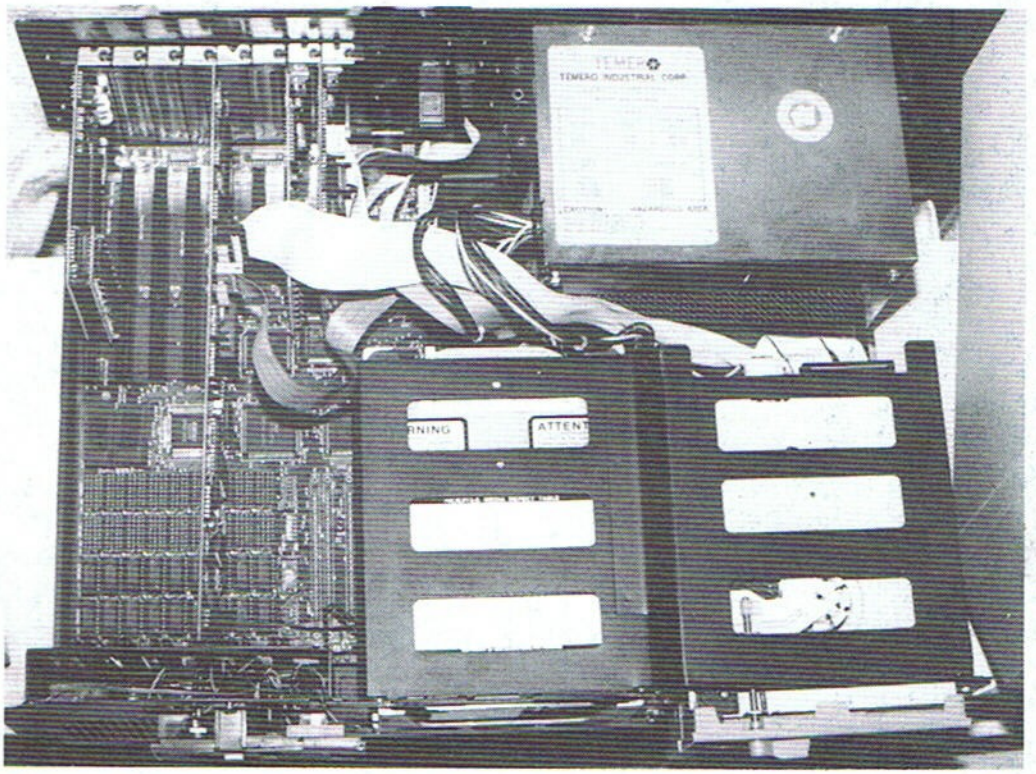
De µP Kenner

Veertiende jaargang nr. 5
December 1990
69



In dit nummer o.a.:

DOS65 3.0?
Assembler op de PC
KGN-68k MINIX Syteem
Datacommunicatie Deel 5
Programmeren op de Macintosh

Inhoud

De μP Kenner

Nummer 69, december 1990
Verschijnt 5 maal per jaar
Oplage: 250 stuks
Druk: FEBO Offset, Enschede

De redactie:

Gert van Opbroek
Bram de Bruine
Antoine Megens
Nico de Vries
Joost Voorhaar

Eindredactie:

Gert van Opbroek

Vormgeving:

Joost Voorhaar
Nico de Vries

Redactieadres:

Gert van Opbroek
Bateweg 60
2481 AN Woubrugge

De μP Kenner nummer 70 verschijnt op 16 februari 1991.

Kopijsluitingsdatum voor nummer 70 is vastgesteld op 2 februari 1991.

Vereniging

Uitnodiging voor de clubbijeenkomst 5
Verslag algemene ledenvergadering d.d. 17 november 1990 .. 6
Begroting 1991 7
Van de bestuurstafel 53
Informatie 54

Algemeen

Redactioneel 4

Talen/Software

Het standaard Operating System "UNIX" (Deel 2) 9
To Share Or Not To Share, That's The Question 21
Het programmeren van de 8088 in de IBM (Deel 3) 25

Systemen

Ervaringen met de installatie van MINIX 1.5.10 8
Het programmeren van de Mac, een "event driven system" .. 13
De IBM-PC en z'n klonen (Deel 11) 37

Hardware

KGN-68k MINIX systeem 33

DOS65

Wilt u versie 3 van DOS-65? Reageer! 36
SUFAST t.b.v. de real time clock MC146818 43

Datacommunicatie

Methoden en technieken voor datacommunicatie (Deel 5) .. 49

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

De HCC-dagen zijn weer achter de rug. Ook dit keer was het weer twee dagen een drukte van belang. Voor het eerst sinds enkele jaren had de KGN weer een eigen stand en we hebben in die twee dagen een groot aantal clubleden en ex-clubleden mogen ontmoeten. Verder waren er ook een aantal mensen die geïnteresseerd in de club waren waarbij vooral het op stapel staande MINIX hardwareproject veel belangstelling trok. Hoe het daarmee gaat kunt u lezen in één van de artikelen in dit blad. Op de HCC-dagen was onze belangrijkste trekpleister een robotarm die tegen de computer zat te schaken. Met behulp van de robotarm werden de zetten van de computer uitgevoerd waarna de robotarm zijn eigen zet intoetste op een terminal. Dit geheel was door Joost Voorhaar geprogrammeerd en draaide op een MINIX systeem. Volgend jaar kunnen we hetzelfde misschien doen waarbij de programmatuur hopelijk op binnen de club ontwikkelde hardware draait. Het is namelijk de bedoeling de KGN-68k processorkaart volgend jaar op de HCC-dagen draaiend te hebben.

Behalve de maand van de feestdagen is december ook de laatste maand van het jaar en daarmee is het decembernummer van de μP Kenner de laatste uitgave van de veertiende jaargang. Deze veertiende jaargang stond in het teken van een aantal veranderingen. We zijn overgegaan van een opmaak middels knip- en plakwerk naar een opmaak door middel van Desktop Publishing. Hierin heeft vooral Joost Voorhaar een heel belangrijke rol gespeeld. Hij is degene geweest die de layout van de eerste drie uitgaven verzorgd heeft. De laatste twee uitgaven zijn opgemaakt door Nico de Vries die hierbij gecoached wordt door Joost. Ook de inhoud van het blad is gaandeweg iets veranderd. We proberen namelijk bewust ons wat meer met de fundamenten van computers bezig te houden. Verder is ook de omslag gewijzigd. Enerzijds is de stijl wat moderner geworden, anderzijds is het vier-kleurenomslag vervangen door een zwart-wit omslag met één steunkleur. Dit vooral uit kosten-overwegingen. Kortom hoewel het blad goedkoper geworden is, ziet de opmaak en inhoud er professioneel uit. Het is daarom ook zeker op zijn plaats Joost een Nico een compliment te maken voor de veertiende jaargang en hartelijk te bedanken voor de hoeveelheid werk die ze het afgelopen jaar voor het blad verzet hebben. Ook Jacques Banser wordt vanuit de redactie heel hartelijk bedankt. Hij doet namelijk, met zijn familie, de afwikkeling van het blad. Hij onderhoudt de contacten met de drukker, doet het blad in enveloppen en zorgt voor de verzending. Kortom ook een behoorlijke portie werk dat hij naast zijn bestuurstaak als penningmeester en zijn werk als Sysop doet.

Deze uitgave van de μP Kenner bevat uiteraard weer een vervolg op de artikelenreeks van Nico over de PC en zijn klonen. Dat is al weer de elfde aflevering. Volgens Nico gaat deze serie nog bijna het hele volgende jaar door, hij is namelijk nog steeds niet uitgepraat over de PC en de AT en alles wat daar bij hoort. Nu kan het zijn dat er mensen zijn die niet alle afleveringen in hun bezit hebben omdat ze nog niet zo lang lid zijn. Speciaal voor die mensen zijn de reeds verschenen afleveringen, netjes opgemaakt in een DTP-pakket, beschikbaar voor fl. 5,-- plus fl. 2,50 verzendkosten, totaal dus fl. 7,50. Als u dit bedrag overmaakt op de girorekening van de KGN met de vermelding "PC-serie" dan krijgt u de eerste 10 afleveringen thuisgestuurd. Uiteraard kunt u ook wachten tot het einde van de serie die dan compleet als "special" beschikbaar komt.

Verder kan ik tot mijn grote vreugde vaststellen dat we 1990 met een iets groter aantal leden afsluiten dan 1989. Het lijkt er dus op dat de daling van het ledental tot staan gebracht is. Ik denk dat nu de tijd gekomen is om gezamelijk te proberen het ledental verder op te schroeven. Tenslotte moet het toch mogelijk zijn enkele tientallen mensen in Nederland te interesseren voor de μP Kenner, ons Bulletin Board the Ultimate, onze bijeenkomsten en onze hardware-projecten. Ik denk dat ons grootste probleem bij de ledenwerving het feit is dat men ons eigenlijk niet kent. Dat moeten we met z'n allen op gaan lossen. Het bestuur moet meer naar buiten treden maar ook de leden. Laat collega's, vrienden, familie en kennissen eens een clubblad zien, vertel over de club en wat we zoal doen en als de mensen geïnteresseerd zijn, geef dan naam en adres door aan één van de bestuursleden zodat we aanvullende informatie en een aanmeldingsformulier kunnen sturen.

Tenslotte wens ik u allen, namens de redactie, hele prettige feestdagen toe en een zeer voorspoedig 1991. Ik hoop dat u in het komende jaar minstens net zoveel plezier van uw computerhobby en de KGN zult hebben als in 1990.

Gert van Opbroek

Uitnodiging voor de clubbijeenkomst

Datum: 19 januari 1991
Locatie: Nieuwe kantine FORBO-Krommenie
Industrieweg 12
1566 JP Assendelft
Telefoon: 075-291911

Thema's: Financieel verslag 1990
MINITEL

Entreeprijs: fl. 10,--

Routebeschrijving

Per auto

1. Uit de richting Amsterdam: Coentunnel door en de Coentunnelweg helemaal afrijden. Aan het einde rechtsaf (water aan de linkerzijde). Dan de 1e afslag rechtsaf, richting Uitgeest-Alkmaar. Doorrijden tot aan de stoplichten. Linksaf de spoorbaan over.

2. Na 75 meter linksaf: Industrieweg. Links aanhoudende komt men dan op het FORBO-terrein.

3. Uit de richting Alkmaar: Snelweg Alkmaar-Haarlem. Afslag Uitgeest-Zaandam. Bij kruising linksaf. Bij de 3e stoplichten rechtsaf, de spoorbaan over. Verder volgens punt 2.

Per trein

Station Krommenie-Assendelft. Rechtsaf tot over de spoorwegovergang. Zie verder punt 2.

Programma

9:30 Zaal open. Ontvangst met koffie.
10:00 Opening door de voorzitter en verwelkoming door de gastheer:Co Filmer.
10:10 Diashow over FORBO-Krommenie.
10:30 Ledenvergadering
Agenda:
- Opening ledenvergadering
- Verslag vergadering 17 november 1990
- Behandeling jaarverslag 1990
- Rondvraag

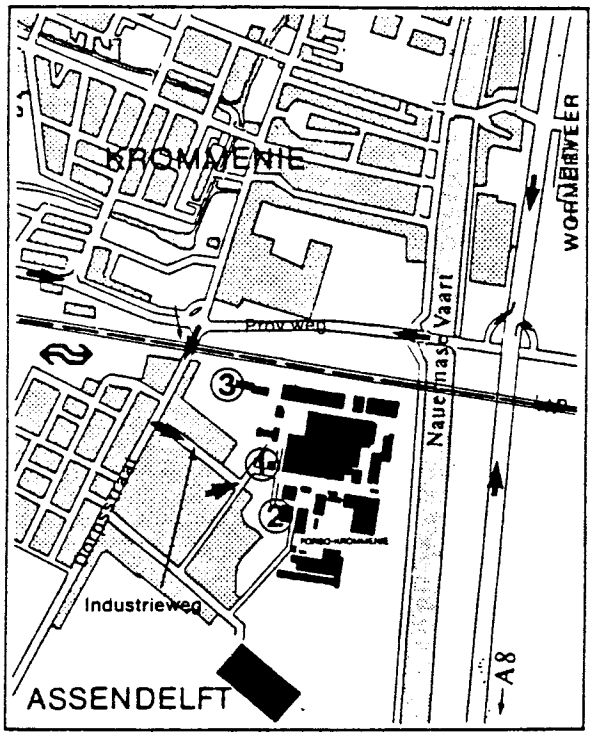
11:00 Koffiepauze.
11:15 Spreekbeurt van Coen Kleipool. Titel: MINITEL en het franse digitale telefoonnet (met misschien demonstraties).
12:15 Forum en markt
12:30 Lunch, aangeboden door FORBO-Krommenie.

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.

Attentie

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



① Portier / Portier / Portier / Gateman / Portero
② Ontvangstcentrum / Salle de reception / Empfangsraum / Reception building / Sala de recepcion
③ Kantoor / Bureaux / Büros / Offices / Oficinas
④ Centraal magazijn / Magasin central / Zentrallager / Central warehouse / Almacen central

Treinverb. Amsterdam-Alkmaar (half uurdienst)
Chemin de fer Amsterdam-Alkmaar (toutes les demi-heures)
Eisenbahn Amsterdam-Alkmaar (jede halbe Stunde)
Railway Amsterdam-Alkmaar (half hour service)
Linea ferroviaria Amsterdam-Alkmaar (cada media hora)

Verslag van de algemene ledenvergadering d.d. 17 november 1990.

1. Opening

De voorzitter (Nico de Vries) opent de vergadering om 11:00 uur. Notulist is Gert van Opbroek.

2. Verslag van de ledenvergadering d.d. 20-01-90

Het verslag wordt zonder op- of aanmerkingen goedgekeurd.

3. Concept-begroting 1990

Als toelichting op de begroting worden door het bestuur de volgende feiten opgemerkt:

- Na de daling van het ledental in de afgelopen jaren, zit er weer een lichte stijging in het aantal leden.
- Door de μP Kenner bij een andere drukker te laten drukken en door de nieuwe vormgeving is het blad tegenwoordig per exemplaar aanzienlijk goedkoper dan in voorgaande jaren.
- Voor een toelichting op de post "Hardware project" wordt verwezen naar punt 6 van de agenda.

De concept-begroting wordt zonder verdere discussie goedgekeurd. Deze begroting staat in de bijlage afgedrukt.

4. Verkiezing kascontrole-commissie voor het verenigingsjaar 1991

Staande de vergadering hebben de volgende personen zich verkiesbaar gesteld voor de kascontrole commissie:

- Herman Steunenberg
- Frits Herlaar

Deze twee personen zijn door de vergadering benoemd tot kascommissaris.

5. Verkiezing bestuursleden

Van de zittende bestuursleden traden, volgens rooster, af:

- Jacques Banser (Penningmeester)
- Jan Derksen
- Gert van Opbroek (Redactie)

Al deze personen waren herkiesbaar. Er waren geen tegenkandidaten gesteld. Bovengenoemde personen zijn verder zonder stemming herverkozen in het bestuur.

6. De toekomst van de club.

Binnen het bestuur zijn momenteel enkele kleinere en een wat groter hardware projecten bekend. Het betreft een parallelle I/O-kaart voor MS-DOS machines. Verder voor deze of elke andere parallelle IO-kaart een sampler. Het grotere project betreft een processorkaart t.b.v. van MINIX.

Gert van Opbroek heeft een toelichting gegeven op het op dit laatste hardware project. Aangegeven is dat er een projectgroep gevormd wordt die het project uit gaat voeren. Ten aanzien van dit onderwerp zijn de volgende besluiten genomen.

- De projectgroep rapporteert bij elke uitgave van de μP Kenner de stand van zaken naar de leden.
- De projectgroep moet de belangstelling voor het nabouwen van de hardware onder de leden peilen om zeker te stellen dat een eventuele investering van verenigingsgeld in dit project verantwoord is.
- Gert van Opbroek zegt, namens de projectgroep, toe dat deze zaken in het plan van aanpak opgenomen zullen worden.

Adri Hankel biedt een binnen Besamu ontwikkelde seriële interface monitor als hardware project voor de club aan. Dit aanbod wordt uiteraard in dank aangenomen.

7. Rondvraag

Nico de Vries heeft als voorstel aan de vergadering het erelidmaatschap van Rinus Vleesch Dubois ingebracht. Als toelichting werd aangegeven dat Rinus zeer lang voorzitter van de club geweest is en gedurende die periode veel voor de club gedaan heeft en de club door een aantal moeilijke perioden heeft geloodst. De vergadering heeft unaniem besloten de voordracht aan te nemen. Aan Rinus Vleesch Dubois zal op zeer korte termijn het ere-lidmaatschap aangeboden worden.

8. Sluiting

De vergadering wordt om ca. 12:30 gesloten.

Woubrugge, 2 december 1990

Gert van Opbroek.

Begroting 1991

BATEN :	
Contributie: 170 * ƒ 50,00	ƒ 8500,00
Losse μP Kenners	- 100,00
Bijeenkomsten	- 400,00
Reclame	- 250,00
DOS65	- 250,00
Hardware project	- 1500,00
Totaal	ƒ 11000,00
LASTEN :	
μP Kenner drukken/verzenden: 5 * ƒ 1000,00	ƒ 5000,00
Afschrijving inventaris	- 1000,00
Bulletin Board The Ultimate	- 1000,00
Bestuurs kosten	- 500,00
Reclame - ledenwervingsactie	- 500,00
Hardware project	- 2000,00
Onvoorzien	- 1000,00
Totaal	ƒ 11000,00

De begroting is dit jaar gemaakt met de gedachte dat het volgend jaar een beter jaar wordt voor de club. Hadden we vorig jaar nog op de begroting een aangenomen leden aantal van 120 daar hebben we dit jaar 170 van gemaakt. (Het ledental van de club kwam dit jaar uit op een goede 160 leden).

- Vorig jaar voorspelden we dat de μP Kenner niet meer gedrukt zou worden. Daar zijn we bij nader inzien toch maar van terug gekomen. (Wel gaat onze dank uit naar Ernst Elderenbosch die nummer 63 geheel gestencild had (kosten +/- ƒ 200,00)) We hebben dus een drukker gevonden die het voor de helft van de prijs kon doen t.o.v. de vorige drukker. Waar dan wel tegenover staat dat we er iets meer achteraan moeten lopen voor een "perfecte" afwerking...

- De kosten voor een advertentie hebben we fors naar beneden gebracht. Hopelijk kunnen we in het komende jaar nog andere bedrijven vinden die in ons perfecte blad willen adverteren.

- De losse verkoop van μP Kenners ligt zo goed als plat. Het komt nog sporadisch voor dat men een los nummer koopt. Dit gebeurt dan meestal op aanvraag via het BBS, hoewel de frequentie niet van betekenis is. Misschien dat we daar nog iets aan kunnen doen tijdens de komende HCC dagen.

- De hardware projecten die we gaan starten moeten een nieuwe opleving in de club teweeg brengen. Om te kunnen starten met het projecten komen diverse aanloop kosten om de hoek kijken. Ik hoop dat ik het met de ƒ 2000,00 niet al te verkeerd heb ingeschat. Ik verwacht dat er, net als tijdens het DOS65 project, weer een paar centen in de club-kas kunnen komen tijdens de verkoop van de onderdelen van de diverse projecten.

Enschede, 10 Nov 1990

Uw penningmeester,

Jacques Banser

Het standaard Operating System "UNIX" (deel 2)

Vorige keer heb ik het heel oppervlakkig gehad over een aantal standaard UNIX-commando's. Deze keer nemen we een kijkje in het UNIX systeem zelf. We lopen "even" door de standaard directory tree en bekijken en passant de belangrijkste files op het systeem.

UNIX is geen Disk Operating System in de zin dat we niet direct met device names geconfronteerd worden. Een device in UNIX is altijd een speciale file. Een disk wordt meestal ergens in het file system gehangen, zodat we, zonder dat we de devicename kennen, op een disk kunnen schrijven en/of van lezen. Daartoe moet het device "gemount" worden aan de subdirectory waar het device in het filesystem gehangen dient te worden. Alles wat na het mounten in de gemounte subdirectory geschreven wordt komt op de disk terecht. Hebben we de disk niet meer nodig, dan moet hij afgemeld worden bij het systeem, ge-"unmount". De opdrachten voor het mounten en unmounten zijn respectievelijk mount en umount. Dit laatste is *geen* schrijffout; de opdracht heet werkelijk umount en geen unmount.

Van dit mounten en unmounten heb je in de praktijk weinig last. Mounten van standaard subdirectories wordt meestal in het bootproces al gedaan en u(n)mounten vindt vaak pas plaats bij een system shutdown.

Ieder UNIX systeem heeft een bepaalde vaste directory structuur. Sommige subdirectories lijken dubbel voor te komen in het systeem. Dit verschijnsel heeft

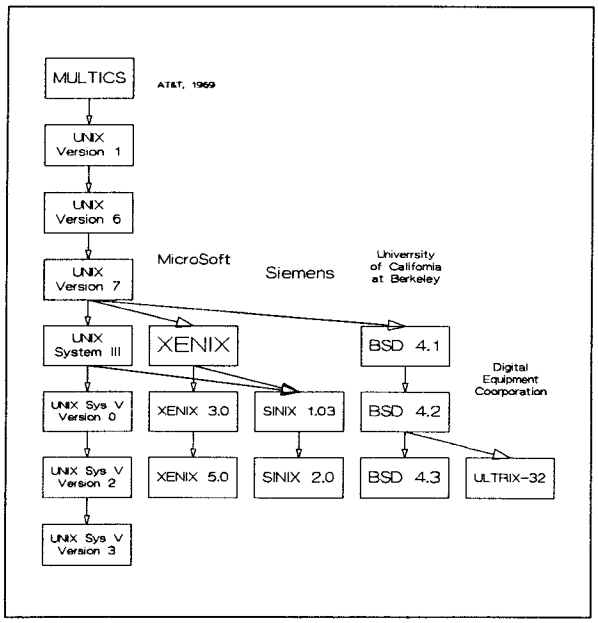
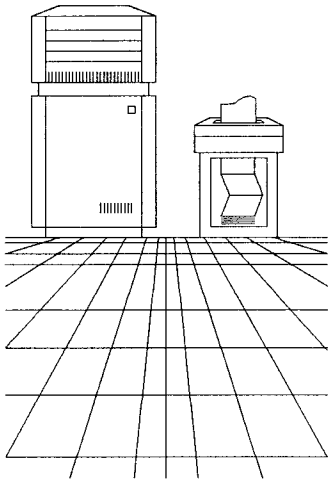


Fig. 1: Historisch overzicht van de diverse UNIX versies



een historische achtergrond. In het prille begin van UNIX waren externe randgeheugens duur. Vooral de disks, die relatief snel waren ten opzichte van tapes waren exorbitant hoog geprijsd. Daarom werden de utilities die veel gebruikt werden op snelle disks gezet en de utilities die minder vaak gebruikt werden hield men beschikbaar op tape. De "snelle" disks werden direct onder de root gezet; de langzame en goedkopere disks zaten meestal onder de /usr subdirectory.

De root van het gemiddelde UNIX systeem ziet er als volgt uit:

```
drwxr-xr-x   2 bin   288  Sep 13  10:53  bin
drwxr-xr-x   2 bin   480  Sep 6   17:26  dev
drwxr-xr-x   2 bin   416  Sep 16  20:52  etc
drwxr-xr-x   2 bin    32  Sep 12  13:40  mnt
drwxrwxrwx   2 bin    80  Sep 17  11:31  tmp
drwxr-xr-x   7 bin   128  Sep 13  14:53  user
drwxr-xr-x  12 bin   192  Jan 4   19:17  usr
```

De betreffende subdirectories hebben alle een eigen functie binnen het systeem. We zullen ze hieronder stuk voor stuk even nalopen.

We beginnen bij /bin. In de directory /bin staan de meest gebruikte utilities zoals "login", "ls" en "sh" (de shell). Deze directory maakt meestal deel uit van het standaard PATH.

In de /dev directory staan de verschillende devices. Ruwweg kunnen de devices in twee groepen onderverdeeld worden. We onderscheiden character- en blockdevices. Character devices zijn devices die per character een invoer stroom verwerken of genereren. Voorbeelden van character devices zijn bijvoorbeeld de terminals en de printer. Blockdevices zijn

devices die data per block verwerken. Een blockdevice heeft meestal een grootte; dit in tegenstelling tot een character device. De enige manier om het einde van een file te bepalen die van een character device komt is door het definiëren van een EOF (End Of File) character. In UNIX is EOF standaard ^D. Een blockdevice heeft, zoals gezegd, een grootte. Een blockdevice kan meerdere files bevatten en kan onderdeel uit maken van het filesystem. Disks en tapes zijn voorbeelden van blockdevices. In de file entry zijn devices herkenbaar aan een vlaggetje. Als het eerste vlaggetje in een directory listing een "c" is, hebben we te maken met een character device. Is het eerste vlaggetje een "b", dan is de file entry een blockdevice. In UNIX kan er wel van een blockdevice gelezen worden alsof het een character device is; we spreken dan van "raw" mode. Er wordt dan direct van het device gelezen.

De derde directory draagt de illustere naam "/etc". In de /etc directory staan allerlei systeem informatie files en vaak een hoop programma's die alleen door de root uitgevoerd mogen worden zoals "mkfs" (Make File System), "mount" en "umount". De belangrijkste files in de /etc subdirectory zijn de password- en group files en de terminal setup files. Over de password- en group files zal ik verder uitwijden aan het einde van deze aflevering. Voor nu zullen we volstaan met te vermelden dat de files slechts door de root geschreven mogen worden; echter: lezen mag iedereen! In /etc/passwd (de password file) staan de naam, user id, group id, home directory, shell en password van alle users. Het password is gecrypt volgens een one-way algorithme; een password kan wel gecrypt worden, maar men zegt dat er geen weg terug is. Ondanks het feit dat de password file alleen door de root te schrijven is, moeten users hun password kunnen veranderen. Ze hebben zelf geen write-access op /etc/passwd, dus zou onder normale omstandigheden de password file niet aangepast kunnen worden. Daarvoor heeft UNIX een fraai protectie algorithme aan boord: het SetUID en SetGID bit. De utility waarmee het password aangepast kan worden (passwd) heeft in plaats van de normale "executable" bitjes de mode -rwsr-sr-x. De eerste s geeft aan dat de utility in "Set UID" mode loopt, hetgeen betekent dat als effectieve user id de id van de owner van de file aangenomen wordt. De tweede s geeft aan dat de utility in "Set GID" mode loopt. De effectieve group id wordt dan tijdens het uitboeren van het programma gelijk gemaakt aan de group id van de owner. De passwd utility is meestal van de root en zet de effectieve user id van de user die passwd runt gelijk aan de user id "root".

Het password is gecrypt volgens een one-way algorithme; een password kan wel gecrypt worden, maar men zegt dat er geen weg terug is.

De volgende directory in de root is /mnt. Deze subdirectory is niet in alle systemen aanwezig, soms heeft het ding een andere naam en soms bevindt'ie zich op een heel andere plaats. De /mnt directory wordt gebruikt voor het mounten van floppies en/of tapes om bijvoorbeeld nieuwe programma's of datafiles van andere systemen te kunnen copiëren naar het file system van ons UNIX systeem.

De functie van de /tmp directory zal duidelijk zijn; de naam wijst er al op: temporary files worden hier neergezet. Dit is ook de enige directory onder de rootdirectory waar iedereen in mag schrijven.

De /user directory in het tabelletje is niet standaard. In mijn systemen heb ik onder /user alle home-directories van de verschillende users gehangen. In meer grote systemen kan dit een bijzonder complexe boom worden. Sommige sysops prefereren hun user subdirectory onder /usr te hangen; anderen hebben er weer andere ideeën over. In principe geldt dat de user directory ergens in het systeem uit moet hangen, maar ook dit is geen wet van buigen of barsten.

De laatste subdirectory in de root is /usr. Als ik de maker van UNIX was geweest had ik 't ding /sys genoemd, want onder de /usr directory hangen een aantal belangrijke systeem directories. De listing van /usr laat in mijn systeem zien:

```
drwxr-xr-x  2 bin   48  Sep 11  22:22  adm
drwxr-xr-x  2 bin 2832  Sep 10  22:27  bin
drwxr-xr-x  2 bin   32  Sep 6   17:34  doc
drwxr-xr-x  4 bin  688  Sep 12  18:42  include
drwxr-xr-x  3 bin  400  Sep 6   18:10  lib
drwxr-xr-x  2 root 160  Sep 13  10:45  local
drwxr-xr-x  2 bin   64  Sep 1   10:17  man
drwxr-xr-x  6 bin   96  Sep 6   17:34  spool
drwx------  9 bin  176  Sep 6   19:59  src
drwxrwxrwx  2 bin   80  Sep 17  16:52  tmp
```

Deze listing is al minder standaard dan die in de root directory. In alle UNIX systemen is wel een /usr/bin, /usr/include, /usr/lib, /usr/spool en een /usr/tmp. Meestal zijn ook /usr/adm, /usr/doc en /usr/man aanwezig. Soms ook nog /usr/local en slechts zelden een /usr/src. We beginnen bij de meest voorkomende subdirectories.

De subdirectories "bin" en "tmp" zijn twee van die directories die een voortzetting vormen van de gelijknamige directories onder de root. Ze zijn een direct gevolg van de historie van UNIX. In /usr/bin staan de minder vaak gebruikte commando's en in /usr/tmp worden de minder intensief geraadpleegde temporary files ondergebracht.

Speciaal voor de C-compiler en eventuele andere programmeertalen zijn de lib- en include directories. Ze bevatten de C-libraries en de header files voor C.

In /usr/spool staan een aantal andere subdirectories. Het zijn directories die gebruikt worden door "deamon" processen die in de achtergrond draaien om (tussen-) resultaten in op te slaan die niet in /tmp thuishoren. Bijvoorbeeld omdat niemand ze mag lezen en/of weggooien. In /usr/spool zijn subdrectories voor de postverbindingen, de postbussen voor de lokale users, voor batchjobs en dat soort zaken. De meeste zitten veilig op slot.

In het gemiddelde UNIX systeem wordt heel wat systeem administratie gedaan. De database files die daar uit ontstaan staan meestal onder /usr/adm.

De /usr/man direcory bevat de online manuals van het UNIX systeem. al naar gelang de inspanningen van de sysop en de noodzaak om documentatie bij de hand te hebben is deze manual meer of minder goed gevuld. Het "man"-commando slaat in /usr/man zijn datafiles op. In /usr/doc staan vaak de lokale aanpassingen en de documentatie files van lokale utilities.

We gaan even terug naar de /etc subdirectory. De interessantste files voor de beginnende systeem manager zijn hier te vinden. We kijken eerst eens naar /etc/passwd. In deze file staat voor iedere user een regel in de vorm zoals gegeven in figuur 2.

De naam, uid en gid zijn verplichte velden; de andere zijn optioneel. Het tweede veld, "ecpasswd" bevat het encrypted password. Zodra een user inlogt en zijn/haar password invoert, wordt dat gecrypt en vergeleken met het tweede veld van de regel in /etc/passwd die begint met de username (gevolgd door een dubbele punt!). Is dat password correct, dan wordt de shell opgestart (default: /usr/bin/sh of /bin/sh). Deze shell krijgt user- en group id zoals gedefinieerd in respectievelijk het derde en vierde veld van de /etc/passwd-regel. Tussen het opstarten van de controleren van 't password en het opstarten van de shell wordt de huidige directory gelijk gemaakt aan de directory zoals gespecificeerd in het zesde veld in de password file. Het "special" veld kan door de sysop gedefinieerd worden. Eigenlijk is het een commentaar veld. Meestal wordt er de volledige naam van de user en/of zijn/haar telefoonnummer in opgeslagen.

Een speciale mogelijkheid is, in het shell-field een asterisk (*) te zetten. Er wordt dan geen shell opgestart, maar er wordt een "chroot" uitgevoerd en een nieuwe login opgestart. "chroot" staat voor "change root" en geeft aan dat van de gespecificeerde directory een volledige nieuwe filestructuur verwacht wordt. De user kan de parent files en de vertakkingen die onder de betreffende directory zitten niet zien. Deze mogelijkheid wordt voornamelijk gebruikt in heel grote systemen waar bijvoorbeeld de afdelingen "teksschrijvers" en "financiën" van dezelfde computer gebruik maken. De systeembeheerder kan op deze manier het systeemgedeelte van "authors" volledig onzichtbaar maken voor "financials" en vice versa.

In het gemiddelde UNIX systeem wordt heel wat systeem administratie gedaan.

Een andere belangrijke file is /etc/group. In deze file staat precies omschreven welke groep welke naam heeft en wie toegang heeft tot welke groepen. Het is namelijk mogelijk in UNIX met behulp van het "newgrp" commando (tijdelijk) in een andere groep ingedeeld te worden. De inhoud van de file /etc/group bestaat uit regels (voor iedere groep één) met de volgende vorm:

<group name>:<ecpasswd>:<gid>:<userlist>

De userlist bestaat uit een rij usernames zoals gedefinieerd in /etc/passwd, van elkaar gescheiden met behulp van komma's. Stel dat er in /etc/group de volgende regel staat:

staff::100:ast,sir

De users "ast" en "sir" mogen nu het commando "newgrp staff" uitvoeren waarmee hun GID op 100

<name>:<ecpasswd>:<uid>:<gid>:<special>:<homedir>:<shel

Fig. 2: formaat van de file /etc/passwd

gezet wordt. De user "fvk" mag het commando niet uitvoeren. Althans, het commando mag wel uitgevoerd worden, maar het effect zal óf een foutmelding óf een botte weigering zijn.

Deze aflevering sluit ik af met een lijst van datafiles zoals die in de /etc directory zouden kunnen voorkomen en hun betekenis. Volgende keer ga ik dieper in op de shell commando's en een aantal standaard utilities.

gettydefs	omschrijving van de terminal settings
group	de lijst met users die naar een bepaalde groep mogen met "newgrp"
logmessage, ident of issue	de file die naar de terminals gestuurd wordt vóór de login: prompt
motd	tekstfile die ná de login, maar voor de .profile naar de user gestuurd wordt
mtab	een overzicht van de file systems zoals die aan het systeem gemount zijn
passwd	de password/user file
rc	het shell script dat automatisch opgestart wordt zodra het systeem geboot is en vlak voordat het "multiuser gaat"
termcap	tekst file waarin (escape-) codes voor de verschillende terminals omschreven staan
ttys	de file die aangeeft welke terminals aangesloten zijn en wat er moet gebeuren om ze aan te sturen
ttytype	geeft aan welke terminal types op welke lijnen verondersteld worden
utmp	user accounting file, nodig voor o.a. "who" en "last"

Literatuur:

1: G.J.M. Austen, UNIX: Het standaard operating system, Academic Service (ISBN 90-6233-217-X)

2: Eric Foxley, UNIX For Super-users, Addison-Weshley; International Computer Science Series (ISBN 0-201-14228-7)

Joost Voorhaar

Het programmeren van de Macintosh, een "event driven system"

De Macintosh en zijn grafische user interface hebben sinds hun verschijnen in 1984 een diepgaande invloed op de computerwereld gehad. Apple heeft, met de interface ideeen die oorspronkelijk in het fameuze Xerox-Palo Alto Research Centre werden ontwikkeld, voor een benadering gekozen die hemelsbreed verschilt van de commandline georienteerde IBM-DOS omgeving. Het gebruik van de muis, windows, icons, en gedeeltelijk gestandaardiseerde pull-down menu's zorgen voor een uniforme en logische interaktie tussen gebruiker en systeem. Het succes van deze benadering blijkt ook al uit IBM's pogingen een soortgelijk platform te lanceren (OS/2), vooralsnog met maar matig succes. Eerlijkheidshalve moet wel vermeld worden dat de grote gebruiksvriendelijkheid van Mac programma's bereikt wordt via moeizame arbeid van de kant van de programmeur. De Mac staat -niet geheel ten onrechte- bekend als een systeem dat moeilijk te programmeren is. Het is echter zo dat veel van de problemen uit oude gewoontes voortkomen, en wanneer men eenmaal met de principes van een "event driven" systeem vertrouwd geraakt is verdwijnen de meeste obstakels als sneeuw voor de zon. Ikzelf ben in ieder geval na een zeer kritisch begin (de Mac is een computer voor domme mensen, etc.) nu volledig verkocht, en ik zou voor geen goud meer naar de C> prompt terug willen. Vanuit de programmeur gezien is de grote kracht van de Macintosh de "User Interface Toolbox". De toolbox bevat honderden kant en klare routines in ROM, niet alleen voor het creeren en onderhouden van de gebruikers interface maar ook voor meer algemene taken zoals memory management. Tenslotte bevat de ROM ook een set snelle beeldschermroutines Quickdraw genaamd, die het hart vormen van het hele grafische systeem. In dit stuk zal ik verder niet te diep op de toolbox ingaan maar me beperken tot het bespreken van de basisstructuur van ieder Mac programma aan de hand van het standaard beginners programma "Hello world". Om de gedachte te bepalen, echter eerst een kort overzicht van het functioneren van een Macintosh gezien vanuit de gebruiker.

Verdwenen zijn ook de bekende 25 regels van 80 karakters, op de Mac is alles grafisch, ook tekst.

Na het booten springt de machine automatisch naar de "Finder". Dit is eigenlijk een normaal programma, maar de meeste gebruikers realiseren zich dit niet, en zien de Finder als het operating systeem van de Mac. Op zich is dit geen onlogische gedachte, aangezien de Finder inderdaad veel taken uitvoert die door bijvoorbeeld een PC op het operating systeem niveau gedaan worden. Het feitelijke Mac operating systeem is echter een deel van de toolbox en geheel onzichtbaar voor diegenen die niet zelf programmeren.

Het eerste wat na het booten opvalt is het ontbreken van de vertrouwde commandprompt in de Finder. Verdwenen zijn ook de bekende 25 regels van 80 karakters, op de Mac is alles grafisch, ook tekst. Het komt verder slechts zelden voor dat een programma direkt op het scherm schrijft (dat doet alleen de Finder zelf). Alle andere programma's sturen hun output naar een of meer windows, die gezien kunnen worden als vellen papier op een bureaublad. Dit is de zogenaamde "Desktop metafore". Harde schijf en floppy's (let op: niet de drives, maar de floppy's zelf) worden op de desktop door icons (kleine plaatjes) weergegeven. Rechtsbeneden op het scherm staat een prullenmand icon (de trash) waarin objecten geplaatst kunnen worden die men wil deleten. Twee maal klikken op een icon van bijvoorbeeld de harde schijf opent een window met daarin voor iedere subdirectory (meestal folder genoemd) en voor iedere file een icon. Ieder programma heeft zijn karakteristieke icon, terwijl ook de bijbehorende datafiles (bijvoorbeeld textfiles van een wordprocessor) een eigen icon hebben. Twee maal klikken op een programma icon start dat programma. Twee maal klikken op een datafile start het bijbehorende programma en opent daarna direkt de aangeklikte file. Voor het kopiëren van een file naar een andere folder of floppy hoeft alleen het icon met de muis naar de bestemming gesleept te worden, waarna het copy commando automatisch wordt uitgevoerd. Nooit meer spelfouten in de filename!

Om orde te kunnen scheppen in de stapels overlappende windows kunnen deze met de muis op verschillende manieren bewerkt worden. Klikken in een window maakt dit window aktief, en plaatst het boven alle andere windows. Langs de bovenrand van ieder window is verder een "dragbar" aangebracht, waarmee het window over de desktop geschoven kan worden. Het is ook mogelijk het aktieve window groter of kleiner te maken met de muis door de rechterbenedenhoek (de growbox) naar een nieuwe positie te schuiven. De linkerbovenhoek blijft hierbij op zijn plaats.

Links in de dragbar van de windows is een klein symbooltje aanwezig (de closebox) dat wanneer het

wordt aangeklikt het window sluit. Langs de bovenkant van het hele scherm loopt een "Menubar" met een aantal pull down menu's waaruit met de muis een keus gemaakt kan worden. Na het indrukken van de mouse button in een van de menu's worden de menu items (keuzes) er onder zichtbaar, Het loslaten van de muis in een item resulteert in het uitvoeren van het gekozen commando.

Apple doet veel moeite om programmeurs te overtuigen van de noodzaak te standaardiseren (de term "evangelisatie" die weleens wordt gebruikt geeft de gedrevenheid van dit proces goed weer). De "User interface guidelines" beschrijven nauwkeurig een aantal eigenschappen die ieder "echt" Macintosh programma behoort te hebben. Zo behoort het eerste menu in de menubar altijd het zgn. Apple menu te zijn. Hierin kan de gebruiker (met een daarvoor beschikbaar utility programma) verschillende hulpprogramma's zetten, bijvoorbeeld een calculator, een klok, een kladblok enz. Verder heeft ieder programma een "Edit menu" op een vaste plaats met ten minste Cut, Copy, Paste en Clear commando's.

Terug naar "Hello world"

Hoewel het tegenwoordig mode is om in C te programmeren ben ik een overtuigd Pascal fan (assembler is natuurlijk nog mooier, maar waar haal je van daag de dag de tijd vandaan). Het wordt dus een Pascal programma.

In zijn eenvoudigste vorm ziet het programma er ongeveer zo uit:

```
program Hello_World;
begin
    writeln("I hate IBM");
end.
```

Het eerste probleem dat opdoemt is dat het volstrekt onduidelijk is waar de output van het writeln statement heen zal gaan. Pascal implementaties voor de Mac lossen dit meestal op door een eigen tekst window te definiëren waarheen de output gaat, in feite een beeldscherm in een window. Het zal duidelijk zijn dat dit beslist geen fraaie oplossing is. Het is maar afwachten wat de Pascal voor mogelijkheden biedt om het window te verplaatsen, van grootte te veranderen, of later iets met de tekst te doen. Bovendien, wat moeten we doen wanneer ook grafische output gewenst is? In het programma definiëren we daarom ons eigen window, en houden zo alle mogelijkheden voor de toekomst open. Het programma wordt wel wat omvangrijker, maar daarmee ook leerzamer, inspirerender en bovenal veel flexibeler. Om de zaken op dit moment niet al te ingewikkeld te maken begin ik verder direkt met zondigen tegen de user-interface guidelines: het programma krijgt (nog) geen menubar. Wat overblijft is een enkel window dat na het starten op het scherm komt. In dit window wordt de "Hello world" tekststring geschreven. Het window kan met de muis over de desktop geschoven kunnen worden en naar behoefte groter en kleiner gemaakt. Het spreekt verder van zelf dat wanneer andere windows ons window hebben overlapt, de tekststring "ge-update" moet worden. Klikken in de closebox tenslotte beeindigt de executie van het programma, en geeft de controle terug aan de Finder.

Het gewenste gedrag van het programma is nu bekend, voor de verdere invulling is het nodig meer te weten over de manier waarop de Macintosh intern georganiseerd is. De term "event driven" is al enige keren gevallen, en het is nu tijd daar (wat) dieper op in te gaan. Hier komt weer een principieel verschil naar voren tussen de PC en de Macintosh. In het doorsnee PC programma is het meestal de computer die de orders uitdeelt:

'Please type a filename'
'Answer Yes or No'
'Type < CR > to continue'

Mocht de gebruiker wat anders in de zin hebben dan is een foutmelding zijn deel en meestal laat het programma zich niet overhalen verder te gaan voordat de gevraagde informatie ingegeven is. Het doorsnee Mac programma laat daarentegen aan de gebruiker over wat hij wil doen. In plaats van instructies te geven wat er dient te gebeuren accepteert het systeem opdrachten van de gebruiker betreffende de volgende stap. De gebruiker slaat toetsen aan, bedient de muis, wisselt floppies uit enz., en het systeem (let op, NIET het programma) creëert voor ieder van deze akties (Events genoemd) een Event-record en zet dit in een wachtrij, de Event-queue. De programmeur is hiermee dus niet belast, alle events verschijnen automatisch in de queue. (voor diegene die het naadje van de kous willen weten, de "vertical retrace manager" in de ROM controleert via een interrupt regelmatig de VIA's waaraan muis, toetsenbord etc. zijn aangesloten, en genereert de events).

Het programma (in Apple terminologie de "applicatie" genoemd) bestaat in principe uit een oneindige lus waarin events van de eventqueue worden opgehaald en afgehandeld totdat de gebruiker opdracht geeft te stoppen. Het verschuiven van een window dat een ander window gedeeltelijk bedekt veroorzaakt ook een event (een "update" event) om de applicatie er op te wijzen dat de inhoud van het onderliggende window vernieuwd moet worden. Tenslotte worden ook events gegenereerd als een window actief of inactief gemaakt wordt (per definitie is alleen het bovenste window actief, alle anderen

zijn inactief). Een voorbeeld is geselecteerde tekst in een tekst editor window. Het geselecteerde gedeelte wordt alleen geïnverteerd weergegeven als het window actief is. Dit helpt de gebruiker te bepalen op welk window zijn acties betrekking hebben. Tenslotte bestaat er een null-event dat aangeeft dat er niets gebeurd is, en dat er daarom niets te doen valt.

Samenvattend kunnen dus de volgende events in de event queue gevonden worden:

NullEvent	niks gebeurd
KeyDown	toets ingedrukt
KeyUp	toets losgelaten
autoKey	bij vasthouden toets
Mousedown	muis knop ingedrukt
MouseUp	muis knop losgelaten
UpdateEvt	vernieuw inhoud window
ActivateEvt	activeer/deactiveer window

Er zijn in feite nog meer events maar ze spelen voor ons programma geen rol en een volledige bespreking voert hier te ver.

Het is nu mogelijk het hoofdprogramma van "Hello world" te schrijven. Dat ziet er uit zoals in figuur 1 gegeven.

Er zijn drie globale variabelen gedefinieerd, ten eerste een vlag die gezet kan worden als de gebruiker het programma wil verlaten, dan een pointer die wijst naar de datastructuur van het outputwindow (waarover later meer), en tenslotte een eventrecord om de events die van de eventqueue gehaald worden op te slaan.

De "event driven" structuur van het programma is duidelijk herkenbaar, na een eerste initialisatie (waarin we o.a ons output window zullen definiëren) worden net zo lang events afgehandeld in de procedure DoEvent, totdat de gebruiker besluit dat het welletjes is geweest, en de done vlag opzet. Het aanroepen in de lus van de routine Systemtask (een toolbox routine) is noodzakelijk om periodieke acties van de (in de inleiding genoemde) hulpprogramma's onder het applemenu mogelijk te maken.

De initialisatie routine is in ons geval heel eenvoudig. Er hoeven eigenlijk maar twee taken uitgevoerd te worden: het initialiseren van de done vlag, en het creeren van ons window. In sommige pascal versies is het ook nog nodig de toolbox managers die later worden aangeroepen te initialiseren. maar we nemen hier aan dat dit door de compiler gedaan wordt. De initialisatie procedure ziet er als volgt uit: (zie figuur 2 op de volgende pagina).

De lange rij variabelen voor de functie call die het window creëert is hier alleen als illustratie opgenomen. Normaal wordt deze informatie niet direct in de source gecodeerd, maar opgeslagen in een "resource" in het resource gedeelte van de applicatie file. Deze scheiding van code en data maakt het mogelijk later de eigenschappen van het window te veranderen (met een resource editor), zonder opnieuw te hoeven compileren. Het is standaard praktijk dat ook alle tekst in het programma in resources wordt opgeslagen om een aanpassing aan een andere taal (bijvoorbeeld Chinees) later eenvoudig mogelijk te maken.

De functie Newwindow geeft een pointer terug die naar een windowrecord wijst. In dit record staat alle informatie die voor het window van belang is. De positie van het window wordt meegegeven als een

```
program Hello_World

var
    done : Boolean;          { true if user wants to quit      }
    theWindow: WindowPtr;    { pointer to our output window    }
    event:   Eventrecord;    { contains info about event       }

begin

    Initialise;              { set up our world                }

    Repeat
        Systemtask;          { periodic tasks of the system    }
        DoEvent;             { handle events                   }
    Until done;              { exit if done flag true          }

    Cleanup;                 { don't leave garbage around      }

end. {Hello_world}
```

Fig. 1: hoofdprogramma "Hello World"

```
procedure Initialise;

var
    storage: Ptr;                { pointer to storage for window    }
    windowrectange: Rect;        { window's position on screen      }
    title: Str255;               { window's title                   }
    visible: Boolean             { window's initial visibility      }
    windowtype: Integer;         { window's borders                 }
    behindwindow: WindowPtr;     { window put behind this           }
    hasClose: Boolean;           { close box in dragbar?            }
    refcon: Longint;             { constant, free to use            }

begin

    done := False                { not yet done with eventloop }

    { Set up variables for window, this information would       }
    { normally come from a special part of the file,            }
    { called the "resource fork"                                }
    storage := nil;                      { system can allocate it  }
    title := "Window on the world';      { window's title          }
    visible: = FALSE                     { window not visible yet  }
    windowtype := 0                      { normal window           }
    Behindwindow: = pointer(-1);         { window on top of pile   }
    hasClose := TRUE;                    { close box in dragbar    }
    refcon := 0;                         { constant, not used      }

                                         { initialise position..   }
                                         { left,top,right,bottom   }
    setrect(windowrectangle,50,100,300,200);
                                         { now create the window }
    theWindow := NewWindow(storage, windowRectangle, title, visible,
        windowtype, behindwindow, hasclose, refcon);
    ShowWindow(thewindow);     { draw window on screen }

end; {Initialise}
```

Fig. 2: initialisatie routine

rectangle. Dit type variabele is gedefinieerd in het grafische pakket Quickdraw, en bestaat uit vier integers die de schermpositie aangeven (in pixels) voor top, left, bottom en right (het punt 0,0 ligt linksboven op het scherm). Met de toolbox procedure "Setrect" wordt dit rectangle geïnitialiseerd, waarbij nogmaals opgemerkt moet worden dat deze informatie eigenlijk uit een resource zou moeten komen. Tenslotte wordt met de toolbox procedure ShowWindow het (nu nog lege) window op het scherm gezet.

Op dit punt aangekomen zijn we eigenlijk terug bij de programmaversie die helemaal aan het begin van dit artikel beschreven werd, met dit verschil dat het output window nu door ons zelf gedefinieerd is. Als nu de gewenste textstring in het window zou worden geschreven kan het programma direct verlaten worden zonder verder met events bezig te hoeven zijn (wat een opluchting). Voor de doorzetters en diegenen die meer toeters en bellen willen toevoegen, is het nodig een "main eventloop" op te zetten, en daarin de events in de eventqueue af te handelen.

Voordat we onze routine daarvoor (DoEvent) meer in detail gaan bekijken zullen we de structuur van de eventrecords die door het systeem in de queue worden gezet nader onder de loep te nemen.
Het record is als volgt gedefinieerd:

```
type
    eventrecord = record
    what      : Integer  { event type              }
    message   : Longint  { specific info           }
    when      : Longint  { time of event           }
    where     : Point    { mouse position          }
    modifiers:Integer    { state of modifier keys  }
end;
```

Het event.what veld geeft aan wat voor type event het betreft. In het systeem zijn voor all types events (zoals KeyDown en MouseDown) constanten gedefinieerd. De informatie in het event.message veld hangt af van het type event. Voor een KeyDown event bevat het de aangeslagen toets, voor een Mousedown event is het veld niet van belang, voor een update event of een activate staat er een pointer naar het betreffende window in. De event.when en event.where velden bevatten respectievelijk de tijd van optreden van het event en de positie van de muis. Tenslotte bevatten de bits in het event.modifiers veld vlaggetjes die onder andere aangeven of er speciale toetsen zoals de shift of control waren ingedrukt tijdens het event.

Gewapend met deze kennis kunnen we onze procedure DoEvent nader bezien. Het is eigenlijk niet meer dan een verdeelstation. In de routine wordt een event van de eventqueue gehaald (met behulp van de toolboxroutine GetNextEvent), waarna in een case statement voor ieder type event een afhandel routine wordt aangeroepen: (zie figuur 3).

De toolboxroutine GetNextevent krijgt twee variabelen mee, onze global "event" om het event in op te slaan, en een masker dat het mogelijk maakt om bepaalde types van events uit te filteren. Aangezien dit in ons geval niet van belang is, geven we het masker "everyEvent" mee, zodat alle events opgehaald worden. De functie geeft een boolean terug die aangeeft of het voor het programma nodig is om op het event te reageren (er staan ook events in de queue die niet voor ons bedoelt zijn). Als Getnextevent False teruggeeft wordt DoEvent daarom zonder verdere actie verlaten.

Na de call staat het event in ons eventrecord. In dit simpele programma zijn maar twee soorten events belangrijk: mousedown events en update events.

Terugdenkend aan het gewenste gedrag van ons programma kunnen we vaststellen dat de DoMousedown routine een viertal verschillende situaties moet onderscheiden:
1) Mousedown BUITEN window, hier hebben we niks mee te maken.
2) Mousedown in de DRAGBAR van ons window, de gebruiker wil het window verplaatsen.
3) Mousedown in de GROWBOX van ons window, de gebruiker wil het window van grootte veranderen.
4) Mousedown IN ons window, ook hiermee hoeven we (nog) niets te doen.
Het onderscheiden van al deze mogelijkheden kan heel comfortabel met een enkele toolbox call gerealiseerd worden:

Position:= Findwindow(event.where,theWindow);

De eerste parameter bevat de plaats van de mousedown (die in het "where" veld van het eventrecord beschikbaar is). Als de mousedown in een window plaatsvond is na terugkeer de variabele thewindow gevuld met de betreffende windowpointer, als de mousedown niet in een window lag krijgt deze variabele de waarde NIL. De functie zelf geeft een integer resultaat terug dat codeert op wat voor soort locatie de mousedown plaatsvond. De bovengenoemde 4 mogelijkheden voor de plaats van de mousedown hebben ieder hun eigen integer constante. De structuur van de DoMousedown procedure wordt nu duidelijk: Na de findwindow call kan met

```
Procedure DoEvent;

begin {DoEvent}

    if GetnextEvent(everyEvent,event) then

        Case event.what of      { get eventtype from eventrecord   }

        Mousedown:
            DoMousedown;        { handle the  mousedown            }

        UpdateEvt:
            DoUpdate;           { update window's content          }

        Otherwise
        ;                       {Ignore all other events           }
    end; { case on event    }
end; {DoEvent}
```

Fig. 3: de procedure DoEvent

```
Procedure DoMousedown;

var
    position      : integer      {which part of the window    }
    thewindow     : windowPtr    {mousedown in this window    }

begin

    Position: = Findwindow(event.where,theWindow);

    Case position of
        InDesk:                  { in desktop,not in a window  }
    ;                            { we can ignore this          }
        Incontent:               { inside window               }
    ;                            { ignore this also            }
        Indrag:                  { in window's dragbar         }
            DoDrag(thewindow):   { drag the window             }
        Ingrow:                  { in window's growbox         }
            DoGrow(thewindow)    { grow the window             }
        InGoaway:                { in window's closebox        }
            done: = TRUE;        { user wants to quit          }
                                 { so set flag true            }
    end; {case}
end; { doMousedown}
```

Fig. 4: DoMousedown routine

een case statement bepaald worden wat de door de gebruiker gewenste aktie is: (zie figuur 4, hierboven)

Na deze selectie komen we dan nu eindelijk aan het eigenlijke werk toe, het verschuiven en van grootte veranderen van het window. We beginnen met de DoDrag routine. Het begint eentonig te worden, maar ook hier staat ons een toolbox call ter beschikking, met de voor de hand liggende naam DragWindow, die alles doet wat nodig is. De call krijgt drie parameters mee, natuurlijk de windowpointer, dan de positie van de muis en tenslotte een rechthoek die grenzen stelt aan de verschuiving. Dit laatste is nodig om te voorkomen dat de gebruiker in zijn enthousiasme het window geheel buiten het scherm probeert te schuiven waar het voorgoed buiten zijn bereik zou zijn. Om zeker te zijn dat het programma op alle Mac's kan draaien, dus ook die met een groter scherm, wordt deze rechthoek (limitsrect) afgeleid van de door het systeem gedefinieerde globale variabele screenbits.bounds. deze variabele bevat altijd een rechthoek ter grootte van het hele scherm. Met de toolboxcall Setrect kan onze limits-

```
procedure DoDrag(window:windowptr);

var
    limitsrect: Rect;     { limits window size and move      }
begin
    { init the limits for size and move of our window, using the   }
    { rectangle screenbits.bounds that the system keeps for us     }

    With sceenbits.bounds do
        setRect(limitsrect,left + 4,top + 40,right-4,bottom-4);
    { the next call does all the dragging }
    Dragwindow(window,Event.where,limitsrect);

end;{Dodrag}
```

Fig. 5: DoDrag routine

rect eenvoudig gevuld worden met een iets kleinere versie van screenbits.bounds

De DragWindow call doet al het eigenlijke werk (lang leve de toolbox), terwijl de windowmanager van het systeem automatisch na de verschuiving het scherm weer up to date brengt.

Dan nu de DoGrow routine. Het window dat we gecreëerd hebben wordt in principe door het systeem onderhouden (de dragbar met de titel en de randen). Wat voor ons overblijft is alleen het mededelen aan het systeem wat de nieuwe maat en positie van het window is geworden na onze schaling, en het schoonvegen van de gehele inhoud, zodat onze update routine straks met een schone lei kan beginnen. Tenslotte moet het systeem er ook van in kennis gesteld worden dat we behoefte hebben aan een update event. (Het formaat van het window is immers veranderd, en we willen dat eventueel nieuw zichtbaar geworden delen ook daadwerkelijk op het scherm verschijnen. De volgende 6 regels code (alle toolbox calls) verzorgen deze zaken (figuur 6).

De functie GrowWindow doet alles wat nodig is voor de schaling van ons window. Zolang de gebruiker de muisknop ingedrukt houdt, wordt een gestippelde nieuwe omtrek van het window getoond. Als de gebruiker tevreden is met het nieuwe formaat laat hij de muisknop los en de routine geeft de nieuwe positie van de rechter benedenhoek terug in de twee bytes van size. Met de calls Hiword and LoWord komen de benodigde coördinaten beschikbaar voor Sizewindow, de toolboxcall die het systeem de nieuwe maten doorgeeft.

Om het window schoon te maken halen we de benodigde rechthoek uit het windowrecord. Met de ^ (dereference) krijgen we via de windowpointer toegang tot het window record zelf. Dit gecompliceerde record wordt hier verder niet besproken, maar het portrect veld bevat de rechthoek die de inhoud van ons window omsluit (deze rechthoek wordt door het systeem bijgehouden, ook na verschuiven of schalen van het window). EraseRect veegt dus ons window geheel schoon. Invalrect genereert het update event dat nodig is om de window inhoud opnieuw te tekenen. (zoals al opgemerkt zorgt het systeem voor het tekenen van de dragbar en de randen van het window) Dan komt nu de Update procedure aan de beurt. Behalve de feitelijke inhoud van het window moeten we ook de growbox tekenen. De growbox is het symbool in de rechter benedenhoek van het window dat de gebruiker aanklikt als hij de grootte wil veranderen. De growbox is alleen aanwezig ten behoeve van visuele feedback naar de gebruiker, het wel of niet tekenen er van heeft geen invloed op het functioneren van het window. De gebruiker verwacht echter een growbox. De DoUpdate routine bestaat alweer uit toolbox calls (figuur 7, volgende pagina).

De parameter "window" die al deze calls meekrijgen wordt afgeleid uit het event.message veld van het eventrecord. Bij update events staat hierin een pointer naar het windowrecord van het te updaten wind-

```
Procedure doGrow

var
     size: Longint;
     limitsrect: Rect;      { limits window size and move      }

begin
     { init the limits for size and move of our window, using the    }
     { rectangle screenbits.bounds that the system keeps for us      }
     With sceenbits.bounds do
          setRect(limitsrect,left + 4,top + 40,right-4,bottom-4);

     size: = Growwindow(window,event.where,limitsrect);

     { The new position of the bottom-right corner is returned in    }
     { the high and low byte of size                                 }
     if size < > 0 then begin
          sizewindow(window,LoWord(size), HiWord(size), TRUE);
          Eraserect(window^.portrect); { clear the window }
          Invalrect(window^.portrect); { update event needed      }
     end;
end;
```

Fig. 6: DoGrow routine

ow. voor het gemak maken we een lokale kopie van deze pointer in de variabele window. De setPort call die nu volgt zorgt er voor dat de grafische commando's (onze tekst string is op de Mac ook grafisch) ook daadwerkelijk op de juiste plaats (ons window dus) aankomen. Eigenlijk is deze call maar een keer nodig, en zou dus in Initialise routine geplaatst kunnen worden, maar het is een goede gewoonte om voor ieder gebruik van Quickdraw routines zeker te stellen dat onze output naar het juiste window gaat. Ons window is een zelfstandige grafische eenheid, met een eigen lokaal coördinaten systeem, een eigen pen positie, pen kleur etc. Voor we de "Hello world" tekst in het window kunnen zetten moeten we daarom eerst de pen positioneren met een "MoveTo" call (je raadt het al, ook een toolbox routine). Daarna kan de gewenste tekst in het window geschreven worden. Tenslotte wordt de growbox getekend met de toolbox routine DrawGrowIcon. Deze call heeft een windowpointer als parameter nodig, waarvoor we natuurlijk onze lokale kopie van event.message weer gebruiken.

Hiermee hebben we alle benodigde code gehad, de cleanup procedure is alleen voor de volledigheid opgenomen. Omdat we zelf geen geheugen gereserveerd hebben valt er ook niets op te ruimen. Het systeem zorgt automatisch voor het vrijgeven van de ruimte die gebruikt is voor het windowrecord enz.

Wanneer het hele programma nu bekeken wordt kunnen een aantal conclusies getrokken worden: Het overgrote deel bestaat uit overhead van de windowmanagement en gebruikersinterface. Dit is vrijwel altijd zo, gebruiksvriendelijkheid moet nu eenmaal betaald worden. Daartegenover staat dat het programma eigenlijk niet meer dan een reeks toolboxcalls is, hetgeen het schrijven zeer vereenvoudigd. Verder kunnen met betrekkelijk weinig moeite toevoegingen gedaan worden, zonder dat aan de basis structuur gesleuteld hoeft te worden. Om bij ons voorbeeld te blijven, door het toevoegen van een "DoInContent" procedure in de DoMousedown routine kan een reactie op klikken met de muis in ons window geïmplementeerd worden.

Tenslotte moet nog opgemerkt worden dat de hier beschreven principes niet uniek zijn voor de Mac, andere "Graphical User interfaces" (GUI's) zoals OS/2's Presentation manager, Microsoft Windows, maar ook een op workstations gangbaar systeem als X-windows, zijn Event Driven. gezien de grote en nog steeds groeiende invloed die de GUI's loont het dus zeker de moeite je eens wat verder in deze materie te verdiepen.

Literatuur

Inside Macintosh Volume 1-5, Addison-Wesley, 1985. ISBN 0-201-17731-5
De bijbel van iedere Macintosh programmeur, absoluut onmisbaar wanneer je werkelijk wat wilt doen op de Mac.

Programmers introduction to the Macintosh family. Addison-Wesley, 1988. IBSN 0-201-19254-3
Een goede inleiding, beschrijft de basistechnieken en de ideeen achter de gebruikersinterface.

Knaster, S. (1987) Macintosh programming secrets. Adisson-Wesley. IBSN 0-201-06661-0
Dit boek, vol van merkwaardige humor, beschrijft minder bekende technieken en mogelijkheden, het veronderstelt wel enige kennis van de Mac en de toolbox.

Zeer aanbevolen.

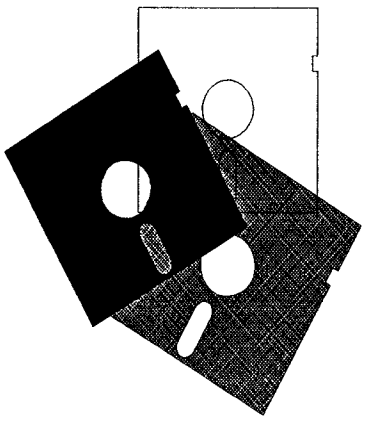
Peter Roessingh

```
Procedure DoUpdate;

var
    window: Windowptr;              { window to update      }

begin
    window: = event.message;        { window to update      }
    setPort(window);                { draw in this window   }
    MoveTo(20,20);                  { position pen          }
    BeginUpdate(window);            { clears event          }
        Drawstring('I hate IBM');   { print string          }
        DrawGrowIcon(window);       { draw growbox          }
    EndUpdate(window);
end {DoUpdate}
```

Fig. 7: DoUpdate routine

To Share Or Not To Share, That's The Question

De HCC-dagen zijn net voorbij. Ik heb mij, net als vele anderen, wat nieuwe apparatuur aangeschaft, waaronder een nieuw AT-moederbord. Het ding werd mij verkocht als een 16 Mhz moederboard, maar op het kristal staat "25.5 MHz". Jawel, 16 MHz Landmark natuurlijk. De vraag die onmiddellijk rees was dan ook hoe snel het ding werkelijk zou zijn. De doos met floppen werd er dus maar eens bij gepakt en de verschillende testen bestudeerd. Bij de beoordeling van het moederbord vielen in de verschillende testen redelijk grote afwijkingen te constateren. Reden genoeg om eens een blik te werpen op de testtechnieken van de beroemde benchmarks en eens te zien wat er in het PD/Shareware circuit aan testen beschikbaar is.



Bij een snelheidstest moeten we van tevoren vragen stellen omtrent de testmethoden. "Wat gaan we testen" en "Hoe moet een representatieve test er uitzien?" zijn de belangrijkste twee aandachtspunten. De verschillende testen zijn onder te verdelen in drie categorieën: er zijn microtesten, macrotesten en de applicatietesten. De microtesten geven testresultaten weer over verschillende kleine onderdeeltjes van het systeem. De macrotesten geven een oordeel over het gehele systeem op basis van microtesten en de applicatie gerichte testen geven een waardeoordeel over het systeem met betrekking tot bepaalde applicaties. Helaas blijken in de praktijk de populairste testen in de groep "macrotesten" te vallen. Helaas, want vaak wordt het testresultaat opgehangen aan een te beperkt aantal snelheidsbepalende factoren om de test als zinvol te bestempelen. Een goed voorbeeld van een dergelijke test is Norton's SysInfo test. Deze is gebaseerd op een klein aantal multiply- en divideinstructies die in een dubbele lus opgenomen zijn. De test geeft hooguit wat informatie over de rekenkundige snelheid van de processor. Dat is de reden waarom bijvoorbeeld de processoren uit de V20- en V30 series van NEC zo snel lijken te zijn; NEC heeft de divide en multiply routines in hardware gebouwd in plaats van in microcode omschreven. Ook andere, meer gerenommeerde testen lijden aan dit euvel. Bekend is bijvoorbeeld de Landmark speed-test, waar veel fabrikanten en "vakhandels" mee adverteren. Een gunstige uitzondering op de wildgroei in benchmarks is het totaaltestpakket "Checkit". Checkit checkt niet alleen de snelheid van het apparaat maar vooral het functioneren van de verschillende onderdelen. De benchmarks zijn slechts kadootjes die "per ongeluk" in dezelfde verpakking opgenomen zijn.

In de PD- en sharewaresfeer zijn een aantal bijzonder goede testen te vinden. Voor het testen van een moederbord is vooral de CPU-snelheid belangrijk en de mate waarin de CPU met het geheugen en over de bus met randapparatuur babbelt. Voor wetenschappelijke toepassingen kan het ook belangrijk zijn eventuele floating point co-processoren te testen. Helaas ben ik niet zo rijk dat ik mij een FPU kan veroorloven.

De benchmarks op de testbank zijn de testprogramma's van PC-Magazine (PC-Lab), de Byte en een kleine CPU-benchmark van Richard B. Johnson ("Design"). Deze laatste bekijkt slechts de verschillende soorten instructies t.o.v. de PC/XT. De resultaten worden gegeven in clockticks en omgerekend naar snelheidspercentages ten opzichte van de originele PC. Het eindoordeel is een overall-percentage à la Norton. De grote afwijkingen in de eindresultaten (test-noise) duiden op een slechte meetmethode en het programma valt voor mij dan ook af als betrouwbare benchmark.

Van geen der genoemde programma's is een source beschikbaar, wat best jammer is daar een source veel duidelijk kan maken over de gebruikte testmethoden. Nu moeten we er maar van uitgaan dat de gebruikte algorithmes ook daadwerkelijk zin hebben en geen ongewenste effecten opleveren van de toegepaste compilers.

De Byte-testen

Het testprogramma van de Byte is ondergebracht in één grote uitvoerbare file en een aantal kleine datafiles. Het programma draagt versienummer 2.1 en is gedateerd "augustus 1990". De test is opgedeeld in een viertal testcategorieën: CPU, floating point, disk en video.

De CPU testen behelzen de beroemde zeef van Eratosthenes, een tweetal sorteer algorithmes (quicksort en shellsort), een integer berekening en een byte-georiënteerde stringmove test. De testen staan

uitgebreid omschreven in de Byte van juni 1988. In deze omschrijving wordt ook melding gemaakt van word/odd en word/even stringmove testen. Deze zijn echter in versie 2.1 niet meer terug te vinden.

Voor de floating point testen is een 80x87 processor nodig. Daar ik die niet had (en nog steeds niet heb... uhhh - stille hint?) kan er over deze test geen uitsluitsel gegeven worden. De Byte omschrijft ze zelf als "math error", "sine(x)" en "exponential" testen.

Disks kunnen ook getest worden door de Byte. De testen omvatten file I/O lees- en schrijftesten, "actual throughput" en een seektest. De file I/O testen hebben veel diskruimte nodig; de vier vrije Megabytjes op de testmachine waren niet toereikend om de test goed te laten verlopen. De DOS file I/O gaf resultaten... daar sla je stijl van achterover: 600 kB per seconde! En dat terwijl het ding (een RLL-controller) slechts kans ziet zo'n 270 kB per seconde te verstouwen... De low-level testen leverden slechts machine-crashes op (die niet het gevolg zijn van de programmatuur maar van een aantal te langzame RAMmetjes in de machine).

De video test van byte is wel aardig. Het ding zoekt zelf uit wat voor videoadapter in de machine hangt en gebruikt zijn eigen routines voor graphics modes. De textmodes worden zowel via BIOS als met behulp van direct screen I/O getest.

De low-level testen leverden slechts machine-crashes op.

De eindresultaten kunnen naar believen opgeslagen worden in een file of naar de printer gestuurd worden. Daar dit soort testen eigenlijk alleen interessant is in vergelijking met andere apparaten is het belangrijk dat de programmatuur zelf vergelijkingsmogelijkheden biedt. Bij de benchmarks van de byte kunnen twee vergelijkingsmachines gekozen worden uit een zestal mogelijkheden. De voorgeprogrammeerde resultaatmachines doorlopen het hele scala van PC/AT-8 tot en met een 25-MHz 80486 doos.

PC-Lab

PC-Magazine heeft een eigen testprogramma ontwikkeld met de naam "PC-Lab". De versie die hier op de testbank ligt draagt versienummer 5.5. Het hele programma is ondergebracht in verschillende files waarvan de oudste uit oktober 1989 komt. De nieuwste files stammen uit mei 1990. De opbouw doet vermoeden dat het pakket een muis ondersteunt, maar door hardware problemen kon dat helaas niet nagegaan worden. Het hoofdscherm van PC-Lab is een beetje rommelig; de menupunten staan boven in beeld met een hoop nutteloze informatie eronder. Het hoofdmenu biedt de keuze uit de opties "File", "View", "Performance", "Compatibility", "Quality", "Set" en "Help". De echte benchmarks hangen hier onder het performance menu. De andere opties dienen voornamelijk voor het instellen van parameters en het bewaren van testresultaten. De Compatibility test doet vermoeden dat er nog getest kan worden op compatibiliteit, maar helaas: alleen VGA hardware! Waar het ons om gaat is voornamelijk het performance submenu. Dit submenu heeft als keuzemogelijkheden: "Processor", "Coprocessor", "Disk", "Video", "Memory", "Printer" en "Battery Rundown". Op zich geeft ieder menupunt weer een sub-submenu, waarin gekozen kan worden tussen de echte tests. Voor de processor is dat bijvoorbeeld een instructie-mix, een 128kB NOP-loop, wat integer- en floating point berekeningen, string handling en meer van dat soort rommel.

De testen doen zo sterk denken aan de testen van Byte dat men geneigd is het als "kopieerwerk" af te doen. Alleen de screen-layout en menuindeling is verschillend. Verder heeft PC-Magazine een printer-test en een batterij-test voor portables verzonnen.

Naast de processortest is er ook een disk-test. Deze is behoorlijk uitgebreid, met name op het DOS-vlak. Er kan lekker ge-experimenteerd worden met verschillende record-grootten en het aantal records dat geschreven, gelezen of gezocht moet worden. Deze aanpak heeft tot gevolg dat de combinatie controller/harddisk veel meer uiteengerafeld wordt dan bij de Byte testen het geval is. De resultaten van deze test zijn met name interessant als men van plan is de machine te versnellen door extra hardware toe te voegen of bestaande hardware te vervangen. In de testmachine bleek bijvoorbeeld de harddisk een blok aan het been te zijn. Dat klopt ook wel, want die komt uit een XT en is zeker geen wonder van snelheid. Dit verschijnsel komt met name uit de doeken als er grote records sequentieel weggeschreven worden; dan is de disk eenheid opeens bloedsnel. Moet er veel gezocht worden, dan wordt hij akelig traag.

Het videodeel doet ook bijzonder sterk denken aan de Byte testen. Er wordt flink gescrollt door het BIOS en ook de programmering op de kale hardware wordt niet vergeten. Jammer is wel dat de grafische mode niet even getest kan worden van Hercules. Voor alle andere soorten schermen staat er een uitgebreid scala aan tests ter beschikking, maar de Hercules komt er wat magertjes af.

Besproken produkt	: Byte benchmarks
Categorie	: Test programmatuur
Registratie	: Geen (public domain?)
Auteur/Leverancier	: Byte Magazine
Verkrijgbaarheid	: The Ultimate, utility area Bbench21.Zip, ca. 51 kB.

Minimale systeemeisen

Niet gespecificeerd

Algemene beoordeling

Documentatie	: Geen. Uitgebreid toegelicht in "Byte", juni '88.
Online help	: Redelijk, voor ieder menupunt is er een korte toelichting.
Gebruikersinterface	: Keyboard, ouderwetse keuze menu's
Muisondersteuning	: Geen.

Positief

Compact
Breed scala aan testen beschikbaar
Goed vergelijkingsmateriaal
Test ook Hercules graphics mode

Negatief

Geen documentatie
Te weinig informatie over vergelijkings systemen
Geen floating point testen zonder 80x87 mogelijk

Eindbeoordeling

Stabiliteit	: 7
Bruikbaarheid	: 8
Totaalresultaat	: 8

Het programmeren van de 8088 in de IBM (Deel 3)

In deze aflevering zien we de source van HEXDUMP, een programma dat van elke willekeurige file een hex-dump afdrukt. De bedoeling is dat we het programma al lezend en begrijpend invoeren, assembleren en testen. We gaan het nu niet regel voor regel behandelen, maar besteden onze aandacht aan een groot aantal nieuwe verschijnselen.

Symbolen van het type TEXT

Als een bepaalde operand steeds weer moet worden gebruikt, is het handig hem te vervangen door een symbool van het type TEXT. Zo kennen we intussen de operatoren SEG en OFFSET. Waarom niet OFS voor OFFSET? Dat is een kwestie van smaak natuurlijk, maar als we het met die lange kreet niet eens zijn, dan veranderen we dat gewoon! U ziet hoe dat kan door OFS met een EQU directive equivalent te verklaren aan OFFSET

De JMP instructie

We maken nu kennis met de onvoorwaardelijke sprongopdracht, de JMP. Veel valt er niet meer over te vertellen. In de vorige aflevering hebben we de CALL leren kennen en JMP kent exact dezelfde adresseermethoden. Ook nu blijft de JMP FAR nog even buiten beschouwing. Naast de onder CALL besproken mogelijkheden kent de JMP instructie echter nog een speciale adresseermogelijkheid. Het is de onvoorwaardelijke versie van de relatieve sprongopdrachten en het doeladres moet dus binnen -128 of +127 bytes liggen. Als een onvoorwaardelijke sprong terug moet gaan behoeft de programmeur zich niet druk te maken over het kiezen van de meest efficiente JMP. De assembler doet dat voor hem. Is de sprong voorwaarts, dan reserveert de assembler echter altijd 3 bytes gedurende pass 1 en vult daar dan zomogelijk een twee bytes relative versie in waarbij het derde byte met een NOP (No OPeration) instructie wordt opgevuld. De programmeur kan dit voorkomen door i.p.v. een JMP MY_LABEL te schrijven: JMP SHORT MY_LABEL. Daarbij neemt hij het risico van een foutmelding al het doeladres te ver weg ligt. TASM biedt de mogelijkheid de assembler te laten waarschuwen als een JMP kan worden vervangen door een JMP SHORT. Overigens komt u deze JMP short in het programma HEXDUMP tegen onder een eigen mnemonic: JMPS.

Een eenvoudige macro

We hebben de in de vorige paragraaf genoemde JMP SHORT wat lelijk gevonden en zelf een nieuw mnemonic voor JMP SHORT gedefinieerd. Een macrodefinitie begint met de macronaam, in dit geval dus JMPS, gevolgd door het woord MACRO, gevolgd door de parameters van de macro. Hier is slechts een parameter TARGET gespecificeerd, als dummy operand. Deze parametersymbolen zijn volkomen lokaal binnen de macrodefinitie en behoeven dus naar buiten niet uniek te zijn. ENDM geeft het einde van een macrodefinitie aan.

Diverse conditionele sprongen

De 8088 kent een uitgebreid scala van conditionele sprongopdrachten. De belangrijkste zijn:

JZ	if zero.(Indien de ZF vlag gezet is)
JC	if carry.(Indien de CF vlag gezet is)
JS	if sign.(Indien de SF vlag gezet is)
JP	if parity.(Indien de PF vlag gezet is)
JO	if overflow.(Indien de OF vlag gezet is)
JAE	if above or equal.

Van al deze sprongopdrachten bestaan ook de tegenovergestelde versie die we simpel schrijven met een N achter de J, dus JNZ,JNC ezv. Bovendien hebben al deze mnemonics wel een handig alias. Zo bestaat voor JZ ook JE (Equal). Handig zijn JB (Jump if below) en JA (Jump if above). Een buitenbeentje is de JCXZ die we in de vorige aflevering al leerden kennen. Het gaat te ver om alle mogelijkheden hier te noemen. Bestudeer de instructieset zoals deze is opgenomen in de handleiding van de assembler.

Het hoofdprogramma

We treffen dezelfde elementen aan die we in de vorige aflevering leerden kennen. Initialiseren van DS, Overtollig geheugen teruggeven en later het terugkeren naar DOS. Bovendien slaan we nu de paragraaf waarop het PSP begint op in een variabele. Het hoofdprogramma bestaat verder uit de aanroep van twee procedures. GETFILE leest de parameter uit het PSP en DUMPFILE opent, dumpt en sluit de file gespecificeerd door die parameter.

Primitieve stringinstructies

Bij het ophalen van de parameter uit het PSP maken we gebruik van een krachtig mechanisme in 8088 om geheugenblokken te benaderen. Allereerst zijn daar de instructies LODS (LOaDString) en STOS (STOreString). LODS leest een byte of woord in AL maar doet dat op een krachtige manier. Het byte of woord wordt namelijk door DS:SI geindexeerd. In feite komt de LODS instructie voor een byte overeen met de navolgende opeenvolging van instructies:

```
MOV AL,[SI]
INC SI
```

Evenzo komt de LODS instructie voor een woord overeen met navolgende opeenvolging:

```
MOV AX,[SI]
ADD SI,2
```

Na het lezen van een byte of woord wordt dus automatisch het indexregister SI verhoogd. MASM kent twee verschillende mnemonics voor deze instructie: LODSB en LODSW om resp. een byte of een woord te lezen. Nu hebben we aangenomen dat SI steeds opgehoogd wordt, maar het is ook mogelijk om "achterstevoren" te lezen. De leesrichting wordt bepaald door de DF vlag in het statusregister. Normaal dien DF=0 te zijn en we moeten dat zekerstellen met de instructie CLD (CLear Direction flag). Om te bereiken dat na elke LODS indexregister SI met een of twee wordt verminderd, moeten we de "direction flag" juist zetten met de instructie STD (SeT Direction flag). Om een byte of woord na eventuele bewerking weer ergens op te slaan dient de tegenhanger van LODS die STOS heet (STOre String). In MASM weer STOSB of STOSW. Deze instructie werkt niet met het SI maar met het DI register. Hieraan danken deze registers dan ook hun namen "Source Index" en "Destination Index". De STOS instructie behoort tot de buitenbeentjes die ik in de eerste aflevering aankondigde. STOS slaat namelijk geen data op in het datasegment maar in het extrasegment! Derhalve is de STOSB instructie equivalent aan:

```
MOV ES:[DI],AL
INC DI
```

De LOOP instructie

Een instructie die alles te maken heeft met het CX register in zijn functie van "counter" is de LOOP instructie. Het is in feite een conditionele sprong die eerst het CX register met een verminderd waarna de sprong wordt uitgevoerd zolang CX ongelijk 0 is. Hierbij heeft geen van de vlaggen in het statusregister enige invloed. Anders is dat bij de variant LOOPZ (of LOOPE welke een alias voor LOOPZ is) die bovendien eist dat de ZF vlag gezet is. LOOPZ beëindigd dus een loop indien CX=0 of indien ZF=0. LOOPZ heeft als tegenhanger LOOPNZ (Of LOOPNE) die de loop eveneens pas afbreekt als CX=0 maar eveneens stopt als ZF=1. Op een rijtje:

```
LOOP      Totdat CX=0
LOOPZ     Totdat CX=0 OF ZF=0
LOOPNZ    Totdat CX=0 OF ZF=1
```

Combinatie van LODS en STOS

Het zal voorkomen dat we data willen verplaatsen van een locatie in het datasegment naar een locatie in het extrasegment. Dat zou kunnen met de opeenvolging van LODS en STOS. De opeenvolging van deze twee is echter ter beschikking als een krachtige instructie: MOVS. Ook hier weer twee aparte mnemonics voor byte of woord: MOVSB en MOVSW. MOVSB is dus equivalent aan:

```
MOV AL,[SI]
INC SI
MOV ES:[DI],AL
INC DI
of aan:
LODSB
STOSB
```

Stel nu dat we een aantal bytes aangegeven in CX willen verplaatsen. Dat zou dan als volgt kunnen:

```
MOVE_BLCK: MOVSB
LOOP       MOVE_BLCK
```

Maar speciaal voor deze loop bestaan er versies van de LOOP instructie die met elke primitieve stringinstructie kan worden gecombineerd. Dat zijn de instructies: REP, REPZ en REPNZ welke gelijk zijn aan de overeenkomstige LOOP opdrachten doch bovendien gecombineerd worden met een primitieve string instructie. Nadat SI,DI,ES,CX en de DF vlag zijn geïnitialiseerd bestaat zo'n loop dus kortweg uit:

```
REP
MOVSB
```

en toegestaan is:

```
REP MOVSB
```

De procedure GETFILE

De procedure GETFILE in het programma HEXDUMP maakt gebruik van primitieve stringinstructies om de naam van de te dumpen file uit de command tail in het PSP te halen. Merk op dat de "source" in dit geval niet het datasegment is en dat de "destination" nu juist wel het datasegment is. ES moet dus op het datasegment worden gericht, terwijl we DS tijdelijk op het PSP moeten richten. Dit is levert een gevaarlijke valkuil op waar al heel wat programmeurs ingevallen zijn. We zijn namelijk gewend een variabele in het datasegment te adresseren als waren we een oude 8 bits processor aan het programmeren. Als de paragraaf in DS niet op dat datasegment wijst gaat die vlieger niet meer op! Verander de waarde in DS dus niet voordat dat echt nodig is en zet hem weer terug zodra dat mogelijk is. Als de naam van de te dumpen file is opgehaald zet-

ten we er nog een 0 achter omdat de ontwerpers van MS-DOS dachten dat 00H of NUL de betekenis van einde string heeft. We moeten er maar mee leven. Als geen parameterstring blijkt te zijn gespecificeerd keert de procedure terug met CF=1, anders met CF=0. Het gebruik van de carry als foutvlag is in de MS-DOS omgeving een standaard geworden.

De procedure DUMPFILE

Voor het eerst openen we nu een file. DOS-functie 3DH opent alleen bestaande files. In AL wordt aangegeven of de file moet worden geopend om te lezen (AL=0) of om te schrijven (AL=1) of beiden (AL=2). Sedert DOS versie 3.00 zijn er nog een aantal andere mogelijkheden die buiten het bestek van deze artikelen vallen. DS:DX moet op de file naam wijzen (Path). Na aanroep van de functiemanager geeft de carry aan of alles naar wens is verlopen. CF=1 is fout en in dat geval staat de foutcode in AX. Anders staat de handle die DOS aan de file heeft toegekend in AX. Met functie 3FH kunnen we de file lezen. Het gewenste aantal bytes moeten we daartoe aangeven in CX, BX moet de handle bevatten en DS:DX moet aangeven waar de te lezen bytes naartoe moeten. Weer geeft CF=1 aan dat er een foutcode in AX staat. Anders bevat AX het aantal bytes dat daadwerkelijk werd gelezen. Is dat minder dan het gevraagde aantal in CX, dan waren dit de laatste bytes van de file. Door met het handle nummer in BX functie 3EH aan te roepen sluiten we de file. De procedure leest steeds 16 bytes en dumpt dan een regel.

Enkele standaard routines

De procedure WRWORD kent een aantal externe ingangen waar de aanhangers van "gestructureerd programmeren" van zullen gruwen. Deze verzameling routines om een woord, byte of digit in hex af te drukken komt in dezelfde vorm in veel andere software voor evenveel andere processors voor. Er valt niets aan te wijzigen of te onderhouden dus de kans op problemen is nihil. Daartegenover staat compactheid en vooral snelheidswinst. In de routine WRBYT word vier maal achtereen een register 1 bit naar rechts geschoven. Dit zou ook kunnen door het met CL=4 een enkele SHR AL,CL uit te voeren. I/O routines die het CX register aantasten zijn echter niet gezien. Volgens de DOS specificaties worden door de functiemanager geen andere registers dan AX veranderd. Dat wil zeggen dat alle registers ofwel gedefinieerd zijn ofwel onveranderd zijn met uitzondering van AX. Indien AX niet is gespecificeerd is AX ongedefinieerd!

Merk tenslotte op dat we functie 02H gebruiken om een karakter naar het Standaard Output Device te sturen. Deze functie is een niet meer aanbevolen oudgediende uit het CP/M tijdperk. Functie 40H is echter voor het verzenden van een enkel teken buitengewoon omslachtig. In volgende afleveringen zullen we overigens aandacht gaan besteden aan I/O buiten DOS om via de BIOS.

Ruud Uphoff

Fig. 1: assembler brontekst van het programma HEXDUMP.EXE

```
                NAME    HEXDUMP

COMMENT @       Het programma HEXDUMP drukt van een willekeurige file een
                hexdump af op het "Standard Output Device"
                @
                ;---------------------- CONSTANTEN ----------------------------
DISPHAND        EQU     2               ;Handle naar display
CR              EQU     13              ;ASCII voor CR
LF              EQU     10              ;ASCII voor LF
OFS             EQU     OFFSET          ;Geen zin om voluit OFFSET te tikken

JMPS            MACRO TARGET            ;JMPS bespaart ook type werk
                JMP     SHORT TARGET
                ENDM

                ;---------------------- DATASEGMENT ---------------------------
DATA            SEGMENT

PSPSEG          DW      (?)             ;Paragraaf van het PSP segment
```

```
FILEOFS     DW      (?)             ;Offset in de te dumpen file
BYTES       DW      (?)             ;Aantal gelezen bytes uit de file
BUFFER      DB      128 DµP (?)     ;Buffer voor diverse toepassingen
ERRMSG      DB      ENDMSG-ERRMSG-1                 ;Foutmelding
            DB      'Missing file name or file not found',CR,LF
ENDMSG      EQU     $
DATA        ENDS

            ;---------- CODESEGMENT ----------
CODE        SEGMENT
            ASSUME CS:CODE, DS:DATA, SS:PGMSTACK

            ;Procedure GETFILE haalt de parameter ( naam van de file die
            ;moet worden gedumpt ) uit het PSP. De parameters staan in
            ;een string op offset 80H in het PSP. Deze wordt COMMAND TAIL
            ;genoemd.

GETFILE     PROC    NEAR
            PUSH    DS              ;ES:DI naar buffer laten wijzen
            POP     ES
            MOV     DI,OFS BUFFER   ;DI is nu echt "Destination Index"
            PUSH    DS              ;DS bewaren
            MOV     SI,80H          ;En DS:SI op PSP:80H richten
            MOV     AX,PSPSEG       ;Verander DS daarbij het laatst!
            MOV     DS,AX           ;SI in nu echt de "Source Index"
            CLD     ;Richtingsvlag wissen
            LODSB   ;Lengte van de parameter uit PSP halen
            MOV     CL,AL           ;en naar CX overbrengen
            XOR     CH,CH
UNLEAD:     STC     ;Foutflag zetten voor als het mis gaat
            JCXZ    NONAME          ;Lengte 0 ? Dat is dus goed fout!
            LODSB   ;Lees anders een karakter
            DEC     CX              ;en tel het
            CMP     AL,20H          ;Een voorafgaande spatie ?
            JZ      UNLEAD          ;daar hebben we niets aan
            STOSB   ;Hoera! het eerste goede karakter
            REP     MOVSB           ;Dan willen we de rest ook hebben
            XOR     AL,AL           ;ASCIIZ terminator toevoegen
            STOSB
            CLC     ;Foutflag wissen
NONAME:     POP     DS              ;Altijd DS terughalen
            RET
GETFILE     ENDP

            ;CRLF op het Standard Output Device
CRLF        PROC    NEAR
            MOV     AL,CR           ;Print CR
            CALL    CHAROUT
            MOV     AL,LF           ;en dan LF
            JMPS    CHAROUT         ;Equivalent aan CALL CHAROUT, dan RET
CRLF        ENDP

            ;Drie spaties naar het Standard Output Device
BLANK3      PROC    NEAR
            CALL    BLANK           ;Druk een spatie
```

```
                ;Externe ingang: Twee spaties naar het Standaard Output Device
BLANK2:         CALL    BLANK           ;Druk een spatie

                ;Externe ingang: Spatie naar het Standaard Output Device
BLANK:          MOV     AL,20H
                JMPS    CHAROUT
BLANK3          ENDP

                ;Schrijf het woord in AX als 4 hex-digits
WRWORD          PROC    NEAR
                PUSH    AX              ;Word van AX bewaren
                MOV     AL,AH           ;AH naar AL
                CALL    WRBYT           ; en afdrukken
                POP     AX              ;Word terughalen en AL afdrukken

                ;Externe ingang WRBYT in routine WRWORD
                ;schrijft het byte in AL als twee hex-digits
WRBYT:          PUSH    AX              ;AL bewaren
                SHR     AL,1            ;Hoogste digit naar recht schuiven
                SHR     AL,1
                SHR     AL,1
                SHR     AL,1
                CALL    DIGOUT          ;en hex-digit afdrukken
                POP     AX              ;Byte terughalen van stack
                AND     AL,0FH          ;Hoogste digit weggooien

                ;Externe ingang in WRWORD en WRBYT
                ;Schrijft het hex-digit in AL bit 0..3 als karakter
DIGOUT:         CMP     AL,0AH          ;0A..0F,
                JB      DECDIG
                ADD     AL,7            ;vereist correctie met 7 voor ASCII
DECDIG:         ADD     AL,30H          ;Altijd ASCII voor '0' optellen

                ;Externe ingang in WRWORD, WRBYT en DIGOUT
                ;Schrijft het karakter in AL op het Standard Ouput Device
CHAROUT:        PUSH    DX
                MOV     DL,AL           ;Karakter naar DL
                MOV     AH,02H          ;DOS functie 2 schrijft karakter in DL
                INT     21H
                POP     DX
                RET
WRWORD          ENDP

                ;Open de file, dump in hex en sluit hem.

DUMPFILE        PROC    NEAR
                MOV     DX,OFS BUFFER ;DS:DX wijst naar de ASCIIZ naam
                XOR     AL,AL           ;AL = 0 voor openen in 'Read only' modus
                MOV     AH,3DH          ;DOS Functie 3DH opent de file
                INT     21H
                JC      ERREXIT
                MOV     BX,AX           ;Handle in BX
                MOV     FILEOFS,-16     ;Offset teller in de file initialiseren

NEXT16:         ADD     FILEOFS,16      ;Offset in de file bijwerken
                MOV     AX,FILEOFS      ; in AX halen
```

```
                CALL    WRWORD          ; en afdrukken
                CALL    BLANK2          ; gevolgd door twee spaties
                MOV     CX,16           ;We vragen om 16 bytes uit de file
                MOV     DX,OFS BUFFER   ; in het buffer
                MOV     AH,3FH          ;Functienummer voor "file lezen"
                INT     21H
                JC      ERREXIT
                MOV     BYTES,AX        ;Aantal gelezen bytes noteren
                MOV     CX,AX           ;en als teller in CX zetten
                JCXZ    DMPDONE         ;Geen verdere actie bij einde file
                MOV     SI,OFS BUFFER   ;DS:SI wijst naar het buffer

NXTBYT:         LODSB   ;byte uit buffer halen
                CALL    WRBYT           ; en afdrukken
                CALL    BLANK           ; gevolgd door een spatie
                LOOP    NXTBYT          ;en dat CX keer
                MOV     CX,16           ;Bereken verschil 16-aantal bytes
                SUB     CX,BYTES
                JZ      DOASCII         ;indien geen 16 (laatste regel)

MATCH:          CALL    BLANK3          ;Dan aanvullen met 3 spaties
                LOOP    MATCH           ; voor elk ontbrekend byte

DOASCII:        MOV     CX,BYTES        ;Opnieuw aantal bytes in CX
                MOV     SI,OFS BUFFER   ;Opnieuw DS:SI naar buffer laten wijzen
NXTCHAR:        LODSB   ;bytes uit buffer halen
                CMP     AL,20H          ;Indien kleiner dan ASCII voor spatie
                JAE     NALTER
                MOV     AL,'.'          ;dan in punt veranderen
NALTER:         CALL    CHAROUT         ;Deze keer afdrukken als karakter
                LOOP    NXTCHAR         ;en dat weer CX keer
                CALL    CRLF            ;CRLF na elke regel van 16
                CMP     BYTES,16        ;Waren het er 16 ?
                JZ      NEXT16          ;dan volgende 16

DMPDONE:        CALL    CRLF            ;Extra CRLF ter afsluiting
                MOV     AH,3EH          ;Met functie 3EH de file sluiten
                INT     21H
                RET
DUMPFILE        ENDP

                ;Foutmelding afdrukken

ERREXIT         PROC    NEAR
                PUSH    AX              ;Bewaar foutcode van AL
                MOV     DX,OFS ERRMSG+1         ;DS:DX wijst naar foutmelding
                MOV     CL,ERRMSG       ;Lengte van de foutmelding in CX
                XOR     CH,CH
                MOV     BX,DISPHAND     ;Handle naar display in BX
                MOV     AH,40H          ;Foutmelding afdrukken
                INT     21H
                POP     AX              ;Foutcode in AL achterlaten
                MOV     AH,4CH          ;en programma afbreken.
                INT     21H
ERREXIT         ENDP
```

```
                ;Het hoofdprogramma

DUMP            PROC    NEAR
                MOV     AX,SEG DATA     ;DS initialiseren
                MOV     DS,AX
                MOV     PSPSEG,ES       ;Paragraaf van het PSP bewaren
                MOV     BX,SEG ZZZZZZZZ                 ;Overtollig geheugen vrijgeven
                MOV     AX,ES
                SUB     BX,AX
                MOV     AH,4AH
                INT     21H
                CALL    GETFILE         ;Filenaam van commando ophalen
                JC      ERREXIT         ;Geen naam? Dan foutmelding en afbreken
                CALL    DUMPFILE        ;File openen, dumpen en weer sluiten
                MOV     AX,4C00H        ;Terug naar DOS met foutcode 0
                INT     21H
DUMP            ENDP

CODE            ENDS

                ;---------------------- STACKSEGMENT ---------------------------

PGMSTACK        SEGMENT STACK
                DW      100H DUP (?)
PGMSTACK        ENDS

                ;---------------------- VRIJ GEHEUGEN --------------------------

ZZZZZZZZ        SEGMENT
ZZZZZZZZ        ENDS
```

Ik heb interesse in de KGN en wil

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

Naam : _____

Adres : _____

Postcode en woonplaats : _____

Datum : _____ Handtekening : _____

Dit strookje kunt u ingevuld opsturen aan het secretariaat van: KIM Gebruikersclub Nederland
Postbus 99650
1000 NA Amsterdam

KGN-68k MINIX systeem

Rapportage project-groep

Inleiding

Enkele maanden geleden is vanuit het bestuur het idee gekomen "iets" met MINIX te gaan doen. Dat iets zou dan in eerste instantie moeten bestaan uit het leggen van contacten met de MINIX gebruikersgroep om te onderzoeken of MINIX iets voor de KGN zou kunnen zijn. Uit deze contacten kwam naar voren dat het voor de KGN zeer interessant zou kunnen zijn MINIX te gaan ondersteunen. Het grote voordeel van MINIX is namelijk dat het operating systeem op diverse hardware-platforms draait zoals bijvoorbeeld IBM-compatibles, Amiga, Atari-ST, Macintosh etc. Gezien de veelvoud van systemen die binnen onze club aanwezig is, is dat uiteraard een groot voordeel. Een tweede aspect is dat er voor MINIX veel software beschikbaar is terwijl MINIX toch nog heel veel vrijheid geeft voor het zelf aanpassen van het systeem aan specifieke wensen. Verder is het een groot voordeel dat een belangrijk deel van de software in source code beschikbaar is; kortom mogelijkheden genoeg om de soft- en eventueel de hardware naar eigen inzichten te veranderen.

Nadat de eerste contacten gelegd zijn, heeft Joost Voorhaar, in samenwerking met Fred van Kempen een voordracht over MINIX gehouden op de clubbijeenkomst in Haarlem in september j.l. Op deze bijeenkomst is het idee geboren, behalve met de MINIX-software ook iets aan de ontwikkeling van speciale hardware voor MINIX te gaan doen. De eerste ideeën waren nogal wild. Gedacht werd aan een systeem met een super-processor (een 680x0 of 80x86 waarbij x minimaal een 2 maar nog liever een 3 of 4 was) draaiend op heel veel megahertzen, losse geheugenkaarten met meerdere megabytes per kaart, SCSI-drives en losse seriële kaarten voor de communicatie met dit systeem. Kortom, eigenlijk een relatief groot UNIX-systeem, gebaseerd op een VME-bus. Een dergelijk systeem kan uiteraard in de winkel gekocht worden voor pakweg fl. 15.000,-. Het zelf bouwen van een dergelijk systeem is nog veel duurder en voor het gemiddelde clublid niet haalbaar.

Op een bestuursvergadering in oktober zijn we toch nog eens verder gegaan met het brainstormen. Het bestuur heeft de indruk dat het voor de toekomst van de club zeer interessant kan zijn weer eens een wat groter hardware-project te starten. Verder bestaat er al een aantal jaren de wens eens een CPU-kaart te ontwikkelen en bouwen. Nu ben je er natuurlijk niet met alleen een CPU-kaart. De kaart moet uiteraard hulp krijgen van onder andere disk controllers en I/O-kaarten. Nu kun je die natuurlijk ook allemaal mee ontwikkelen alleen heeft dat als nadeel dat je wel heel veel in één keer moet ontwerpen en bouwen. Verder is ook het kostenaspect niet verwaarloosbaar. De door de KGN ontwikkelde hardware moet concurreren met goedkope PC-hardware uit het verre oosten. Al brainstormend kwam het bestuur op het idee het beste uit twee werelden te combineren. We maken gebruik van een deel van de goedkope hardware uit de PC-wereld en ontwikkelen daarbij een aantal kaarten, waaronder een processorkaart, die speciaal toegesneden zijn op het MINIX-systeem. Op deze manier zijn we gekomen op de AT-bus. Dat wil zeggen dat de AT-bus in het systeem gebruikt gaat worden om de processor te laten communiceren met de kaarten uit de PC-wereld. Dit kan dan op twee manieren gedaan worden. In de eerste plaats is het bij een AT mogelijk vanaf een uitbreidingskaart de processor stil te zetten. Dit betekent dat vanaf de KGN-processorkaart de 80286 (80386) hardwarematig stilgezet wordt waarna de processor op de uitbreidingskaart MINIX opstart. Een alternatief is het gebruiken van een passieve AT-busprint. Hierbij komt dus het AT-moederbord te vervallen. Wordt voor de eerste mogelijkheid gekozen, dan blijft het natuurlijk mogelijk in plaats van de processorkaart de AT te gebruiken om MS-DOS of zelfs AT-MINIX te draaien.

Projectgroep

Op de bestuursvergadering in november is MINIX weer agendapunt geweest. Enkele bestuursleden hadden intussen wat voorwerk gedaan en eens gekeken naar de mogelijkheden. Dat zag er op zich redelijk interessant uit waarna het bestuur besloten heeft een projectgroep samen te stellen. Het eerste probleem dat toen naar voren kwam was het feit dat een meerderheid van het bestuur zitting wilde gaan nemen in de projectgroep en dat gaat natuurlijk ook niet omdat het bestuur dan zijn controle-mogelijkheden kan verliezen. De projectgroep dient onafhankelijk van het bestuur te kunnen opereren en moet rapporteren aan het bestuur die vervolgens op de gang van zaken in de projectgroep in moet kunnen grijpen.

Vanuit het bestuur hebben de volgende personen zitting genomen in de projectgroep:

Geert Stappers:	Projectleider
Gert van Opbroek:	Secretaris
Jan Derksen:	Beheerder MINIX soft- en hardware

Verder wordt de projectgroep uitgebreid met leden van de KGN. Het doel is namelijk een zodanige projectgroep samen te stellen dat de voor het project

benodigde kennis en ervaring voldoende afgedekt is. Een kleine inventarisatie van wat we denken nodig te hebben:

- Electronica t.b.v. computersystemen
- Print layouts
- Software/operating systems
- Kennis (hardware en software) over de te gebruiken processor
- IBM-AT kennis en ervaring
- Communicatie met de ontwikkelaars van MINIX aan de Vrij Universiteit Amsterdam
- Contact met Prentence-Hall c.q. Fred van Kempen
- MINIX/UNIX ervaring
- Schrijven van documentatie
- Public Relations
- enz. enz.

We zijn daarom ook heel blij dat we de projectgroep al uit hebben kunnen breiden met Pieter de Visser die zelf al een 68020-systeem ontwikkeld heeft en Ad Brouwer, één van de ontwerpers/auteurs van DOS-65. Als er verder binnen de leden nog mensen zijn die denken een bijdrage te kunnen leveren, dan worden die verzocht contact op te nemen met Geert Stappers.

Systeemeisen

Als je met een dergelijk project begint, is het heel verstandig aan het begin vast te gaan leggen wat je precies wilt. Op dit moment zitten we in dat stadium. We zijn binnen de projectgroep aan het inventariseren welke eisen we aan het systeem stellen. Deze eisen worden onderverdeeld in drie categorieën te weten:

1: Noodzakelijk
2: Zeer wenselijk
3: Nice to have

Het opstellen van de systeemeisen willen we voor een belangrijk deel in overleg met de leden doen. Dat zullen tenslotte de afnemers van het systeem gaan worden. Vervolgens is het aan de projectgroep, in overleg met het bestuur, de systeemeisen te wegen en de specificatie op te stellen. Daarna gaat de projectgroep op basis van de opgestelde specificatie aan de gang. Het is dus zeer belangrijk dat als u, als lid, wensen/eisen heeft ten aanzien van het project u dat ons laat weten. In het volgende nummer van de μP Kenner zullen we hier op terugkomen.

Processor

Hoewel het bij het vastleggen van een dergelijk systeem niet gebruikelijk is al in een vroeg stadium de processor vast te leggen is dat in dit geval wel gebeurd. Gekozen is namelijk voor een processor uit de 680x0 lijn van Motorola. Enerzijds vanwege het feit dat een 68000-achtige processor de 6502 mensen iets beter ligt als een Intel-processor, anderzijds omdat het eigenlijk toch ook een beetje vreemd is een AT met een 80286 te verbouwen om een andere 80x86 processor MINIX te laten draaien terwijl MINIX voor een AT beschikbaar is. Alternatieven zijn of technisch niet mogelijk (6502), of veel te duur (Sparc, transputers) of de benodigde kennis en ervaring is momenteel niet één twee drie voorhanden (processoren van National Semiconductors). De Motorola 68000-lijn heeft als voordelen dat ze redelijk goed aansluit bij de 6502 en 6809, dat er binnen de club een groot aantal mensen zijn die al ervaring met de 68000-lijn hebben en, ook een belangrijke reden, MINIX is voor 68000-processoren beschikbaar. Voor welke processor uit de 68000-lijn gekozen wordt is nog niet definitief bepaald. Elke processor heeft zijn eigen specifieke voor- en nadelen. Het is een zaak van de ontwerpers om deze voor- en nadelen tegen elkaar af te wegen waarbij de ideeën momenteel wel in de richting van een 68020 gaan.

Voorlopige opbouw van het systeem

In figuur 1 is schematisch een overzicht van de mogelijke opbouw van het systeem getekend. We zien hier de AT-bus met daarop aangesloten een optioneel AT-moederbord. Verder zijn daar als uitbreidingskaarten een gecombineerde Floppydisk/Harddisk controller en een Hercules kaart aanwezig. De taak van de disk controller is duidelijk, de hercules kaart is voornamelijk opgenomen omdat deze een printerpoort heeft. Verder kan via de Hercules kaart eventueel een beeldscherm aangestuurd worden. Als derde kaart is de KGN-processorkaart aanwezig.

Als er een AT-moederbord aanwezig is, dan staat het geheugen en de aanwezige periferie in principe ook ter beschikking van de processorkaart. Er is echter wel een vrij ernstige complicatie. Om het ge-

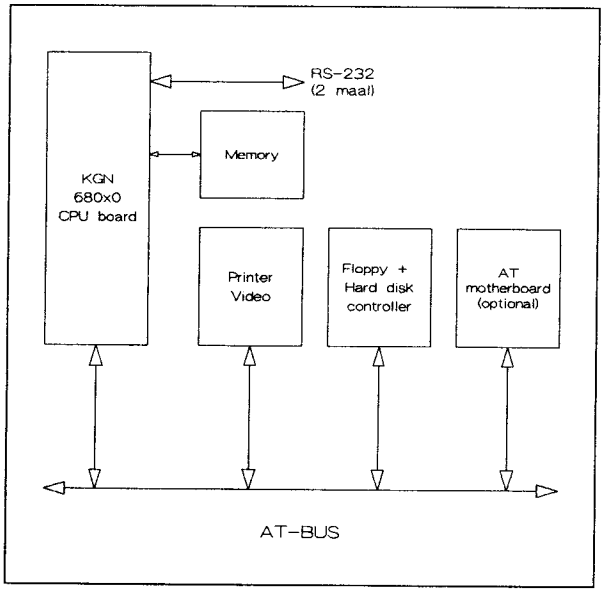


Fig. 1: het KGN-68k bord omringd door AT-hardware

heugen te kunnen gebruiken moet de processorkaart er voor zorgen dat elke 15 usec. het geheugen gerefreshed wordt. Dit lijkt momenteel erg problematisch. Waarschijnlijk is het wel mogelijk het toetsenbord te gebruiken zodat in dat geval de console gevormd wordt door het AT toetsenbord en de hercules kaart.

Behalve een complicatie met het geheugen op het moederbord is er ook een complicatie bij de keuze van de uitbreidingskaarten. aangezien de processorkaart beschikt over een 680x0 processor, is het niet mogelijk Intel-code uit te voeren. Er bestaan echter AT kaarten, bijvoorbeeld een EGA of VGA kaart die op de kaart een EPROM hebben met routines (BIOS) om de kaart te besturen. Deze routines zijn uiteraard in Intel-code. Zouden we dergelijke kaart willen gebruiken, dan moeten deze routines dus herschreven worden in 68000-code. en tweede nadeel is dat, door de aanwezigheid van kaart-afhankelijk BIOS er geen "standaard" EGA of VGA-kaart is. Dat betekent dat de gemaakte 68000-routines misschien wel werken op de EGA-kaart van fabrikant X maar niet op die van fabrikant Y. Kortom, kaarten met een eigen BIOS worden, zeker in de beginfase, niet ondersteund. Kaarten zonder een eigen BIOS zijn onder andere de gecombineerde HD/FD controllers, Hercules en CGA kaarten.

Het standaard-systeem zoals dat momenteel gedefinieerd is bestaat uit de getekende kaarten (eventueel in combinatie met een AT-moederbord doch niet noodzakelijk) een 3.5 inch 1.44 MB diskette drive en een winchester met voldoende capaciteit (minimaal 10 MB). Als console device wordt in het begin een losse terminal of computer met terminal-emulatie gebruikt. Later zullen een eventueel aanwezig toetsenbord en de herculeskaart ook bruikbaar zijn.

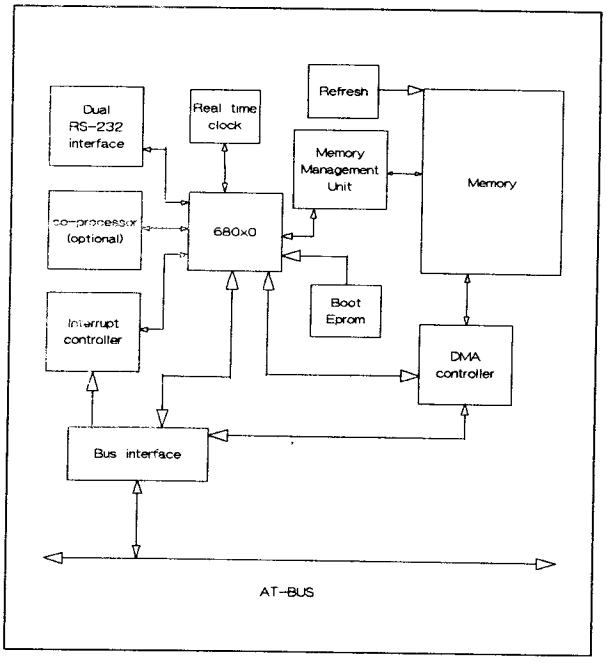


Fig. 2: mogelijk blokschema van het KGN-68k bord

De processorkaart

In figuur 2 is schematisch en mogelijke processorkaart getekend. In een blokschema zijn de functieblokken weergegeven.

De kaart communiceert met de AT kaarten via een bus-interface. Dit bus-interface koppelt de bus van de 680x0 aan de AT bus waarbij de signalen die de 680x0 nodig heeft omgezet worden naar AT-signalen en andersom.

Verder zien we uiteraard een processor en een hoeveelheid geheugen. De breedte van dit geheugen is afhankelijk van de gebruikte processor. Wordt gekozen voor een 68010, dan is de breedte 16 bits. Wordt er van een 68020 of 68030 processor gebruik gemaakt, dan kan de geheugenbreedte ingesteld worden op 8 bits, 16 bits of 32 bits. Vooral de mogelijkheid om geheugen met een breedte van 8, 16 en 32 bits door elkaar te kunnen gebruiken maakt de 68020 zeer interessant. In dat kan de breedte van het geheugen eventueel door middel van een draadbrug ingesteld worden zodat de bouwer, afhankelijk van de inhoud van zijn portemonnee de breedte kan kiezen. Een breedte van 32 bits betekent dan een aanzienlijk sneller systeem dan een breedte van 8 bits. Het feit dat de 68020 verschillende breedtes van geheugen door elkaar kan gebruiken heeft ook grote voordelen bij de communicatie met de AT kaarten. Een deel van de kaarten heeft namelijk een busbreedte van 8 bits, een deel heeft een breedte van 16 bits. Bij een 68020 kan dit allemaal eenvoudig in hardware geregeld worden.

Naar aanleiding van enkele contacten die we tijdens de HCC-dagen gehad hebben, is tussen processor en geheugen een MMU (Memory Management Unit) opgenomen. Een dergelijke bouwsteen kan er voor zorgen dat de software aanzienlijk eenvoudiger wordt omdat de MMU een groot aantal zaken m.b.t. adressering gewoon regelt. Of deze bouwsteen daadwerkelijk opgenomen wordt is momenteel nog niet besloten.

Naast de processor is een optionele coprocessor getekend. Dit is een 68881 of 68882 die als het ware parallel aan de 68020 geschakeld wordt. Aangezien voor deze processor geen extra hardware nodig is, is het zinvol op de print plaats voor dit rekenwonder te reserveren. Het is echter niet noodzakelijk de coprocessor aan te schaffen om het systeem te laten draaien.

De interrupt controller zorgt er voor dat de interrupts van de AT-kaarten netjes omgezet worden naar interrupts die de 680x0 processor kan afhandelen. Hierbij genereert de controller de vectoren aan de hand waarvan de processor weet welk randapparaat het interrupt gegenereerd heeft.

Bij de DMA controller zijn nog wel een aantal problemen op te lossen. Zoals al is aangegeven, kent de AT uitbreidingskaarten met een busbreedte van 8 bits en een busbreedte van 16 bits. Gebruiken we geheugen met een breedte van 32 bits, dan moet de controller er dus voor zorgen dat de bytes toch op de juiste plaats terechtkomen. Ook bij een geheugen breedte van 16 bits bestaat dit probleem, zij het in iets mindere mate.

Omdat de processorkaart ontworpen wordt voor gebruik zonder AT moederbord, is er een Real time clock en een dubbele RS-232 interface voorzien. Op dit interface kunnen twee terminals of andere computers aangesloten worden waarbij voorlopig één van de twee als console wordt gebruikt.

Mechanische opbouw

Het is waarschijnlijk niet mogelijk alle benodigde hardware op een AT uitbreidingskaart onder te brengen. Er wordt overwogen het geheugen op een aparte print onder te brengen dat als een imperiaalbordje op de uitbreidingskaart gestoken wordt. De rest van de hardware komt dan op een AT-uitbreidingskaart die in een AT-slot geplaatst kan worden. Verder wordt uiteraard gebruik gemaakt van een AT-kast met voeding.

Er leven momenteel wel ideeën de processor-kaart ook nog te voorzien van een aansluiting waarmee de 680x0-bus naar buiten gevoerd wordt. Deze aansluiting is dan bedoeld om voor specifieke 68000 uitbreidingskaarten niet afhankelijk te zijn van de AT-bus. Hoe dat echter uitgevoerd gaat worden is nog niet bekend.

Afsluiting

Met dit eerste artikel hoop ik enigzins duidelijk gemaakt te hebben wat voor ideeën er leven over het hardware-project. We weten onderhand uit reacties op de bijeenkomst in Almelo en van de HCC-dagen dat er zowel vanuit de leden als vanuit de rest van de wereld zeer veel belangstelling voor het project is. Toch staan we uiteraard open voor alle vormen van kritiek vanuit de leden. Het is tenslotte een project dat door de leden gedragen (en betaald!) moet worden. Kortom, nogmaals een oproep om uw ideeën en meningen aan de projectgroep kenbaar te maken waarbij wij uw bijdragen het liefst schriftelijk of in magnetische vorm zouden willen ontvangen. Verder kunt u op bijeenkomsten natuurlijk altijd de leden van de projectgroep over dit onderwerp aanspreken.

Namens de projectgroep KGN-68k MINIX,

Gert van Opbroek

Wilt u versie 3 van DOS-65? Reageer!

Op de bijeenkomst in Almelo ben ik door twee leden benaderd met de opmerking dat DOS-65 versie 3 eigenlijk beschikbaar zou moeten komen. Ik weet verder dat meer mensen graag verder zouden willen met DOS-65.

Zoals bekend, is het voor versie 3 van DOS-65 van groot belang dat er een virtuele geheugenkaart in het systeem zit en dat is dan ook precies de reden dat versie 3 nooit vrijgegeven is. Pogingen om vanuit het bestuur een virtuele geheugenkaart ter beschikking te stellen zijn steeds op niets uitgelopen. Nu blijken er echter een aantal mensen te zijn die toch een geheugenkaart gebouwd hebben en juist deze mensen zouden ook graag DOS-65 versie 3 willen gebruiken.

Op de HCC-dagen heb ik met Ad Brouwer over deze zaak gesproken en naar aanleiding van dat gesprek deze oproep:
Willen degenen die DOS-65 versie 3 willen gaan gebruiken zich via postbus 99650 1000 NA Amsterdam bekend maken? In overleg met die mensen willen we dan gaan onderzoeken wat de mogelijkheden zijn om DOS-65 versie 3 vrij te geven. Overleg is namelijk noodzakelijk omdat er geen "standaard" versie van de virtuele ramkaart bestaat. Verder zal er ook nog wat werk in gestoken moeten worden om het DOS draaiende te krijgen en om de utilities over te zetten. Bij voldoende belangstelling zal er waarschijnlijk een vergadering belegd worden om dit alles eens door te spreken.

Gert van Opbroek

compatibel met de PC/XT, ofschoon niet alle waarden in de tabel worden gebruikt.

Niet alleen het BIOS, maar ook de BASIC-ROMs met de cassette-BASIC zitten in de AT. Op dezelfde adressen: F600:0000-7FFF, of iets anders: F000:6000-DFFF. Dus midden in het AT-BIOS, dat van F000:0000 tot F000:FFFF loopt. Met die BASIC-ROMs kan de AT op zich niets beginnen: de cassette-poort ontbreekt, net als in de PC/XT. DE ROMs zijn echter nodig om IBMs BASIC.COM en BASICA.COM te kunnen draaien. Een ieder die dat al eens geprobeerd heeft op een niet-IBM machine had meestal een hangende machine als resultaat. Dit is dan ook de reden van het bestaan van GWBASIC.EXE: klonen en compatibles missen immers de BASIC-ROMs.

Compatibiliteit alom dus. Ook bij de BIOS functies INT 10h tot en met INT 1Fh: die zijn opwaarts compatibel. Wat er in de PC(/XT) kan, kan de AT dus ook. Maar eerst maken we een uitstapje naar de POST.

De POST in de PC/AT

De PC/AT heeft meer hardware op het moederbord, en alleen daarom al is de POST veel uitgebreider. Verder heeft IBM ervoor gezorgd dat er maar 1 jumper op het moederbord voorkomt: de AT kent geen DIP-switches meer. Dat komt de door de 146818 real time clock, die nog wat RAM bevat. Hierin wordt de machine configuratie bewaard, en een van de taken van de POST is de opgeslagen configuratie te controleren. Onder die ene jumper kon IBM niet uit. Het is de display-jumper. Deze jumper bepaalt, welke display adapter er actief is bij power-on als de configuratie data niet klopt, en er twee displayadapters (CGA en MDA) in de machine zitten. Klopt de configuratie data wel, dan wordt de setting van de jumper alleen maar gecontroleerd.

De POST verloopt geheel langs dezelfde lijnen als in de PC(/XT): de volgorde van gebeurtenissen is gelijk. Er is eigenlijk maar 1 ding bijgekomen: een tweede geheugentest. Die test hangt trouwens naadloos vast aan de eerste, en test het RAM-geheugen boven de 1 Mbyte-grens. Naar de smaak van ondergetekende heeft IBM dat niet slim opgelost: het BIOS geeft altijd de totale hoeveel RAM in de machine weer. Heeft de AT 512k basic geheugen beneden de 1 Mbyte grens en 1.5 Mbyte daarboven, dan meldt het BIOS 2048 kbyte geheugen. Dat doet het BIOS echter ook bij 640 kbyte basis geheugen en 1408 kbyte geheugen boven de 1 Mbyte. Overigens wordt het RAM beneden de 1 Mbyte base memory genoemd en alles boven de 1 Mbyte heet extended memory. Dat laatste geheugen is zoals we al gezien hebben, uitsluitend te adresseren als de processor in protected mode staat.

Die protected mode wordt trouwens ook getest, net als de real address mode. De gehele RAMtest wordt in protected mode afgewerkt, waarbij het afdrukken van de geheugenhoeveelheid in real-address mode geschiedt: na iedere geteste 64k volgt er dus een shutdown!

Wat verder nieuw is, dat beide floppy drives worden getest: er wordt gekeken hoeveel tracks iedere drive heeft. Dat lijkt heel knap, maar het gaat met de botte bijl. Eerst wordt de drive naar track 0 gestuurd. Daarna naar track 48. Die bestaat echter niet op een 360k drive, die dan ook ergens in de buurt van track 42 of 43 tegen een mechanische eindstop aanloopt en een aantal tracks slipt. Vervolgens stuurt de controller de drive weer naar track 10, ofwel 38 tracks naar binnen. Een 1.2M drive staat nu op track 10, een 360k drive echter niet: die staat in de buurt van track 5 of 6. Nu wordt er steeds 1 track naar binnen gestept, en wordt er gekeken of de drive track 0 geeft. Doet de drive dat bij minder dat 10 steps, dan is de drive 40 track. Doet hij dat na exact 10 steps, dat is de drive 80 tracks. Zijn er meer steps nodig, dan meldt het BIOS een floppy-probleem: error 601. Gevoelsmatig moet je dit niet te vaak doen met een 360k drive: die zou wel eens mechanisch kunnen verlopen. In de praktijk blijkt dat mee te vallen: iedere AT doet het op deze manier en er worden nooit klachten over 360k drives gehoord.

Er wordt gekeken hoeveel tracks iedere drive heeft.Dat lijkt heel knap, maar het gaat met de botte bijl.

Winchesters, voor zover aanwezig, worden ook getest: of de drive ready is, en of track 0 te lezen is. Uiteraard vergeet de POST ook de controller niet.

Ook de keyboard test is uitgebreid: dankzij het tweeweg-verkeer tussen toetsenbord en machine is het nu mogelijk om aan het keyboard te vragen wat voor toetsenbord het is. Blijkt het een "Enhanced Keyboard" te zijn, dan schakelt het BIOS automatisch de NumLock in: dat keyboard heeft immers een apart cursorveld.

Aan het einde van de POST volgen, indien noodzakelijk de foutmeldingen. Er was ruimte over in het BIOS, dus werden de nietszeggende getallen uitgebreid met een begrijpelijke tekst. Als de configuratie data niet klopt komt er een melding bij: "Please run

SETUP program". Deze verschijnt in het geval dat de POST een andere, geldige configuratie vindt als opgeslagen is in de batterijklok. Het SETUP-programma werd door IBM apart op een floppy disk meegeleverd. Behalve de configuratie kon het programma ook de tijd en datum in de klok instellen. Omdat de naam van het programma SETUP is, spreken de meeste mensen van "de SETUP" in plaats van de configuratie data.

De CMOS SETUP

Behalve de tijd en de datum heeft de 146818 klok nog een paar features. Als eerste kan hij dienst doen als wekker: er kan een alarmtijd worden ingesteld. Wordt dat tijdstip bereikt, dan volgt er een interrupt, die netjes door het BIOS wordt ondersteund. Het tweede feature is een vijftigtal RAM locaties, die worden vastgehouden met behulp van de batterij in de machine. In dat RAM wordt de configuratie opgeslagen. De volgende zaken worden bewaard:

- Aanwezigheid van coprocessor
- Power-on video adapter
- Aantal floppy drives
- Type floppy drive A:
- Type floppy drive B:
- Aantal winchesters
- Type winchester drive 0
- Type winchester drive 1
- Hoeveelheid basisgeheugen
- Hoeveelheid extended geheugen
- Shutdown nummer
- Nummer van de eeuw

Het type van floppies en de winchesters wordt als een vier bits getal opgeslagen waardoor voor twee floppies en twee winchesters slechts twee bytes nodig zijn. Dit geeft de mogelijkheid om per drive 16 verschillende typen aan te geven. IBM beschouwt type 0 in beide gevallen als "Not Installed" zodat er nog 15 overblijven. Bij floppies is dat genoeg, want de typeverdeling is als volgt:

No.	Type
1	360 kbyte, 5.25 inch
2	1.2 Mbyte, 5.25 inch
3	720 kbyte, 3.5 inch
4	1.44 Mbyte, 3.5 inch
5+	Niet gedefinieerd

Als het type niet bepaald is, of als er sprake is van een checksumfout, is het default type floppy drive een 1.2 Mbyte drive. Op die manier kan de machine ook van floppy booten als de configuratie niet klopt. Dat moet ook, want het SETUP programma staat immers op floppy.

Bij winchesters geldt iets dergelijks. In het oudste BIOS was plaats van 14 typen: 0 is niet aanwezig en IBM gaf aan type 15 de typering "Reserved" mee. Dat reserved bleek nodig bij latere BIOS-versies: als er type 15 staat, is er type hogere dan 15 geïnstalleerd, en staat het nummer elders in het CMOS RAM. De parametertabel in het BIOS kan maximaal 46 typen omvatten. Het hoogste type wat men dan ook theoretisch kan tegenkomen is 46. De parameters van type 47 zouden een compatibiliteits JMP overschrijven. IBM machines gaan niet verder dan type 23. Het heeft nauwelijks zin om in dit verhaal de parameters van alle 22 typen op te sommen: binnen de IBM-wereld is dit weliswaar consequent hetzelfde, maar in kloon-BIOSsen worden nogal eens afwijkende parametertabellen opgenomen.

Over het geheel van de configuratie (uitgezonderd de eeuw en de shutdown) wordt een 16-bit checksum opgeslagen, zodat er enige zekerheid bestaat omtrent de geldigheid van de data. Een checksum van nul is niet geldig: die treedt namelijk ook op als de batterij leeg is. Ik ken echter maar 1 BIOS die daarop test, en die is niet van IBM...

Als de batterij leeg raakt wordt in 1 van de status registers in de klok een vlaggetje gezet. Het BIOS vraagt om SETUP te draaien als:

- De batterij leeg is
- De checksum niet klopt
- De configuratie afwijkt van die gevonden door de POST.

Hiermee is de POST, of althans de nieuwe dingen erin, samen met de CMOS SETUP geheel bekeken. Dus gaan we verder met:

De nieuwe BIOS functies

Net als in het verhaal over het PC(/XT) BIOS zullen we de diverse functies stuk voor stuk even langslopen.

INT 10h, het video BIOS kreeg er 1 functie bij: AH = 13h. Dit is de functie write string. ES:BP wijst het begin van de string aan, CX is de lengte. In DH en DL kunnen rij en kolom waar de string moet worden afgedrukt worden opgegeven. AL bepaalt het formaat van de string en of de cursor meebeweegt, of op zijn plaats blijft staan. Dit laatste wordt door bit 0 bepaald. Bit 1 bepaalt het formaat van de string. Is het nul, dan bestaat de string uitsluitend uit ASCII karakters. In het tegenovergestelde geval bestaat de string uit afwisselend een attribuutbyte en een ASCII karakter, te beginnen met een karakter. Op die manier is het dus mogelijk verschillend gekleurde strings af te drukken. BH geeft de video-pagina aan waar de string naar toe moet, en BL geeft het attribuut in het geval van een ASCII-string.

INT 11h en INT 12h slaan we maar over: die bleven precies hetzelfde.

INT 13h zouden we eigenlijk moeten splitsen in floppy- en winchesterfuncties. Omdat de floppy wordt omgelegd naar INT 40h, doen we dan hier ook maar. Omdat de winchesterfuncties nog niet genoemd zijn, hier een lijstje:

AH	Functie
00h	Reset disk systeem
01h	Lees status
02h	Lees sector(en)
03h	Schrijf sector(en)
04h	Verifieer sector(en)
05h	Formatteer een track
(06h	Formatteer een slechte track)
(07h	Formatteer de gehele drive)
08h	Haal drive parameters
09h	Initialiseer parametertabel
0Ah	Lees sector met ECC bytes
0Bh	Schrijf sector met ECC bytes
0Ch	Seek
0Dh	Alternatieve disk reset
(0Eh	Lees sector buffer)
(0Fh	Schrijf sector buffer)
10h	Test drive ready
11h	Recalibrate drive
(12h	Controller RAM test)
(13h	Drive test)
14h	Controller test
15h	Haal drive type
[16h	Haal disk change status]
[17h	Set disk type]
[18h	Set disk type voor formatteren]

De waarden tussen haakjes zijn functies die wel bij de PC/XT voorkomen, maar die bij de AT zijn vervallen. De functies tussen teksthaken ([]) zijn functies die alleen voor floppies van toepassing zijn. In het algemeen is wel zo, dat de overeenkomstige floppy- en winchesterfuncties gebruik maken van de dezelfde functienummers.

Functie 08h is de eerste nieuwe. Hiermee kunnen we aan het BIOS vragen hoeveel cilinders, koppen en sectoren een drive heeft. Verder geeft de functie het totale aantal drives in de machine en het totale aantal sectoren op de drive, zodat de capaciteit in 1 keer bepaald kan worden.

Functie 09h initialiseert twee tabellen waarin de drive parameters worden opgeslagen. Deze functie wordt door DOS gebruikt om intern zo'n tabel te creeren. De interrupts INT 41h en INT 46h worden gebruikt om het adres door te geven voor de tabel van drive 0 respectievelijk drive 1.

De functies 0Ah en 0Bh zijn verwant met de functies 02h en 03h, met dit verschil dat de controller niet automatisch voor de ECC bytes zorgt: dat moet de gebruik nu zelf doen. Dit kan onder andere gebruikt worden om het ECC mechanisme te testen en om defecte sectoren toch van schijf te kunnen lezen.

Functie 0Ch beweegt de kop van de drive naar een bepaalde cilinder. Deze functie wordt maar zelden gebruikt: de lees-, schrijf- en verifyfuncties doen automatisch een seek indien nodig.

Functie 0Dh is officieel bedoeld om 1 enkele drive te resetten. In de praktijk is de functie geheel gelijk aan functie 00h.

Functie 10h kijkt of een drive gereed is, functie 11h stuurt de koppen van een drive naar track 0. Functie 14h vraagt aan de controller een interne test uit te voeren. Dit wordt o.a. tijdens de POST gebruikt en soms door programma's die zware eisen aan de betrouwbaarheid van het harde schijfsysteem stellen zoals interleave changers.

Functie 15h lijkt hetzelfde als functie 08h maar is het niet: deze functie doet een uitspraak over het soort drive. Dat kan zijn: niet aanwezig, floppy zonder change lijn, floppy met change lijn of harde schijf. Deze functie wordt door DOS gebruikt om te bepalen wat een floppy en wat een winchester is.

Sommige kloon-BIOSsen hebben ook nog functie 19h: parkeer de koppen. Deze functie stuurt de koppen van een winchester naar de zogenaamde transportcilinder. Dit kan vaak ook bewerkstelligd worden met functie 0Ch, maar alleen als de parkeer cilinder de laatste is. Ligt de cilinder verder naar binnen dan moet het met functie 19h: seek gaat niet verder dan de laatste geldige cilinder op de drive.

INT 14h (RS-232 services) is hetzelfde gebleven in de AT.

INT 15h was in de PC de woonplaats van de cassettefuncties. Die zijn er in de AT ook niet, maar INT 15h wordt nu gebruikt als verzamelbak van allerlei functies. De cassettefuncties geven, net als in een PC/XT "Bad Command" als antwoord terug. De volgende functies zijn echter nieuw en uniek voor de AT (zie de volgende bladzijde):

AH	Functie
4Fh	Keyboard OS hook
80h	Device open OS hook
81h	Device close OS hook
82h	Device program terminate
83h	Wait for event
84h	Lees joystick
85h	SysReq OS hook
86h	Wacht CX:DX microseconden
87h	Block move
88h	Haal extended memory grootte
89h	Ga naar protected mode
90h	Device busy OS hook
91h	Zet vlag en maak interrupt af
C0h	Get system identifier table address

Met deze verzameling wordt INT 15h zoals gezegd een echte vergaarbak. Alles wat niet duidelijk onder een bestaande verzameling calls valt, zit hier. Opvallend zijn de "OS hooks". Dit zijn kapstokken waarmee een operating system kan inbreken in het BIOS. Hiertoe zijn op bepaalde plaatsen in het BIOS een aantal INT 15h instructies ingevoegd die allemaal met een bepaalde waarde van AH werken. Het werkt net zo als de INT 1Ch timer tick interrupt. AH=4Fh treedt dus voor iedere keyboard-interrupt op, om het operating system als eerste de kans te geven iets met de scan codes te doen. AH=80h, AH=81h en AH=82h zijn bedoeld als kapstok om een multitasking operating system te helpen bij task-switching. Binnen het AT-BIOS worden deze drie functies niet gebruikt. AH=83h en AH=86h lijken op elkaar: wacht een bepaalde tijd. Bij AH=83h stop het wachten als het byte aangewezen door ES:BX het MS-bit gezet heeft. AH=84h leest de huidige stand van de joystick uit. AH=85h treedt op als op de SysReq-toets gedrukt wordt. Bij indrukken wordt AL=0, bij loslaten is AL=1. AH=87h is een bijzondere: dit is een universele memory block move routine. Omdat de routine in protected mode werkt, kan al het geheugen binnen de PC/AT bereikt worden. Om de routine te laten werken moet eerst in een stuk geheugen een zogenaamde descriptor table worden gezet waarin alle adres- en hoeveelheidsinformatie in verwerkt is. ES:SI moet naar het eerste byte in de tabel wijzen. AH=88h is een broertje van INT 12h: haal de extended memory grootte, ook hier in kilobytes. AH=89h schakelt de processor in protected mode. Merk op dat er geen call is om terug te komen, ofschoon dat dankzij het shutdown mechaniek wel gekund had. AH=90h en AH=91h zijn broertjes.

Opvallend zijn de "OS hooks". Dit zijn kapstokken waarmee een operating system kan inbreken in het BIOS.

AH=90h geeft de start van een wachttijd aan. Verstrijkt de tijd, dan volgt er een INT 15h met AH=91h. In AL wordt een procesnummer meegegeven, waaruit het operating system kan opmaken wat het gedurende de wachttijd kan doen. AL=0 t/m AL=7Fh zijn devices die het operating system zelf moet arbitreren. Hieronder vallen ook de BIOS wachtlussen voor harde schijf (AL=0), floppy wachttijd (AL=1), en toetsenbord (AL=2). Voor AL=80h t/m AL=BFh geldt, dat de devices re-entrant zijn: er hoeft niet gearbitreerd te worden. Voor AL=C0h t/m AL=FFh geldt dat het operating moet wachten totdat de wachttijd verstreken is. Dit geldt voor de harde schijf reset (AL=FCh), floppy motor start (AL=FDh) en de printer (AL=FEh). In de praktijk wijzen wijst INT 15h AH=90h en AH=91h naar een IRET, en doen deze functies dus niets. Multitaskers als DesQview gebruiken deze kapstokken echter wel.

De hekkesluiter is AH=C0h. Deze call is later toegevoegd. Hij levert een adres dat naar een tabel verwijst waarin de machine beschreven staat. Binnen de IBM-wereld is het namelijk lange tijd moeilijk geweest exact te identificeren op welke machine een programma nu eigenlijk liep. De hoofdregel is nu: als INT 15h, AH=C0h een gezette carry oplevert, is het een PC, een PC/XT, of een PCjr. In de andere gevallen valt uit de tabel op te maken welke machine het nu wel is.

INT 16h is en was het keyboard-BIOS. Er zijn een paar functies bijgekomen: AX=0305h zet bijvoorbeeld de repeat-rate van het toetsenbord. In BX staan dan de delay en de snelheid. Met AH=05h kunnen toets-waarden in de keyboardbuffer gezet worden, waarop het systeem ze zal behandelen alsof ze op het toetsenbord werden ingetypt. Heeft de machine een "Enhanced" toetsenbord, dan komen er nog drie functies bij. AH=10h is hetzelfde als AH=0 (haal een toets) maar geeft ook de extended scancodes door zodat de gebruiker bijvoorbeeld onderscheid kan maken tussen de twee PgDn toetsen of de twee Enter toetsen op het toetsenbord. AH=11h is analoog, maar dan equivalent met AH=1h. AH=21h haalt, net als AH=2 de shiftstatus op, maar nu van de extra toetsen op het "Enhanced" toetsenbord: in AH worden de standen van de SysReq, CapsLock, NumLock, ScrollLock, rechter Alt, rechter Ctrl, linker Alt en linker Ctrl toetsen doorgegeven. Om verwarring te voorkomen: in AL zitten de toestanden van de Lock toetsen. In AH

kun je zien of de toets ingedrukt, en dat is wat anders.

INT 17h, het printer BIOS, bleef hetzelfde, net als INT 18h, call BASIC en INT 19h, boot DOS.

INT 1Ah is de verzameling tijdservices. Naast de bestaande calls voor de tick-clock (AH = 0 en AH = 1) zijn er services bijgekomen voor de CMOS 146818 klok. Alle features van de klok worden ondersteund: lees de tijd (AH = 2), zet de tijd (AH = 3), lees de datum (AH = 4), zet de datum (AH = 5), zet het alarm (AH = 6) en zet het alarm af (AH = 7). In tegenstelling tot de eerste twee calls werken alle batterijklok calls in BCD en niet in clock ticks. Met ingang van DOS 3.30 is het zo, dat de TIME en DATE commando's niet alleen de tick klok veranderen, maar ook de 146818 klok. Daarvoor kon dat alleen maar met SETUP.

En daarmee zijn we bijna door het BIOS heen, want de overige calls (INT 1Bh t/m INT 1Fh) zijn hetzelfde als bij de PC/XT. Ook in de AT is geen uitgebreide CGA karaktergenerator opgenomen, ofschoon daar wel plaats voor was geweest.

Blijft nog over: INT 40h, de floppy disk services. Ook die groeiden fantastisch: het is grootste BIOS-deel na de POST. Dan kan ook haast niet anders: er worden vier nogal verschillende drive-types gebruikt, waarvan er twee ook nog eens twee verschillende soorten diskettes accepteren. De eerste zes calls bleven hetzelfde. De rest ziet er zo uit:

AH	Functie
08h	Haal drive parameters
15h	Haal drive type
16h	Haal disk change status
17h	Set disk type
18h	Set disk type voor formatteren

Er zitten behoorlijk wat gaten in de nummering, maar dat komt door de compatibiliteit met de services voor de harde schijf (zie aldaar). AH = 08h doet hetzelfde: het meldt het aantal tracks, koppen en sectoren/track. Verder wijst ES:DI naar de parametertabel voor de drive. AH = 15h geeft weer de soort drive aan. Je kunt zo te weten komen of de drive een changelijn heeft. Met AH = 16h kun je dan te weten komen of er van diskette gewisseld werd. Dit gaat overigens alleen op 1.2M en 1.44M drives.

AH = 17h vertelt het BIOS wat voor schijfje er in een drive zit, zodat de controller omgeprogrammeerd wordt. De mogelijkheden zijn:

AL	Type
0	Geen diskette aanwezig
1	360k diskette in 360k drive
2	360k diskette in 1.2M drive
3	1.2M diskette in 1.2M drive
4	720k diskette in 720k drive
5	720k diskette in 1.44M drive
6	1.44M diskette in 1.44M drive

AH = 18h deelt het BIOS mee hoe er geformatteerd moet gaan worden als AH = 5 wordt gebruikt. Dit is noodzakelijk omdat de diskette in de drive blank is: er kan dus niet uitgevogeld worden wat de drive/diskette combinatie is. Als antwoord geeft de functie of de gevraagde combinatie mogelijk is of niet. Is het mogelijk, dan wordt de controller zonodig omgeprogrammeerd.

De volgende keer...

bomen nog even door over het BIOS, maar nu van andere fabrikanten. En bespreken we dat andere soort geheugen: LIM-EMS of expanded memory. We zullen doorgaan...

Nico de Vries

Ervaringen met de installatie van MINIX 1.5.10

In september hoorde ik op de bijeenkomst in Haarlem voor het eerst van MINIX. Omdat de presentatie positief op mij overkwam en ik vanwege mijn werk bekend ben met Unix heb ik mij daar ook een pakketje aangeschaft voor gebruik op mijn AT-PC. Dat was nog de versie 1.3. Omdat de nieuwere versie 1.5 er zat aan te komen heb ik op de bijeenkomst ook meteen die update besteld.

Thuisgekomen ben ik aan het werk gegaan met versie 1.3. Eerst moest mijn PC enigszins worden aangepast, want ik had geen 5 1/4-inch drive als A-drive. Wanneer je eenmaal van de flop kunt opstarten gaan een heleboel dingen vanzelf goed. Wanneer je niet kunt opstarten ligt het aanzienlijk moeilijker, want de handleiding staat op één van de floppen, die je dan dus niet kunt lezen. Ook het werken op een 2^e geinstalleerde harddisk leverde geen noemenswaardige problemen op.

Tijdens het gebruik van MINIX bleven er toch wel een aantal vragen open. Zonder manual kom je daar niet zo makkelijk uit. Gelukkig kon ik die (in Amsterdam) bestellen, hoewel mij door Fred van Kempen was verteld dat de levering wel eens moeilijk zou kunnen zijn. Dat manual gaf inderdaad antwoord op bijna al mijn vragen en verstrekte bovendien een heleboel achtergrondinformatie. Ik zou dan ook iedereen die met MINIX aan het werk gaat willen aanraden dat manual aan te schaffen.

Na ongeveer 2 maanden kwam dan de langverwachte update naar 1.5.10. Een kompleet nieuwe versie bestaande uit een zevental floppen. Naarstig ging ik aan de slag met die nieuwe versie, en binnen de kortste keren had ik, vanwege de onbekendheid, mijn installatie-flop om zeep geholpen. Dom natuurlijk om niet eerst een kopie van de installatieflop gemaakt te hebben, maar je weet hoe dat gaat! Voorlopig kon ik dus even niet verder. Er zat niets anders op dan Fred te vragen de flop opnieuw te beschrijven. Na de andere floppen "write protect" te hebben gemaakt ging ik deze toch maar eens bekijken en kwam tot de ontdekking dat de Supplementdiskette een MS-DOS indeling had. Op die diskette staat ook de handleiding en een aantal nuttige utilities. Jammer dat die informatie niet was vermeld in het begeleidende schrijven bij de update. Dat had waarschijnlijk voorkomen dat mijn installatie-flop werd opgeblazen. Binnen enkele dagen ontving ik van Fred een opnieuw geschreven installatie-flop. Nu kon het dus echt beginnen!

Misschien helaas ging dat toch aanzienlijk minder snel. Het eerste probleem waar ik onmiddellijk tegenaan liep was, dat ik het installatie-programma met geen mogelijkheid kon vertellen dat ik MINIX op mijn 2^e harddisk wilde plaatsen. Ik heb veel geprobeerd, maar het is mij niet gelukt en ik ben dan ook tot de konklusie gekomen dat dat helemaal niet kan. Dan maar de 1^e harddisk anders organiseren om daar ruimte te maken. Na de nodige backups gemaakt te hebben ging ik met Speedstore aan de slag om mijn 1^e harddisk te initialiseren en te partioneren. Dat gaat eenvoudig, maar dan! Het MINIX-installatieprogramma zag geen kans om voor mij een werkbare MINIX-partitie te maken. Ook na tientallen keren, met verschillende parameters en na herhaald initialiseren en partitioneren van de harddisk, niet. Nog niet geheel ten einde raad ben ik gewoon gaan partitioneren met DOS-fdisk en formatteren met DOS-format. Dat bracht mij al een stuk verder. Toch bleken er nog meer addertjes onder het gras te zitten. Het Minix-installatie programma stelt een aantal vragen. Het geeft daarbij ook evenzovele default-antwoorden. Die wilde ik niet gebruiken. Naar mij later gebleken is, leest het installatie programma de antwoorden van het scherm en schrijft die EENMALIG op de installatie-flop. Een 2^e keer andere antwoorden invoeren blijkt dus geen zin te hebben! Ook leest het installatie programma bij wijziging van de default getallen niet het gehele getal terug (zoals ik veronderstelde) maar alleen de gewijzigde cijfers uit het getal, hoewel het de indruk wekt dat wel te doen. Ook bleek het niet te lukken om op de wijze zoals in de handleiding vermeld de MINIX-partitie "aktief" te maken, zodat je van je harddisk kunt booten om het 2^e gedeelte van de installatie te kunnen verrichten. Dat ging wel door in het MINIX-fdisk programma de "a"-optie te kiezen, of door dit naderhand te doen met de DOS-fdisk versie.

Uiteindelijk is het mij dan toch gelukt om MINIX te booten van mijn (1^e) harddisk. De rest van de installatie ging zonder meer vlekkeloos. Rest mij om te zeggen, dat ik MINIX toch een goed pakket vind. Vooral de 1.5-versie lijkt heel veel op UNIX. Ik heb dit stukje danook niet geschreven om het pakket schade te doen, maar om anderen, die misschien dezelfde problemen zullen tegenkomen, te helpen. Dat is de enige achtergrond van mijn bijdrage.

Hans Hartwijk

DIL IS WEGENS BALANSEN GESLOTEN van 31 dec. t/m 4 jan. Zaterdag 5 jan. 1991 zijn wij weer met raad-en-d(r)aad voor u paraat!

Wij wensen al onze klanten knusse en knutselige feestdagen en een knallend begin van 1991.

CHIP SELECT 91-92

Datasheets en -boeken leveren vaak net niet de informatie die nodig is om een IC optimaal toe te kunnen passen. Een overzicht van een aantal toepassingen (applikaties) kan echter zeer verhelderend werken. Voor dit boek werden meer dan honderd "chips" geselekteerd uit de ontwerpen die de afgelopen vijf jaar in Elektuur zijn verschenen. Van elke chip wordt naast de technische gegevens een aantal door de fabrikant voorgestelde toepassingen getoond. Het resultaat is enerzijds dit onafhankelijke naslagwerk, dat bijdragen van een groot aantal fabrikanten in zich verenigt. Anderzijds krijgt de Elektuur lezer zo een beter beeld van de achtergronden van de in Elektuur gepubliceerde schakelingen.
Dit 500 pagina's tellende boek omvat een totale selektie van meer dan honderd IC's en rond de duizend applikaties.

Duidelijk en overzichtelijk

Voor de beschrijving van de IC's zijn 450 redaktionele pagina's gebruikt. Er worden verschillende IC-typen en hun toepassingsmogelijkheden beschreven. Een Nederlandstalige introduktie met penlayout, gaat vooraf aan een pagina elektrische specifikaties en een pagina applikaties.

Wat heeft Chip Select te bieden?

- Zeer overzichtelijk door het numeriek- en funktieregister. Hierdoor is alles makkelijk en snel te vinden.
- Niet merkgebonden; u krijgt een overzicht van merken en typen.
- Méér dan 700 figuren.
- Nederlandstalige introduktie van de IC's.
- Lineaire en digitale IC's worden behandeld.
- Talloze schakelingen die direkt toepasbaar zijn.
- Alle relevante aansluitgegevens van ieder IC staan overzichtelijk bij elkaar.

75.- inkl. BTW

Terug van weggeweest:

BUIZEN-VERSTERKERS!

Het "warme" en "wollige" buizengeluid is weer in pakketvorm leverbaar bij DIL. Voorlopig een drietal ontwerpen, t.w. 1 voorversterker en 2 eindversterkers. Elke bouwkit wordt geleverd inkl. zwarte 19" behuizing, sub-chassis, frontplaat en eerste klas komponenten, o.a. ringkern-trafo's in beide eindversterkers.
Er moet nog wel enig mechanisch werk worden gedaan (boren) o.a., wat niet moeilijk is omdat feitelijk alle komponenten op de print worden gemonteerd.

De **VOORVERSTERKER**, beschreven in Elektuur jan./febr. 1990, is een perfekte stereovoorversterker zonder (!) tegenkoppeling en voorzien van ingangen voor CD, tuner, tape in/uit, MD-PU en aux (relais-bestuurd!). Lees de beschrijving in Elektuur en beslis zelf.
Inkl. 19" kast (matzwart), hoogte ca. 13 cm, diepte van de kast 25 cm.

Bestelkode: **87006-T** **695.-** inkl. BTW

De **100W-EINDVERSTERKER**, beschreven in Radio Bulletin oktober 1987, is een uitstekende mono-eindversterker met ringkern-trafo's voor de voeding en de uitgang, en wordt geleverd inkl. 19" kast (matzwart) van ca. 17 cm hoog, diepte 25 cm; eindtrap met 4 x EL34. Lees de beschrijving eerst in Radio Bulletin.

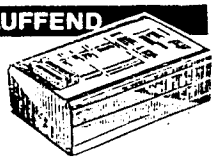
Bestelkode: **RB100EV** **749.-** inkl. BTW

Zijn kleinere broer, een **40W-EINDVERSTERKER (mono)** met 4 x EL84, vindt u terug in Radio Bulletin september 1985. Desgewenst hebben wij deze beschrijving voor u ter inzage. Ook hier weer ringkern-trafo's en het principe (bijna) alles op de print. 19" kast van 13 cm hoog en 25 cm diep, matzwart front.

Bestelkode: **RB40EV** **525.-** inkl. BTW

De prijzen van deze eindversterkers zijn niet laag, voornamelijk omdat (veel) aandacht besteed is aan met name de trafo's, en die vormen dan ook een forse onkosten-post.
De 100W eindversterker is desgewenst ook nog leverbaar met een nog zwaardere, ultralineaire uitgangstrafo, type XC462. Meerprijs is *f* 50,- zodat een bouwset dan *f* 799,- gaat kosten.

VERBLUFFEND

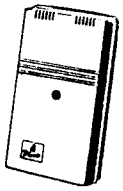


Steeds meer worden in moderne elektronische apparatuur Eproms en Eeproms toegepast. Tot vandaag was het programmeren van die ROM's vaak een lastige klus. U moest ervoor van huis om die te laten programmeren door uw leverancier. Maar vanaf vandaag hoeft er geen belemmering meer te zijn.
U kunt het voortaan zelf!

Met de EPP-1 Eprom programmer kunt u alle gangbare Eproms van 2716 tot en met 27512 programmeren (dus in 25,21 en 12,5 Volt versie) en bovendien de Eeprom 2864.
De programmer wordt aangesloten op een RS 232 poort van een computer en is dus rechtstreeks geschikt voor alle gangbare computers. Besturing van de programmer gebeurt met een eenvoudig communicatie programma. Voor MS Dos computers is bovendien een menugestuurd hulpprogramma meegeleverd.

410.- inkl. BTW (346.- exkl. BTW)

MINI Telefooncentrale



Dit apparaat maakt van uw telefoontoestellen een compleet communicatiecentrum.
U kunt 2 tot 4 telefoontoestellen aansluiten. U kunt ALLE binnenkomende en uitgaande gesprekken naar elk toestel doorverbinden, in de wacht zetten of overzetten naar elk gewenst toestel. Elk model toestel is aansluitbaar, de toestellen die u reeds in huis heeft, kunt u dus gewoon blijven gebruiken. Zonder extra kosten maakt u van de telefoontoestellen een intercom. Elk vertrek in huis te bereiken zonder telefoonkosten bij een intern gesprek. De 4-Phone is officieel toegelaten door het Ministerie van Verkeer en Waterstaat onder het nummer NL-89051201.
1-2-3-4: interne toestelnrs.
5: oproep alle toestellen
6: volgstand, om een aangenomen gesprek op een ander toestel voort te zetten.
0: aanvragen extern gesprek.

BIJ ALLE HANDELINGEN HEEFT U VOLLEDIGE PRIVACY, MEELUISTEREN ZOWEL INTERN ALS EXTERN IS NIET MOGELIJK.

Het telefoongemak van grote bedrijven bij u thuis. Maak van uw telefoontoestel een communicatiecentrum.

OPBELLEN: Draai een '0' + telefoonnummer.
NORMALE BELRITME: Externe oproep, neem de hoorn op.
WISSELEND BELRITME: Oproep van huisgenoot, neem de hoorn op.
OVERZETTEN GESPREK: Draai een '6' en leg de hoorn neer, u kunt het gesprek voortzetten op elk gewenst toestel door de hoorn op te nemen.
DOORVERBINDEN: Draai 1-2-3-4 of 5 om ander toestel te bereiken. Met of zonder intern overleg kan huisgenoot gesprek overnemen door het draaien van de '0'. Wilt u alleen intern overleg plegen zonder het gesprek door te verbinden, dan draait u nà intern overleg zelf de '0' om de buitenlijn terug te halen.
INTERCOM/HUISTELEFOON: Draai het interne telefoonnummer van een huisgenoot, 1-2-3 of 4. De huisgenoot hoort aan het belntme dat het een interne oproep is. Als u niet weet in welk vertrek uw huisgenoot is, draai dan een '5', alle toestellen bellen dan als algemene oproep.

Bestelkode: **TELCEN/4** **399.-** inkl. BTW

GE-LASER.....

ELV heeft een aantal ontwerpen voor het maken van een betaalbare laser-lichtshow met een rode laser-buis.

Zie de beschrijving in Elektuur december 1990. Uiteraard gaan wij deze kits opnemen in ons verkoopprogramma. Geïnteresseerd in de ELV folder en de (Duitstalige) bouwbeschrijving? Aanvragen per bnef(kaart) a.u.b.! (niet telefonisch).

ELEKTUUR BOUWPAKKETTEN

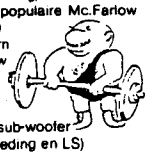
Elektuur bouwpakketten worden strikt geleverd volgens de lijst in het Elektuur bouwpakketten worden strikt geleverd volgens de lijst in het blad, inkl. voetjes voor alle IC's. DE PRINT IS BIJ DE PRIJS INBEGREPEN! Nieuwe bouwpakketten kunnen in prijs nog worden gewijzigd indien de Elektuur vooraf-informatie afwijkt van de publikatie in het blad. Aanzienlijke prijswijzigingen bij onze inkoop (o.a. geheugen IC's) worden in de pakketten doorberekend. De bouwbeschrijving mogen wij u NIET meer toezenden van de uitgever van dit blad, de eerste vier cijfers in de omschrijving geven het jaar en de maand aan van de publikatie in Elektuur.

87192	8711	**BASIC-BESTURINGSCOMPUTER** met INTEL 8052AH-BASIC	279.00
900020	9009*	**GITAARSTEMMER** excl. kast	89.50
900031-T	9009	**SPOELEN-Q-METER** exkl. meter, inkl. adaptor, front en kast	215.00
900042	9010	**ELEKTR. BELASTINGSWEERST.** inkl. koelplaten en hoekprof.	159.00
900057	9012	**GELUIDSDEMODULATOR** voor satelliet-ontvangers	45.00
900078-T	9009	**HFE-METER** komplete kit inkl. meter, kastje, front	199.00
900081	9010	**BASIC-TELEFOONCENTRALE** inkl. relais en ESS5943	325.00
900082-T	9010	**SUPER-VOEDING**, dubb. uitv. inkl. kast enz., exkl. meters	749.00
900083	9009*	**DIMLICHT AUTOMAAT** exkl. kast	45.00
900098	9010	**MONO-EINDVERSTERKER** exkl. kast, exkl. voeding	225.00
900098-MV	9010	**MONO-VOEDING** voor 900098 exkl. kast	217.50
900098-SV	9010	**STEREO-VOEDING** van 900078 exkl. kast	275.00
900104-T	9011	**REGELB. WISSELSPANN.VOED.** inkl. kast, front en meter	229.00
900106	9011	**TERMOMETER PT-100** exkl. opnemer	69.00
900111	9011	**MINUTIEUZE MD-VOORVERST.** inkl. trafo en exkl. behuizing	179.00
900114	9012	**POLARITEITS-INDIKATOR** (Fase-test) exkl. kastje	78.75
900134	9012	**UNIVERSELE AKKU-LADER** exkl. kast, front en trafo	159.00
904039	9009	**INGANGSKEUZE-SCHAKELAAR** exkl. relais	89.95
910004-T	9011	**MILLI-OHM METER BOUWKIT** inkl. kast, front & adaptor	189.00

LUIDE SPREKERS.....

Wij leveren u alle luidsprekers en filters die recent in Elektuur zijn beschreven:

SEAS/K29F	**TWEETER** voor het "triplet"-systeem (Elektuur mei 1990) en het aktieve **SUB-WOOFER** ontwerp (Elektuur dec. 1990)	47.00
SEAS/W111	**WOOFER** voor het "triplet"-systeem en het aktieve **SUB-WOOFER** concept, zie Elektuur december 1990	115.00
SEAS/WD215	**SUB-WOOFER** voor het "triplet" ontwerp in Elektuur mei 1990	175.00
890013-1	**FILTERPRINT** voor "triplet" (sub-woofer passief)	70.00
890013-2	**FILTERPRINT** voor "triplet" (midden/hoog scheiding)	65.00
MCF/H2590	**DOME-TWEETER** voor het inmiddels berucht-populaire Mc.Farlow "gevouwen hoorn" ontwerp, Elektuur april 1990	60.00
MCF/TO860	**LAGE TONEN SPEAKER** voor Mc.Farlow-hoorn	72.00
300047	**FILTER-ONDERDELEN** (geen print!) Mc.Farlow	22.35
FOC/10V516	**FOCAL SUB-WOOFER SPEAKER** voor de aktieve box beschreven in Elektuur november/december 1990	299.00
900122-1	**FILTER/REGELPRINT** voor bovengenoemde sub-woofer	65.00
900122-2	**SUB-WOOFER EINDVERSTERKER** (exkl. voeding en LS)	79.50



DIL elektronika

TELEFOON 010 - 4854213 / TELEFAX 010 - 4841150
JAN LIGTHARTSTRAAT 59-61, 3083 AL ROTTERDAM

*** partikulieren:**
Per brief met ingesloten EUROCHEQUE, GROENE BANKBETAALKAART of GIROBETAALKAART. (ondertekenen en pasnummer invullen) verzendkosten f 6,50 GEEN minimum orderbedrag
Door VOORUITBETALING op onze postgirorekening 649943 of ons bankrek. nr. 69.45.65.644 verzendkosten f 6,50 GEEN minimum orderbedrag
Per telefoon: levering geschiedt onder REMBOURS. Orders boven f 100,- verzendkosten f 10,- Voor kleine orders: verzendkosten f 15,-

*** openingstijden en winkelverkoop:**
DINSDAG t/m VRIJDAG 9.00 - 17.30 uur ZATERDAG: 9.00 - 16.00 uur. GESLOTEN: op maandag en vrijdagavond.

*** levering volgens onze standaardleveringsvoorwaarden**

*** bedrijven/instellingen:**
Toezending per PTT of NPD na ontvangst van uw bestelbon of uw opgave per telefax. Orders boven f 100,-: verzendkosten f 7,50 Voor kleinere orders: verzendkosten f 15,-
BALIEVERKOOP (voor levering 'op rekening' altijd een bestelbon of zakelijke legitimatie meenemen).
Na vooraf gaande afspraak is maand-fakturering mogelijk voor diegenen die geregeld kleine aantallen komponenten nodig hebben.

AL ONZE PRIJZEN ZIJN INKL. BTW (tenzij anders vermeld).

*** voor België Elektro-8000 PVBA.**
Langestraat 108. B 8000 Brugge. Tel. 050 - 341007 / Fax. 050 - 341168

SUFAST t.b.v. de real time clock MC146818

Dit programma dient om de real time clock te gebruiken in Dos65, net als de bestaande utility SETUP.

SETUP is geschreven door HaViSoft 1985/86.

Het verbaasde mij dat Dos65 bij het opstarten nogal lang bezig was met "iets". Na het geheel leeg maken van login.com bleek dat e.e.a. aanzienlijk sneller ging. Het meest tijdrovend bleek te zijn: SETUP. Een nadere bestudering daarvan leert dat er in SETUP nogal veel moet gebeuren. Het meeste is wel nuttig maar het hoeft niet elke keer als het systeem opgestart wordt. Dus SETUP blijft bestaan en er is een utility SUFAST gemaakt voor allerdaags gebruik.

Een overzicht van wat SUFAST doet in vergelijking met SETUP:

- Er wordt door SETUP bepaald welke dos versie u heeft. Daartoe wordt door het gehele geheugen gezocht naar de text Dos65. SUFAST doet dit niet, als u het toch wilt weten gebruik dan SETUP. (Nb: dit is de belangrijkste oorzaak dat SETUP zo traag is. Beide programma's SETUP en SUFAST controleren of er echt een rtc is.)
- De rtc wordt zodanig ingesteld dat er elke 500 ms een interrupt is. Waarom is mij niet bekend. De tijd verandert toch pas na 1 seconde. Daarom maakt SUFAST gebruik van een interrupt na iedere seconde. Die arme 6502 heeft het toch al zo druk, dit scheelt mooi een irq in elke seconde.
- Er wordt bijgehouden hoelang het systeem in totaal aan heeft gestaan. Dit doet SUFAST ook om te zorgen dat hier geen rare dingen gebeuren bijv. als meestal SUFAST gebruikt wordt, en soms eens SETUP. Dan zal SETUP dus toch de juiste tijd weergeven.
- De totale systeem gebruikstijd wordt door SETUP op het display gezet. Dit doet SUFAST niet want hoe kleiner het programma hoe sneller het in te lezen is van disk. Het scheelt natuurlijk ook in rekentijd.
- SETUP kontroleert of de rtc zonder stroom heeft gestaan. Kan gebeuren als deze bijvoorbeeld nieuw in het systeem zit. In dit geval zorgt SETUP voor het correct instellen van de rtc. SUFAST controleert dit wel de rtc en geeft dan een waarschuwing, maar kan in zo'n geval niet de rtc instellen. Gebruik daarvoor SETUP.
- SETUP zet een welkomst text op het display met enige versiering. Daarbij staat ook wanneer het systeem voor het laatst gebruikt is. Om tijd en ruimte te besparen is dat weggelaten in SUFAST.

Henk Speksnijder

```
;file sufast.mac                                1989 04 30

;H.Speksnijder
;A.Cuypstraat 43
;2902 GA Capelle a/d IJssel

                ;system
clock           equ     $E120           adres van rtc

                ;io65
datupd          equ     $E77E           variabele voor datum
day1            equ     $E77F           variabele voor dag
month1          equ     day1+1          variabele voor maand
year1           equ     day1+2          variabele voor jaar
hour1           equ     day1+3          variabele voor uur
minut1          equ     day1+4          variabele voor minuut
second1         equ     day1+5          variabele voor seconde
intvec          equ     $E7AF           unreg.interrupt vector

                ;dos65
out             equ     $C023           print char
```

```
crlf        equ     $C02F       print CR + LF
spa         equ     $C032       print spatie
hnout1      equ     $C035       print A hex of nibble
hexout      equ     $C038       print accu als byte
text        equ     $C03B       print string tot nul
hexdec      equ     $C044       convert hex naar decimaal

;interrupt voor aanpassen elke seconde:
areg        equ     $20         32 kHz zonder periodieke interrupt
breg        equ     $17         UIE, DM, 24/12, DSE enable
creg        equ     $10         vlag: einde update
                                ;opmerking: UIE = update interrupt enable

;variabelen in het rtc-ram (beschermde gedeelte)
;elke seconde aangepast:
;           $10     updating year
;           $11     updating month
;           $12     updating day
;           $13     updating hour
;           $14     updating minutes
;           $15     updating seconds
;gebruikt om systeem gebruiks tijd te berekenen:
;           $16     start uur v.h. gebruik
;           $17     start minuut  v.h. gebruik
;           $18     (2 bytes)totaal aantal uren systeem gebruik
;           $1A     total aantal minuten systeem gebruik
;voor status v.d. rtc. ( om te bepalen of deze geheel nieuw is )
;           $1B,$1C programmeer status
;           $1D     rtc             aangesloten status


xstore      macro
            stx     clock
            sta     clock+1
            endm

ystore      macro
            sty     clock
            sta     clock+1
            endm

getrtc      macro
            stx     clock
            lda     clock+1
            endm

            org     $A000
helptxt     jmp     sufast
            fcc     $C8,$C5,$CC,$D0
            fcc     $4C,$4c,$4c                         dummy
            fcc     '\rsupport real time clock MC146818'
            fcc     '\rsyntax: SUFAST'
            fcc     '\rno options, no abbreviation'
            fcc     '\rnote: if there is no rtc then an error message'
            fcc     '\rwill be displayed and the program aborted.\r'
```

```
                fcc     '\rnote: SETμP is needed to clear the rtc the first time'
                fcc     0
;variabelen voor tijdelijk gebruik in werkgeheugen
year            fcc     1               jaar
month           fcc     1               maand
day             fcc     1               dag
hour            fcc     1               uur
minut           fcc     1               minuut
second          fcc     1               seconde
sthour          fcc     1               start uur
stminut         fcc     1               start minuut
uhourl          fcc     1               ( 2 bytes) uren syseem tijd
uhourh          fcc     1
uminut          fcc     1               minuten systeem tijd
statusl         fcc     1               status
statush         fcc     1               status

;_____________________MAIN PROGRAM_____________________

sufast          lda     #$C0            test of er een rtc is
                ldx     #$1D
                xstore                  schrijf $C0 in de rtc
                getrtc                  lees het terug van de rtc
                cmp     #$C0
                bne     1.f             indien verschillend dan fout
                lda     #$0C
                xstore                  schrijf $0C in de rtc
                getrtc
                cmp     #$0C
                beq     main1           ok en ga door met de rest
1               jsr     text
                fcc     '\rNo real time clock'
                fcc     '\rsetupfast aborted',0
                rts
;kopieer jaar, maand, dag etc van rtc in werkgeheugen
main1           ldy     #$00
                ldx     #$10
main2           getrtc
                sta     year,y
                iny
                inx
                cpx     #$1D
                bne     main2           herhaal tot alles gedaan is
;kontroleer status v.d. rtc
;statusl moet $AA zijn  en statush moet $55 zijn
;als dat niet zo is: gebruik dan eerst SETμP voor initialisatie v.d. rtc
                lda     statusl
                cmp     #$AA
                bne     1.f
                lda     statush
                cmp     #$55
                beq     cusetime        indien ok dan doorgaan
1               jsr     text
                fcc     '\rtry SETUP',0
                rts
```

```
;bereken nieuwe tijd in gebruik
;   usetime: = endtime - starttime
;controleer of een minuut of uur gepasseert wordt
cusetime lda    minut
         cmp    stminut
         bcs    10.f
         lda    minut
         adc    #60
         sta    minut
         dec    hour
         cmp    #$FF
         bne    10.f
         lda    #23
         sta    hour
10       lda    hour
         cmp    sthour
         bcs    20.f
         lda    hour
         adc    #24
         sta    hour
;compute new usetime
20       sec
         lda    minut
         sbc    stminut
         clc
         adc    uminut
         sta    uminut
         cmp    #60
         bcc    40.f
         lda    uminut
         sbc    #60
         sta    uminut
         inc    uhourl
40       sec
         lda    hour
         sbc    sthour
         clc
         adc    uhourl
         sta    uhourl
         cmp    #100
         bcc    50.f
         lda    uhourl
         sbc    #100
         sta    uhourl
         inc    uhourh
;kopieer nieuwe gebruikstijd van werkgeheugen naar rtc-ram
50       ldx    #$18
         ldy    #$00
1        lda    uhourl,y
         xstore
         inx
         iny
         cpy    #$03
         bne    1.b
;haal de actuele tijd uit de rtc en zet dat in het rtc-ram
```

```
                ldx     #$04
                ldy     #$16
                getrtc
                ystore
                dex
                dex
                iny
                getrtc
                ystore
;de interrupt routine omleggen
                sei
                lda     intvec
                cmp     #interr&255
                bne     1.f
                lda     intvec+1
                cmp     #interr>>8
                beq     2.f
1               lda     intvec          kopie oude vector naar extra interrupt routine
                sta     lexint
                lda     intvec+1
                sta     hexint
                ;
                lda     #interr&255     zet nieuwe vector naar extra interrupt routine
                sta     intvec
                lda     #interr>>8
                sta     intvec+1
                ;
2               ldx     #$0A
                lda     #areg
                xstore                  reg. A in rtc = interrupt every update
                ldx     #$0B
                lda     #breg
                xstore                  reg. B in rtc = UIE, DM, 24/12, DSE enable
                cli
                rts
;aanpassen tijd en datum van de rtc
                org     $E600           nieuwe interrupt routine
interr          ldx     #$0C
                getrtc                  haal register C van rtc
                and     #creg
                bne     rtcint          als interrupt van rtc dan ga naar rtcint
                pla                     anders herstel registers
                tay
                pla
                tax
                pla
                fcc     $40             code voor jmp....
lexint          fcc     1               naar oude interrupt routine
hexint          fcc     1
rtcint          ldx     #$07
                getrtc                  haal dag
                cmp     day1
                beq     10.f
                pha
                lda     #$FF            zet vlag voor aangepaste datum
```

```
            sta     datupd
            pla
            sta     day1
            ldy     #$12
            ystore                  plaats dag ook in rtc-ram
            inx
            dey
            getrtc
            sta     month1
            ystore                  plaats maand ook in rtc-ram
            inx
            dey
            getrtc
            sta     year1
            ystore                  plaats jaar ook in rtc-ram
10          ldx     #$04
            ldy     #$13
            getrtc
            sta     hour1
            ystore                  plaats uur ook in rtc-ram
            dex
            dex
            iny
            getrtc
            sta     minut1
            ystore                  plaats minuut ook in rtc-ram
            dex
            dex
            iny
            getrtc
            sta     second1
            ystore                  plaats seconde ook in rtc-ram
nocop       pla
            tay
            pla
            tax
            pla
            rti
```

Fig. 1: source tekst voor SUFAST

Methoden en technieken voor datacommunicatie (Deel 5)

Inleiding

In de voorgaande aflevering is onder andere het zeven laags OSI model behandeld. In deze aflevering zullen we ons concentreren op één van die zeven lagen, namelijk op laag 4 de transport layer. Dit is de laag waar we te maken hebben met de zogenaamde protocollen. In deze aflevering zullen we het dan ook gaan hebben over protocollen, waar ze voor dienen en hoe ze, in algemene zin, in elkaar zitten.

Om het geheugen nog even op te frissen worden de voorgaande afleveringen van deze serie nog even in het OSI model geplaatst. In aflevering 1 hebben we het gehad over seriële en parallelle communicatie en over onder andere de RS-232 standaard. Dit zijn zaken die duidelijk thuishoren in laag 1, de Pysical Layer. In aflevering 2 hebben we een uitstapje gemaakt naar het openbare telefoonnet en hoe dat zo'n beetje in zijn werk gaat. Dit zijn zaken uit laag 0, het medium. In aflevering 3 en ook nog in een deel van aflevering 4 hebben we het gehad over de werking van modems. Deze horen duidelijk thuis in de zogenaamde Datalink layer, laag 2 dus.

Binnen de wereld van de hobby- en personal computers speelt laag 3, de Network layer niet zo'n belangrijke rol. Over het algemeen wordt er rechtstreeks tussen twee computers gecommuniceerd waarbij er meestal aan beide kanten maar één of hooguit twee communicatiepoorten aanwezig zijn. Bij een Bulletin Board kan de Network layer wel een belangrijke rol spelen. Er zijn namelijk Bulletin Boards die meer dan één lijn tegelijk bedienen en waarmee dus meerdere computers gelijktijdig kunnen communiceren. In dat geval zorgt de Network layer ervoor dat de informatie steeds op de juiste plaats komt en dat iedere gebruiker die informatie krijgt die voor hem of haar bestemd is. Op laag 3 zullen we in de toekomst nog een keer terugkomen.

Laag 5, de session layer, laag 6, de presentation layer en laag 7 de application layer zijn we in deze serie nog niet tegengekomen. Ik ben er echter van overtuigd dat ze nog wel een keer ter sprake zullen komen, bijvoorbeeld als we het gaan hebben over het uitwisselen van mails over netwerken van bulletin boards en over points.

Waarom protocollen?

Als we informatie van de ene computer naar de andere computer willen sturen, dan hangt het van de soort informatie af hoe ernstig een verminking van de informatie is. Als de een bericht naar een menselijke gebruiker sturen, dan is een verminking vaak niet zo erg kijk maar:

Deze zi@!!n bevat e#n aantl verminki*ngen mar is toc no$g leesbar (met moeite)

Dat komt omdat geschreven tekst veel zogenaamde redundantie bezit. Dat wil zeggen dat lang niet alle leestekens nodig zijn om de bedoeling duidelijk te maken. En computerprogramma's, zeker als het om een programma in uitvoerbare vorm gaat, bevat echter lang niet zoveel redundantie. Als we in een programma één bit veranderen, zal het programma in het algemeen niet meer datgene doen waarvoor het ontwikkeld is.

Als we in een programma één bit veranderen, zal het programma in het algemeen niet meer datgene doen waarvoor het ontwikkeld is.

Een ander voorbeeld van een geval met veel en met weinig redundantie vinden we in de wereld van de telegrafie. Hier wordt gebruik gemaakt van het zogenaamde Morse alfabet. In het Morse alfabet heeft elke letter een eigen patroon van strepen en punten of te wel van lange en korte piepjes. Ook de cijfers hebben een eigen patroon. Toch worden die cijfers meestal niet gebruikt. Als er namelijk een storing is of de ontvangende telegrafist is even afgeleid, dan kan een streep gemakkelijk voor een punt worden aangezien en dat veranderd bijvoorbeeld een 3 in een 4. Om dit probleem op te lossen worden de cijfers meestal als woorden (dus d-r-i-e of t-h-r-e-e) geseind. Hiermee neemt de redundantie toe en daarmee de storingsgevoeligheid sterk af.

Zoals al is aangegeven, zit er in computerprogramma's maar heel weinig redundantie. Bovendien is een computer lang niet zo slim als een mens. Een mens kan uit de context waarin de informatie staat bij een verminking meestal toch nog wel nagaan wat er bedoeld wordt een computer kan dit, afgezien van enkele expertsystemen, niet. Aangezien echter de communicatie tussen computers niet altijd vlekkeloos verloopt, zijn er manieren ontwikkeld om fouten te detecteren en te corrigeren. Deze methoden worden vastgelegd in een protocol en als nu beide kanten maar hetzelfde protocol gebruiken, dan kan er vrijwel niets meer misgaan.

Opbouw van een protocol

Om informatie over te sturen wordt het bericht of bestand meestal opgedeeld in stukjes. Dit zijn de zogenaamde packets. Een dergelijk stukje kan bestaan uit een vast aantal tekens, een complete opdracht voor het andere systeem of bijvoorbeeld uit een regel uit een tekstfile. Wat in de oudere protocollen nog wel eens voorkomt is een lengte van 80 of 132 tekens. Die 80 tekens zijn afkomstig uit de wereld van de ponskaarten. Dit was een stukje karton met een gaatjespatroon. Elk teken had een vast patroon van gaatjes toegewezen gekregen. Op een ponskaart gingen 80 tekens, werd niet de hele kaart gebruikt, dan zaten er in de rest geen gaatjes en dit was precies de gaatjescode voor een spatie. Een ponskaart gaf dus altijd stukken van 80 posities door, bevatte de informatie op de kaart minder tekens, dan werd het aangevuld met voldoende spaties. Een dergelijk bericht wordt wel een card image genoemd. 132 tekens vindt ook zijn oorsprong in de wereld van het mainframe. De regellengte van een lineprinter was namelijk 132 tekens. In dit geval wordt een kortere regel niet aangevuld met spaties maar afgesloten met een carriage return (CR) en eventueel een linefeed (LF). In dit geval is dus de maximale lengte voor een packet vastgelegd doch kunnen ook kortere packets overgestuurd worden.

Beide bovengenoemde methoden worden tegenwoordig nog gebruikt. Packets met een vaste lengte vinden we bijvoorbeeld in XMODEM en de daarvan afgeleide protocollen, een variabele lengte met een maximum komt bijvoorbeeld voor in Kermit. Wel is de maximale lengte van een packet tegenwoordig meestal groter. Een lengte van 1 of 2 kilobyte is eerder regel dan uitzondering.

In de hobby-wereld wordt een protocol het meest gebruikt bij het oversturen van files van de ene computer naar de andere. Dit betekent dus dat er meerdere packets voor één stuk informatie (file) nodig zijn. Hierbij wordt voor de lengte van het packet meestal de maximale lengte gekozen die het protocol toelaat en is het aantal packets afhankelijk van de hoeveelheid informatie. In de industrie komen er ook nog andere gevallen voor. Denk bijvoorbeeld aan een robotarm die een bepaalde beweging uit moet voeren. In dat geval bestaat een packet vaak uit een complete opdracht. De lengte van het packet (we spreken dan meestal van een telegram) is afhankelijk van de lengte van de opdracht. Voor elke beweging wordt dan een nieuw telegram gestuurd.

Wat alle protocollen gemeen hebben is het feit dat de ontvanger van een bericht door middel van het terugsturen van één of meer afgesproken tekens, aan kan geven of hij het packet goed ontvangen heeft of dat de inhoud van het packet niet deugt en dat hij verzoekt het packet nogmaals te sturen. Om dit te kunnen, moet de ontvanger natuurlijk wel kunnen zien of de inhoud van het packet deugt. Hiervoor wordt extra informatie aan het packet toegevoegd, een soort controle-getal. Dit controle-getal kan uitgerekend worden aan de hand van de rest van de informatie in het packet. Verderop in het artikel kom ik hier nog op terug.

De uitwisseling tussen zender ontvanger gaat dan als volgt:

```
Zender                              Ontvanger
-------------------------------------------------------------
Packet n      -----> <-----         Packet n correct
Packet n+1    -----> <-----         Packet n+1 niet OK
Packet n+1    -----> <-----         Packet n+1 correct
Packet n+2    ----->
etc.
```

Hierin gaat er dus een informatiestroom van de zender naar de ontvanger. Per packet geeft de ontvanger aan of het packet goed of niet goed ontvangen is. Het komt echter ook voor dat beide kanten de zender van informatie en de ontvanger kunnen zijn. Als voorbeeld weer de robotarm. De arm is bijvoorbeeld in staat 10 opdrachten in zijn geheugen op te slaan die hij achter elkaar uitvoert. Verder geeft de robotarm na het uitvoeren van een commando aan dat hij het commando uitgevoerd heeft, dit noemen we een gereedmelding. Het kan dus voorkomen dat de computer een nieuw commando wil sturen terwijl de robotarm een gereedmelding wil sturen. Hiervoor zijn in principe twee mogelijkheden. De eerste mogelijkheid gaat uit van het principe dat beide kanten ombeurten een telegram mogen sturen. Hebben ze niets te sturen, dan geven ze dit door middel van een code aan. De systemen krijgen wij wijze van spreken om beurten een stukje communicatietijd, een zogenaamde time-slice:

```
Computer                            Robot
-------------------------------------------------------------
niets te sturen   -----> <-----     Niets te sturen
niets te sturen   -----> <-----     Opdracht X gereed
goed ontvangen    ----->
niets te sturen   -----> <-----     Niets te sturen
Opdracht Y        -----> <-----     Goed ontvangen
                         <-----     Niets te sturen
Opdracht Z        -----> <-----     Goed ontvangen
                         <-----     Gereedmelding Y
etc.
```

Een variant op deze methode is de zogenaamde Master - Slave verhouding. Hierin is de computer meestal de master en het onderliggende systeem de slave. In dat geval mag de master altijd een bericht

sturen en de slave alleen als de master daarom vraagt. De master polt dus de slave.

Master		Slave
Heeft u iets?	----->	<----- Niets te sturen
Opdracht Y	----->	<----- Goed ontvangen
Heeft u iets	----->	<----- Gereedmelding X
Goed ontvangen	----->	
etc.		

Deze twee methoden hebben het voordeel dat het nooit voorkomt dat beide partijen gelijktijdig informatie uit wille wisselen. Een tweede voordeel is dat continu bewaakt wordt of er communicatie tussen de beide systemen mogelijk is. Het nadeel van dit systeem is dat er meestal wat tijd over heen gaat voordat een bericht verstuurd mag worden. In het tweede geval geldt dit overigens alleen voor berichten van de slave naar de master. Verder is het zo dat er constant berichten over en weer gestuurd worden en dat geeft enige belasting van de systemen.

Een andere mogelijkheid voor het uitwisselen van telegrammen is een manier die meer met een interrupt te vergelijken valt. Als één van de twee systemen de behoefte krijgt een bericht te zenden, dan vraagt hij hiervoor aan het andere systeem om toestemming. In de meeste gevallen zal deze toestemming verleend worden. Kan die toestemming niet verleend worden dan wordt meestal een code gestuurd die betekent "wacht even en probeer het straks nog eens". Het kan in dit geval echter ook voorkomen dat beide systemen gelijktijdig een aanvraag sturen. In dat geval spreken we van een zogenaamde collission (botsing). Om dit op te lossen moet één van de twee systemen de aanvraag met een toestemming beantwoorden terwijl het andere systeem de aanvraag gewoon naast zich neerlegt. Het systeem dat de toestemming moet geven heeft dan een lage prioriteit, het andere systeem een hoge. Het protocol loopt dus als volgt af:

Systeem A		Systeem B
Mag ik ?	----->	<----- Gaat uw gang
Opdracht Y	----->	<----- Goed ontvangen
Mag ik ?	----->	<----- Gaat uw gang
Opdracht Z	----->	<----- Niet ontvangen
Opdracht Z	----->	<----- Goed ontvangen
		<----- Mag ik ?
Gaat uw gang	----->	<----- Gereedmelding X
Goed ontvangen	----->	
etc.		

Voor de sturing in al deze protocollen kunnen in principe alle tekens en combinaties van tekens gebruikt worden. In de praktijk komen de volgende tekens het meeste voor:

Teken	Omschrijving
STX ($02)	Mag ik ?
DLE ($10)	Gaat uw gang/goed ontvangen
ACK ($06)	Goed ontvangen
NAK ($15)	Niet ontvangen

Inhoud van een packet

Zoals al is aangegeven, kunnen packets altijd een vaste lengte hebben of er zijn zogenaamde variabele packets. Packets die een vaste lengte hebben, hoeven in principe niet met een aparte code afgesloten te worden. In de praktijk wordt dit echter vaak wel gedaan om een extra controle-mogelijkheid te hebben. Bij packets met een variabele lengte, moet in het begin van het packet de lengte staan of het packet wordt afgesloten door een bepaalde code. In de praktijk wordt een dergelijk packet meestal afgesloten met DLE ETX en het controle-getal of door CR LF. Komt in het packet het teken DLE ook voor, dan wordt deze DLE twee maal gestuurd. De ontvanger weet dan dat hij niet te maken heeft met het einde van het bericht.

Bij packets met een vaste lengte moet uiteraard wel een vulteken afgesproken worden voor het geval het bericht korter is dan een geheel aantal packets. Afhankelijk van het gekozen protocol wordt een vulteken gedefinieerd.
Bij informatie die uit meerdere packets bestaat, zoals bij file-transfer, wordt aan het begin van het packet bijna altijd het volgnummer van het packet doorgegeven. De ontvanger kan dan zien of hij de packets in de juiste volgorde binnenkrijgt. Ontvangt hij een packet twee maal, dan detecteert hij dat ook en zal één van de twee negeren. Verder bevat een packet vaak een code die de afzender identificeert en een code die de geadresseerde identificeert. Andere systemen die dat packet dan eventueel ontvangen weten dan dat het niet voor hen bestemd is en kunnen het packet doorsturen of negeren. Dit wordt in de network layer van die systemen opgelost.
Het packet wordt tegenwoordig vrijwel altijd afgesloten met een controle-getal van één, twee of meer tekens. Aan de hand van dit controle-getal kan de ontvanger bepalen of de inhoud van het packet in orde is.

Iedereen die wel eens een programma van een bulletin board gehaald heeft, weet dat bij de meeste protocollen de filenaam bij de ontvanger dezelfde wordt als bij de afzender. Verder wordt vaak aangegeven hoe groot de file is etc. Dergelijke informatie wordt

in een apart packet die aan het begin van de file gestuurd wordt overgeseind. Aan de hand van deze informatie weet de ontvanger wat hij met de file moet doen en hoe lang de transmissie waarschijnlijk gaat duren. Een dergelijk HEADER-packet komt voor bij de meeste protocollen. Uitzondering hierop is het protocol XMODEM.

Het controle-getal

Zoals al is aangegeven, zit er in packet bijna altijd een controle-getal om te kunnen onderzoeken of de inhoud van het packet in orde is. De werking van dit controle-getal is vergelijkbaar met de werking van een pariteitsbit bij het oversturen van losse bytes dat aangeeft of de databits in orde zijn (zie deel 1). Het pariteitsbit wordt berekent door de exclusive or van de databits te nemen. Is dit resultaat een één en is de pariteit even, dan wordt het pariteitsbit ook een één, is de pariteit odd, dan wordt het pariteitsbit een nul. Is het resultaat van de exclusive or een nul, dan is het pariteitsbit respectievelijk een nul of een één.

Een dergelijke methode wordt ook vaak gebruikt voor de berekening van het controle-getal. In dit geval praten we dan over de checksum. Van het packet worden alle bytes door middel van een exclusieve or samen genomen. Het resultaat is vervolgens de checksum. Als voorbeeld:

- We nemen een packet met als inhoud "Einde.", het packet wordt afgesloten met een <CR> en een <LF>. We gebruiken zeven bits ASCII met een even pariteit. Gevraagd de checksum.

E	1 100 0101
i	0 110 1001
n	1 110 1110
d	1 110 0100
e	0 110 0101
.	0 010 1110
CR	0 000 1100
LF	0 000 1010
k	1 110 1011

De checksum is bepaald door van elke kolom bit voor bit de exclusive or te nemen. In dit geval is dat de letter "k". Het komt in de praktijk wel eens voor dat de checksum een niet printbaar teken is. Dit gebeurt als het teken kleiner dan $20 is (de spatie). In sommige gevallen is het niet mogelijk dergelijke tekens te versturen omdat de hardware iets met een dergelijk teken doet, bijvoorbeeld de communicatie afsluiten. In dat geval wordt de checksum in twee delen opgesplitst, een high nibble (0110 in ons voorbeeld) en een low nibble (1011) in ons voorbeeld. Merk op dat het pariteitsbit hierbij komt te vervallen. Vervolgens worden bij deze waarden een andere waarde (bijvoorbeeld $20) waarna de checksum als twee aparte bytes, elk met hun eigen (opnieuw bepaalde) pariteitsbit. Ook een dergelijke afspraak is onderdeel van het protocol.

Evenals bij een enkel pariteitsbit slechts fouten in één bit van het databyte geconstateerd kunnen worden en er geen mogelijkheid is om fouten te corrigeren, is het bij een checksum ook slechts mogelijk een fout in één bit in een kolom te detecteren en is correctie ook niet mogelijk. Fouten in twee bits heffen elkaar namelijk op. Modernere technieken maken daarom ook geen gebruik meer van de beschreven methode van checksum. Tegenwoordig wordt bijna altijd gebruik gemaakt van een zogenaamde CRC of Clyclic Redundancy Check. Hierbij worden 2, 3 of meer extra bytes toegevoegd die ook uit het bitpatroon van de andere bytes bepaald zijn. Het verschil is echter dat er niet een bit is die bit-0 van alle bytes bewaakt en een bit voor bit-1 etc. doch dat elk bit in de CRC meerdere bits in de bytes bewaakt en dat elk bit in een byte door meerdere bits in de CRC bewaakt wordt. Op deze manier kan men fouten in meer dan één bit zitten ook detecteren en soms zelfs herstellen. In de praktijk laat men bij CRC de pariteitscontrole op de databytes daarom ook achterwege. Voorbeelden van dergelijke CRC berekeningen zijn te vinden in referentie 1 en 4.

Afsluiting

In deze aflevering is gesproken over de overdracht van informatie met behulp van een protocol. Er is aangegeven waarom een protocol gebruikt wordt en hoe iets dergelijks in zijn werk gaat.
In de volgende aflevering zullen we eens gaan kijken hoe enkele specifieke protocollen opgebouwd zijn. Als voorbeelden worden daarbij XMODEM en ZMODEM behandeld. Wil iemand voor februari al een uitgewerkt voorbeeld, dan verwijs ik naar referentie 2 waar het Kermit protocol beschreven is.

Tot de volgende keer dus.

Literatuur

Aanvullende informatie over de behandelde onderwerpen is o.a. te vinden in de volgende artikelen/boeken.

1: Gert Klein: Kermit een file transfer protocol. De 6502 Kenner 54, februari 1988.
2: Bram de Bruine: Datacommunicatie met micro's. De μP Kenner 60, februari 1989.
3: P.C. den Heijer/R. Tolsma: Datacommunicatie. Kluwer PC boeken.
4: Andrew S. Tanenbaum: Computer Networks. Prentice-Hall.

Gert van Opbroek.

Van de bestuurstafel

U heeft hem inmiddels al weer bijna uitgelezen. Het laatste nummer van de veertiende jaargang. Aangaande de planning zou je kunnen zeggen dat dit nummer tussen de chocoladeletters en de kerstkransen door is gemaakt. Onder verhoogde tijdsdruk, zoals blijkbaar alles aan het einde van het jaar.

We zijn nog maar net bekomen van de HCC-dagen. Die waren trouwens heel gezellig, en zeer geslaagd. Een van belangrijkste conclusies die na twee dagen wel kon trekken is wel, dat hardware werkelijk helemaal niets meer kost. Floppy drives, vroeger schreeuwend duur, gaan al over de toonbank voor rond de fl. 100,-. Voor diezelfde honderd piek kon je zaterdagmiddag zelfs een complete PC kopen, nieuw in doos, compleet met MS-DOS 1.25 en 128k RAM. HCC staat voor Hobby Computer Club. Een tweede vaststelling is, dat die hobby weer een beetje terug komt. Zo kon je op zeer veel kramen defecte hardware voor een krats kopen. Wat dacht u bijvoorbeeld van een monochrome-monitor, brandnieuw doch defect voor fl. 15,-? De totale reparatie vergde 1 diode van 50 cent... Behalve de koopjes was er ook gelegenheid voor andere zaken: we hebben we een aantal mensen in de kraam gehad die wellicht lid zullen worden. En daar ging het toch uiteindelijk om. Iedereen van de stand-bemensing (bemanning was deze keer gewoon een foute benaming) heeft het enorm naar zijn zin gehad. Al met al voor herhaling vatbaar. We waren trouwens allemaal gestoken in een blauw T-shirt met daarop het clublogo. Er zijn nog een paar T-shirts over. U kunt ze bij de penningmeester bestellen.

Al die computers bijelkaar in de Jaarbeurs is natuurlijk techniek, techniek en nog eens techniek. Iedereen lijkt overal verstand van te hebben. Totdat je twee heren van middelbare leeftijd met gefronst voorhoofd naar onze publiekstrekker ziet kijken. Dat was een stokoude Teletype, die lekker stond te ratelen. Zegt de ene heer tegen de andere: "Dit is het eerste dat ik vandaag zie waarvan ik nog wat begrijp..."

Voordat de HCC-dagen losbraken, heeft het bestuur weer de gebruikelijke twee-maandelijkse vergadering belegd. Hierover kan gemeld worden dat het MINIX-project inmiddels in volle gang is gezet (zie ook het verslag van de projectgroep in dit nummer). Verder zijn de taken binnen het bestuur opnieuw gerangschikt. Jan Derksen vervult thans de taak van secretaris. Geert Stappers maakt zich nu sterk voor het MINIX-hardware project. Verder hebben we besloten om de vorige voorzitter op de ledenvergadering in Almelo voor te dragen als ere-lid. In het verslag van die vergadering valt te lezen, dat Rinus Vleesch-Dubois unaniem werd gekozen. Rinus: van harte gefeliciteerd.

Ook bij deze club steekt de hobby weer de kop op. Het MINIX-project belooft heel wat te gaan worden: zowel op de ledenvergadering in Almelo als op de HCC-dagen waren de reacties zonder uitzondering positief: de soldeerbout kan weer eens een keer aan, en dat houdt in: volop ouderwets hobby-en! En dat is iets wat we al bijna vergeten waren hoe dat er ook weer uitzag. Over hobby gesproken: zelfs DOS65 blijkt nog springlevend: de vraag naar versie 3, die onder andere multi-tasking zou worden is weer aan de orde. Wie had dat gedacht...

We gaan bijna aan het derde lustrum van de club beginnen. De vooruitzichten zijn goed. Het ledental stijgt langzaam, maar zeker. U blijft dit jaar toch hoop ik ook lid? Meer leden zijn natuurlijk altijd welkom. Kijk eens rond in uw kennissenkring of daar potentiële leden zitten. Hoe meer zielen, hoe meer vreugd. En hoe meer we voor de leden kunnen doen. De penningmeester gaat de accept-giro's voor de contributie binnenkort verzenden. U doet hem een plezier door op korte termijn het lidmaatschapsgeld over te maken. Voor het geld hoeft u het niet te laten lijkt me: we zijn zo langzamerhand de goedkoopste club van heel Nederland geworden...

Uiteraard wenst het gehele bestuur een ieder uiterst plezierige feestdagen toe. En natuurlijk een uitbundig 1991. Tot volgend jaar maar weer.

Uw veurzitter,

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 dediverse systemen in stand gehouden.

Landelijke bijeenkomsten:

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

Bulletin Board:

Voor het uitwisselen van mededelingen, het stellen en beantwoorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board beschikbaar gesteld.
Het telefoonnummer is: 053-303902.

Software Bibliotheek en Technisch Forum:

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

Correspondentie adres

Alle correspondentie betreffende verenigingszaken kan gestuurd worden aan:

KIM Gebruikersclub Nederland
Postbus 99650
1000 NA Amsterdam

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.

Nico de Vries (voorzitter)
Mari Andriessenrade 49
2907 MA Capelle a/d IJssel
Telefoon 010-4517154

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

Jan D.J. Derksen(secretaris)
C.P. Soeteliefstraat 41
1785 CC Den Helder
Telefoon 02230-35002

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

Mick Agterberg
Davidvosstraat 29
1063 HV Amsterdam
Telefoon 020-131538

Geert Stappers
Engelseweg 7
5825 BT Overloon
Telefoon 04788-1279

Ton Smits
De Meren 39
4731 WB Oudenbosch

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