordt deze data bij het laden wel geïnitialiseerd, namelijk op de waarde 0.

Het veld met de naam entry_point spreekt waarschijnlijk voor zich. Dit v byte veld kan het startadres van het programma bevatten.

De twee laatste velden in de header geven aan hoeveel ruimte er gebruikt wordt voor de zogenaamde relocatie informatie. Met behulp van deze informatie kan de linker de verwijzingen naar externe symbolen netjes invullen. Hierover in de volgende paragraaf.

Relocatie

In de voorbeelden uit de vorige paragraaf worden diverse symbolen benaderd waarvan het adres bij het assembleren nog niet bekend zijn. In Hello, world zijn dit de symbolen LC0, ___main en _serial_put. Het symbool LC0 is geen extern symbool maar moet toch nog ingevuld worden omdat dit symbool met behulp van een absolute adressering benaderd wordt. De symbolen ___main en _serial_put zijn externe symbolen.

```
Voorbeeld in C:
extern int    a;
short         b = 3;        /* Initialized data */
extern int    f (int);

void g (void)
{ b = f(a);
}

Voorbeeld in assembler:
              NAME        Extern g
                          BIN         GNU                    ; GNU output format
;                         ORG         $0000
                          OPT         -oe                    ; Do not optimize
                          CPU         $08                    ; 68030 CPU
;
gcc2_compiled.:
;                         Declarations global labels
                          local       gcc2_compiled.
                          global      _b
                          global      _g
                          global      _a
                          global      _f
;
;                         initialized data segment:
_b:                       dc.w        3
;
;                         unintialized data
;                         (leeg)

;                         Text segment
_g:                       link        a6,#0
                          move.l      _a,-(sp)
                          bsr.l       _f
                          addq.w      #4,sp
                          move.w      d0,_b
L1:                       unlk        a6
                          rts
                          END
```

Fig. 6: voorbeeld met initialized data

In de header staat dat de lengte van de text relocation information 18 hex = 24 byte bedraagt. Voor elke relocatie een entry van 8 byte. In figuur 8 is de definitie van de relocatie entries weergegeven.

In het eerste longword staat welk adres gereloceerd moet worden. In Hello, world zijn dit de (relatieve) adressen 14, 1A en 3A. Voor al deze adressen geldt dat er vier byte aangepast moeten worden (r_length = 2). Meteen volgend op het adres staat een 3 byte constante die aangeeft welk symbool we gebruiken. Voor externe symbolen (niet in deze object gedefinieerd) zijn dit de volgnummers van de gebruikte symbolen uit de symbol table. In ons voorbeeld zijn dit symbool 3 (___main) en 4 (_serial_put). Voor de interne symbolen (LC0) geeft deze constante aan van wat voor type deze relocatie entry is. In ons voorbeeld is dit type 2: een absolute adressering waarvan de offset t.o.v. het begin is opgegeven.

In de bits die volgen op de 3 byte constante staat precies wat er met dit adres aan de hand is en hoe de linker het adres aan moet passen; of hij de absolute waarde in moet vullen of dat hij het verschil

```
Addr  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F      0123456789ABCDEF
-------------------------------------------------------  ----------------
0000  00 00 01 07 00 00 00 1C 00 00 00 04 00 00 00 00     ................
0010  00 00 00 3C 00 00 00 00 00 00 00 18 00 00 00 00     ...<............
0020  4E 56 00 00 2F 39 00 00 00 00 00 61 FF FF FF FF F4  NV../9.....a.....
0030  58 4F 33 C0 00 00 00 1C 4E 5E 4E 75 00 03 00 00     XO3.....N^Nu....
0040  00 00 00 14 00 00 06 20 00 00 00 0C 00 00 04 D0     ................
0050  00 00 00 06 00 00 03 50 00 00 00 04 04 00 00 00     .......P........
0060  00 00 00 00 00 00 00 13 07 00 00 00 00 00 00 1C     ................
0070  00 00 00 16 05 00 00 00 00 00 00 00 00 00 00 19     ................
0080  01 00 00 00 00 00 00 00 00 00 00 00 1C 01 00 00     ................
0090  00 00 00 00 00 00 00 1F 67 63 63 32 5F 63 6F 6D     ........gcc2_com
00A0  70 69 6C 65 64 2E 00 5F 62 00 5F 67 00 5F 61 00     piled.._b._g._a.
00B0  5F 66 00 00 00 00 00                                _f.....
```

Fig. 7: object behorend bij figuur 6

tussen de waarde en de program counter moet gebruiken.

Symbol table

Na de informatie met behulp waarvan de linker de relocatie uitvoert, komen er twee gebieden die samen de symbol table vormen. In deze gebieden staan de naam en enkele gegevens over de externe symbolen opgesomd. Hierbij is een extern symbool een symbool dat niet in de object gedefinieerd is (bijvoorbeeld de routine ___main) of een symbool dat wel in het object is gedefinieerd maar die vanuit een ander object benaderd kan worden.

Laten we voor het gemak beginnen met het laatste deel van de object file. Dit is het deel dat vier byte voor de tekst "gcc2_compiled." begint. Van dit gebied geeft het getal in de eerste vier byte aan hoe lang dit gebied is. Vervolgens komen achter elkaar de namen van alle externe symbolen en van de symbolen die van buiten benaderd kunnen worden. Deze namen worden gescheiden door een nul byte. Het symbool gcc2_compiled. is niet een extern symbool en kan ook niet van buiten worden benaderd. Dit is meer een tekst die aangeeft dat de object file m.b.v. een GNU compiler en assembler is ontwikkeld.

Zo nu hebben de hele object file besproken op het gebied tussen de relocatie informatie en de zogenaamde tekst tabel na. In dit gebied staat voor elk extern symbool (en voor gcc2_compiled.) een stukje data met een omvang van 12 byte. De formele beschrijving van dit gebied staat in figuur 9.

Kijken we eerst even naar de waarden die de variabele n_type aan kan nemen. Het minst significante bit geeft aan of het symbool een zogenaamd extern symbool is. Dit wil zeggen dat het symbool niet in de functie gedefinieerd wordt (n_type = N_UNDEF + N_EXT = $01) of dat het symbool in de object file wordt gedefinieerd en die vanuit een andere object benaderd kan worden (bijvoorbeeld n_type = N_DATA + N_EXT = $07.

De verwijzing volgens char *name zul je in een object file niet tegenkomen. Dit is bedoeld voor eenzelfde datastructuur die bij het assembleren in het

```
typedef struct reloc_info_68k {
  long          r_address;              /* address which is relocated              */
  unsigned short  r_symbolnum_h:1,      /* high order part of local symbol         */
                  r_relative:1,         /* relative relocation                     */
                  r_jumptable:1         /* pc-relative to jump table               */
                  r_baserel:1,          /* linkage table relative                  */
                  r_extern:1,           /* does not include value of symbol        */
                  r_length:2,           /* 0 = byte, 1 = word, 2 = long            */
                  r_pcrel:1,            /* was relocated pc relative already       */
                  r_symbolnum_l:8;      /* local symbol ordinal                    */
      } t_reloc_info;
```

Fig. 8: definitie van een relocatiestring

```
/* Simple values for n_type   */

#define N_UNDEF          0x0        /* undefined                    */
#define N_ABS       0x2         /* absolute                  */
#define N_TEXT      0x4         /* text                         */
#define N_DATA      0x6         /* data                         */
#define N_BSS       0x8         /* bss                       */
#define N_COMM           0x12       /* common                       */
#define N_FN             0x1f       /* file names symbol            */
#define N_EXT       01          /* external bit or'ed in     */
#define N_TYPE      0x1e /* mask for all the type bits     */

typedef struct nlist {
      union {
            char  *n_name;          /* for use when in-memory       */
            long  n_strx;                 /* index into file string table      */
         } n_un;
      unsigned    char  n_type;           /* type flag                 */
                  char  n_other;
                  short n_desc;
      unsigned    long  n_value;          /* value of this symbol or offset    */
      } t_nlist;
```

Fig. 9: formele beschrijving van de symbol table

geheugen staat. In een object file wordt dezelfde geheugenruimte (vanwege de union definitie) gebruikt door een vier byte getal. Hierin staat een verwijzing naar de naam van het symbool in de zojuist besproken tekst tabel. Zo staat voor het symbool gcc2_compiled. in dit veld de waarde 4. De eerste vier byte geven namelijk de lengte van de tekst tabel aan waarna op de vierde positie de tekst staat. Hetzelfde geldt voor alle andere symbolen.

De twee velden volgend op n_type zullen ook ongetwijfeld iets betekenen. Over deze velden heb ik geen informatie. Wel is vastgesteld dat ze bij de voorbeelden die ik heb altijd de waarde nul bevatten. Misschien heeft iemand anders over de velden n_desc en n_other nog aanvullende informatie?

Het laatste veld is wederom vier byte groot. Voor gedefinieerde externe symbolen bevat dit veld de waarde van het symbool. Dit kan uiteraard ook het adres van een functie zijn, gerekend vanaf het begin van de object (bijvoorbeeld de waarde van symbool _main op adres 0E in het eerste voorbeeld).

Het symbool gcc2_compiled. komt zowel in de symbol table als in de tekst tabel voor. Dit symbool is echter geen extern symbool en kan dus niet van buiten worden benaderd. Dat kun je zien aan de waarde van n_type (= 04). Met de definitie uit figuur 10 erbij kun je zien dat dit een symbool in het TEXT gebied is en dat het symbool niet extern is. De linker weet nu dat hij niets met dit symbool behoeft te doen en dat het dus toegestaan is dat er meerdere object files dit symbool bevatten.

Afsluiting

Ik ben momenteel zo ver dat de KGN68k assembler precies net zo'n object file genereert als de GNU 68020 assembler. In dit artikel heb ik getracht een stukje van de ervaringen die ik met mijn GNU-object adventure heb opgedaan weer te geven.

Nu is het uiteraard niet zo dat elke linker op elk systeem dezelfde object gebruikt. Toch hebben de object-formaten van de diverse systemen veel met elkaar gemeen. De opdracht is namelijk ook zeer vergelijkbaar. Een linker moet uit een aantal object files (vaak afkomstig van een mix van compilers en een assembler) een hoeveelheid code maken die op de computer als programma uitgevoerd kan worden. Hierbij moet hij allerlei verwijzingen van de ene file naar de andere file en weer terug zien in te vullen. Van het systeem dat gebruikt wordt om dat in een object file vast te leggen is dit artikel een voorbeeld.

Gert van Opbroek

Van de bestuurstafel

Beste mensen dit wordt voorlopig mijn laatste schrijven op deze plaats. Op de Algemene Leden Vergadering heb ik kenbaar moeten maken dat het om privé redenen voor mij helaas niet meer mogelijk is om mijn functie naar behoren te kunnen vervullen. Dat doet me eerlijk gezegd wel een beetje pijn, want het besturen is toch een leuke taak. Gelukkig is tijdens de vergadering een aantal mensen opgestaan om het bestuur te komen versterken zodat een crisis is uitgebleven. Dat heeft me toch ook weer een beetje goed gestemd omdat ik vind dat een goede vereniging als de onze niet onbestuurbaar mag worden.

Op de vergadering was een agendapunt dat wij als bestuur met gemengde gevoelens hebben geplaatst. Toch is er gebleken dat er onder de leden voldoende belangstelling is om over problemen mee te praten. Het project KGN68k heeft van de leden voor de tijd van een jaar zijn fiat gekregen. Wij vonden het belangrijk dat de leden dat zouden doen en zijn ze ook dankbaar dat er op een positieve manier is meegedacht.

Toch is er gebleken dat er onder de leden voldoende belangstelling is om over problemen mee te praten.

Deze maand staan de HCC-dagen weer in de belangstelling. Of wij daar als vereniging weer aanwezig zijn kan ik op het moment dat ik dit schrijf nog niet bevestigen. Maar als dat wel zo is staat dat wel elders in ons lijfblad. Het wordt elk jaar weer moeilijk omdat te realiseren. De leden hebben ons in de enquête laten weten dat de kosten wel binnen de perken moeten blijven. Daar zal het bestuur dan ook zeker gevolg aan geven. Helaas is het zo dat de HCC steeds meer commercieel denkt en handelt. Dat betekent dat wij elk jaar weer creatief moeten denken om een gratis plaatsje te kunnen krijgen. Maar dat betekent wel dat we lang in onzekerheid worden gelaten of de HCC dat wel of niet accepteert. Hopelijk lukt het dit jaar toch weer.

Uit de enquête, waarvan de uitwerking elders in dit blad staat, is gebleken dat er leden zijn die wel eens wat anders in ons lijfblad willen zien. Het bestuur en de redactie juicht dit van harte toe. Hopelijk zijn de leden net zo realistisch als tijdens de enquête en doen ze er zelf ook iets aan. Jullie kunnen in alle redelijkheid niet van ons verlangen dat wij alle bladvulling voor onze rekening nemen. Door een enkeling wordt daaraan nu al gehoor gegeven, zij worden daarvoor vanaf deze plaats van harte bedankt. Toch kunnen er nog wel wat kleine artikeltjes door anderen worden geschreven. Ik denk dan aan nieuwtjes die jullie lezen waarvan je een klein of gemiddeld artikel zouden kunnen schrijven. Maar misschien kun je ook wat over je werk vertellen of de manier waarop er bij jou wordt gewerkt. Een uitwerking van een idee of de vraag naar een idee als jezelf de uitwerking niet weet vraag dan eens om hulp aan je medeleden. Als je niet weet hoe je dat het best kunt doen mag je altijd contact opnemen met de redactie of een van de bestuursleden. We willen jullie daar van harte mee helpen. Denk niet dat je niet kunt schrijven want dat kunnen we allemaal.

Hiervoor heb ik al laten weten dat ik mijn bestuursfunctie zal moeten neerleggen. Ik wil op deze plaats mijn overige bestuursleden bedanken voor de prettige manier waarop we de afgelopen twee jaar hebben samengewerkt. Maar ook voor de manier waarop we elkaar hebben gesteund, zonder elkaar soms ook eens te vertellen waar het echt op stond. Het nieuwe bestuur wens ik veel succes toe. Dat succes wordt pas een echt succes als jullie hen daar allemaal in steunen. Het beste toegewenst en tot ziens op onze clubdagen.

Tonny Schäffer.

Voortgang DOS65

Een kort berichtje deze keer: de printen voor de prototypen van de 1 Mbyte RAMkaart zijn besteld. De werkgroep gaat proberen u in het decembernummer alles over de definitieve kaart te kunnen vertellen, als de prototypen gebouwd en ontluisd zijn.

Dus niet alle centjes op de HCC-dagen uitgeven!
Namens de DOS65-werkgroep,

Nico de Vries

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 (BBS) 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 BBS.

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)
Looweg 10
3853 JV Ermelo
Telefoon geen

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

Gert van Opbroek (secretaris)
Del Del 16
5071 TT Udenhout
Telefoon 04241-3795

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

Jan Veninga
Klimopstraat 51
7601 SJ Almelo
Telefoon 05490-?

Henk Speksnijder
Albert Cuijpstraat 43
2902 GA Capelle aan den IJssel
Telefoon 010-4586879

Nico de Vries (redactie μP Kenner)
Van der Waalsstraat 46 (m.i.v. 25-12-1993)
2984 EP Ridderkerk
Telefoon 01804-n.n.b.

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