

De μ P Kenner



Zestiende jaargang nr. 1
Februari 1992
75



In dit nummer o.a.:

PASCAL en ANSI
MSDOS naar OS/9
Schizofrene bestanden
MINIX software gevraagd
Terugkeer van een koude Kermis

Inhoudsopgave

De μ P Kenner

Nummer 75, februari 1992
Verschijnt 6 maal per jaar
Oplage: 250 stuks
Druk: FEBO Offset, Enschede

De redactie:

Joost Voorhaar
Gert van Opbroek
Geert Stappers
Nico de Vries

Eindredactie:

Joost Voorhaar

Vormgeving:

Joost Voorhaar
Nico de Vries

Redactieadres:

Joost Voorhaar
Jekerstraat 67
7523 VP Enschede

De μ P Kenner nummer 76 verschijnt op
18 april 1992.

Kopijsluitingsdatum voor nummer 76 is
vastgesteld op 4 april 1992.

Vereniging

Uitnodiging voor de clubbijeenkomst	5
Financieel Jaarverslag 1991	6
Van de bestuurstafel	37

Algemeen

Redactioneel	4
Halen, Contant of per Cheque	7

Talen/Software

To Share Or Not To Share, That's The Question... ..	8
Gespleten files maken	10
Device drivers in TurboPascal	21
MS-DOS C-source naar OS-9/68k	28
Liftend langs Internet	30
Terugkeer na een koude kermis	32
Minix software?	35

Hardware

Hardware beschrijving KGN30 (1)	31
---------------------------------------	----

De μ P Kenner is het huisorgaan van de KIM gebruikersclub Nederland en wordt bij verschijnen gratis toegezonden aan alle leden van de club. De μ P 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.

De redactie houdt zich het recht voor kopij zonder voorafgaand bericht niet of slechts gedeeltelijk te plaatsen of te wijzigen. Geplaatste artikelen blijven het eigendom van de auteur en mogen niet zonder diens voorafgaande schriftelijke toestemming door derden gepubliceerd worden, in welke vorm dan ook.

De redactie noch het bestuur kan verantwoordelijk gesteld worden voor toepassing(en) van de geplaatste kopij.

Redactioneel

Voor je 't weet is 't al weer zo ver: de volgende μ P Kenner moet eigenlijk al bij u, de lezer, op de mat liggen en er is dus nog helemaal niets aan kopij binnengekomen. Kortom: het wordt weer crisis in de redactielokalen en de kleine oogjes der redactieleden verraden dat het nog erg vroeg is voor een zondagochtend. Nou ja, het zei zo. Dan had ik maar een andere job moeten bedingen... Het resultaat ligt voor u: een dunne deze keer. Een zorgelijk dunne...

Ik had gehoopt deze keer een leuk (klein) hardware projectje te kunnen aankondigen: de seriële interface monitor SIMON. Dat is een klein apparaatje waarmee je heel eenvoudig RS-232 signalen kunt monitoren. En dan bedoel ik niet die doosjes met een zevental LEDs erop, maar een kastje met een LCD-schermpje, een toetsenbordje en een klein stukje intelligentie aan boord; in SIMON wordt gebruik gemaakt van een zogenaamde "Single Chip" computer. Da's een microprocessor waar wat ROM, een stukje RAM en wat I/O poorten op het silicium samenge-raapt zijn. Adri zal daar in Geldrop meer van vertellen. Helaas, het verhaal dat bij de hardware hoorde was niet op tijd af doordat de auteur dringende bezigheden had in een ander deel van de wereld!

Het gaat dus niet zo goed met de μ P Kenner, maar gelukkig gaat het beter met allerlei andere activiteiten binnen de club. Het bulletin board loopt beter dan ooit, het KGN68k/30 project kachelt vrolijk verder en de geluiden uit de DOS-65 hoek... uhhh... ja. Of eigenlijk: nee, dat wil nog niet zo hard vloten. Tijd, meneertje, tijd, dat hebben de meeste leden van de werkgroep(en) maar in beperkte mate beschikbaar.

Wat natuurlijk ook heel belangrijk is, zijn de activiteiten die zich buiten de club afspelen. Als oudste

computervereniging kun je natuurlijk niet achterblijven op de actuele stand van zaken in de grote boze wereld. We hebben gezien dat de HCC een vrijage aanging met de bladen "PC-thuis" en "PC-World", hetgeen resulteerde in de "Computer! To-taal". Als je de afkorting van dat blad bekijkt, doet die bijzonder sterk denken aan een Duitstalig blad dat veel meer aanzien heeft dan de originele drie Nederlandse bladen (HCC-nieuwsbrief, en de bovengenoemde) bij elkaar ooit gehad hebben.

Uit een andere hoek waait de wind op "Het netwerk", het Internet dus. In comp.os.minix probeert daar iemand zijn zelfgebrouwde "Linux" aan de man te brengen. Een UNIX-like OS dat specifiek op de 80386 microprocessor geschreven is. De auteur beweert dat zijn operating system stukken beter is dan Minix, en nog gratis ook! En dat is natuurlijk tegen het zere been van de trouwe schare Minix-fans die onder aanvoering van Tanenbaum himself hard van leer trekken tegen de onfortuinlijke Lin-uxer. Ik heb he operating system nog niet in levende lijve mogen aanschouwen, maar het beloofd heel wat te worden. Bij voldoende belangstelling gaan we proberen dit OS van het Internet te halen; maar dat kan wél even duren!

Voorlopig blijft het binnen de KGN echter allemaal nog bij het oude. Hoewel... we gaan wel wat meer activiteiten ontplooiën! Zo hebben we diskettes in de aanbieding (3,5" HD schijfjes) voor een prettige prijs en is er een initiatief op Minix-gebied. Awel, zo hebben we wellicht nog wel wat aangename verrassingen voor u in petto. Ik zal u hier niet langer ophouden en wens u weer veel leesplezier met deze uitgave van de μ P Kenner. Tot ziens in Geldrop!

Joost Voorhaar.

Datum: 21 maart 1991
Lokatie: gebouw 't Kruispunt
Slachthuisstraat 22
5664 EP Geldrop
Telefoon: 040-857527
Thema: Netwerken
Entree: f 10,--

Datum: 21 maart 1991
Lokatie: gebouw 't Kruispunt
Slachthuisstraat 22
5664 EP Geldrop
Telefoon: 040-857527
Thema: Netwerken
Entree: f 10,--

Per trein:

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

Per auto:

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

Vanaf Eindhoven door het centrum van Geldrop richting Heeze. Na winkelstraat en daarna het ziekenhuis aan de rechterzijde de eerste straat links

bij de stoplichten. Dit is de Laan der vier Heemskinderen. Zie verder boven.

Programma:

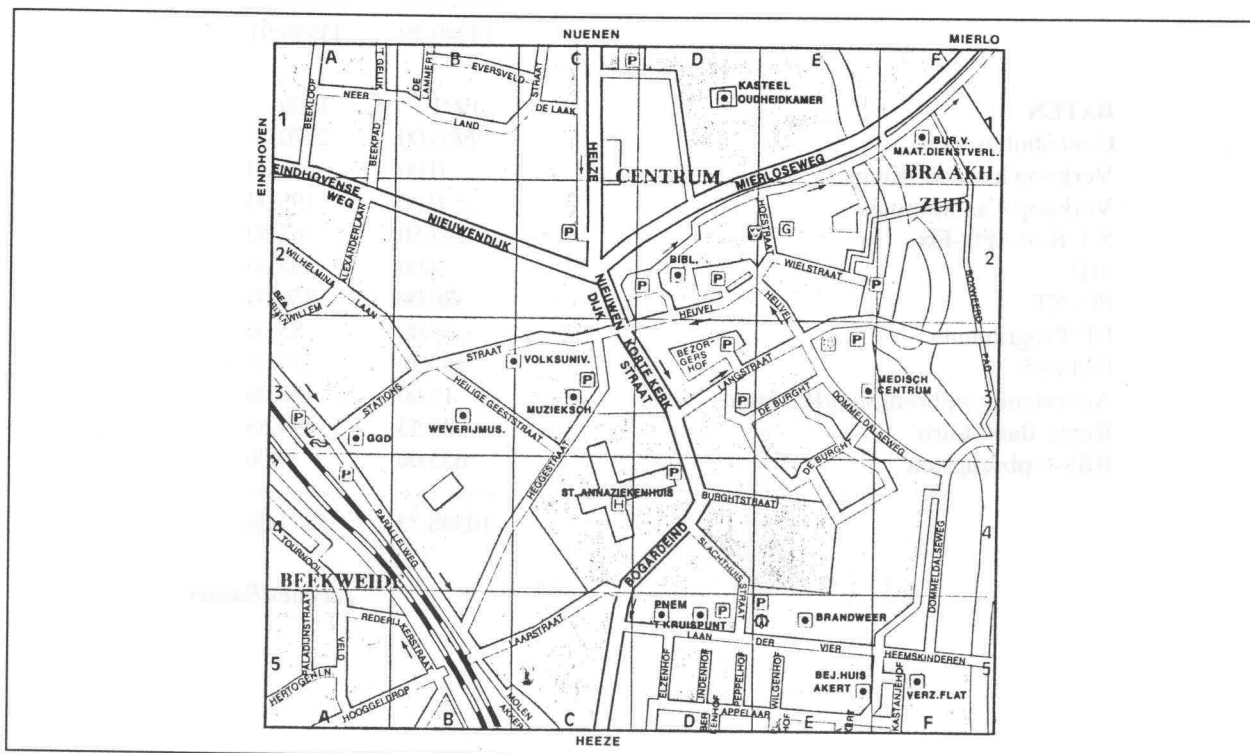
- | | |
|-------|---|
| 9:30 | Zaal open met koffie |
| 10:15 | Opening |
| 10:30 | Voordracht van Adri Hankel over SI-MON: een Seriële Interface MONitor. |
| 11:30 | Stand van zaken t.a.v. het project KGN-68k; gelegenheid voor het stellen van vragen aan en een discussie met vertegenwoordigers van de projectgroep |
| 12:00 | Forum en markt |
| 12:15 | Lunchpauze |

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

17:00 Sluiting.

Let op:

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



KIM GEBRUIKERSCLUB NEDERLAND JAARSTUKKEN 1991

Balans per 31 december 1991

AKTIVA	1991	1990
Inventaria	3332.00	1815.00
Voorraden	pm	pm
Te ontvangen posten	0.00	0.00
Geldmiddelen	9107.45	10978.16
	-----	-----
	12439.45	12793.16

PASSIVA	1991	1990
Vrije Reserve	12039.50	11436.87
Vooruitontvangen contributie	315.00	350.00
Te betalen posten	84.95	1006.29
	-----	-----
	12439.45	12793.16

STAAT VAN BATEN EN LASTEN OVER 1991

LASTEN	1991	1990
Drukkosten μ P Kenner	4554.82	4666.28
Verzendkosten	1416.82	1118.79
Redactie kosten	45.00	436.95
Bestuurskosten	1561.16	626.35
HCC-dagen	1029.50	1591.54
Afschrijving inventaris	1005.00	1200.00
SysOp kosten	3101.69	2340.00
Bijeenkomsten	36.50	0.00
Projectgroep 68000	198.80	0.00
Netwerk TU	300.00	0.00
	-----	-----
	13249.29	11979.91

BATEN	1991	1990
Contributies	7965.00	2770.00
Verkoop losse μ P Kenners	0.00	5.00
Verkoop Cas-Manuals	0.00	195.00
XT-Bios / PC-Fix	197.50	67.50
SID	0.00	295.00
PC-XT	700.00	850.00
EP-Programmer	85.00	85.00
DOS-65	255.70	359.00
Advertentie-opbrengsten/Reklame	15.00	150.00
Rente Bank-Giro	523.53	494.58
BBS-Opbrengsten	655.00	0.00
	-----	-----
	10396.73	5271.08

Jacques Banser

Halen, Contant of per Cheque

Door de drukte van de feestdagen en een bestuurswisseling bent u de vorige μ P Kenner verstoken gebleven van een verslag over de HCC-dagen. Twee jaarlijks terugkerende dagen waarop de KGN zich bekend probeert te maken bij een groot publiek. Twee dagen ook waarop de standbemanning vroeg uit de veren moet om op tijd aanwezig te zijn. Op die dagen maak je uren waarop je baas jaloers zou worden. Dat alles tegen betaling van een vrijkaart om de beurs te mogen betreden. Maar ook het plezier waarop je getraakteerd wordt door naar al die mensen te kunnen kijken. Mensen die soms voor een prikkie eigenaar worden van een computer of die lang gekoesterde harddisk. Ik blijf het een pracht gezicht vinden om naar die gelukkige gezichten te kijken van mensen die met een zware doos of dozen voor hun buik lopen. Vele bezoekers van de HCC-dagen zie je binnenkomen met een blik van: nu moet ik kopen, goedkoper wordt het nooit. Soms zien ze al na enkele dagen dezelfde PC bij de computerboer om de hoek nog enkele snipjes goedkoper. U begrijpt nu ook waarom uw scribent zich de vrijheid nam om de letters HCC te vertalen zoals in de kop van dit artikel.

In de onze KGN-stand stond dit jaar een PC met een deel van het BBS. Men kon daarop lokaal inloggen en rond kijken. En eventueel de aanwezige files downloaden naar een flop. Er was een PC waarop het besturingssysteem MINIX draaide en waaraan een demo zelfbouw 'toetsenbord' hing. Dat laatste trok veel belangstelling bij de dames. Die waren namelijk in de veronderstelling dat ze met die computer en dat toetsenbord de was automatisch konden doen. Van de firma "Besamu" stond er een lichtkrant die een constante stroom informatie gaf over de KGN. Doordat de krant open was en het geflikker van de ledjes, werd de aandacht bij veel bezoekers getrokken. Een van de bestuursleden had een draagbaar appelkistje meegenomen waarop o.a. wat spelletjes stonden. Daarbij lagen er een aantal P Kenners en t-shirts welke door de mensen gekocht konden worden. Tevens lag er wat informatie ter inzage over het KGN-68k MINIX project. Dat alles bij elkaar leverde een bijna constante stroom van bezoekers aan onze stand. Mensen die gewoon even een blik op het e.e.a. wilden werpen. Maar ook mensen die meer informatie vroegen over hetgeen stond uitgesteld en over het doen en laten van de KGN. Dat laatste is voor de vereniging natuurlijk het be-

langrijkste. In een persoonlijk gesprek en met de aanwezige "handel" kun je goed aanschouwelijk maken waar de KGN voor staat. Klein, Goed en Nette-verbeteren.

Wat leveren die dagen nu op? Allereerst een aantal nieuwe leden of mensen die dat in de loop van het jaar worden. Het is niet altijd duidelijk of nieuwe aanmeldingen door deze dagen komen, omdat we ook ons bulletin bord profileren. Van onze sy-sop/penningmeester/ledenadministrateur/... heb ik vernomen dat het opvalt dat er na de HCC-dagen altijd een verhoogde belangstelling is voor het BBS. Die belangstelling wordt niet altijd direct omgezet in aanmeldingen, maar het gebeurt ook later. We moeten ook niet vergeten dat de stand een aanloop en ontmoetingsplek is voor leden.

Je kunt daar even uitrusten bij een lekker bakkie koffie of thee. Als er iets gekocht is kan het eventueel uitgeprobeerd worden. Dat laatste is overigens geen overbodige luxe. Twee aangeschafte modems bleken niet te doen wat er van verwacht mag worden. De leverancier was bereid het geld terug te geven omdat ze de voordeur van het Jaarbeurscomplex nog niet gepasseerd waren. Bij een SVGA-kaart bleek de bijbehorende software niet goed te werken. Hiervoor is een oplos-

sing gevonden waar beide partijen tevreden mee waren. Voor zover ik weet werkt thans alles en is de koper tevreden. Uw columnist en de secretaris werden, door de toenmalige voorzitter, gewezen op een muis voor slechts fl. 25,-. De baasjes zijn er zeer tevreden mee. De poezen wat minder want die worden nu niet meer bij de PC toegelaten.

Als laatste wil ik u nog een geheimpje verklappen. Tijdens die HCC-dagen liet de toenmalige voorzitter weten dat hij niets aan wilde schaffen. Hij had de centjes nodig voor een nieuwe auto. Groter, comfortabeler, luxer en duurder. Nou heeft ie toch stiekem een paar oude monitoren meegesleept. We weten dat hij ze gerepareerd heeft en verkocht. Sindsdien hebben we hem niet meer gezien. Zou die zoveel winst gemaakt hebben dat hij nu die nieuwe Rolls Royce aan het inrijden is? Wie het weet mag het zeggen, anders niet verder vertellen hoor.

Tonny Schäffer

Vele bezoekers van de HCC-dagen zie je binnenkomen met een blik van: nu moet ik kopen, goedkoper wordt het nooit.

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

Heeft u nou ook zo'n hekel aan dat eentonige beepje dat IBM als "welkomstwoord" ten gehore brengt bij het starten van uw PC? De hoge heren ontwerpers bij IBM hadden blijkbaar geen al te hoge dunk van computertoepassingen in de muziekwereld toen ze de PC ontwierpen...

Met wat eenvoudige hardware en de juiste programmatuur is zelfs van zo'n eentonige PC nog een redelijk muziekinstrument te maken. In deze aflevering van "To Share..." presenter ik een tweetal programma's die met deze hardware overweg kunnen en waarmee heel aardige resultaten te halen zijn: modplay om MOD-files mee af te spelen en ModEdit om ze aan te passen en/of eigen muziek te maken.

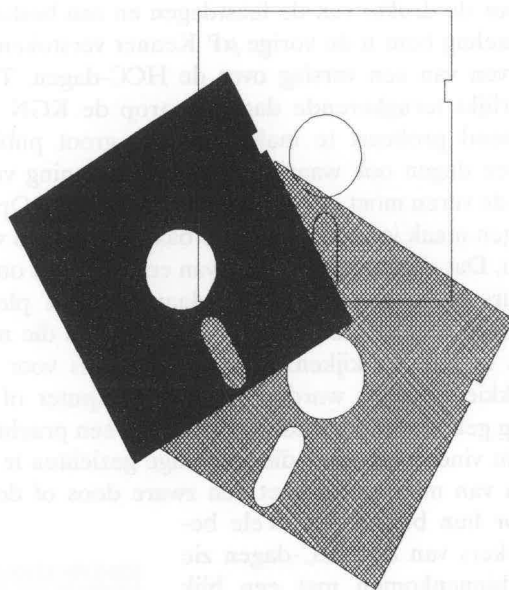
Sound files

Om modplay en ModEdit te kunnen gebruiken heb je in eerste instantie een aantal .MOD-sound files nodig. Deze files bestaan uit een aantal samples van muziekinstrumenten, gevolgd door een omschrijving van de noten die met de gedefinieerde instrumenten gespeeld moeten worden. Iedere MOD-file kan 15 of 31 verschillende instrument-samples bevatten en, willen ze geschikt zijn voor het afspelen op de PC, mag niet meer dan maximaal 4 tracks beschrijven.

Op het bulletin board van de vereniging is een uitgebreide verzameling MOD-files beschikbaar, zodat er naar hartelust geëxperimenteerd kan worden.

De programmatuur

ModPlay heeft minimaal een 10 MHz PC/AT nodig. Wil je kwalitatief iets meer dan het uiterste minimum, dan is een 12 of 16 MHz AT of een nog sneller beest aanbevelenswaard. Voor ModEdit geldt dezelfde beperking.



Met behulp van modplay kan muziek afgespeeld worden via de interne speaker, een eventuele soundblaster of via een zogenaamde covox module. Zo'n covox module is in feite niet meer dan een D/A omzetter die aan de parallele poort gehangen kan worden. De analoge uitgang moet dan verbonden worden met een versterker. De meest goedkope vorm van een covox module bestaat uit een weerstandsnetwerkje, dat met een beetje zorgvuldig solderwerk en wat gefrommel in de marge zelfs nog wel in de behuizing van een standaard 25-pens plug past. Dit weerstandsnetwerkje moet echter bijzonder goed uitgebalanceerd zijn wil de muziek een beetje ruis- en vervormingsarm geproduceerd kunnen worden. Veel betere resultaten kunnen behaald worden als er voor een "echte" D/A omzetter gekozen wordt (zie figuur 1). Daar de originele Amiga MOD-files allemaal 10 bits breed zijn en onze parallele poort slechts 8 bits, wordt wel iets aan dyna-

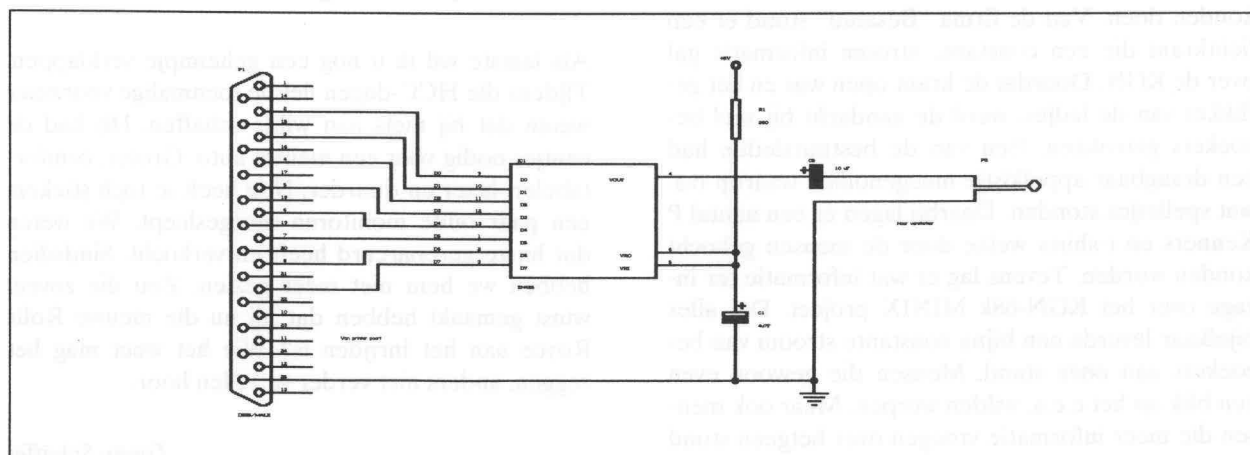


Fig. 1: covox in mono uitvoering

miek opgeofferd, maar een kniesoor die daar op let natuurlijk.

Het kan ook nog luxer dan met een enkele D/A omzetter. In figuur 2 staat het schema voor een stereo-uitvoering, die (op mono-niveau) 9 bits breed werkt.

Het geluid dat door de interne speaker voortgebracht wordt valt natuurlijk een stuk slechter uit. In de documentatie wordt gesproken over een totale breedte van "maar liefst" 5 bits breed... niet echt om over naar huis te schrijven dus. En dan moet je ook nog een goede speaker in je PC hebben... heb je een laptop of een Headstart-achtig geval met zo'n piëzo zoemertje, dan kun je het wel vergeten met die interne speaker.

Met modplay kunnen samples alleen afgespeeld worden. Wil je ze aanpassen of de instrumenten gebruiken om je eigen muziekstukje in elkaar te schroeven, dan heb je ModEdit nodig. Met ModEdit kun je samples bij elkaar rapen om zo een "pattern" te bouwen, een bij elkaar behorende blok noten. Ieder pattern mag 64 noten (lees: samples) omvatten. De totale file bestaat uit maximaal 32 van deze patterns, waarmee een maximum van 128 patterns vastgelegd is. Op het eerste gezicht lijkt dat erg weinig, maar als je bedenkt dat iedere sample ook al uit stukken muziek die meerdere noten omvatten kan bestaan, valt dit gegeven in de praktijk wel mee.

Het werken met modplay is bijzonder eenvoudig. Met behulp van een selectiebalk kun je een modfile selecteren. Met behulp van de C kan voor andere outputdevices gekozen worden (speaker, soundblaster, covox op 1 poort, covox op twee poorten, covox stereo etc.).

Tijdens het afspelen kan je nog wat rommelen met de snelheid waarmee 't hele zaakje loopt. In de praktijk zal je dit bijna nooit doen, behalve dan mis-

schien om eens te horen wat voor effect dat heeft.

ModEdit is wat spartaans uitgerust voor een editor. Het hele programma draait om een wel heel erg sobere menustructuur die niet echt prettig werkt. De menu's bestaan steeds uit een enkel window die in het midden van een verder leeg scherm geplaatst wordt. Niet echt "state-of-the-art" dus, maar wel werkbaar. ModEdit kan overigens uit zichzelf geen geluid produceren, daar is een meegeleverde TSR (= "Terminate stay Resident", een residente devicedriver dus) voor nodig.

Documentatie

De handleiding van ModPlay is klein maar handzaam. De paginanummering ontbreekt, maar dit laat zich in een dergelijk klein document niet als storend betitelen. ModEdit is een stuk complexer, en aan de handleiding is dan ook wat meer zorg besteedt. Alles wat je zoekt staat in de 32 pagina's tellende docfile, maar echt overzichtelijk is het hele zaakje niet. Naast deze twee "handleidingen" is er nog een klein tekstbestandje met informatie over de covox-hardware en over de TSR.

Conclusie

Met ModPlay kan een heel aardige geluidskwaliteit behaald worden. In combinatie met ModEdit laat zich zelfs nog het een en ander aanpassen. Wat duidelijk ontbreekt in deze programmatuur is de ondersteuning van bijvoorbeeld synthesizers en/of samplers om zelf samples en muziek te kunnen maken. Best jammer, maar hiervoor heb je óf hele dure programmatuur nodig óf een andere computer. De covox module wordt steeds meer door allerlei spelletjes ondersteund, en 't ding kost heel wat minder dan een dure soundblaster.

Joost Voorhaar

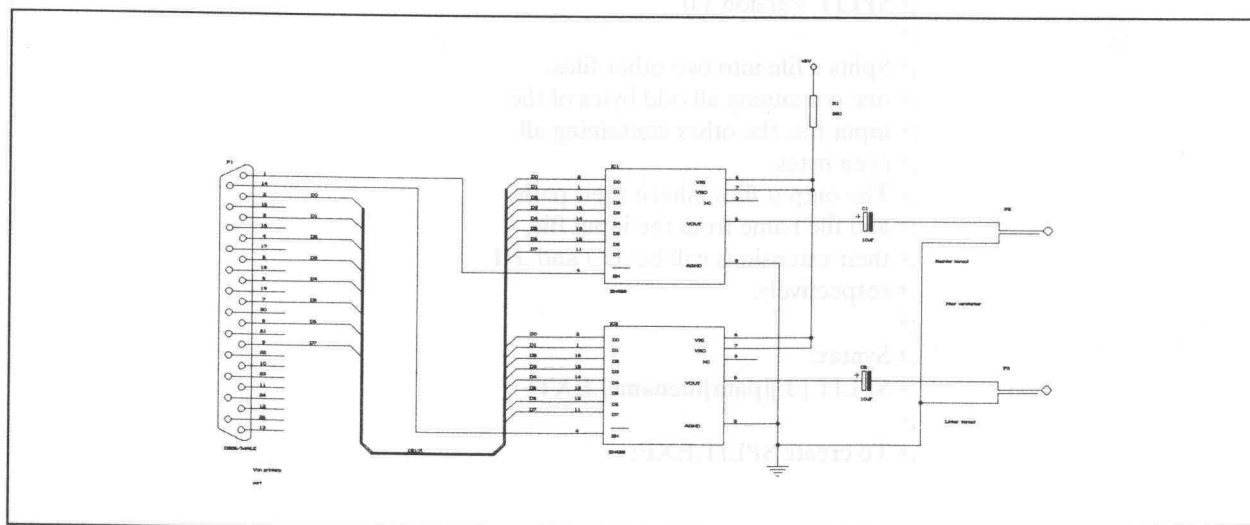


Fig. 2: covox in stereo uitvoering

Gespleten files maken

Alweer een heel tijdje terug was ondergetekende in de weer met een AT-BIOS dat uitgebreid werd met een diagnose-programma. Het gevolg daarvan was, dat het geheel een omvang kreeg van 64k byte. Op zich is dat geen wereldschokkend nieuws, want een beetje programma is tegenwoordig groter dan de in een normale MS-DOS machine beschikbare 640k byte. Toch was er een vuiltje aan de lucht.

Dat vuiltje bestond uit de EPROM-programmer EP in DOS65. Gemaakt met veel overleg en geduld. Kan ook met 16-, en zelfs met 32-bit data overweg dankzij de instelbare address increment. Maar de RAMbuffer van EP was en is slechts 32k byte...

Inmiddels was in mijn PC echter een EPROMprogrammer verschenen van een degelijk en gerespecteerd merk. De fabrikant was echter "vergeten", om mogelijkheden voor 16- en 32-bit data in te bouwen, want wilde ook graag zijn duurdere modellen die dat wel konden aan de man zien te brengen. Leuk voor de fabrikant, maar niet voor mij, want mijn budget is opgegaan aan een wanstaltige auto.

Zelf doen

Het BIOS van een AT moet dus gesplitst worden in een file met alle low-bytes, en een file met alle high-

bytes erin. Dat is niet moeilijk: een bestand openen om te lezen, en twee om te schrijven. Lees steeds een byte uit het invoerbestand, en stamp dat om beurten in het ene, of in het andere uitvoerbestand. Na een hele middag knutselen en proberen was het onderstaande stukje software het resultaat.

Het programma werkt met zogenaamde "handles", dat zijn words waarmee je aan MS-DOS kunt vertellen met welke file je bezig bent. Eerst moet je aan MS-DOS vragen om zo'n handle, door de filenaam op te geven. Als je van MS-DOS een handle hebt gekregen, hoef je alleen maar de handle op te geven in plaats van de filenaam. In sjieke talen als PASCAL en zo maakt dat niet zo veel uit want daar springen ze met strings net zo gemakkelijk om als schoonmoeders met erfenissen. Maar zoals hieronder te zien is, is dat handle-gedoe in assembler reuze handig. Voor de studiefreaks: vat een MS-DOS functie-tabel, en begin met lezen. Voor de luiaards: de sourcetekst staat ook op het Ultimate BBS.

Een volgende keer komt het tegenovergestelde: maak van twee files die low- en high-bytes bevatten weer het oorspronkelijke blok bytes.

Nico de Vries

```

NAME      SPLIT
PAGE      66,131
TITLE     SPLIT ---- Split File into odd and even parts

;*****
;*
;* SPLIT Version 1.0
;*
;* Splits a file into two other files:
;* one containing all odd bytes of the
;* input file, the other containing all
;* even bytes.
;* The output files inherit their path
;* and file name from the input file,
;* their extensions will be .LO and .HI
;* respectively.
;*
;* Syntax:
;* SPLIT [d:][path]filename[.EXT]
;*
;* To create SPLIT.EXE:

```

Fig. 1: sourcetekst van SPLIT.EXE

```

;* MASM SPLIT;
;* LINK SPLIT,TEMP;
;* EXEPACK TEMP.EXE SPLIT.EXE
;*
;*****

CR      EQU      0DH      ;ASCII carriage return
LF      EQU      0AH      ;ASCII line feed
TAB      EQU      09H      ;ASCII TAB code
BLANK    EQU      20H      ;ASCII space code

COMMAND EQU      80H      ;buffer for command tail

B_SIZ   EQU      16384     ;size of records

OUTPUT_HANDLE EQU      1   ;handle of standard output device
ERROR_HANDLE  EQU      2   ;handle of standard error device

CSEG      SEGMENT PARA PUBLIC 'CODE'
            ASSUME      CS:CSEG,DS:DATA,ES:DATA,SS:STACK

;*****
;*
;* Start of program. Test for correct
;* DOS version, because SPLIT uses
;* 'handle calls' not available in
;* DOS 1.X.
;*
;*****

SPLIT      PROC      FAR
            PUSH      DS      ;save DS:0000 for final
            XOR      AX,AX    ;return to DOS in case
            PUSH      AX      ;function 04CH cannot be used
            MOV      AX,Seg DATA ;make our own data segment
            MOV      ES,AX     ;accessible via ES register
            MOV      AH,030H   ;check version
            INT      21H       ;of DOS
            CMP      AL,2      ;if version 1.X
            JAE      SPLIT1
            MOV      DX,Offset DOS_ERR ;point to DOS 1.X error message
            PUSH      ES       ;point DS
            POP      DS       ;to data segment
            MOV      AH,9      ;cannot use modern function here
            INT      21H       ;print message
            RET              ;and exit to caller

;*****
;*
;* Read file name from input buffer.
;*
;*****

SPLIT1:    CALL      GET_FILENAME ;get path and file name spec
            PUSH      ES       ;point DS
            POP      DS       ;to own data segment
            JNC      SPLIT2    ;if no error, proceed
            MOV      DX,Offset SIGN_ON ;point to description message
            MOV      CX,SIGN_ON_LENGTH ;get length

```

```

JMP          SPLIT16                ;and print message
;*****
;*
;* Try to open input file.
;*
;*****
SPLIT2:      CALL      OPEN_INPUT    ;try to open input file
JNC          SPLIT3                ;if OK, proceed
MOV          DX,Offset FILE_NT_FND  ;point to message
MOV          CX,FILE_NT_FND_L       ;get length
JMP          SPLIT16                ;and print message
;*****
;*
;* Get length of input file. Print a
;* warning if the file holds an odd
;* number of bytes, as this will result
;* output files with different lengths.
;*
;*****
SPLIT3:      CALL      GET_FILE_LENGTH ;get input file length
JC           SPLIT4                ;if error exit with message
TEST         Word Ptr IN_SIZE,0001H ;if file length is even
JE           SPLIT5                ;proceed
MOV          DX,Offset ODD          ;point to 'File holds odd
MOV          CX,ODD_LENGTH          ;get length
CALL         WRITE_STD              ;print message
JMP          Short SPLIT5            ;and proceed
SPLIT4:      MOV          DX,Offset MSG4 ;point to 'Input file is empty'
MOV          CX,MSG4_LENGTH         ;get length
JMP          SPLIT15                ;and print message
;*****
;*
;* Create the output file names.
;*
;*****
SPLIT5:      CLD                    ;set direction forward
MOV          DI,Offset OUTL_NAME     ;point DI to output name low
CALL         MAKE_OUTNAME            ;create output file name low
JC           SPLIT6                ;if no usable file name, exit with error
MOV          AX,'OL'                ;else get extension
STOSW                           ;move too
XOR          AL,AL                  ;get a zero
STOSB                           ;and terminate outputname low
MOV          DI,Offset OUTH_NAME     ;and DI to output name high
CALL         MAKE_OUTNAME            ;create output file name hi
MOV          AX,'IH'                ;get extension
STOSW                           ;move too
XOR          AL,AL                  ;get a zero
STOSB                           ;and terminate outputname high
;*****
;*
;* Create the output files.
;*
;*****
MOV          DX,Offset OUTL_NAME     ;get file name address
MOV          AL,1                    ;access is write
MOV          CX,0                    ;normal attribute

```

```

MOV     AH,03CH           ;create file
INT     21H               ;via DOS
JC      SPLIT6            ;if error, print it
MOV     OUTL_HANDLE,AX    ;else save handle
MOV     DX,Offset OUTH_NAME ;get file name address
MOV     AL,1              ;access is write
MOV     CX,0              ;normal attribute
MOV     AH,03CH           ;create file
INT     21H               ;via DOS
JNC     SPLIT7            ;if no error, proceed
SPLIT6: JMP     SPLIT14    ;and print message
SPLIT7: MOV     OUTH_HANDLE,AX ;else save handle
;*****
;*
;* Read 1st block of input file into
;* input buffer. Check if DOS returns
;* an error. If so, report empty input
;* file.
;*
;*****
CALL     READ_BLOCK       ;read 1st block of data
JNC     SPLIT8            ;if OK, proceed
MOV     DX,Offset MSG4    ;point to 'Input file is empty'
MOV     CX,MSG4_LENGTH    ;get length
JMP     SPLIT14           ;and print message
;*****
;*
;* Perform the actual split.
;*
;*****
SPLIT8: MOV     Word Ptr OUTL_PTR,0 ;set output byte counter low
SPLIT9: MOV     Word Ptr OUTH_PTR,0 ;set output byte counter high
CALL     GET_CHAR         ;read 1 byte from input
JC      SPLIT12           ;if error (input exhausted), exit
MOV     DI,Offset OUTL_BUF ;get buffer address
MOV     BX,OUTL_PTR       ;get buffer offset
MOV     [DI+BX],AL        ;put byte in buffer
INC     OUTL_PTR          ;adapt output byte counter
DEC     IN_SIZEL          ;count input bytes
JNE     SPLIT10           ;if low zero,
MOV     AX,IN_SIZEH       ;get hi
OR      AX,AX             ;if zero
JE      SPLIT12           ;write remaining bytes
DEC     IN_SIZEH          ;else adapt count hi
SPLIT10: CALL    GET_CHAR  ;read 1 byte from input
MOV     DI,Offset OUTH_BUF ;get buffer address
MOV     BX,OUTH_PTR       ;get buffer offset
MOV     [DI+BX],AL        ;put byte in buffer
INC     OUTH_PTR          ;adapt output byte counter
INC     BX                ;and BX
DEC     IN_SIZEL          ;count input bytes
JNE     SPLIT11           ;if low zero,
MOV     AX,IN_SIZEH       ;get hi
OR      AX,AX             ;if zero too
JE      SPLIT12           ;write remaining bytes
DEC     IN_SIZEH          ;else adapt count hi
SPLIT11: CMP     BX,B_SIZE/2 ;if buffer not full yet

```



```

JNE          SPLIT9          ;continue to fill
CALL         WRITE_BLOCK     ;else write buffer to files
JMP          SPLIT8          ;and loop
;*****
;*
;* Input file completely read. Flush data
;* remaining in output buffers in
;* output files.
;*
;*****
SPLIT12:     MOV             BX,OUTL_PTR      ;get buffer offset low
MOV          AX,OUTH_PTR     ;get buffer offset high too
OR           AX,BX           ;if no more bytes to do
JE           SPLIT13         ;exit now
CALL         WRITE_BLOCK     ;else write output files
SPLIT13:     CALL         CLOSE_INPUT        ;close input file
CALL         CLOSE_OUTPUT    ;close output files
MOV          DX,Offset MSG5   ;point to 'SPLIT succesful.'
MOV          CX,MSG5_LENGTH   ;get length
CALL         WRITE_STD       ;print message
MOV          AX,04C00H        ;command is exit DOS
INT          21H              ;with error level 0
;*****
;*
;* Error exit: problem with output
;* files. Report problem through error
;* output handle and close all files.
;*
;*****
SPLIT14:     CALL         CLOSE_OUTPUT        ;close both output files
MOV          DX,Offset WR_ERR ;point to 'Error writing output file'
MOV          CX,WR_ERR_LENGTH ;get length
;*****
;*
;* Error exit: problem with input file.
;* Report problem through error output
;* handle and close the file.
;*
;*****
SPLIT15:     CALL         CLOSE_INPUT        ;close input file
;*****
;*
;* Error exit: problem with input file
;* name. Report problem through error
;* output handle.
;*
;*****
SPLIT16:     CALL         WRITE_ERROR        ;write error message
MOV          AX,04C01H        ;exit to DOS
INT          21H              ;with error level 1

SPLIT        ENDP

;*****
;*
;* Move input file description to
;* internal buffer, stripping it from

```

```

; * spaces and TAB characters.
; * If no filename is found, an error is
; * flagged.
; *
; *****
GET_FILENAME PROC NEAR
MOV SI,Offset COMMAND ;point SI to command line
MOV DI,Offset INPUT_NAME ;point DI to buffer for name
CLD ;set direction forward
LODSB ;get length byte
OR AL,AL ;if no name
JE GET_FILENAME5 ;exit with error
GET_FILENAME1: LODSB ;else get command line byte
CMP AL,CR ;if carriage return
JE GET_FILENAME5 ;exit with error
CMP AL,BLANK ;if space
JE GET_FILENAME1 ;get next character
CMP AL,TAB ;if TAB
JE GET_FILENAME1 ;get next character
GET_FILENAME2: STOSB ;else store file name character
GET_FILENAME3: LODSB ;get next
CMP AL,CR ;if last
JE GET_FILENAME4 ;exit without error
CMP AL,BLANK ;if space
JE GET_FILENAME3 ;skip it
CMP AL,TAB ;if not TAB
JNE GET_FILENAME2 ;store and get next character
GET_FILENAME4: CLC ;flag no error
RET ;and exit
GET_FILENAME5: STC ;flag error
RET ;and exit
GET_FILENAME ENDP

; *****
; *
; * Open input file. If an error occurs
; * carry will be set.
; *
; *****
OPEN_INPUT PROC NEAR
MOV DX,Offset INPUT_NAME ;point DX to filename
MOV AL,0 ;read only
MOV AH,03DH ;open handle
INT 21H
MOV INPUT_HANDLE,AX ;store handle number
RET ;and exit
OPEN_INPUT ENDP

; *****
; *
; * Read file length of input file. If
; * an error occurs carry will be set.
; *
; *****
GET_FILE_LENGTH PROC NEAR
MOV AL,2 ;get end file position
MOV AH,042H ;move file pointer

```

```

MOV     BX,INPUT_HANDLE    ;get handle
XOR     DX,DX              ;get zero
XOR     CX,CX              ;offset
INT     21H                ;get file length
JC      LENGTH_ERROR       ;if error exit
MOV     IN_SIZEH,DX         ;save MSW
MOV     IN_SIZEL,AX         ;save LSW
MOV     AL,0               ;get begin file position
MOV     AH,042H            ;move file pointer
MOV     BX,INPUT_HANDLE    ;get handle
XOR     DX,DX              ;get zero
XOR     CX,CX              ;offset
INT     21H                ;get file length
LENGTH_ERROR: RET          ;and exit
GET_FILE_LENGTH ENDP

```

```

;*****
;*
;* Create output file name. The input
;* file name string is copied to a
;* buffer upto the end, or the period
;* at the start of extension. The new
;* name string is terminated with a
;* period.
;*
;*****
MAKE_OUTNAME PROC NEAR
CLD                                ;set direction forward
MOV     SI,Offset INPUT_NAME      ;point SI to input name
LODSB                                ;get character
CMP     AL,','                    ;if 1st is not a period
JNE     MAKE_NAME3                ;parse input string
STOSB                                ;else store character
MAKE_NAME1: LODSB                  ;get next
STOSB                                ;move to buffer
OR      AL,AL                     ;if at end now
JE      MAKE_NAME5                ;exit
CMP     AL,'\'                    ;if not backslash
JNE     MAKE_NAME1                ;loop
MAKE_NAME2: LODSB                  ;get next character
CMP     AL,','                    ;if at period of extension
JE      MAKE_NAME4                ;stop now
MAKE_NAME3: STOSB                  ;else move character
OR      AL,AL                     ;if not at end now
JNE     MAKE_NAME2                ;loop
MOV     AL,','                    ;else get period
MAKE_NAME4: STOSB                  ;in output name
CLC                                ;flag no error
RET                                ;and exit
MAKE_NAME5: STC                    ;flag error
RET                                ;and exit
MAKE_OUTNAME ENDP

```

```

;*****
;*
;* Close input file. If an error occurs
;* carry will be set.

```

```

;
;*****
CLOSE_INPUT PROC NEAR
    PUSH    AX
    PUSH    BX
    MOV     BX,INPUT_HANDLE ;get file handle
    MOV     AH,03EH         ;close handle
    INT     21H
    POP     BX
    POP     AX
    RET
;and exit
CLOSE_INPUT ENDP

;*****
;
;* Close output files. If an error
;* occurs, carry will be set.
;
;*****
CLOSE_OUTPUT PROC NEAR
    PUSH    AX
    PUSH    BX
    MOV     BX,OUTL_HANDLE ;get file handle
    MOV     AH,03EH         ;close handle
    INT     21H
    MOV     BX,OUTH_HANDLE ;get file handle
    MOV     AH,03EH         ;close handle
    INT     21H
    POP     BX
    POP     AX
    RET
;and exit
CLOSE_OUTPUT ENDP

;*****
;
;* Get character from the input file
;* buffer. If the buffer is empty, it
;* will be filled reading from disk.
;* If the input file is completely read
;* carry will be set.
;
;*****
GET_CHAR PROC NEAR
    PUSH    BX ;save BX
GET_CHAR1: MOV     BX,INPUT_PTR ;get current buffer address
    CMP     BX,B_SIZ ;if not through
    JNE     GET_CHAR2 ;get character from buffer
    MOV     INPUT_PTR,0 ;else zero buffer address
    CALL    READ_BLOCK ;read in next block
    JNC     GET_CHAR1 ;if no error, loop
    POP     BX ;restore BX
    RET ;else exit
GET_CHAR2: MOV     AL,[INPUT_BUFFER + BX] ;get buffer character
    INC     INPUT_PTR ;adapt buffer address
    POP     BX ;restore BX
    CLC ;flag no error
    RET ;and exit

```

GET_CHAR

ENDP

```

;*****
;*
;* Fill the buffer by reading the input
;* file from disk. If the file is read
;* up to the end, carry will be set.
;*
;*****

```

READ_BLOCK

```

PROC      NEAR
MOV       BX,INPUT_HANDLE    ;get file handle
MOV       CX,B_SIZ           ;and byte count
MOV       DX,Offset INPUT_BUFFER
MOV       AH,03FH            ;read from handle
INT       21H
MOV       INPUT_PTR,0        ;zero buffer pointer
OR        AX,AX              ;if any read in
JNZ       READ_BLOCK1
STC                           ;else flag error
RET                           ;and exit
ENDP

```

READ_BLOCK1:
READ_BLOCK

```

;*****
;*
;* Write current output buffers to the
;* output files. If an error occurs
;* carry will be set.
;*
;*****

```

WRITE_BLOCK

```

PROC      NEAR
MOV       BX,OUTL_HANDLE     ;get file handle
MOV       CX,OUTL_PTR        ;and byte count
MOV       DX,Offset OUTL_BUF
MOV       AH,040H            ;write to handle
INT       21H
JC        WRITE_BLOCK1       ;if error, exit
MOV       BX,OUTH_HANDLE     ;get file handle
MOV       CX,OUTH_PTR        ;and byte count
OR        CX,CX              ;if zero bytes to write
CLC                           ;flag no error
JE        WRITE_BLOCK1       ;and exit
MOV       DX,Offset OUTH_BUF
MOV       AH,040H            ;write to handle
INT       21H
RET                           ;and exit
ENDP

```

WRITE_BLOCK1:
WRITE_BLOCK

```

;*****
;*
;* Write to the standard output device.
;* This output can be redirected.
;*
;*****

```

WRITE_STD

```

PROC      NEAR
MOV       BX,OUTPUT_HANDLE   ;get handle
MOV       AH,40H             ;write to handle
INT       21H

```



```

WRITE_STD      RET      ;and exit
                ENDP

                ;*****
                ;*
                ;* Write to the standard error device.
                ;* This output can not be redirected.
                ;*
                ;*****

WRITE_ERROR    PROC      NEAR
                MOV      BX,ERROR_HANDLE ;get handle
                MOV      AH,40H          ;write to handle
                INT      21H
                RET
                ENDP

WRITE_ERROR    PROC      NEAR
                MOV      BX,ERROR_HANDLE ;get handle
                MOV      AH,40H          ;write to handle
                INT      21H
                RET
                ENDP

CSEG           ENDS

                ;*****
                ;*
                ;* Work and buffer area.
                ;*
                ;*****

DATA           SEGMENT PARA PUBLIC 'DATA'

                ;*****
                ;*
                ;* Buffers for file names.
                ;*
                ;*****

INPUT_NAME     DB      64 DUP (0)        ;buffer for input name
INPUT_HANDLE   DW      0                 ;buffer for handle number
INPUT_PTR      DW      0                 ;pointer into input buffer
IN_SIZEH       DW      0                 ;file length, hi word
IN_SIZEL       DW      0                 ;file length, lo word

OUTL_NAME      DB      64 DUP (0)        ;buffer for input name
OUTL_HANDLE    DW      0                 ;buffer for handle number

OUTL_PTR       DW      0                 ;pointer into input buffer
OUTH_PTR       DW      0                 ;pointer into input buffer

OUTH_NAME      DB      64 DUP (0)        ;buffer for input name
OUTH_HANDLE    DW      0                 ;buffer for handle number

                ;*****
                ;*
                ;* Buffers for data processing.
                ;*
                ;*****

INPUT_BUFFER   DB      B_SIZ DUP (?)     ;input buffer for DOS read
OUTL_BUF       DB      B_SIZ/2 DUP (?)   ;low output buffer
OUTH_BUF       DB      B_SIZ/2 DUP (?)   ;high output buffer

                ;*****
                ;*

```

```

;* Text messages.
;*
*****
SIGN_ON      DB      CR,LF,'SPLIT V1.0'
              DB      CR,LF,CR,LF,'by NdV, for KIM Gebruikersclub Nederland.'
              DB      CR,LF,'Splits input file into two output files,'
              DB      CR,LF,'one holding all odd, and one holding all'
              DB      CR,LF,'even bytes.'
              DB      CR,LF,'Output files inherit their path and name'
              DB      CR,LF,'from the input file, but will have the'
              DB      CR,LF,'extensions .LO and .HI by default.',CR,LF
              DB      CR,LF,'Usage: SPLIT [d:][path]filename[.ext]',CR,LF
MSG2          DB      CR,LF,'The input file name is missing.',CR,LF
SIGN_ON_LENGTH EQU    $-SIGN_ON
MSG2_LENGTH  EQU    $-MSG2

FILE_NT_FND   DB      CR,LF,'Cannot find input file.',CR,LF
FILE_NT_FND_L EQU    $-FILE_NT_FND

DOS_ERR       DB      CR,LF,'Incorrect DOS version, must be DOS 2.0'
              DB      CR,LF,'or higher.',CR,LF,'$'

MSG4          DB      CR,LF,'Nothing to do: input file is empty!',CR,LF
MSG4_LENGTH  EQU    $-MSG4

MSG5          DB      CR,LF,'SPLIT succesfully completed.',CR,LF
MSG5_LENGTH  EQU    $-MSG5

ODD           DB      CR,LF,'WARNING: Input file holds odd number'
              DB      CR,LF,'of bytes. The output files will differ'
              DB      CR,LF,'in length. Split will be done.',CR,LF
ODD_LENGTH   EQU    $-ODD

WR_ERR        DB      CR,LF,'Error writing output file(s).',CR,LF
WR_ERR_LENGTH EQU    $-WR_ERR

DATA          ENDS
*****
;*
;* The stack.
;*
*****
STACK        SEGMENT PARA STACK 'STACK'

              DB      64 DUP (?)

STACK        ENDS

              END      SPLIT

```

Device drivers in TurboPascal

Turbo Pascal is, zoals bekend, een stevige PASCAL variant van Borland International. Naast de ISO-standaard PASCAL heeft turbopuke een uitgebreide library en een aantal extensies die het geheel tot een redelijk bruikbare programmeertaal maken. Eén van die uitbreidingen is natuurlijk het UNIT-principe. Een UNIT bestaat uit gecompileerde stukken sourcecode, die meegelinkt kunnen worden in een totaalprogramma. Op deze manier kun je heel eenvoudig speciale toolboxes en libraries opbouwen die dan niet iedere keer opnieuw gecompileerd hoeven te worden.

Een andere sterke kant van TurboPascal die (he-las!) erg weinig wordt gebruikt, zijn de device drivers. Je kunt in TP 4.0 en hoger namelijk tamelijk eenvoudig een device driver schrijven die allerlei handige klusjes voor je opknapt. Teksten naar de seriële poort? Via een device driver! Een speciaal netwerk gebouwd dat via de parallele poort werkt? Via een device driver! Hoe zo'n device driver nu precies in elkaar steekt wordt hier omschreven aan de hand van het bijgaande voorbeeld: een device driver voor het implementeren van ANSI-sequences.

ANSI (een acroniem voor "American National Standardization Institute") heeft een aantal richtlijnen opgesteld volgens welke terminals functies kunnen uitvoeren die niet in de standaard character sets opgenomen zijn. De UNIT die we hier gebruiken als voorbeeld ondersteunt een subset van het totale ANSI-gebeuren, de zogenaamde ANSI-BBS subset. Deze subset omschrijft de codes zoals die op de meeste bulletin boards in gebruik zijn. De ANSI-BBS codes beginnen altijd met een escape character (ASCII 27), gevolgd door een vierkante haak openen. De codes die daarachter volgen zijn bepalend voor de functie van de code (zie tabel 1).

Terug naar de device driver. Een device driver bestaat in Turbo Pascal uit een aantal functies en meestal één procedure. Deze procedure wordt gebruikt om een textfile aan de devicedriver te knopen zoals de "Assign" procedure dat doet. Voor iedere devicedriver is dan een speciale uitvoering van die assign functie (bijvoorbeeld AssignCrt om de "snelle" crt routines aan een textfile te verbinden). De taak van deze procedure bestaat uit het initialiseren van een speciaal record, het textrecord. Dat textre-

cord wordt door de compiler gebruikt om te weten waar de functies zich bevinden die de werkelijke devicedriver implementeren en bevat een aantal gegevens omtrend te status van de driver. Verder heeft het ding nog een buffer en een aantal vrij te definiëren velden beschikbaar. Dit textrecord is als volgt gedefinieerd:

```
TYPE    textbuf = ARRAY [0..127] OF char;
        textrec = RECORD
            handle    : word;
            mode      : word;
            bufsize   : word;
            private   : word;
            pufpos    : word;
            bufend    : word;
            bufptr    : ^textbuf;
            openfunc  : pointer;
            inoutfunc : pointer;
            flushfunc : pointer;
            closefunc : pointer;
            userdata  : ARRAY [1..16] OF byte;
            name      : ARRAY [0..79] OF char;
            buffer    : textbuf;
        END;
```

Dit type is standaard gedefinieerd in de DOS UNIT van TurboPascal 4.0 en nieuwer. Het record is samengesteld uit DOS-informatie (handle, mode en name) en uit data die TurboPascal nodig heeft om de bijbehorende tekstfile te bewerken. Daartoe is een buffer gedefinieerd met een aantal pointers die het begin, einde en de huidige buffer positie markeren (bufpos, bufend en bufptr). Als je nu iets naar een textfile schrijft, wordt deze data door TurboPascal in de buffer gezet en de devicedriver wordt aangeroepen via een viertal speciale functies. Deze functies kunnen een file openen, sluiten, lezen en/of schrijven en, last but not least, flushen.

In het bijgaande voorbeeld wordt met behulp van deze structuur een ANSI-filter gedefinieerd. Natuurlijk kan ik helemaal gaan zitten uitleggen hoe 't één en ander exact in elkaar steekt, maar wellicht vind u het veel aardiger om zelf eens de handen uit de mouwen te steken...

Joost Voorhaar

```

{ ANSISEQ - Use ANSI sequences on console, Version 1.00
{ Copyright (c) J.Voorhaar, 1991 - All rights reserved
{
{ The following codes are implemented (spaces must be ignored):
{
{ Cursor motion:
{   ESC [ pn A          Cursor up pn times
{   ESC [ pn B          Cursor down pn times
{   ESC [ pn C          Cursor right pn times
{   ESC [ pn D          Cursor left pn times
{   ESC [ pl ; pc H      Set cursor position, pl - line, pc = column
{   ESC [ pl ; pc f      Set cursor position, pl - line, pc = column
{   ESC [ s              Save current cursor position
{   ESC [ u              Restore previously save cursor position
{
{ Character attributes:
{   ESC [ pn m, where pn is a sequence of numbers, seperated by semicolons.
{
{ Attributes:
{       0      Turn off all attributes, normal video
{       1      Highlight
{       2      Normal intensity
{       4      Underline
{       5      Blink
{       7      Inverse video
{       8      Invisible
{
{ Colors (add 10 for background colors:
{       30     Black
{       31     Red
{       32     Green
{       33     Yellow
{       34     Blue
{       35     Magenta
{       36     Cyan
{       37     White
{
{ Erasing:
{   ESC [ 0 K          Erase to end of line (inclusive)
{   ESC [ 1 K          Erase to beginning of line (inclusive)
{   ESC [ 2 K          Erase entire line (cursor doesn't move!)
{   ESC [ 0 J          Erase to end of screen (inclusive)
{   ESC [ 1 J          Erase to beginning of screen (inclusive)
{   ESC [ 2 J          Erase entire screen and home cursor
{
UNIT ansisec;

INTERFACE

VAR ansi : TEXT;

IMPLEMENTATION

USES crt;

```

Fig. 1: sourcetekst van ANSISEQ.PAS

```

CONST fmClosed = $D7B0;
      fmInput  = $D7B1;
      fmOutput = $D7B2;
      fmInOut  = $D7B3;

TYPE TextBuf = ARRAY[0..127] OF char;
      TextRec = RECORD
        Handle: word;
        Mode: word;
        BufSize: word;
        Private: word;
        BufPos: word;
        BufEnd: word;
        BufPtr: ^TextBuf;
        OpenFunc: pointer;
        InOutFunc: pointer;
        FlushFunc: pointer;
        CloseFunc: pointer;
        UserData: ARRAY[1..16] OF byte;
        Name: ARRAY[0..79] OF char;
        Buffer: TextBuf;
      END; { REC TextRec }

FUNCTION Asc2Int(VAR strg : string) : integer;
  VAR result, index : integer;
  BEGIN
    result := 0;
    index := 1;
    WHILE (index < M => length(strg)) AND (strg[index] <> ';') DO
      BEGIN
        result := result * 10 + ord(strg[index]) - 48;
        inc(index)
      END; { WHILE }
    delete(strg, 1, index);
    Asc2Int := result
  END; { FUNC Asc2Int }

{$F+}

FUNCTION DevInOut(VAR f : TextRec) : integer;

  CONST lastchar : char = ';';
        active   : boolean = FALSE;
        sequence : string[32] = '';
        storedx  : integer = 0;
        storedy  : integer = 0;

  VAR index, loop : integer;
      parm1, parm2 : integer;
  BEGIN
    WITH f DO
      BEGIN
        FOR index := 0 TO bufpos - 1 DO
          BEGIN
            IF active THEN
              CASE BufPtr^[index] OF
                '0'..'9', ';': sequence := sequence + bufptr^[index];

```

```

'A' : BEGIN
    parm1 := asc2int(sequence);
    gotoxy(wherex,wherey - parm1);
    active := FALSE
END;
'B' : BEGIN
    parm1 := asc2int(sequence);
    gotoxy(wherex,wherey + parm1);
    active := FALSE
END;
'C' : BEGIN
    parm1 := asc2int(sequence);
    gotoxy(wherex + parm1,wherey);
    active := FALSE
END;
'D' : BEGIN
    parm1 := asc2int(sequence);
    gotoxy(wherex - parm1,wherey);
    active := FALSE
END;
'H','f' : BEGIN
    parm1 := Asc2Int(sequence);
    parm2 := Asc2Int(sequence);
    gotoxy(parm2,parm1);
    active := FALSE
END;
'm' : BEGIN
    REPEAT
        CASE Asc2Int(sequence) OF
            0 : BEGIN textattr := 7; lowvideo END;
            1 : highvideo;
            2 : lowvideo;
            4 : IF lastmode = Mono THEN
                textattr := textattr AND $F0 OR blue;
            5 : textattr := textattr OR $80;
            7 : textattr := (textattr AND $80) OR
                (textattr AND 7 SHL 4) OR
                (textattr AND $70 SHR 4);
            8 : textattr := textattr AND $80;
            30 : textattr := textattr AND $F0 OR black;
            31 : textattr := textattr AND $F0 OR red;
            32 : textattr := textattr AND $F0 OR green;
            33 : textattr := textattr AND $F0 OR yellow;
            34 : textattr := textattr AND $F0 OR blue;
            35 : textattr := textattr AND $F0 OR magenta;
            36 : textattr := textattr AND $F0 OR cyan;
            37 : textattr := textattr AND $F0 OR lightgray;
            40 : textattr := textattr AND $8F OR black SHL 4;
            41 : textattr := textattr AND $8F OR red SHL 4;
            42 : textattr := textattr AND $8F OR green SHL 4;
            43 : textattr := textattr AND $8F OR yellow SHL 4;
            44 : textattr := textattr AND $8F OR blue SHL 4;
            45 : textattr := textattr AND $8F OR magenta SHL 4;
            46 : textattr := textattr AND $8F OR cyan SHL 4;
            47 : textattr := textattr AND $8F OR
                lightgray SHL 4
        END { CASE }
    UNTIL FALSE
END;

```



```

        UNTIL sequence = ";
        active := FALSE
    END;
    's' : BEGIN
        storedx := wherex;
        storedy := wherey;
        active := FALSE
    END;
    'u' : BEGIN
        gotoxy(storedx,storedy);
        active := FALSE
    END;
    'K' : BEGIN
        CASE Asc2Int(sequence) OF
            0 : ClrEol;
            1 : BEGIN
                parm2 := wherex;
                gotoxy(1,wherey);
                FOR loop := 1 TO parm2 DO
                    write(' ');
                    gotoxy(parm2,wherey);
                END;
            2 : BEGIN
                parm1 := wherex;
                gotoxy(1,wherey);
                ClrEol;
                gotoxy(parm1,wherey)
            END
        END; { CASE }
        active := FALSE;
    END;
    'J' : BEGIN
        CASE Asc2Int(sequence) OF
            0 : BEGIN
                ClrEol;
                parm1 := wherex;
                parm2 := wherey;
                FOR loop := parm2 TO Hi(WindMax) DO
                    BEGIN
                        gotoxy(1,loop + 1);
                        clreol
                    END; { FOR }
                gotoxy(parm1,parm2)
            END;
            1 : BEGIN
                parm1 := wherex;
                parm2 := wherey;
                FOR loop := 1 TO parm2 - 1 DO
                    BEGIN
                        gotoxy(1,loop);
                        clreol
                    END;
                gotoxy(1,parm2);
                FOR loop := 1 TO parm1 DO
                    write(' ');
                    gotoxy(parm1,parm2)
                END;
            END;
        END;
    END;

```

```

                2 : ClrScr;
                END; { CASE }
                active := FALSE
            END;
            ELSE BEGIN write(#27,['',sequence); active := FALSE END;
        END { CASE }
    ELSE
        IF (BufPtr^[index] = '[') AND (lastchar = #27) THEN
            BEGIN
                active := TRUE;
                sequence := ''
            END { IF }
        ELSE
            IF (BufPtr^[index] #27) THEN write(BufPtr^[index]);
            lastchar := BufPtr^[index];
        END; { FOR }
        bufpos := 0
    END; { WITH }
    DevInOut := 0
END; { FUNC DevInOut }

FUNCTION DevNotOpen(VAR f : TextRec) : integer;
BEGIN
    DevNotOpen := 103
END; { DevNotOpen }

FUNCTION DevClose(VAR f : TextRec) : integer;
BEGIN
    WITH f DO
        BEGIN
            mode := fmclosed;
            inoutfunc := @DevNotOpen;
            flushfunc := @DevNotOpen;
            closefunc := @DevNotOpen
        END; { WITH }
        DevClose := 0
    END; { FUNC DevClose }

FUNCTION DevFlush(VAR f : TextRec) : integer;
BEGIN
    DevFlush := 0
END; { FUNC DevFlush }

FUNCTION DevOpen(VAR f : TextRec) : integer;
VAR result : integer;
BEGIN
    WITH f DO
        IF mode = fminput THEN
            BEGIN
                result := 12; { Invalid file access code }
                mode := fmclosed
            END { IF }
        ELSE
            BEGIN
                mode := fmoutput;
                inoutfunc := @DevInOut;
                flushfunc := @DevInOut;
            END
        END
    END
END

```

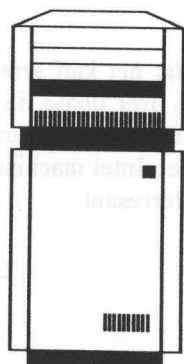
```

closefunc := @DevClose;
result := 0
END; { ELSE }
DevOpen := result;
END; { FUNC DevOpen }

BEGIN
WITH textrec(ansi) DO
BEGIN
mode := fmclosed;
bufsize := sizeof(buffer);
bufptr := @buffer;
openfunc := @DevOpen;
inoutfunc := @DevNotOpen;
flushfunc := @DevNotOpen;
closefunc := @DevNotOpen;
name[0] := #0
END; { WITH }
rewrite(ansi);
END. { UNIT ansiseq }

```

BBS "The Uitimate" For all systems



Telefoon 053-303902 & 053-328506

- 053-303902 - (2 lijnen!)

V22, V22bis, V23, V32bis, 9600/HST, V42bis, MNP5

- 053-328506 -

V21, V22, V22bis & V23

MS-DOS C-source naar OS-9/68k

Het fileformaat van DBase van Asthon Tate schijnt de defacto standaard van data base files in de MS-DOS wereld te zijn. Hij wordt ook in ieder geval gebruikt in "Clipper" van Nantucket en in "Fox-base" van een software huis waar me de naam van ontschoten is. Er is zelfs een "Fox-base" versie voor de Apple Machintosh, die ook dezelfde DBase bestandsstructuur gebruikt.

Over de DBase files zelf wil ik het niet hebben, daar voor verwijs ik naar het boek "Clipper" van Rick Spence, Uitgeverij Sybex, ISBN 90-5160-317-7. In dat boek staan achter in C-sources die zonder meer op een IBM-compatible te compileren en te draaien zijn. Een van deze sources heb ik geporteerd naar een OS-9/68k systeem, daarbij kwamen enkele (leuke) problemen aan het licht. Over de oplossing van die problemen gaat dit artikel. Het resultaat is in de header file msdos2osk.h.

Het overhalen van het programma en oefen bestandje gebeurde gewoon met een seriële kabel en met communicatie programma's. De overbodige linefeeds en de niet standaard ASCII tekens, tekens boven de 127, zijn er in de umacs-editor uit gehaald. Verder is het streven geweest om de file zo veel mogelijk in zijn originele staat te laten.

De onbekende library files die door `#include <FOEI>` aangevraagd werden aangeroepen zijn al op voorhand buitenspel gezet door de compiler directives in de `#ifdef & #endif` in samenwerking met de niet gedefinieerde variabele `not_now`. Dat stukje source ziet er nu dus zo uit:

```
#include <MIO> .h
#ifdef not_now
#include <BI> .h
#include <FOL> .h
#include <MIO> .h
#endif
```

Bij het eerste keer compileren kwamen de onbekende file permission flags naar boven. De juist vlaggen voor OS-9/OSK komen uit de file modes.h, vandaar dat die in de header file msdos2osk.h ge-included wordt. `O_RDONLY` moet men als `READ ONLY` lezen, en `O_BINARY` schijnt in te houden dat teken 13 een carriage return blijft en er geen carriage return line feed van gemaakt wordt.

Het compileren lukte, maar de uitvoer was erg vreemd. Er kwamen grote negatieve getallen uit, terwijl ik kleine positieve getallen verwacht had. Er bleken twee dingen aan de hand te zijn. De unsigned van de MS-DOS C-compiler is 16-bits en 32-bits bij de OS9/68k C-compiler en de bytes staan in de verkeerde volgorde! Om alle unsigned te veranderen in short wilde ik niet doen, en `#define unsigned short` ook niet. Ik heb alle unsigned veranderd in unsigned en de header file `typedef short MSunsigned;`. In de source kan men dus zien wat eerst een unsigned was en nu een short is. De lengte van de variabele klopt nu wel maar, de bytes moeten nog steeds van plaats verwisseld worden. Nog even vooraf in C worden alleen de waarden van de parameters doorgegeven. Omdat `mswap_unsigned & mswap_long` analoog aan elkaar zijn zal ik alleen `mswap_unsigned` toelichten. De naam zou eigenlijk `ms_swap_unsigned` moeten zijn, "Swap van Micro Soft unsigned". De functie geeft een `MSunsigned` terug en moet aangeroepen met een pointer naar een `MSunsigned`. Binnen de functie zijn nog een union en een char nodig. De union laat een `MSunsigned` en een structure elkaar overlappen. De struct is opgebouwd uit twee characters. Verder is er nog variabele van het type char nodig, omdat die even groot is als een byte. De ingrediënten zijn er, er kan begonnen worden. De union wordt gevuld als `MSunsigned` met de waarde die staat op de plaats aangegeven door de pointer `*to_swap`. Het echte verwisselen van de bytes (chars gebeurd in de volgende drie regels. Het resultaat wordt op het moment van `return(to_bes_swapped.msu)` teruggeven als een `MSunsigned`. Het aanroep van de functie gebeurt nadat de variabelen hun "MS-DOS" waarde hebben gekregen door de dbase file in te lezen, natuurlijk voordat met de invoer begonnen is. Een voorbeeld van zo een aanroep: `dbf_head.rec_grootte = mswap_unsigned(&dbf_head.rec_grootte);`.

Ter afsluiting

Dit is zo'n geval van dat het kalf groter is dan de koe, je wilt meer weten over dbase files en je komt meer te weten over permission flags, unsigned variabelen en hoe bytes in een Intel machine staan. Och, dat maakt een hobby interessant.

Geert Stappers

```

/* MS-DOS to OS9/68k port utilities */

/* MS-DOS file permissions */
#include <modes.h>

#define O_RDONLY (S_IREAD)
#define O_BINARY 0

/* MS-DOS types, which differ from Microware OS-9/68k C-compiler types*/
typedef short MSUnsigned;

/*
    SWAP routines
*/
MSUnsigned mswap_unsigned(to_swap)
MSUnsigned *to_swap;
{
    struct two_bytes {
        char hi;
        char low;
    };
    union {
        MSUnsigned msu;
        struct two_bytes tb;
    } to_be_swapped;
    char temp;

    to_be_swapped.msu = *to_swap;

    temp          = to_be_swapped.tb.hi;
    to_be_swapped.tb.hi = to_be_swapped.tb.low;
    to_be_swapped.tb.low = temp;

    return(to_be_swapped.msu);
}

MSUnsigned mswap_long(to_swap)
long *to_swap;
{
    struct four_bytes {
        char hihi;
        char hilow;
        char lowhi;
        char lowlow;
    };
    union {
        long lng;
        struct four_bytes fb;
    } to_be_swapped;
    char temp;

    to_be_swapped.lng = *to_swap;

    temp          = to_be_swapped.fb.hihi;

```

Fig. 1: headerfile MSDS2OS9.H

```

to_be_swapped.fb.hihi      = to_be_swapped.fb.lowlow;
to_be_swapped.fb.lowlow    = temp;

temp                        = to_be_swapped.fb.hilow;
to_be_swapped.fb.hilow     = to_be_swapped.fb.lowhi;
to_be_swapped.fb.lowhi     = temp;

return(to_be_swapped.lng);
}

/* EOF */

```

Liftend langs Internet

Iedereen kent de titels wel denk ik: "Hitchhiker's Guide To The Galaxy", "The Restaurant At The End Of The Universe", "Life, The Universe And Everything" en "So long, and Thanks For All The Fish". De beroemde trilogie van Douglas Adams. Deze werkelijk aanbevelendwaardige werkjes zijn niet alleen in druk bij de boekhandel verkrijgbaar, maar ook via een aantal bulletin boards in elektronische vorm. Zo staat de trilogie natuurlijk ook op "The Ultimate"...

Een erg aardige aanvulling op deze trilogie is "The Hitchhiker's Guide To The Net", van Michael Niermann. Geheel in de sfeer van "Hitchhikers Guide To The Galaxy" is dit een werkje dat handelt over een persoon die van "Het Net" (lees: Internet) ontvoerd wordt. De stijl lijkt meer op een toneelspel, wat het verschil tussen Douglas Adams en Michael Niermann in technisch gezien als auteur duidelijk verschillend maakt. In tegenstelling tot "... the Galaxy" is dit werkje niet in de boekhandel verkrijgbaar. Hoewel er duidelijk ook een begin gemaakt is met "The Restaurant At The End Of The Net", is er hier eigenlijk sprake van slechts één verhaal.

Op "het Net" zijn er, zoals te doen gebruikelijk op elektronische netwerken, altijd een aantal groepen gebruikers te onderscheiden. Zo zijn er de "Flamers", mensen die altijd en overal kritiek op hebben.

Ze stoppen pas met herrie schoppen als ze op alle fronten gelijk gekregen hebben. Daarnaast heb je dan bijvoorbeeld de echte eenlingen, die het netwerk alleen maar schijnen te gebruiken om hun passie voor computers te botvieren en de "Extra Commercial", de particulieren en/of bedrijven die het net gebruiken om er zelf (flink wat) rijker aan te worden. Al deze groepen gebruikers worden in "The Hitchhiker's Guide..." op de hak genomen. Iedereen die na lang ploeteren eindelijk op "het Net" terecht gekomen is zal zichzelf ergens in het verhaal onmiddellijk herkennen, soms als "slachtoffer", maar misschien ook wel als "Flamer" of juist als "echte eenling". Fantastische verhalen zijn het, waar om gelachen mag, nee, bijna moet worden!

Eén tipje echter... het eerste deel telt al ruim 138 pagina's, dus wellicht is het niet geheel onverstandig niet de teksten rücksichtloos uit te printen. Er zijn speciale programmaatjes her en der verspreid waarmee u bijvoorbeeld 4 pagina's tekst op één printer-vel kunt printen. En als u nog even wacht komt er wellicht zelfs eentje waarmee je heel eenvoudig zelf een boekje in elkaar kunt sleutelen op mini-pocket formaat. Kortom: een aanrader voor lange avonden zonder televisie en zonder computer...

Joost Voorhaar

Hardware beschrijving KGN30 (1)

Inleiding

In het afgelopen jaar is in het kader van het KGN Minix project een computer systeem ontwikkeld op basis van de Motorola 68030 processor. Dit artikel is de eerste aflevering van een serie waarin de hardware van de KGN30 besproken wordt. In deze eerste aflevering wordt aan de hand van een blokschema het totale systeem behandeld. In de volgende afleveringen wordt elk onderdeel in nader detail besproken.

Blokschema

Figuur 1 geeft schematisch weer waar de KGN30 uit is opgebouwd. De volgende onderdelen zijn te onderkennen:

- **CPU** De Motorola 68030 micro processor vormt het hart van de machine. Enkele kenmerken zijn:
 - 32 bits adresbereik (4 Giga byte);
 - 32 bits brede data bus;
 - ondersteuning van coprocessors;
 - kloksnelheden van 16 tot 33 MHz;
 - instructie en data cache;
 - ingebouwde memory management unit (MMU);
 - ondersteuning van 8 en 16 bits brede interfaces.
- **FPC** De Motorola 68882 floating point coprocessor is transparant voor het systeem. Indien deze niet aanwezig is, zal de 68030 floating point instructies via software emuleren.
- **DRAM** Het dynamisch RAM is het werkgeheugen van de KGN30 en heeft een grootte van 1 M tot 32 Mbyte. Het DRAM is 32 bits breed en is opgebouwd uit de gangbare 256 K, 1 M of 4 Mbyte SIP's.
- **EPROM** De EPROM bevat de 68030 monitor en boot routines, is 8 bits breed en heeft een grootte van 64 K tot 512 Kbyte.
- **SRAM** Het statisch RAM dient als werkgeheugen van de monitor en voor het opslaan van systeem (instel) gegevens. Het SRAM is 8 bits breed en heeft een grootte van 32 K tot 128 Kbyte. Om gegevens te bewaren wordt voedingsbackup verzorgd door de RTC.
- **RTC** De real time clock verzorgt de actuele tijd en de voedingsbackup van het SRAM. De RTC heeft de vorm van een IC voet (het SRAM wordt in de RTC gestoken).
- **FIFO** Het first in first out RAM heeft als doel om data transfer tussen het systeem en de SCSI of FDC interface mogelijk te maken. Het FIFO is 2 Kbyte diep.

- **SCSI** De SCSI interface verzorgt de communicatie met de SCSI hard disk. Daarnaast is het ook mogelijk om via SCSI met andere systemen te communiceren.
- **FDC** De floppy disk controller interface verzorgt de communicatie met floppy disk drives (maximaal 3 stuks). Alle gangbare formaten worden ondersteund (single, double en high density, 5.25 en 3.5 inch; 1.2 Mbyte IBM AT formaat echter niet).
- **Centronics** Deze interface verzorgt de Centronics interface (gebruikt naar voornamelijk printers). Doordat de interface bidirectioneel is, kan er ook ontvangen worden.
- **RS232** Er zijn 2 seriële RS232C interfaces beschikbaar. Beide kunnen geschikt gemaakt worden voor MIDI.
- **Bus Interface** Het systeem is voorzien van een businterface waardoor uitbreiding mogelijk is. Het busprotocol is gebaseerd op het 68030 protocol.

Tot zover de inleiding aan de hand van het blokschema. De volgende keer gaan we nader in op de diverse onderdelen te beginnen met de CPU en FPC.

Ad Brouwer

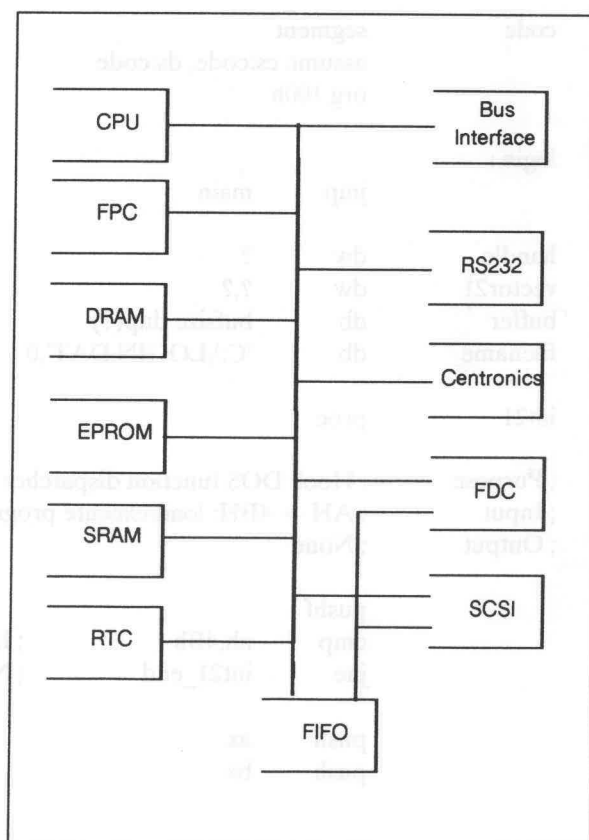


Fig. 1: blokschema van de KGN30

Terugkeer na een koude kermis

Wie assembler programma's ontwikkelt of wel eens stoeit met pointers in C of Pascal kent wellicht het probleem van "de hardnekkige luis". Ergens zit er een fout in het zojuist gecreëerde programma die de hele PC onherroepelijk doet vastlopen. Bij mij resulteert dat dan in -tig keer hetzelfde: programma loopt vast, reset, booter-de-boot, en wat tikwerk: "d:" en "cd \talen\tasm". Aha, kunnen we weer verder... aangezien tikwerk allesbehalve de leukste kant van het programmeerwerk is en ik altijd helemaal gaar wordt van xx maal hetzelfde intikken, heb ik op een mooie zaterdagmiddag ooit eens een utility geschreven. "LogDrive" noemde ik het ding...

De werking van deze utility is heel eenvoudig: als LogDrive opgestart wordt nestelt het zich in 't geheugen van de PC. Daar neemt het dan de kapitale hoeveelheid geheugen in van wel 464 bytes plus wat environment ruimte in beslag. Tegelijk gaat 'ie nog even op zoek naar een bestandje genaamd "C:\Lo-

gin.Dat". Daarin namelijk staat het path waar ik de laatste keer gebleven was. Iedere keer dat een programma middels DOS-functie 4Bh ("Load and execute a program or overlay") gestart wordt, schrijft het residente deel van LogDrive even de huidige drive en directory naar schijf. Op die manier kom ik na een herstart altijd uit in de directory waar ik de laatste keer gebleven was. Eenvoudig, maar handig. Niet dan?

Voor de volledigheid: 't zaakje assembleren met "tasm logdrive,," en linken met "tlink logdrive /t". Of met masm: "masm logdrive,," "link logdrive" en "exe2bin logdrive.exe logdrive.com". Het resultaat is een file-tje genaamd "logdrive.com" van 283 bytes...

Joost Voorhaar

; LogDrive, version 1.10 - Copyright (c) J.Voorhaar, 1991			
msdos	equ	21h	
bufsize	equ	68	
code	segment		
	assume cs:code, ds:code		
	org 100h		
login :			
	jmp	main	
handle	dw	?	
vector21	dw	?,?	
buffer	db	bufsize dup(?)	
filename	db	'C:\LOGIN.DAT',0	
int21	proc		
; Purpose : Hook DOS function dispatcher			
; Input : AH = 4BH: load/execute program or overlay			
; Output : None			
	pushf		
	cmp	ah,4Bh	; Is it function 4Bh?
	jne	int21_end	; No, we're done!
	push	ax	
	push	bx	

Fig. 1: sourcetekst van LOGDRIVE.ASM

```

        push    cx
        push    dx
        push    ds

        mov     ax,cs
        mov     ds,ax      ; DS to my segment

        mov     ax,4408h
        xor     bl,bl
        int     msdos      ; Current disk is changable?

        cmp     ax,1
        jnz     int21_done  ; Yes it is, do not write data

        mov     ah,19h
        int     msdos      ; Get current disk
        add     al,'A'     ; Convert to ASCII
        mov     buffer,al  ; And write it to the buffer

        mov     ax,'\: '
        mov     word ptr buffer + 1,ax      ; Disk separator and root slash

        mov     ah,47h
        lea     si,buffer + 3
        xor     dl,dl
        int     msdos      ; Get current directory

        mov     ah,3Ch
        mov     cx,2
        lea     dx,filename
        int     msdos      ; Create a new handle
        jc      int21_done  ; Can't create a new handle! :-(

        mov     handle,ax   ; Preserve handle number
        mov     ah,40h
        mov     bx,handle
        mov     cx,bufsize
        lea     dx,buffer
        int     msdos      ; Write data

        mov     ah,3Eh
        mov     bx,handle
        int     msdos      ; Close handle

int21_done :
        pop     ds
        pop     dx
        pop     cx
        pop     bx
        pop     ax

int21_end :
        popf
        jmp     dword ptr cs:[vector21]      ; Continue in the original function

int21     endp

```

```

transient      label      near

main :

    mov     ax,3D00h
    lea     dx,filename
    int     msdos          ; Open handle
    jc      main1          ; File not found or other error

    mov     handle,ax      ; Preserve handle number

    mov     ah,3Fh
    mov     bx,handle
    mov     cx,bufsize
    lea     dx,buffer
    int     msdos          ; Read the file

    mov     ah,3Eh
    mov     bx,handle
    int     msdos          ; Close it

    mov     ah,3Bh
    lea     dx,buffer
    int     msdos          ; Change directory

    mov     ah,0Eh
    mov     dl,buffer      ; Get disk name
    sub     dl,'A'         ; Convert to disk number
    int     msdos          ; Select new disk

main1 :

    mov     ax,3521h
    int     msdos          ; Get address of function dispatcher

    mov     vector21,bx
    mov     vector21+2,es   ; Preserve address of original handler

    mov     ax,2521h
    lea     dx,int21
    int     msdos          ; Install the new handler

    lea     dx,transient + 10h
    mov     cl,4
    shr     dx,cl          ; Compute length of resident part

    mov     ax,3100h
    int     msdos          ; Terminate stay resident

code      ends
end        login

```

Minix software?

In de vorige μ P Kenner heb ik mij een beetje ongelukkig uitgedrukt geloof ik. Zo schreef ik in het artikel over "DualBoot", dat het software aanbod voor Minix niet voldoet om een pure Minix machine als werkbaar systeem te laten draaien. Later werd mij verteld dat iemand die dit gelezen had daardoor niet langer meer wachtte op de KGN68k/30, maar bij de computerboer op de hoek een '386 kocht. En dat is natuurlijk eeuwig zonde. Bij het kiezen van een OS voor een nieuwe machine heeft immers de beschikbaarheid van software een doorslaggevende rol gespeeld. De hoogste tijd dus om eens te inventariseren wat er zoal aan "Minix-stuff" beschikbaar is en wat er (eenvoudig) beschikbaar gemaakt zou kunnen worden.

Verschillen

Allereerst moeten we een onderscheid maken tussen te verschillende Minix implementaties. Hoewel UNIX altijd als schoolvoorbeeld van portabiliteit gebruikt wordt, valt deze veelgeroemde systeemafhankelijkheid in de praktijk nog al eens tegen. Zelfs binnen de implementatie van een enkel operating systeem zijn er nog behoorlijk grote verschillen. Zo heeft de PC-implementatie zijn eenvoud als grootste nadeel. ACK (de standaard C-compiler van Minix) is weliswaar een uitstekende compiler voor kleine projecten, maar op de PC kent het ding alleen het zogenaamde small memory model. Dat betekent dat alle code in één segment moet passen, evenals de data en stack. In totaal levert dat een maximale programmagrootte op van $2 \times 64 \text{ kByte} = 128 \text{ kByte}$. Deze restrictie wordt door voor de eenvoud in de kernel doorgevoerd; ook programmatuur die een ander memory model zou gebruiken (assembly!) krijgt maximaal die 128 kByte toegewezen.

De bovenstaande restrictie is op te heffen met behulp van een aantal patches, waarmee een '386-specific Minix/PC-versie gemaakt kan worden. Daarbij is dan wel een andere compiler nodig (de 32-bits versie van Bruce's C-compiler bijvoorbeeld) die zich geen barst meer aantrekt van die vervelende 128 kByte grens.

Daarnaast zijn er een aantal PC-specifieke utilities verkrijgbaar in de vorm van bijvoorbeeld grafische user interfaces die direct gebruik maken van BIOS

en/of direct op de hardware geprogrammeerd zijn. Deze werken overigens meestal alleen in de 386-specific Minix/PC.

Voor de 680x0 systemen ligt de zaak beduidend anders. Doordat de 68000-serie van Motorola een minder brain-dead erfenis met zich mee sleept (concreet: geen segmentatie in de processor) zijn de 68k-implementaties van Minix een stuk flexibeler. Voor Minix/68k zijn de meeste UNIX-programma's zo te compileren. Zeker als ACK vervangen wordt door bijvoorbeeld de GNU C-compiler (GNU is een produktlijn van de "Free Software Foundation", een stichting die zich sterk maakt voor de gedachte dat alle software in principe vrij copieerbaar zou moeten zijn). De grotere UNIX-machines hebben heel vaak een GNU C-compiler draaien, waardoor er in prin-

cipe veel software beschikbaar is voor Minix/68k. De huidige plannen zijn er op gericht de Minix implementatie voor de KGN68k/30 op deze compiler te baseren.

Soorten software

Minix wordt behoorlijk compleet geleverd. Dat betekent in de praktijk dat 90% van de normale (externe) UNIX commando's zo uit de doos beschikbaar is. Utilities die niet in de standaard distributie meegeleverd worden zijn vaak wel door derden geschreven (uiteraard alles

in source) en zijn meestal via het Internet beschikbaar gesteld. Op 't BBS proberen we een collectie van deze utiliteiten te verzamelen en toegankelijk te maken voor "the unfortunates that have no access to Usenet".

Naast deze utiliteiten zijn er een aantal applicaties die heel specifiek zijn voor het complete UNIX gebeuren. Daarbij valt bijvoorbeeld te denken aan uucp ("Unix to Unix CoPy", de programmatuur die de transmissie van elektronische post verzorgt) en een X-windows achtige GUI.

Tenslotte zijn er de echte applicatiesoftware: databases, spreadsheets etc. Deze groep is voor Minix een beetje ondervertegenwoordigd, wat een direct gevolg is van de doelstellingen van Tanenbaum: Minix is in eerste instantie een tool om mee les te kunnen geven. Zelf zegt hij hierover: "Minix is a teachingtool, not a full featured, almost-for-free,

**Later werd mij verteld dat
iemand die dit gelezen had
daardoor bij de
computerboer op de hoek
maar een '386 kocht. En
dat is natuurlijk eeuwig
zonde.**

BSD-lookalike operating system". Tekstverwerkers à la WordPerfect zul je in de UNIX-omgeving tevergeefs zoeken; in "the real world" wordt gebruik gemaakt van het tekstformateersysteem "roff". Daarmee kun je via eenvoudige stuurcommando's (die in de tekst opgenomen worden) je teksten geschikt maken voor alle mogelijke uitvoerapparaten, variërend van beeldschermen tot high-end professionele fotozetters.

Porten

Het omzetten van software van het ene platform naar het andere wordt ook wel "porten" genoemd. Welnu, op het ogenblik krijgen we op het bulletin board van de club zo af en toe sources van de newsgroup "comp.unix.sources" binnen. Tussen deze software zit zo af en toe heel aardig spul. Verder is een hoop software van een beperkt aantal andere operating systems (OS/9 bijvoorbeeld) eenvoudig over te zetten naar Minix. Zo weet ik dat een lid van de werkgroep thuis een OS/9 machine heeft staan waarop een kermit draait die vrijwel hetzelfde is als de kermit die bij Minix meegeleverd wordt. Deze persoon heeft ook een database geport naar die machine. Hij heeft ooit eens gezegd dat hij dit ding ook naar Minix zou gaan porten.

Omdat het jammer zou zijn als iedere fanatieke Minix-hobbyist dezelfde programmatuur naar Minix zou gaan porten stel ik voor een lijst bij te houden van wie waar mee bezig is. Het opgenomen worden in die lijst schept geen enkele verplichting: lukt het niet, of heb je (bijna) geen tijd beschikbaar voor een voorgenomen project(je), dan kun je later altijd weer van die lijst geschrapt worden. Wel wordt door zo'n lijst voorkomen dat we straks tien verschillende implementaties van het mcalc spreadsheet hebben...

Tenslotte

Geïnteresseerden kunnen contact met mij opnemen via het bulletin board, het fidonet (2:512/32.2), het internet (voorhaar@cs.utwente.nl) de telefoon (053-333483) of per post (Jekerstraat 67, 7523 VP Enschede). Een modem of netwerkaansluiting is niet nodig; voor dit project willen we het één en ander ook wel via de telefoon en snail-mail (uhh... tante Pos) afhandelen.

Joost Voorhaar

(advertentie)

De KIM Gebruikersclub Nederland zoekt met spoed een

Bestuurslid voor de Public Relations

Het bestuur is van mening dat zij niet voldoende aandacht kan besteden aan de public relations van de KGN naar leden zowel als naar derden.

Uw taak zal ondermeer bestaan uit het onderhouden van contacten met leden, pers en (potentiële) sponsors. Daarnaast wordt er van de P.R.-functionaris verwacht dat hij (of zij!) zich zal gaan bezighouden met het organiseren van activiteiten die buiten het directe werkgebied van de huidige bestuursleden liggen, waarbij zal hij/zij uiteraard wel hulp zal krijgen van het voltallige bestuur.

Het bestuur verwacht van eventuele gegadigden goede communicatieve vaardigheden, zowel op papier als op verbaal vlak. Ervaring in P.R. is een pré, maar absoluut géén bindende voorwaarde. Daarnaast zal u als toekomstig bestuurslid voldoende tijd moeten willen én kunnen steken in het P.R.-werk. In dit licht wijst het bestuur met nadruk op de komende lustrumactiviteiten. Hierbij valt o.a. te denken aan een symposium, gespecialiseerde studiedagen en misschien zelfs wel een (aantal?) leuke excursie(s).

Van de bestuurstafel

't Was in mijn bescheiden stulpje gisteravond een gezellige drukte. Tot vijf maal toe stond er iemand voor de deur met ogen als chipjes. Je kent dat wel met die stekende pinnetjes. Het was voor mij de eerste bestuursvergadering als voorzitter van de KGN. Met gierende zenuwen opende ik de vergadering nadat ik de koffie had geserveerd. Toch had ik een redelijk veilig gevoel: als er een moeilijk zou gaan doen was ik daar op voorbereid. Mijn huisarts had mij een middelje gegeven: een flesje met smaakloze slaapdruppels. Volgens hem zouden die al na ongeveer 2 minuten moeten gaan werken. Of dat inderdaad zo is kan ik hier nu niet bevestigen omdat ik er geen gebruik van heb behoeven te maken. Integendeel, er was een goede sfeer. We hebben een aantal besluiten en intenties kunnen nemen die voor jullie in de nabije toekomst merkbaar zullen zijn. Hetzelfde geldt voor een mededeling van de sysop m.b.t. het BBS. De huidige sponsor van het 512-net, de PCC, gaat de pijp aan Maarten geven. Gelukkig hoeven we ons daar geen grote zorgen om te maken omdat er al een nieuwe sponsor klaar staat. Het voordeel voor Jacques is dat hij maar één lettertje moet veranderen. De nieuwe sponsor heet PCM. Verder is het de bedoeling om een UNIX file-area op het BBS te zetten. We zouden het op prijs stellen als jullie ons gaan helpen om de software die daarin komt te porten naar MINIX. Alle hulp is daarbij welkom.

De redactie van de μ P Kenner gaat u verzoeken om bijdragen in de hardwarehoek. Het bestuur ondersteunt dit van harte en vraagt u om ook eens een bijdrage te leveren. Het mogen best wel eens kleine projectjes zijn waaraan ook beginners en minder begaafde technici zich kunnen wagen. Met name wil ik hier die leden die hun contributie nog niet betaald hebben vragen om dit eerst zo snel mogelijk te doen. Om de pijn en moeite die dat onze penningmeester bezorgt te verzachten, zouden zij eens iets moois voor de μ P Kenner kunnen inleveren. Bij de ingekomen post zat een uitnodiging voor een HCC-dag in Venlo op 7 maart a.s. De KGN zal daar ook met een stand aanwezig zijn. Wat daarvan de inhoud zal zijn is nog niet bekend. Maar ik heb er alle vertrouwen in dat Geert Stappers er iets moois van zal gaan maken. Dus als je tijd en zin hebt kom dan eens langs.

Bij de aanschaf of registratie van software is het vaak nodig dat je in het bezit bent van een creditcard. Omdat niet iedereen alleen daarvoor zo'n ding wil aanschaffen gaat de vereniging dat doen. Als je dus iets wil aanschaffen of registreren kan dat straks via de KGN. De bedoeling is om te proberen kortingen te bedingen. Jullie horen hier binnenkort meer over. Er lag al geruime tijd een enquête op de plank. De bedoeling is om eens te inventariseren wat er binnen de leden leeft en wat ze van de vereniging verwachten. Binnenkort kunnen een aantal leden dus een telefoontje van mij verwachten, mogelijk ben ik daar

zelfs al mee begonnen op het moment dat u dit leest. Als e.e.a. geëvalueerd is, worden de leden daar van in kennis gesteld. Daarna kunnen we misschien samen gaan discussiëren wat we met de gevonden gegevens moeten gaan doen. Op de clubbijeenkomst in Assendelft werd mij gevraagd of het bestuur ervoor kon zorgen dat er PD en SW software voor handen kan zijn op de bijeenkomsten. Nu blijkt dat onze penningmeester altijd wel wat diskettes bij zich heeft die tegen kostprijs kunnen worden aangeschaft. Hierop vind je thans hoofdzakelijk de algemeen bekende archivers. Het is niet mogelijk om alle voorhanden zijnde software, dat is zo'n 200Mb, telkens weer mee te nemen. Op de komende bijeenkomsten zal er wel een lijst ter inzage liggen. Je kunt dan aangeven waar je belangstelling naar uitgaat. In de toekomst kunnen we dan die software meenemen waar de grootste vraag naar is. En ook software die door leden besteld is. Het bestuur is van mening dat we er voor moeten waken geen copieervereniging te worden. We zouden dan immers niet handelen in de geest van onze statuten. De vergadering werd afgesloten met de rondvraag: 4 bier en twee cola.

Ik refereerde hiervoor al even aan de bijeenkomst in Assendelft. De wegblijvers hebben volgens mij duidelijk iets gemist. We waren te gast bij FORBO-Krommenie dat werd vertegenwoordigd door de heer Co Filmer. Naar mijn mening, waar de andere bestuursleden volmondig mee instemden, een geweldig goede gastheer. Alles was tot in de puntjes verzorgd. Een mooie ruimte voorzien van alle gemakken inclusief een overheadprojector, video installatie en diaprojector. Ook de inwendige mens werd goed verzorgd met een natje en een droogje. Bij de dia-video-presentatie gaf Co een toelichting die elke KIM-liefhebber moet hebben aangesproken. Het gebruik van de KIM is FORBO niet onbekend: er ligt daar in Assendelft een stukje van de roots. Daarna werden we door Gert van Opbroek getrakteerd op een herhaling van zijn artikel in de μ P Kenner van december. Herhalingen zijn meestal niet leuk, maar wat Gert ons voorschotelde was daarop een echte uitzondering. Zijn toelichting liet hij vergezeld gaan van een tekening in kleur, overheadprojecties en twee video-films. Na de lunch werd het samenzijn voortgezet met het uitwisselen van gegevens en het elkaar helpen problemen op te lossen. Een van de leden had 4 IBM terminals en 3 printers waar hij geen weg mee wist. Afgesproken is dat hij ze mee zal nemen naar de bijeenkomst in Geldrop.

Zo het toetsenbord is weer leeg. Rest mij alleen nog jullie allen tot ziens te wensen op 21 maart in Geldrop.

Uw opgerichte voorzitter

Informatie

De μ P 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 μ P 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 μ P 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: 053-328506 of 053-303902.

Software Bibliotheek en Technisch Forum:

Voor het beheer van de Software Bibliotheek en technische ondersteuning streeft het bestuur ernaar zgn. systeemcoördinatoren te benoemen. Van tijd tot tijd zal in de μ P Kenner een overzicht gepubliceerd worden. Dit overzicht staat ook op het Bulletin Board.

Correspondentie adres

Alle correspondentie betreffende verenigingszaken kan gestuurd worden aan:

KIM Gebruikersclub Nederland
Postbus 1336
7500 BH Enschede

Het Bestuur

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

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

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

Gert van Opbroek (secretaris)
Bateweg 60
2481 AN Woubrugge
Telefoon 01729-8636

Joost Voorhaar (redactie μ P Kenner)
Jekerstraat 67
7523 VP Enschede
Telefoon 053-333483

Jan D.J. Derksen (DOS65 coördinator)
Verfailleweg 434
1783 BP Den Helder
Telefoon 02230-28168

Geert Stappers (KGN/68k coördinator)
Engelseweg 7
5825 BT Overloon
Telefoon 04788-1279

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

