De µP Kenner

Vijftiende jaargang nr. 5
December 1991
74

In dit nummer o.a.:

Inhoudsopgave

De μP Kenner
Nummer 74, december 1991
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 μP Kenner nummer 75 verschijnt op 15 februari 1992.

Kopijsluitingsdatum voor nummer 75 is vastgesteld op 1 februari 1992.

Vereniging

Algemeen

Talen/Software

Systemen

Hardware

Datacommunicatie

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

Hmm... dat tikt als vanouds, zou ik bijna zeggen. De eerste keer voor de μP Kenner weliswaar, maar gelukkig hebben we wel meer met het redactionele bijltje gehakt. Allereerst wil ik Gert bedanken voor het werk dat hij de afgelopen jaren als hoofdredacteur heeft verzet. Het blad is onder zijn bezielende leiding gegroeid tot een evenwichtig doch gevarieerd blad met leuke, leesbare en interessante artikelen. Gelukkig maar dat hij heeft toegezegd te blijven schrijven voor de μP Kenner; Een echte redacteur verliest misschien wel zijn streken, maar gelukkig nooit zijn pen!

Verder wil ik u vanaf deze plaats een heel gezond en produktief 1992 toewensen. Een nieuw jaar ligt voor ons, een verenigingsjaar dat bol zal staan van de activiteiten. Naast de bijeenkomsten, het uitkomen van uw lijfblad en het bulletinboard hopen we dit jaar ook de geboorte te mogen beleven van de KGN68k, DOS-65 versie 3.0 en een hoop kleine doch leuke projectjes. Zo heb ik horen fluisteren dat SIMON, de Seriële Interface MONitor, verspreidingsrijp gemaakt wordt en dat het tijd werd voor een flinke geheugensteun voor uw DOS-65 systeem.

Alles mag, van wip-wap systemen voor het hamsterhok tot geautomatiseerde containeroverslag in de Rotterdamse haven.

Al deze projecten zullen, vrees ik, een beetje ten koste gaan van de tijd die men in de μP Kenner kan steken. Het gat dat zo ontstaat in de kopij-berg kan wellicht mede door uw bijdrage gedicht worden. Denk niet te snel dat u niet geschikt bent om te schrijven! Schrijven is iets wat je kunt leren; en ik ben zeker niet te beroerd om een schrijftechnisch "slecht" stuk wat bij te schaven als daar om gevraagd wordt. En onderwerpen zijn er genoeg... W.F.Hermans schreef in zijn boek "Het sadistisch universum" eens het volgende: "Alleen dan is het voor een schrijver de moeite waard geschreven te hebben, als hij de zekerheid heeft hardop uit te spreken wat zijn publiek wel heeft geweten, maar altijd heeft verzwegen; wat het gedroomd heeft, maar bij het ontwaken verdrongen". Kortom: wat u wel weet, en waarvan u denk dat een ander het zeker ook weet, kan toch nog waarde hebben als het aan het papier wordt toevertrouwd. Om een lang verhaal kort te maken: schrijf!

Wat kunt u in de toekomst eigenlijk van de μP Kenner verwachten? Hier is het recept voor de perfecte μP Kenner: Een scheutje software, wat blokjes hardware, een mespuntje humor, een snufje actualiteit en wellicht wat aardig leesvoer. Voeg hier een handjevol illustraties aan toe en breng het geheel in ruime hoeveelheid aan de kook. Zout naar eigen smaak toevoegen graag! Vind u dat er meer listings in de μP Kenner thuishoren? Laat maar horen! Meer hardware misschien? Comin' right up! In ieder geval heb ik het idee dat wat meer "hands on" wat betreft de harde waren voor wat meer aroma zou kunnen zorgen. Verder is het idee geopperd om onder de vlag "toepassingen van (micro) computers" eens wat meer licht te werpen op de zaken waar je een computer nou eigenlijk nog meer voor kunt gebruiken dan slechts als veredelde typmachine of kaartenbak. Alles mag, van wip-wap systemen voor het hamsterhok tot geautomatiseerde containeroverslag in de Rotterdamse haven.

In januari beginnen we traditiegetrouw bij de Forbo fabrieken in Krommenie. Op het moment van schrijven is er nog geen zekerheid over de voordracht die daar plaats gaat vinden, maar naar wat ik gehoord heb gaat het vreselijk mooi worden allemaal. Ik hoop u dan ook in Krommenie te kunnen begroeten. Tot dan!

Joost Voorhaar

Uitnodiging voor de clubbijeenkomst

Datum: 18 januari 1992
Locatie: Nieuwe kantine FORBO-Krommenie
Industrieweg 12
1566 JP Assendelft
Telefoon: 075-291911
Thema: Logistiek en automatisering
Entree: 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. Vervolgens 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	Spreekbeurt van Gert van Opbroek met als titel: Het beheersen van goederenstromen m.b.v. computers. (Met video).
12:00	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.

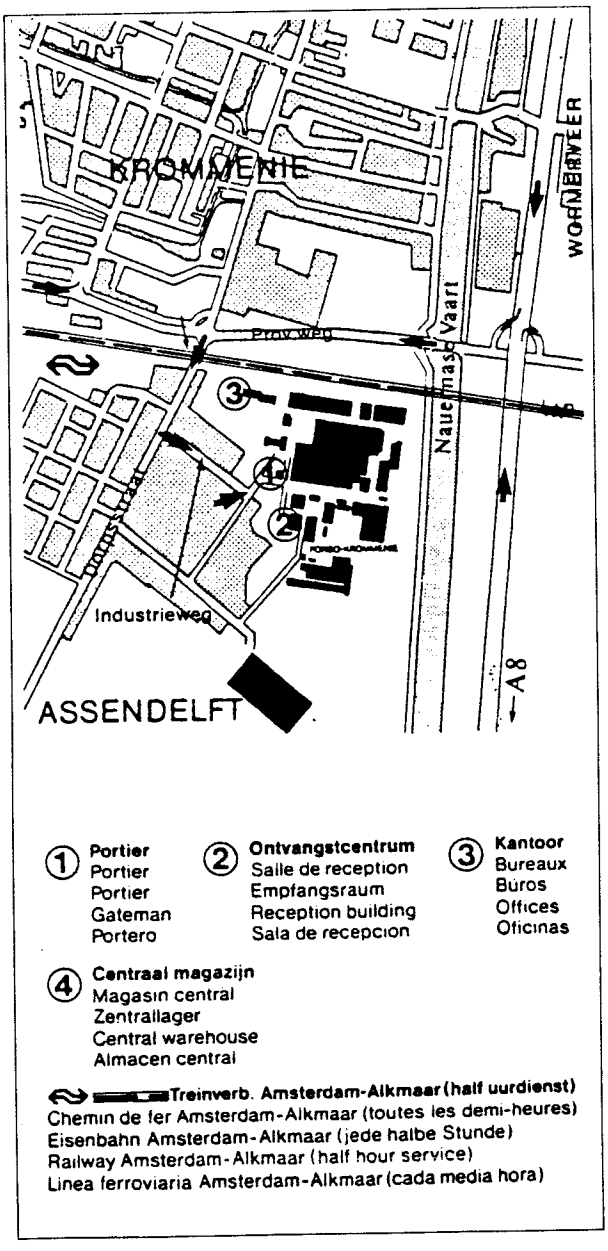
17:00 Sluiting.

Opmerking:

Het bestuur heeft besloten om de behandeling van het financieel verslag over het afgelopen jaar uit te stellen tot de bijeenkomst van maart. Dit heeft als voordeel dat alle noodzakelijke werkzaamheden (opstellen balans, kascontrole etc.) zonder haast en met de nodige accuratesse kunnen plaatsvinden.

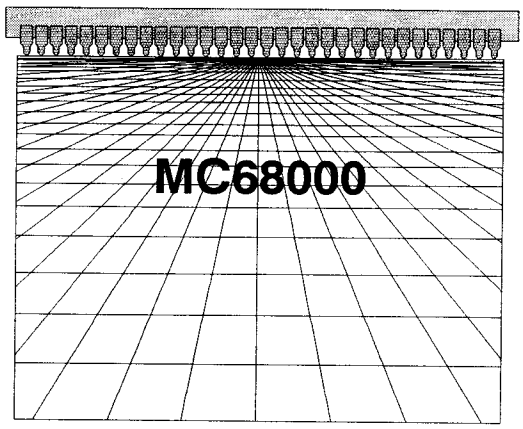
Let op:

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.



Programmeren op de 68000

In 1979 introduceerde Motorola de eerste implementatie van de M68000 16- en 32 bits microprocessor architectuur, de 68000. Deze processor had een 16-bits brede databus en een 24-bits brede adresbus. Later kwam er de 68008, een 8-bits gemultiplexte uitvoering van de 68000 die softwarematig geheel compatibel is met de 68000. Later volgden nog de 68010 die zogenaamde VM-faciliteiten bood, de 68020 (een volledig 32-bits brede processor), de 68030 (met ingebouwde memory manager, ook het hart van de KGN68k/30) en tenslotte de 68040 die een numerieke coprocessor aan boord meekreeg.



In deze serie zal ik proberen de 680x0 nader toe te lichten voor zover het de software betreft. In principe zijn alle 680x0 processoren afgeleid van het basismodel, de M68000. In deze eerste aflevering beginnen we de kennismaking met de 68k-structuur dan ook met deze processor.

Het programmeermodel

De 68000 beschikt over die typische eigenschappen van een op multitasking gerichte processor. Het ding heeft speciale instructies als een "bit test and set" die niet afgebroken kunnen worden door interrupts wat bijzonder aangenaam is bij het switchen tussen verschillende taken. Verder kan de processor zich in user mode of in supervisor mode bevinden. In supervisor mode biedt de processor mogelijkheden die diep in het systeem ingrijpen en die niet in user mode bereikbaar zijn. Normale applicaties lopen dan ook meestal in user mode, terwijl het (multitasking) operating system in supervisor mode loopt.

De registerset van de 68000 is een schoolvoorbeeld van eenvoud (zie fig. 1). Er zijn 8 algemeen toepasbare dataregisters, ieder 32 bits breed. Ieder dataregister kan een 32-, 16- of 8 bits waarde bevatten. Sommige instructies hebben ook betrekking op 1 bit en/of één BCD-value. Ofschoon deze laatste twee waarden wel in de Motorola programmeer handleiding [1] genoemd worden als aparte data-typen vertegenwoordigen ze eigenlijk geen aparte adresseerklasse. In tegenstelling tot Intel noemt Motorola die 32-bits waarden geen "double words" maar "longwords". Deze dataregisters worden simpelweg aangeduid als D0, D1 en zo voorts tot en met D7.

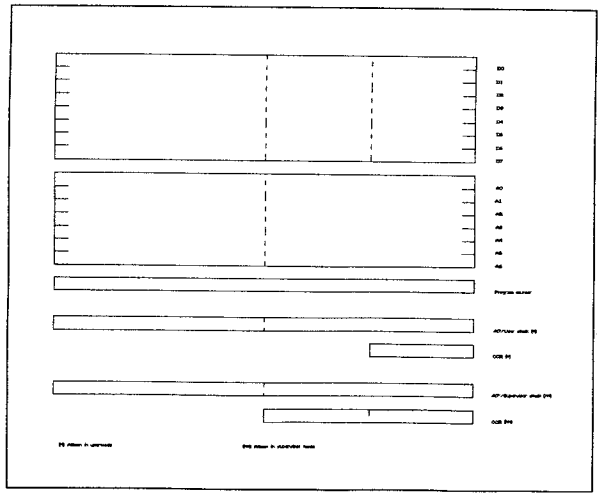


Fig. 1: de 68000 registerset

Naast de genoemde dataregisters heeft de processor een zevental algemene adresregisters, genummerd A0 t/m A6. Er is ook nog een A7, maar die bevat eigenlijk de stackpointer. In veel assemblers kan de stackpointer dan ook aangeduid worden met "A7" zowel als met "SP". Naast A7 wordt er ook een A7' bijgehouden. Deze bevat de waarde van de stackpointer in supervisor mode. Ter onderscheid wordt A7 dan ook wel aangeduid als "USP" (het acroniem voor User Stack Pointer) in user mode of als "SSP" (Supervisor Stack Pointer) in supervisor mode. Al deze adresregisters zijn 32 bits breed. Men kan ze ook als 16-bits registers gebruiken, maar in tegenstelling tot de dataregisters komt een byte-value in de adresregisters nooit voor.

Naast deze registers heeft de processor (natuurlijk!) ook de beschikking over een program counter (eveneens 32 bits) en een Condition Code Register. Deze CCR is in user mode 8 bits breed en in supervisor mode beslaat het ding 16 bits.

Het CCR of statusregister (afbeelding: zie volgende pagina) vat de hele santekraam aan vlaggetjes. Car-

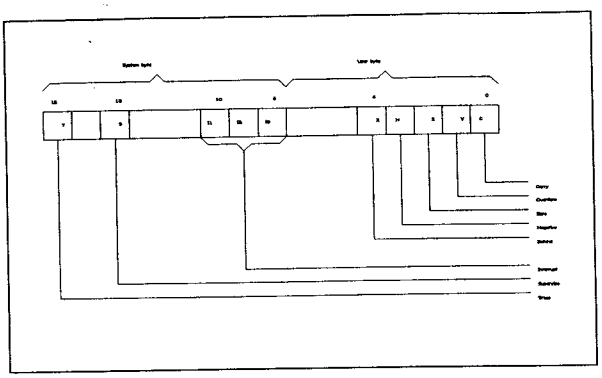


Fig. 2: het statusregister

ry, Overflow, Zero, Negative en de al bijna standaard geworden "Extended Carry". Het hoge byte van het CCR (supervisor mode!) bevat bitjes voor de Interrupt Mask, Supervisor State en het obligate Trace Mode bit.

Adresseringsmogelijkheden

Uit het programmeermodel mag al blijken dat de 68000 geen gesegmenteerd geheugen heeft, maar gebruik maakt van een flat address space. Dit gegeven heeft uiteraard zo zijn invloed op zowel de registerset als de adresseermodes en instructies van de processor. De 68000 biedt keuze uit maar liefst 14 verschillende adresseermethoden. Deze worden onderverdeeld in een zestal basistypen, te weten:

Register direct
- Register indirect
- Absolute
- Immediate
- Program counter relative
- Implied

In het register indirect basistype zijn er mogelijkheden voor post-incrementing, predecrementing, offsets en indexering. In de program counter relative modes kunnen ook indexering en offsets een plaats vinden. Het totale scala aan adresseermethoden ziet er als volgt uit:

Mode	Schrijfwijze
Data register direct	Dn
Addres register direct	An
Addres register indirect	(An)
Address register ind., postincrement	(An) +
Address register ind., predecrement	-(An)
Address register ind., 16 bit offset	d16(An)
Address register ind.,index & offset	d8(An,Xi)
Absoluut short address	xxx.W
Absolute long	xxx.L
Program Counter Relative with offset	d16(PC)
Program Counter Rel., index & offset	d8(PC,Xi)
Immediate data, constants	#xxx
Quick immediate	#d4
Implied	< no ops >

In deze tabel geldt:

Dn	is een willekeurig dataregister
An	is een willekeurig adresregister
Xi	mag elk data- of adres register zijn
PC	is de programcounter
d	staat voor een displacement, oftewel een offset met het gespecificeerde aantal bits.

Traps

De 68000 heeft (bijna natuurlijk) ook een interrupt mechanisme. Een interrupt wordt bij Motorola geen interrupt genoemd (rare jongens, die Motorolen!) maar "trap". Er zijn een aantal hardware traps die door de processor gegenereerd worden. Deze speciale error traps zijn altijd een gevolg van één der volgende condities:

- Woord access op een oneven adres
- Niet toegestane instructies (in usermode...)
- Niet geïmplementeerde instructies
- Niet toegestane address access (bus errors, in usermode)
- Delen door nul
- Overflow detectie (TRAPV)
- Register out of bounds (in geval van een CHK instructie)
- Externe interrupts

Verder heeft men ook gedacht aan een adres-onafhankelijke software service. De hoge heren ontwerpers bedachten dan ook een 16-tal software interrupts waarachter zich allerlei, al dan niet leuke, systeemroutines kunnen bevinden.

Exceptions

De processor kan zich in drie verschillende modes bevinden: normal mode, exception mode en halted mode. Normaliter bevindt de processor zich in normal mode: hij haalt en verwerkt instructies volgens het normale Von Neumann proces. Een speciaal geval van de normal mode is de "stopped mode", waarin de processor zich bevindt na een STOP instructie. In deze speciale mode wordt het geheugen niet meer aangesproken.

Exception mode is een direct gevolg van een interrupt, trap, tracemode of andere buitengewone situatie. Intern kan de exception status een gevolg zijn van een instructie of een abnormale conditie die als gevolg van een instructie ontstaat. Extern kan exception mode worden geactiveerd door een interrupt, een bus error of door een reset.

De halt mode tenslotte (niet te verwarren met "stopped mode"!) is een indicatie van een hardware fout. Als de processor bijvoorbeeld een bus-error genereert tijdens het verwerken van een ander bus-error, neemt de processor aan dat 't zaakje helemaal kaduuk is en stopt 'ie in halted mode. Een processor die eenmaal zover heen is kan slechts door een externe reset weer tot leven gewekt worden.

Op het moment dat een exception ontstaat, wordt de processor automatisch in supervisor mode geschopt. Op deze manier werkt het trace-mechanisme: zet middels een trap instructie het trace bit aan en na iedere instructie komt de processor onmiddelijk in supervisor mode terecht. In deze mode kan dan bijvoorbeeld een debugger aangeroepen worden om registers te tonen en opdrachten te aanvaarden. Meestal zet men dan wel eerst even het supervisor bitje terug...

Volgende keer

In de volgende aflevering ga ik wat specifieker in op de instructieset en de buitengewoon slim opgezette exception verwerking van de 68000. Het is niet mijn bedoeling om de volledige instructieset door te lopen; daar kunt u beter een datasheet voor raadplegen. Tot de volgende keer!

Joost Voorhaar

Literatuur:

1. Motorola Inc. - M68000 16/32-bit microprocessor Programmer's Reference Manual
2. Ir. Meerman - "Assembler programmeren, met als onderwerp de MC68000", moduledictaat "programmatuur 3" van de Hogeschool Enschede.

Toerentalregelaar voor computerventilatoren

PC-voedingen hebben soms de nare eigenschap nogal wat lawaai te produceren. De bron van dat lawaai is natuurlijk de ventilator, die in de meeste PC's heel wat overcapaciteit heeft. Een klein schakelingetje maakt het toerental van de ventilator afhankelijk van de temperatuur in de kast. Het schema is lekker eenvoudig (zie figuur 1) en behoeft nauwelijks enige uitleg. Om kort te blijven: als de temperatuur in de kast stijgt neemt de weerstand van de NTC af. De CE-weerstand van T1 wordt dan groter waardoor de spanning op de basis van T2 toeneemt. Daardoor gaat deze natuurlijk beter geleiden en gaat de motor van de ventilator sneller draaien. Condensator C1 zit nog in het geheel om bij het inschakelen van de PC de motor op de volle 12 volt te laten starten.

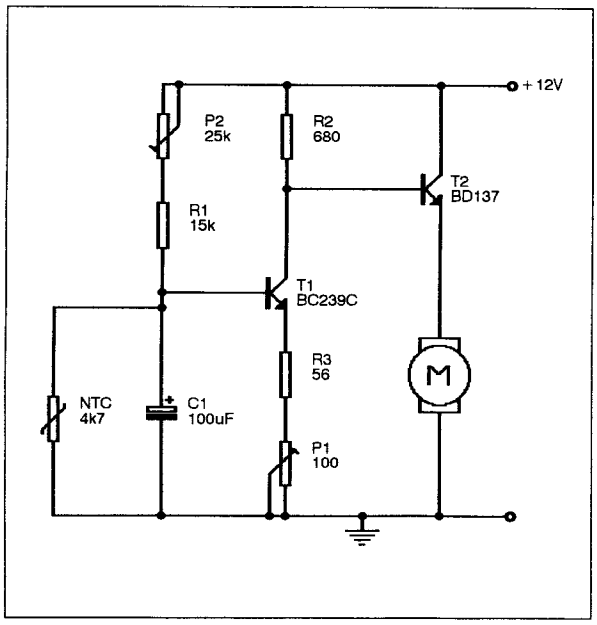


Fig. 1: ventilatordimmer

Het afregelen van het geheel is net zo simpel als de schakeling zelf. We beginnen met beide potmeters in de middenstand. Schakel het geheel in en verwarm de NTC met bijvoorbeeld een föhn tot de maximaal toegestane temperatuur. Draai P2 dan naar de positie waarbij de motor op het maximale toerental loopt. Laat de NTC vervolgens afkoelen tot kamertemperatuur (PC uit!) en stel met P1 de minimaal gewenste draaisnelheid in.

Tenslotte nog een inbouwtip: monteer de schakeling zodanig dat de NTC zich op de warmste plek van de PC bevindt. In de praktijk betekent dat meestal vlak onder de kap, bijvoorbeeld tegen de zijkant van de voeding.

Joost Voorhaar

Literatuur

1. Christian Persson, "Luftbremse", C't, oktober 1989.

Even bij het toetsenbord spieken

Alweer een tijdje geleden wees iemand mij erop, dat lang niet alle BIOSsen in PC's, PC/XT's, PC/AT's en dat soort machines gelijk reageren op bepaalde toetscombinaties. Zo had hij een BIOS gevonden, dat raar deed bij het indrukken van Ctrl-Shift-2 (de 2 bovenaan het toetsenbord). De redenering was: ik wil het karakter NUL invoeren. Dat heeft de ASCII-waarde 0. Dus moet je Ctrl-@ intypen. Alleen @ is de shift van 2 op de meeste IBM-compatibele toetsenborden. Vandaar Ctrl-Shift-2. De oorsprong van de vraag lag in het feit dat de man op NUL testte in zijn programma, en dat de NUL nooit binnenkwam. BIOS-listing van opperhoofd IBM geraadpleegd. Het BIOS moest keurig een ASCII-waarde 0 en een scancode 3 afgeven. Hoe kom je daar nu achter?

Studie van diezelfde IBM BIOS-listing leverde op, dat er nog meer rare toetscombinaties zijn, die tot vreemde waarden in ASCII leiden, zoals de shift-, ctrl- en alt-combinaties van de functietoetsen. In een paar verloren uurtjes een programmaatje in elkaar geklungeld dat keurig netjes laat zien wat het BIOS uitspuugt als je op het toetsenbord zit te taffen. Zoals bij mij gebruikelijk is dat in assembler geschreven. Deze keer niet zo netjes: als het werkte was het goed. Het programmaatje heet eenvoudig KEYBSHOW.COM, en de sourcetekst vindt u hieronder.

KEYBSHOW heeft zijn nut gehad: het BIOS van de vraagsteller deed iets raars bij de beruchte Ctrl-Shift-2: op deze toetscombinatie meldde INT 16h, AH = 1 dat er nog geen toets was binnengekomen, of iets anders uitgedrukt: de toetsindruk werd niet in de toetsenbordbuffer gezet. Daar bleef het niet bij: in het onvolprezen KGN-BIOS bleek ook een heel klein uitglijertje te zitten. Die is bij V2.01 verholpen. En V2.01 is van januari 1988. Voorzover ik weet heeft iedereen allang een nieuwere versie.

De sourcetekst van KEYBSHOW.COM staat ook op het BBS als KEYBSHOW.ZIP. Uw assembler freaker,

Nico de Vries

```
PAGE        65,131
.LIST
;****************************************
;*
;* KEYBSHOW.COM
;*
;* A simple and straightforward program
;* to examine the output of the
;* keyboard-BIOS INT 16h calls.
;*
;* This program was not written as a
;* fine example of suberb assembly
;* language programming, but was merely
;* created to help find an obscure bug
;* in one of the versions of the KGN
;* PC/XT BIOS.
;* It works and does the job required,
;* that's all!
;*
;* This program was written on a sunny
;* monday morning (that sunday I was
;* busy and it was not raining) and is
;* in the Public Domain.
;*
;* Written by Nico de Vries.
;*
;* To create KEYBSHOW.COM:
```

Fig. 1: sourcetekst van KEYBSHOW.COM

```
                ;* MASM KEYBSHOW;
                ;* LINK KEYBSHOW;
                ;* EXE2BIN KEYBSHOW.EXE KEYBSHOW.COM
                ;*
                ;* I hope will enjoy this program as
                ;* much as the author did.
                ;*
                ;****************************************
CODE            SEGMENT  PARA
                ASSUME   CS:CODE,DS:CODE,ES:CODE,SS:CODE
                ;****************************************
                ;*
                ;* Some equates to make life more
                ;* readable.
                ;*
                ;****************************************
LF              EQU      0Ah
CR              EQU      0Dh
H               EQU      100h                    ;moves byte to hi-byte
INS_STATE       EQU      010000000b              ;mask for Ins key
CAPS_STATE      EQU      001000000b              ;mask for Caps key
NUM_STATE       EQU      000100000b              ;mask for Num key
SCROLL_STATE    EQU      000010000b              ;mask for Scroll key
ALT_SHIFT       EQU      000001000b              ;mask for Alt key down
CTL_SHIFT       EQU      000000100b              ;mask for Ctrl key down
LEFT_SHIFT      EQU      000000010b              ;mask for Left Shift key down
RIGHT_SHIFT     EQU      000000001b              ;mask for Right Shift key down
DEL_KEY         EQU      83d                     ;scan code of Del-key

                PAGE
                ORG      0100h                   ;COM-style program
                ;****************************************
                ;*
                ;* Here the misery starts, even the CPU
                ;* goes to work!
                ;*
START:          MOV      AH,0Fh                  ;get
                INT      10h                     ;current video mode
                MOV      AH,0                    ;clear
                INT      10h                     ;screen
                MOV      SI,Offset SIGN_ON_TXT   ;get start text address
                CALL     STROUT                  ;print ASCII$ string at [SI]
                MOV      Byte Ptr OLD_SHIFT_STATE,0 ;clear
                MOV      Byte Ptr SHIFT_STATE,0  ;work area
                CALL     SHOW_SHFT_STATE         ;show shift status
KB_INPUT_LOOP:  MOV      AL,SHIFT_STATE          ;get current shift status
                MOV      OLD_SHIFT_STATE,AL      ;set as old shift status
                MOV      AH,02h                  ;get shift status
                INT      16h                     ;from BIOS
                MOV      SHIFT_STATE,AL          ;store it
                CMP      OLD_SHIFT_STATE,AL      ;if same as old one
                JZ       NO_SHIFT_CHNGE          ;proceed
                CALL     SHOW_SHFT_STATE         ;else update screen shift display
NO_SHIFT_CHNGE: MOV      AH,01h                  ;check if
                INT      16h                     ;a key was pressed
                JZ       KB_INPUT_LOOP           ;if not, loop
```

```
                MOV     AH,0                    ;else get scancode/ASCII
                INT     16h                     ;from BIOS
                CALL    PR_SCAN_ASCI            ;print values on screen
                CMP     AH,DEL_KEY              ;if key was Del
                JNZ     KB_INPUT_LOOP           ;and
                TEST    BYTE PTR SHIFT_STATE,RIGHT_SHIFT+LEFT_SHIFT
                                                ;a shift was down
                JZ      KB_INPUT_LOOP           ;
                MOV     DX,23d*H+0              ;line 23, start
                CALL    POS_CURSOR              ;position cursor
                MOV     AX,4C00h                ;return to
                INT     21h                     ;DOS with error level 0
                ;****************************************
                ;*
                ;* Show shift status on screen.
                ;*
SHOW_SHFT_STATE PROC    NEAR
                MOV     DX,14d*H+3d             ;row 14, column 3
                CALL    POS_CURSOR              ;position cursor
                MOV     AH,OLD_SHIFT_STATE      ;get previous shift state
                AND     AH,LEFT_SHIFT           ;apply mask
                MOV     AL,SHIFT_STATE          ;get current shift state
                AND     AL,LEFT_SHIFT           ;apply mask
                CMP     AL,AH                   ;if no change
                JZ      NO_L_SH_CH              ;position cursor
                MOV     SI,Offset L_SH_TXT      ;else get string address
                TEST    AL,LEFT_SHIFT           ;if down
                JNZ     L_SH_PR                 ;go print
                MOV     SI,Offset BLANK_STR     ;else get new string address
L_SH_PR:        CALL    STROUT                  ;print ASCII$ string at [SI]
NO_L_SH_CH:     MOV     DX,15d*H+3d             ;row 15, column 3
                CALL    POS_CURSOR              ;position cursor
                MOV     AH,OLD_SHIFT_STATE      ;get previous shift state
                AND     AH,RIGHT_SHIFT          ;apply mask
                MOV     AL,SHIFT_STATE          ;get current shift state
                AND     AL,RIGHT_SHIFT          ;apply mask
                CMP     AL,AH                   ;if no change
                JZ      NO_R_SH_CH              ;position cursor
                MOV     SI,Offset R_SH_TXT      ;else get address
                TEST    AL,RIGHT_SHIFT          ;if down
                JNZ     R_SH_PR                 ;go print
                MOV     SI,Offset BLANK_STR     ;else get blanking address
R_SH_PR:        CALL    STROUT                  ;print ASCII$ string at [SI]
NO_R_SH_CH:     MOV     DX,16d*H+3              ;row 16, column 3
                CALL    POS_CURSOR              ;position cursor
                MOV     AH,OLD_SHIFT_STATE      ;get previous shift state
                AND     AH,CTL_SHIFT            ;apply mask
                MOV     AL,SHIFT_STATE          ;get current shift state
                AND     AL,CTL_SHIFT            ;apply mask
                CMP     AL,AH                   ;if no change
                JZ      NO_CTL_CH               ;position cursor
                TEST    AL,CTL_SHIFT            ;if down
                MOV     SI,Offset CTL_SH_TXT    ;get string address
                JNZ     CTL_PR                  ;and go print
                MOV     SI,Offset BLANK_STR     ;else get blanking string
CTL_PR:         CALL    STROUT                  ;print ASCII$ string at [SI]
```

```
NO_CTL_CH:      MOV     DX,17d*H+3d             ;row 17, column 3
                CALL    POS_CURSOR              ;position cursor
                MOV     AH,OLD_SHIFT_STATE      ;get previous shift state
                AND     AH,ALT_SHIFT            ;apply mask
                MOV     AL,SHIFT_STATE          ;get current shift state
                AND     AL,ALT_SHIFT            ;apply mask
                CMP     AL,AH                   ;if no change
                JZ      NO_ALT_CH               ;go position cursor
                TEST    AL,ALT_SHIFT            ;if down
                MOV     SI,Offset ALT_SH_TXT    ;get string address
                JNZ     ALT_PR                  ;go print
                MOV     SI,Offset BLANK_STR     ;else get blanking address
ALT_PR:         CALL    STROUT                  ;print ASCII$ string at [SI]
NO_ALT_CH:      MOV     DX,18d*H+3d             ;line 18, row 3
                CALL    POS_CURSOR              ;position cursor
                MOV     AH,OLD_SHIFT_STATE      ;get previous shift state
                AND     AH,CAPS_STATE           ;apply mask
                MOV     AL,SHIFT_STATE          ;get current shift state
                AND     AL,CAPS_STATE           ;apply mask
                CMP     AL,AH                   ;if no change
                JZ      NO_CAPS_CH              ;position cursor
                TEST    AL,CAPS_STATE           ;if active
                MOV     SI,Offset CAPS_TXT      ;get string address
                JNZ     CAPS_PR                 ;go print
                MOV     SI,Offset BLANK_STR     ;else get blanking string
CAPS_PR:        CALL    STROUT                  ;print ASCII$ string at [SI]
NO_CAPS_CH:     MOV     DX,19d*H+3d             ;line 19, column 3
                CALL    POS_CURSOR              ;position cursor
                MOV     AH,OLD_SHIFT_STATE      ;get previous shift state
                AND     AH,SCROLL_STATE         ;apply mask
                MOV     AL,SHIFT_STATE          ;get current shift state
                AND     AL,SCROLL_STATE         ;apply mask
                CMP     AL,AH                   ;if no change
                JZ      NO_SCROL_CH             ;go position cursor
                TEST    AL,SCROLL_STATE         ;if active
                MOV     SI,Offset SCROL_TXT     ;get string address
                JNZ     SCROL_PR                ;go print
                MOV     SI,Offset BLANK_STR     ;else get blanking string
SCROL_PR:       CALL    STROUT                  ;print ASCII$ string at [SI]
NO_SCROL_CH:    MOV     DX,20d*H+3d             ;row 20, column 3
                CALL    POS_CURSOR              ;position cursor
                MOV     AH,OLD_SHIFT_STATE      ;get previous shift state
                AND     AH,NUM_STATE            ;apply mask
                MOV     AL,SHIFT_STATE          ;get current shift state
                AND     AL,NUM_STATE            ;apply mask
                CMP     AL,AH                   ;if no change
                JZ      NO_NUM_CH               ;position cursor
                TEST    AL,NUM_STATE            ;if active
                MOV     SI,Offset NUM_TXT       ;get string address
                JNZ     NUM_PR                  ;and go print
                MOV     SI,Offset BLANK_STR     ;else get blanking string address
NUM_PR:         CALL    STROUT                  ;print ASCII$ string at [SI]
NO_NUM_CH:      MOV     DX,21d*H+3d             ;row 21, column 3
                CALL    POS_CURSOR              ;position cursor
                MOV     AH,OLD_SHIFT_STATE      ;get previous shift state
                AND     AH,INS_STATE            ;apply mask
```

```
                MOV     AL,SHIFT_STATE      ;get current shift state
                AND     AL,INS_STATE        ;apply mask
                CMP     AL,AH               ;if no change
                JZ      NO_INS_CH           ;exit now
                TEST    AL,INS_STATE        ;else if active
                MOV     SI,Offset INS_TXT   ;get address
                JNZ     INS_PR              ;go print
                MOV     SI,Offset BLANK_STR ;else get blanking string
INS_PR:         CALL    STROUT              ;print ASCII$ string at [SI]
NO_INS_CH:      RET                         ;and exit
SHOW_SHFT_STATE ENDP
                ;****************************************
                ;*
                ;* Show ASCII key value on screen in
                ;* hex, decimal and as character, the
                ;* latter only if character is
                ;* printable.
                ;* Show scancode in both hex and
                ;* decimal.
                ;*
PR_SCAN_ASCI    PROC    NEAR
                PUSH    AX                  ;save character
                MOV     DX,6d*H+17d         ;line 6, column 17
                CALL    POS_CURSOR          ;position cursor
                POP     AX                  ;get character back
                PUSH    AX                  ;save it for later
                CALL    WROB                ;print AL in hex
                MOV     AL,'h'              ;get an h
                CALL    CHROUT              ;print the character in AL
                CALL    PR_SPACE            ;print a space
                POP     AX                  ;get character
                PUSH    AX                  ;save it for later
                CALL    PR_AL_DEC_SP        ;print AL in decimal
                MOV     AL,'d'              ;get a d
                CALL    CHROUT              ;print the character in AL
                CALL    PR_SPACE            ;print a space
                POP     AX                  ;get character
                PUSH    AX                  ;save for later
                CMP     AL,20h              ;if ctl-character
                JB      NOT_PRNTBL          ;or
                CMP     AL,7Eh              ;Del or higher
                JNB     NOT_PRNTBL          ;do not print ASCII
                PUSH    AX                  ;save ASCII
                MOV     AL,27h              ;get a single quote
                CALL    CHROUT              ;print the character in AL
                POP     AX                  ;restore ASCII
                CALL    CHROUT              ;print the character in AL
                MOV     AL,27h              ;get a single quote
                CALL    CHROUT              ;print the character in AL
                JMP     Short PR_SCANC      ;and proceed
NOT_PRNTBL:     CALL    PR_SPACE            ;print
                CALL    PR_SPACE            ;three
                CALL    PR_SPACE            ;spaces
PR_SCANC:       MOV     DX,8d*H+17d         ;line 8, column 17
                CALL    POS_CURSOR          ;position cursor
                POP     AX                  ;get scancode
```

```
                PUSH    AX                      ;save for later
                XCHG    AL,AH                   ;get scancode in AL
                CALL    WROB                    ;print AL in hex
                MOV     AL,'h'                  ;get an h
                CALL    CHROUT                  ;print the character in AL
                CALL    PR_SPACE                ;print a space
                POP     AX                      ;get scancode
                PUSH    AX                      ;save for later
                XCHG    AL,AH                   ;get it in AL
                CALL    PR_AL_DEC_SP            ;print AL in decimal
                MOV     AL,'d'                  ;get a d
                CALL    CHROUT                  ;print the character in AL
                POP     AX                      ;restore scancode
                RET                             ;and exit
PR_SCAN_ASCI    ENDP
                ;***************************************
                ;*
                ;* Position cursor. DL = column, DH = row.
                ;*
POS_CURSOR      PROC    NEAR
                XOR     BH,BH                   ;page zero
                MOV     AH,02h                  ;position
                INT     10h                     ;cursor
                RET                             ;and exit
POS_CURSOR      ENDP
                ;***************************************
                ;*
                ;* Print ASCII$ string pointed to by SI
                ;*
STROUT          PROC    NEAR
                PUSH    AX                      ;save AX
STR_LOOP:       MOV     AL,CS:[SI]              ;get string character
                INC     SI                      ;point to next
                CMP     AL,'$'                  ;if last,
                JE      STR_END                 ;exit now
                CALL    CHROUT                  ;else print it on screen
                JMP     STR_LOOP                ;and loop
STR_END:        POP     AX                      ;restore AX
                RET                             ;and exit
STROUT          ENDP
                ;***************************************
                ;*
                ;* Print a space.
                ;*
PR_SPACE        PROC    NEAR
                PUSH    AX                      ;save AX
                MOV     AL,' '                  ;get a space
                CALL    CHROUT                  ;print the character in AL
                POP     AX                      ;restore AX
                RET                             ;and exit
PR_SPACE        ENDP
                ;***************************************
                ;*
                ;* Print a zero.
                ;*
PR_ZERO         PROC    NEAR
```

```
                PUSH    AX              ;save AX
                MOV     AL,'0'          ;get a zero
                CALL    CHROUT          ;print the character in AL
                POP     AX              ;restore AX
                RET                     ;and exit
PR_ZERO         ENDP
                ;**************************************
                ;*
                ;* Print the character in AL on screen.
                ;*
CHROUT          PROC    NEAR
                PUSH    AX              ;save
                PUSH    BX              ;used registers
                MOV     BX,0007h        ;page 0, white
                MOV     AH,0Eh          ;write TTY
                INT     10h             ;through BIOS
                POP     BX              ;restore
                POP     AX              ;saved characters
                RET                     ;and exit
CHROUT          ENDP
                ;**************************************
                ;*
                ;* Print AL in hex on screen.
                ;*
WROB            PROC    NEAR
                PUSH    AX              ;save AX
                SHR     AL,1            ;get upper
                SHR     AL,1            ;nibble in
                SHR     AL,1            ;lower nibble
                SHR     AL,1            ;
                CALL    WRON            ;print that in hax
                POP     AX              ;restore original AL, and:
                ;**************************************
                ;*
                ;* Print lowest 4 bits of AL in hex on
                ;* screen.
                ;*
WRON            PROC    NEAR
                PUSH    AX              ;save AX
                AND     AL,0Fh          ;get lower nibble only
                CMP     AL,09h          ;if over ten
                JBE     DIGIT           ;
                ADD     AL,07h          ;add letter offset
DIGIT:          ADD     AL,30h          ;convert to ASCII
                CALL    CHROUT          ;print the character in AL
                POP     AX              ;restore AX
                RET                     ;and exit
WRON            ENDP
WROB            ENDP
                ;**************************************
                ;*
                ;* Print AX in decimal, always three
                ;* digits with leading zeroes.
                ;*
PR_AX_ZERO      PROC    NEAR
                CMP     AX,100d         ;if over 100
```

```
                JNB     PR_AX_DEC       ;print in decimal
                CALL    PR_ZERO         ;else print a zero
                CMP     AX,10d          ;if over 10
                JNB     PR_AX_DEC       ;print in decimal
                CALL    PR_ZERO         ;else print a zero
                ;**************************************
                ;*
                ;* Print AX in decimal.
                ;*
PR_AX_DEC       PROC    NEAR
                MOV     CX,10d          ;get divisor
DEC_PR_LOOP:    XOR     DX,DX           ;clear result area
                DIV     CX              ;divide AX by 10
                OR      AX,AX           ;if zero now
                JZ      DEC_EXIT        ;exit printing result
                PUSH    DX              ;else save outcome (digit)
                CALL    DEC_PR_LOOP     ;and repeat
                POP     DX              ;restore digit
DEC_EXIT:       MOV     AX,DX           ;get in AX
                OR      AL,30h          ;convert to ASCII
                CALL    CHROUT          ;print the character in AL
                RET                     ;and exit
PR_AX_DEC       ENDP
PR_AX_ZERO      ENDP
                ;**************************************
                ;*
                ;* Print AL in decimal, always three
                ;* digits with leading spaces.
                ;*
PR_AL_DEC_SP    PROC    NEAR
                PUSH    AX              ;save AX
                XOR     AH,AH           ;get number in AL
                CMP     AL,100d         ;if over 100
                JNB     PR_DEC          ;go print
                CALL    PR_SPACE        ;else print a space
                CMP     AL,10d          ;if over 10
                JNB     PR_DEC          ;print in decimal
                CALL    PR_SPACE        ;else print a space
PR_DEC:         PUSH    DX              ;save registers
                PUSH    CX              ;
                CALL    PR_AX_DEC       ;print AX in decimal
                POP     CX              ;restore
                POP     DX              ;saved registers
                POP     AX              ;and
                RET                     ;exit
PR_AL_DEC_SP    ENDP
                ;**************************************
                ;*
                ;* RAM work area.
                ;*
SHIFT_STATE     DB      ?               ;current shift state
OLD_SHIFT_STATE DB      ?               ;previous shift state
                ;**************************************
                ;*
                ;* Texts.
                ;*
```

```
SIGN_ON_TXT     DB      ' Show Keyboard Status Program  V1.00',CR,LF
                DB      '(C) 1991 KIM Gebruikersclub Nederland',CR,LF,CR,LF
                DB      '    Shift-Del exists to DOS.',CR,LF,CR,LF,CR,LF
                DB      '  ASCII-value :',CR,LF,CR,LF
                DB      '  Scancode   :',CR,LF,CR,LF,CR,LF,CR,LF
                DB      '  Shift & Lock Status:',CR,LF,'$'
L_SH_TXT        DB      'Left Shift Down','$'
R_SH_TXT        DB      'Right Shift Down','$'
CTL_SH_TXT      DB      'Control Down','$'
ALT_SH_TXT      DB      'Alt Down','$'
CAPS_TXT        DB      'Caps Lock Active','$'
NUM_TXT         DB      'Num Lock Active','$'
SCROL_TXT       DB      'Scroll Lock Active','$'
INS_TXT         DB      'Insert Mode Active','$'
BLANK_STR       DB      '                ','$'

CODE            ENDS                                    ;finally...

                END     START
```

Een nieuw virus?

Op mijn werk kwam iemand op mij af. "De beste wensen voor 't nieuwe jaar.", zei hij. "Dank je, voor jou natuurlijk ook", was het antwoord. "Jij weet wel wat van PC's en zo af hè, ik heb iets raars". Het was me inderdaad opgevallen, dat hij wat bleekjes rond z'n neus was, maar och, Kerst, de jaarwisseling, u kent dat wel...

Er was inderdaad iets vreemds aan de hand. De machine is een 386SX op 20 MHz, met een AT-bus harde schijf en draait onder DR-DOS 5.0, met 4DOS 4.00 als commandline interpreter. Tot zover niets vreemds. Vorig jaar draaide het apparaat feilloos. Totdat op 31 december de klok middernacht sloeg en het 1992 werd. Op dat moment was onze gelukkige computerbezitter feest aan het vieren, zodat de verrassing even op zich liet wachten. Op nieuwjaarsdag even een tekstje veranderen. Vreemd... de PC start niet meer op vanaf de harde schijf. SETUP gecontroleerd. Was niets mee aan de hand. Ook de selftest van het BIOS had geen klachten. Dus maar eens van floppy opgestart. Ging wel. Naar drive C:. Ging ook. Na het opvragen van een directory bleek alles aanwezig. Opluchting. Programma opgestart. Lampje van de harde schijf aan. Programma start niet en de prompt komt terug. Zelfde programma gedraaid vanaf floppy disk. Gaat perfect.

Rimpels in het voorhoofd. Harde schijf getest, PCTools gedraaid, CHKDSK geprobeerd, en dit alles meldde geen fouten. 4DOS van de harde schijf gehaald. Maakte niets uit. Alles leek goed. Na een paar uur puzzelen de datum eens teruggezet naar 1991. Vanaf dat moment werkte alles weer perfect. Zodra de datum in 1992 valt, is het mis. Een nieuw virus? Misschien. Machine opgestart zonder AUTOEXEC.BAT en zonder CONFIG.SYS. Alles bleef reageren zoals het reageerde. Er is maar 1 probleempje. De ongelukkige computerbezitter heeft geen originele bootfloppy van DR-DOS 5.00 meer... Dus ook geen virusvrije systeemfiles.

Oproep

Heeft u ook zoiets (in uw omgeving) meegemaakt? En opgelost? Laat het eens weten. Misschien toch een vers virus. Of gewoon een kapotte machine. U hoort er nog van.

Nico de Vries

Nog een extern sorteerprogramma in Pascal

In de vorige uitgave van de μP Kenner heb ik een programma voor extern sorteren gepresenteerd. Dat programma was vrij eenvoudig van opbouw. De informatie werd in een aantal doorgangen verdeeld over twee files en daarna weer samengevoegd tot één file. Het voordeel van een dergelijk eenvoudig programma is dat er geen slimme trucs inzitten en dat je vrij snel begrijpt hoe het programma werkt.

Het programma dat in deze aflevering gepresenteerd wordt, is vrij ingewikkeld. Een volledige beschrijving zou waarschijnlijk de hele μP Kenner kunnen vullen. Toch is dit nog steeds niet het formule 1 programma dat in de vorige aflevering al beloofd werd, nee de turbo ontbreekt nog. Dit programma is een verbeterd programma voor extern sorteren maar maakt nog steeds geen gebruik van extra geheugen om intern en extern sorteren te combineren. Deze turbo zal in een volgende aflevering als uitbreiding gepubliceerd worden zodat dan een compleet formule 1 programma ontstaat.

Hoewel een complete programma-beschrijving niet mogelijk is, die staat overigens in de source opgenomen listing, is het toch wel zinvol enkele regels aan het programma te wijden. In de eerste plaats wordt er gebruik gemaakt van extra files (in het voorbeeld zijn dat er 6). Verder wordt een tape die leeg is, meteen gebruikt als tape waarop gesorteerde informatie wordt teruggeschreven.

Het programma neemt in eerste instantie de ongesorteerde informatie en verdeelt deze over n - 1 = 5 tapes. Hierbij wordt een blok waarvan de regels in volgorde staan een "run" genoemd. Er wordt steeds een run naar één van de 5 tapes gekopieerd en wel zodanig dat het aantal runs op een tape zo goed mogelijk verdeeld zijn volgens de zogenaamde Fibonacci reeks:

level	a(1)	a(2)	a(3)	a(4)	a(5)	Totaal
0	1	0	0	0	0	1
1	1	1	1	1	1	5
2	2	2	2	2	1	9
3	4	4	4	3	2	17
4	8	8	7	6	4	33
5	16	15	14	12	8	65
6	31	30	28	24	16	129

Van elke rij worden de elementen berekend uit het eerste element uit de voorgaande rij en het element uit de voorgaande rij met een index één hoger dan de huidige. Het element met de hoogste index heeft dezelfde waarde als het element met index 1 uit de voorgaande rij. Dus:

$$a(l+1,n) = a(l,1) + (l,n+1)$$
$$a(l+1,5) = a(l,1)$$

Een dergelijke verdeling heeft als voordeel dat er zo optimaal mogelijk gebruik gemaakt wordt van de aanwezige tapes zoals uit het volgende voorbeeld blijkt.

Voorbeeld

We willen een file sorteren waarvan de informatie volgens level 4 verdeeld kan worden. Op de zesde tape gaan we nu steeds elementen van de andere tapes kopiëren en wel zodanig dat we steeds die tape uitzoeken waar het eerste element de kleinste waarde heeft. op een gegeven moment is tape 5 leeg. Op dat moment hebben we van elke tape 4 runs gekopieerd en samengevoegd tot één run waarna de zes tapes de volgende aantallen nog niet gelezen runs bevatten:

Tape	1	2	3	4	5	6
Aantal	4	4	3	2	0	4

hetgeen na wat rangschikken precies de vorige rij uit de tabel blijkt te zijn. Nadat tape 5 voor schrijven en tape 6 voor lezen zijn teruggespoeld, wordt het proces voortgezet waarbij de runs naar tape 5 gekopieerd worden. Op deze manier lopen we de tabel naar boven toe af en houden na 4 slagen een gesorteerde file over, precies op het moment dat alle tapes gelijktijdig leeggelezen zijn.

Nu zal het over het algemeen helaas niet zo zijn dat een file precies voldoende runs bevat. Dit wordt opgevangen door voor elke tape een aantal zogenaamde dummy runs te bepalen zodanig dat het aantal werkelijke runs plus het aantal dummy runs precies het aantal runs volgens de tabel vormt. Zolang een tape een aantal dummy runs groter dan nul heeft, wordt er op dat moment geen informatie van de tape gehaald.

Gert van Opbroek

```
PROGRAM polysort(input,output);

{    Dit progrogramma sorteert een tekstfile d.m.v. een polyphase sort met n tapes.

     De kern van het programma is overgenomen uit:
     Nicklaus Wirth: Algorithns + Data Structures = Programs
     uitgegeven door Prentice-Hall.

     Het programma is enigszins gemodificeerd en aangepast voor Turbo Pascal
     door G. van Opbroek in december 1991
}
CONST n = 6; { aantal tapes = aantal open files }

TYPE    item = string;
        tape = TEXT;
        tapeno = 1..n;

VAR     leng, rand          : INTEGER;          { used to generate file}
        buf                 : item;
        f0                  : tape;             { f0 is the input tape}
        buf_f0              : item;             { file buffer f0       }
        eof_f0              : BOOLEAN;          { end-of-file marker}
        filename            : STRING;           { filename to sort    }
        s_start,s_length    : INTEGER;          { sort key start,length}

PROCEDURE list (VAR f : tape; n : INTEGER);

    { Deze procedure toont een tekstfile. Hij gebruikt worden om het verloop van het sorteerproces
    te tonen.
    }
    VAR     x : item;
            line : INTEGER;

    BEGIN
        writeln('Tape :',n:2); writeln;
        reset(f);
        line := 1;
        WHILE NOT eof(f) DO
        BEGIN
            readln (f,x); writeln(output, line:4, ' : ', x);
            line := succ(line)
        END;
        writeln;
    END; { list }

PROCEDURE polyphasesort;

    VAR i,j,mx,tn       : tapeno;
            k,level     : INTEGER;
            a,d         : ARRAY [tapeno] OF INTEGER;
                                { a[j] = ideal number of runs on tape j }
                                { d[j] = number of dummy runs on tape j }
            dn,z        : INTEGER;
            min,x       : item;
```

Sourcetekst van P_SORT.PAS

```
        last      : ARRAY [tapeno] OF item;
                                    { last[j] = key of tail item on tape j}
        t,ta      : ARRAY [tapeno] OF tapeno;
                                                { mappings of tape numbers}
        f         : ARRAY [1..n] OF tape;       { scratch files  }
        buf_f     : ARRAY [1..n] OF item;       { file buffers f  }
        eof_f     : ARRAY [1..n] OF BOOLEAN;    { end-of-file markers }

FUNCTION compare (x,y : item) : BOOLEAN;

    { Deze functie bepaalt de volgorde na sortering. Aan de hand van
    de globale variabelen s_start en s_length wordt bepaald welke
    delen van de parameters x en y met elkaar vergeleken moeten worden.
    }
    VAR sortkeyx,sortkeyy : STRING;

    BEGIN
        IF s_start  length(x)
        THEN sortkeyx : = ''
        ELSE IF s_start + s_length  length(x) + 1
                THEN sortkeyx : = copy(x,s_start,length(x) - s_start + 1)
                ELSE sortkeyx : = copy(x,s_start,s_length);

        IF s_start  length(y)
        THEN sortkeyy : = ''
        ELSE IF s_start + s_length  length(y) + 1
                THEN sortkeyy : = copy(y,s_start,length(y) - s_start + 1)
                ELSE sortkeyy : = copy(y,s_start,s_length);

            compare  : = sortkeyx  sortkeyy

    END;

PROCEDURE read_buf(n : INTEGER);

    { Omdat je in Turbo Pascal niet in de filebuffer kunt lezen, is er voor
    elke file een intern buffer voor 1 teken. Dit buffer wordt als (eigen)
    filebuffer gebruikt. Behalve een eigen filebuffer, heeft elke
    file ook een eigen EOF-variabele die TRUE wordt als het interne
    filebuffer leeg is.
    }
    BEGIN
        CASE n OF
            0:
                BEGIN
                eof_f0 : = eof(f0);
                IF NOT eof_f0
                THEN readln(f0,buf_f0)
                    ELSE buf_f0 : = ''
                END;
            1..6:
                BEGIN
                    eof_f[n] : = eof(f[n]);
                    IF NOT eof_f[n]
                    THEN readln(f[n],buf_f[n])
                    ELSE buf_f[n] : = ''
```

```
            END;
        END;
    END;

PROCEDURE selecttape;

    VAR i    : tapeno;
             z    : INTEGER;

    BEGIN
        IF d[j]  d[j+1]
        THEN j := j + 1
        ELSE BEGIN
            IF d[j] = 0
            THEN BEGIN
                level := level + 1; z := a[1];
                FOR i := 1 TO n-1
                DO BEGIN
                    d[i] := z + a[i+1] - a[i]; a[i] := z + a[i+1];
                END
            END;
            j := 1
        END;
        d[j] := d[j] - 1;
    END;

PROCEDURE copyrun;

BEGIN { copy one run from f0 to tape j }
    REPEAT
        buf := buf_f0; writeln(f[j],buf);
        read_buf(0);
    UNTIL eof_f0 OR compare(buf,buf_f0);
    last[j] := buf;
END;

BEGIN
    FOR i := 1 TO n
    DO BEGIN { create scratch files }
        filename := 'klad' + chr((i DIV 10) + ord('0')) +
                                + chr((i MOD 10) + ord('0')) +
                                '.del'; assign(f[i],filename);
        IF i  n
        THEN BEGIN
            a[i] := 1; d[i] := 1; rewrite(f[i])
        END ELSE
        BEGIN
            a[i] := 0; d[i] := 0
        END
    END;
    reset(f0); read_buf(0);
    level := 1; j := 1;
    REPEAT
        selecttape; copyrun
    UNTIL eof_f0 OR (j = n-1);
    WHILE NOT eof_f0
```

```
DO BEGIN
    selecttape;
    IF NOT compare(last[j],buf_f0)
    THEN BEGIN                    { continue old run }
        copyrun;
        IF eof_f0
        THEN d[j] := d[j] + 1
        ELSE copyrun
    END
    ELSE copyrun
END;
FOR i := 1 TO n-1
DO BEGIN
    reset(f[i]); read_buf(i);
END;
FOR i := 1 TO n
DO BEGIN
    t[i] := i;
END;
REPEAT                            { merge from t[1] .. t[n-1] to t[n] }
    z := a[n-1]; d[n] := 0; rewrite(f[t[n]]);
    REPEAT                        { merge one run }
        k := 0;
        FOR i := 1 TO      n-1
        DO BEGIN
            IF d[i]  0
            THEN d[i] := d[i] - 1
            ELSE BEGIN
                k := k + 1; ta[k] := t[i]
            END;
        END;
        IF k = 0
        THEN d[n] := d[n] + 1
        ELSE BEGIN { merge one real run from t[1] .. t[k] }
            REPEAT
                i := 1; mx := 1;
                min := buf_f[ta[1]];
                WHILE i  k
                DO BEGIN
                    i := i + 1; x := buf_f[ta[i]];
                    IF compare(min,x)
                    THEN BEGIN
                        min := x; mx := i
                    END;
                END;
                { ta[mx] contains minimal element; move it to t[n] }
                buf := buf_f[ta[mx]];
                writeln(f[t[n]],buf);
                read_buf(ta[mx]);
                IF compare(buf,buf_f[ta[mx]]) OR eof_f[ta[mx]]
                THEN BEGIN                              { drop this tape }
                    ta[mx] := ta[k]; k := k - 1
                END
            UNTIL k = 0
        END;
        z := z - 1;
```

```
            UNTIL z = 0;
            { rotate tapes }
            reset(f[t[n]]); read_buf(t[n]);
            tn := t[n]; dn := d[n]; z := a[n-1];
            FOR i := n DOWNTO 2
            DO BEGIN
                t[i] := t[i-1]; d[i] := d[i-1]; a[i] := a[i-1] - z
            END;
            t[1] := tn; d[1] := dn; a[1] := z;
            { sorted output is on t[1] }
            reset(f[t[1]]); read_buf(t[1]);
            level := level - 1
        UNTIL level = 0;
        { copy sorted file to t0 }
        rewrite(f0); reset(f[t[1]]);
        WHILE not eof(f[t[1]])
        DO BEGIN
            readln(f[t[1]],buf); writeln(f0,buf);
        END;
        { erase scratch files }
        FOR i := 1 TO n
        DO erase(f[i])
    END; { polyphasesort }

BEGIN
    { Hoofdprogramma. Leest de filenaam in en het startpunt en de lengte
        van de sorteersleutel.
    }
    write('Sortfile : ');   readln(filename);
    assign(f0,filename);
    write('startposition of sort key : '); readln(s_start);
    write('lenght of sort key                : '); readln(s_length);
    IF s_start <M= > 0 THEN s_start := 1;
    IF s_length<M= > 0 THEN s_length:= 255;
    list(f0,0); polyphasesort; list(f0,0)
END.
```

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

Eindelijk is het dan zo ver. Deze keer hopelijk zonder zetduiveltje... Lang verwacht, stil gezwegen. Nooit gedacht, maar vorige maand was 'ie dan toch eindelijk daar: versie 4.00 van het onvolprezen 4DOS. Om meteen maar wat verwarring te stichten: op het BBS van de vereniging staan twee verschillende versies van 4DOS 4.00. Allerleerst staat er een soort beta-test copie. Of deze er nog staat als u dit leest valt zwaar te betwijfelen... Daarnaast staat er de officiële distributie versie in de SDN-areas. De beta versie heeft een aantal minimale bugs en de online help is nog niet compleet.

Wat is 4DOS

Voor hen die niet op de hoogte zijn met het verschijnsel 4DOS: 4DOS is een command interpreter, net als Command.Com. Maar dan een stuk uitgebreider, doordachter etc. De lijst met commando's is een stuk langer, maar tevens biedt 4DOS command line recall- editing en expansion, macro's, online help enz. enz.

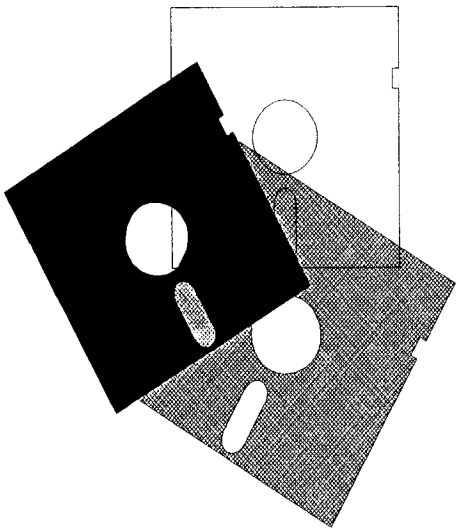
Installatie

De installatie van 4DOS is afhankelijk van de manier waarop u 4DOS aangeleverd kreeg. Als u de "bulletin board" versie heeft moet de installatie met de hand gebeuren. Dat klinkt ingewikkeld, maar valt in de praktijk (mede door de uitstekende documentatie) erg mee. Als je 4DOS op disk aangeleverd krijgt van JP-Software (d.w.z.: per registratie) dan zit er ook een speciaal installatieprogrammaatje bij.

Features

Om naar een nieuwe versie van 4DOS over te schakelen is lang niet altijd nodig. 4DOS is nu ondergebracht in één enkele file, hetgeen een stuk handiger werkt bij formatteer-opdrachten. Bij de "oude" versie was het immers zo dat, na een format /s, alleen 4DOS.Com als shell gecopieerd werd. De .EXE-file moest dan nog met de hand meegecopieerd worden.

Verder is de ondersteuning van DOS 5.00 gerealiseerd in 4DOS 4.00. Dat betekent voor 386-systemen dat 4DOS in upper memory geladen kan worden. Op deze manier neemt 4DOS vrijwel niets meer in van het conventionele geheugen gebied van de PC. Verder heeft men met de invoering van DOS 5.00 een "internal error mechanism" bedacht waarmee de foutmeldingen van de kernel door de commandinterpretor verwerkt worden. In de oude versie van 4DOS kreeg je daarom regelmatig de cryptische melding "Internal error #4" of zo iets. Dit mechanisme wordt nu ook door 4DOS ondersteund.



Andere veranderingen betreffen o.a. een enorme uitbreiding van de set interne functies en variabelen. Verder is het wildcard systeem uitgebreid met UNIX-achtige constructies. Zo kun je nu niet alleen "een willekeurig character" specificeren (het vraagteken in de wildcards), maar ook een character set waaraan voldaan moet worden. Zo kun je bijvoorbeeld alle files die beginnen met een A, B of C te zien krijgen met het commando "dir [a-c]*.*".

De command line history was ook altijd al een sterk punt van 4DOS. Men heeft deze command line history nog eens flink uitgebreid in de nieuwe versie. Zo wordt met PgUp en/of PgDn een klein windowtje geopend waarin de commandline history zichtbaar is. Je kunt er makkelijk doorheen scrollen met behulp van de pijltjestoetsen en door op Enter te drukken kun je er een selecteren. Handiger kon bijna niet!

Inmiddels heeft men ook het verschijnsel "hotkey" geïmplementeerd. Je kunt aan een toets een alias vastknopen waardoor je met behulp van één toetsaanslag een hele rits toetsen kunt simuleren.

Tenslotte heeft men ook nog eens flink zitten sleutelen aan het redirection mechanisme. Geheel in de stijl van UNIX wordt nu een duidelijker onderscheid gemaakt tussen het error-device en het normale "standard I/O" gebeuren. Je kunt foutmeldingen bijvoorbeeld redirecten met "prog & > prog.err".

Help

De online help is (bijna "natuurlijk") ook weer flink uitgebreid. Naast de nieuwe commando's heeft men ook de variabelen, functies, command line editing, instellingen file en ASCII/ANSI tables beschikbaar.

De gehele help is te besturen met de muis, maar echt lekker werkt dat toch nog niet. Helaas heeft men ook bij JP-Software nog geen kaas gegeten van het mode-commando. Men kan met het mode commando bijvoorbeeld de snelheid van het keyboard instellen met "mode con: delay=n rate=m". Wat je dus tevergeefs zoekt in de online help.

Kleuren

Men kan naar hartelust allerlei kleuren instellen in 4DOS. Directories in rood-wit-blauw? Best... popup-help in fel rood? Ook goed... Hoe dit allemaal exact functioneert laat ik over aan de experimentele kronkels van de gebruiker, daar ik zelf niet over een kleurenscherm beschik...

Documentatie

Zoals te doen gebruikelijk is de documentatie van 4DOS groot. Zeg maar gerust... verschrikkelijk groot. Alleen de standaard 4DOS.Doc file is al 367 pagina's (ca. 960 kByte) groot. De bulletin board versie is verpakt in een tweetal files, waarvan de ene net 10% kleiner is dan de ander. De grootste bevat de exe-files, registratie informatie en dat soort zaken, de andere bevat "slechts" de documentatie file...

Die documentatie hoef je in de praktijk (gelukkig maar!) niet helemaal door te lezen. Het is meer een naslagwerk dat je nodig hebt voor eenmalige aangelegenheden als exacte installatie, in- en uitschakelen van features en dat soort zaken.

Conclusie

4DOS is behoorlijk onder handen genomen door Rex Conn. Of de upgrade naar de nieuwe versie verstandig of gewenst is, hangt helemaal af van de manier waarop en waarvoor je 4DOS gebruikt. Zogenaamde "powerusers" zullen deze nieuwe versie ongetwijfeld kunnen waarderen.

Wie van plan is de documentatie op papier te zetten verwijs ik graag nog even naar het programmaatje "Dos4opt". Dit programmaatje is geschreven door één van de points van het bulletin board en je kunt er vier pagina's documentatie op één A4-tje zetten. Zo bespaar je natuurlijk behoorlijk wat papier (en wellicht ook nog heel wat inkt en/of lint!).

Wil je 4DOS echt gaan gebruiken en overweeg je dan ook de zaak te gaan registreren, dan kun je een aardige korting bedingen als je meer dan één copie wilt hebben. De kortingen kunnen dan behoorlijk oplopen. Het plan was dan ook gerezen om dit centraal in de club te doen. Meer informatie over deze centrale registratie-actie is te verkrijgen via het bulletin board (laat gerust eens een berichtje achter!) of via de postbus van de club. Op dit moment zijn de gegevens over de exacte kortingen nog niet beschikbaar. Meer informatie en een aanmeldingsformulier sturen we op aanvraag graag naar u toe...

Joost Voorhaar

Besproken produkt	: 4DOS 4.00
Categorie	: DOS utilities
Registratiekosten	: Maximaal US. $69 (zie tekst)
Auteur/leverancier	: Rex Conn & JP-Software, Arlington, USA.
Verkrijgbaarheid	: The Ultimate, SDN Utilities area Programmatuur in "4dos400d.sdn", 276 kByte Documentatie in "4dos400p.sdn", 258 kByte

Minimale systeemeisen

Standaard PC (Of PS/2. Voor OS/2 is er 4OS2)
PC/MS-DOS
1 diskdrive is voldoende
Beeldscherm: MDA, Hercules, CGA, EGA, VGA, XGA, ...

Algemene beoordeling

Documentatie	: Uitvoerig, Engelstalig
Online help	: Een popup-systeem en een ingebouwd help commando
Gebruikersinterface	: Commandline, een klein beetje window gestuurd (command line history en help)
Muisondersteuning	: Alleen in help en command line history

Positief

Neemt vrijwel geen geheugen meer in beslag
Online help, uitgebreide commando's, etc.

Negatief

Help is nog niet helemaal compleet ("but who needs more?")

Eindbeoordeling

Stabiliteit	: 9
Bruikbaarheid	: 9
Totaalresultaat	: 9

'n Knal assembler

Wel eens de behoefte gehad om uw PC of compatibele computer compleet koud te starten in een batch-file? Ook bedacht dat dat niet ging? Met een paar bytes gaat het wel. Gewoon de re-bootvlag even op nul zetten en naar het resetadres van de CPU springen. Dat is alles. Het kan nog mooier: een AT kan zichzelf resetten. Maar dat is voor een andere keer misschien.

Om COLD.COM te maken, even dit in DEBUG inkloppen:

```
-A 100
XXXX:0100    MOV AX,40 <Enter>
XXXX:0103    MOV DS,AX <Enter>
XXXX:0105    MOV Word Ptr [0072],0 <Enter>
XXXX:010B    JMP F000:FFFF <Enter>
XXXX:0110    <Enter>
-NCOLD.COM <Enter>
-R CX <Enter>
0000: 10 <Enter>
-W <Enter>
Writing 0010 bytes
-Q <Enter>
```

Weer zo'n gigantisch programma (16 bytes!) dat wel eens handig kan blijken.

Nico de Vries

Voortgang KGN-68k

Tussen Kerst en Nieuwjaar bestaat de werkgroep een heel jaar. De optimistische verwachtingen dat we nu klaar zouden zijn, zijn ondertussen achterhaald.

Om deze keer opnieuw enthousiast over de gang van zaken verslag uit te brengen is wat moeilijker dan anders. In de vorige afleveringen kon ik vertellen wat we allemaal bedacht hadden en hoever we het concept omgegooid hadden. De huidige gang van zaken is zo zoals het hoort: "We hebben een goed doordacht ontwerp en dat werken we uit."

Printed Circuit Board

De "drager" van de hardware vraagt nu de meeste aandacht en krijgt die ook. Het is nu ook pas mogelijk omdat volop te doen. Van te voren werd er al flink met de voorlopige versies van het schema gestoeid. De hardware waar het printontwerpprogramma op draait is een echte 386, 386DX dus. "Daar zijn ze toch voor" zult u zeggen, maar het bijzondere is dat de machine niet van de printontwerper is maar van Jan Derksen die zijn eigen systeem hiervoor beschikbaar gesteld heeft.

Het aan elkaar knopen van de eindjes zal nog steeds niet vanzelf gaan. We hebben wel het printoppervlak van een full size AT motherboard maar daar zitten wel her & der bevestigingsgaten. Wij zijn trouwens geïnteresseerd in plaats van de bevestigingspunten in uw AT-kast, laat het ons maar weten en we kunnen ook schetsjes lezen.

Schema

Misschien wel overbodig, maar nog steeds geldt dat het schema voor KGN-leden ter inzage beschikbaar is. Verder kan er nu ook begonnen worden met het documenteren van het schema. Gedeeltes hiervan zullen ook gepubliceerd worden in De μP Kenner.

Software

In het afgelopen jaar hebben we 68000 (cross-)assemblers verzameld en ook monitor programma's in source. Een combinatie van deze twee wordt toegepast om het elementaire programma op een Atari ST aan de praat te krijgen. Vandaar uit is de volgende stap een bootstrap die MINIX laadt zonder de originele ATARI ROMS. Wat ook aan het regelen zijn is de assembler en C compiler van GNU. Deze programma's ondersteunen de extra instructies van MC68030 en worden verspreid in source.

Gratis

Vragen, op en of aanmerkingen, suggesties laat het ons weten. Een Antwoordnummer of een 06 nummer heeft onze club niet, maar het vragen zelf kost voor de rest NIETS.

Geert Stappers

Taakverdeling bestuursleden

Op de ledenvergadering van 16 november j.l. zijn er enkele bestuursmutaties geweest.

Aftredend waren Mick Agterberg, Ton Smits, Geert Stappers en Nico de Vries. Van deze personen was alleen Geert Stappers herkiesbaar. Als nieuwe bestuursleden hadden zich Tonny Schäffer en Joost Voorhaar aangemeld. De aanwezige leden gingen zonder stemming accoord met de voorgestelde bestuursleden.

Op de bestuursvergadering van 13 december zijn de taken binnen het bestuur verdeeld met het volgende resultaat:

Voorzitter:	Tonny Schäffer	
Secretaris:	Gert van Opbroek	
Penningmeester:	Jacques Banser	
Leden:	Jan Derksen	(DOS-65)
	Geert Stappers	(KGN-68k/MINIX)
	Joost Voorhaar	(Redactie)

Verder is er in het bestuur een vacature voor Public Relations en ledenwerving.

Uiteraard worden alle gekozen bestuursleden van harte gefeliciteerd met hun verkiezing en worden de afgetreden bestuursleden heel hartelijk bedankt voor hetgeen ze in de functie van bestuurslid voor de KGN gedaan hebben.

Ontwikkelaars geven toe dat UNIX en C een grap zijn

Nu de KGN zo druk bezig is met UNIX, en daarmee samenhangend met de programmeertaal C, mag ik u het volgende verhaal niet onthouden. Het gaat hier om een vertaling van een artikel het blad "Computerworld" van 1 April 1991.

In een bekendmaking die de computerwereld versteld deed staan, gaven Ken Thompson, Dennis Ritchie en Brian Kernighan toe dat het door hun ontworpen operating system UNIX en de programmeertaal C beide zorgvuldig opgezette 1-aprilgrappen zijn die ruim 20 jaar stand hebben gehouden. Tijdens het afgelopen "UnixWorld Software Development" forum onthulde Thompson het volgende:

"In 1969 had AT&T net de werkzaamheden met het GE/Honeywell/AT&T Multecs project afgesloten. Brian en ik waren begonnen te werken met een vroege evaluatie Pascal-versie van professor Nichlaus Wirth's ETH Laboratorium in Zwitserland, en waren diep onder de indruk van de kracht en elegantie van Pascal. Dennis had net het boek "Bored Of The Rings" uitgelezen, een parodie op de "Lord Of The Rings" trilogie van Tolkien. Bij wijze van geintje besloten we een parodie op de Multics omgeving en Pascal te bouwen. Dennis en ik waren verantwoordelijk voor de systeemomgeving. We namen Multics als uitgangspunt en maakten het nieuwe systeem zo complex en cryptisch mogelijk om de gebruikers zo veel mogelijk te frustreren. We doopten het nieuwe systeem "UNIX" als parodie en dubbelzinnige zinspeling op "Multics". Vervolgens zetten Dennis en Brian zich aan het werk om een zo verwrongen mogelijke variatie op Pascal te bedenken en noemden hun produkt "A". Toen het duidelijk werd dat anderen daadwerkelijk probeerden te programmeren met A, voegden we zo snel mogelijk nieuwe cryptische eigenschappen aan de taal toe en noemden het achtereenvolgens B, BCPL en uiteindelijk C. We hielden pas op toen we de volgende regal syntaxtisch correct konden compileren:

```
for(;P("\n"),R-;P("|"))for(e=C;e-;P("__"+(*u
++/8)%2))P("| "+(*u/4)%2);
```

We konden ons met geen mogelijkheid voorstellen dat een taal die dit accepteerde ooit door moderne programmeurs gebruikt zou gaan worden. We hebben zelfs overwogen om deze taal aan de Sovjets te verkopen om hun computeronderzoek tenminste twintig jaar terug te zetten. U kunt zich onze verrassing voorstellen toen AT&T en andere Amerikaanse firma's begonnen UNIX en C daadwerkelijk te gebruiken! Het heeft ze twintig jaar gekost om voldoende kennis op te bouwen om met behulp van deze uit 1960 stammende technologische parodie iets zinnigs te kunnen doen, maar we zijn diep onder de indruk van de vasthoudendheid (en gezond verstand) van de gemiddelde UNIX- en C programmeur. In ieder geval hebben Brian, Dennis en ik de afgelopen twintig jaar alleen in Pascal op de Apple Macintosh geprogrammeerd en voelen we ons behoorlijk schuldig aan de chaos, verwarring en werkelijk slecht programmeerwerk als een gevolg van onze 1-april grap."

De grootste firma's en gebruikers van UNIX en C, inclusief AT&T, Micrsosoft, Hewlett-Packard, GTE, NCR en DEC hebben tot nog toe geweigerd enig commentaar te geven. Borland International, marktleider van Pascal- en C tools en ontwerper van de bekende Turbo-Pascal, Turbo-C en C++ produktenlijn liet weten dat ze iets dergelijks al enige jaren vermoedde, dat ze verder zou gaan met het uitbouwen van hun Pascal produkten en de verdere ontwikkeling van C zou stoppen. Een woordvoerder van IBM barstte slechts in een ongecontroleerde lachbui uit en stelde vervolgens een persconferentie uit waarin het lot van de RS-6000 uit de doeken gedaan zou worden met de gevleugelde zinsnede "VM will be available Very Soon Now". Professor Wirth van het ETH instituut, de geestelijk vader van o.a. Pascal, Modula-2 en Oberon, liet op cryptische wijze weten dat P.T. Barnum volledig gelijk had.

In een ander verhaal dat wat later op de redactie binnenkwam wordt melding gemaakt van het feit dat een gelijksoortige bekentenis van William Gates op handen is betreffende het operating system MS-DOS en de MS-Windows-omgeving. Een IBM woordvoerder tenslotte ontkende dat het Virtual Machine concept (VM) op gelijksoortige achtergrond had.

Met dank aan Bernard L. Hayes voor het originele verhaal en Peter Danevicius voor de verspreiding op fidonet.

Joost Voorhaar

Computersystemen voor het beheer van goederen

Inleiding

In de rubriek "toepassingen" is er ruimte om een (technische) toepassing van (micro-)computersystemen te presenteren. In deze aflevering wil ik graag iets vertellen over de wijze waarop computersystemen ingezet worden bij het beheren van goederen die in opslag in een magazijn liggen. Dit artikel is gebaseerd op enkele projecten waaraan ik heb meegewerkt bij mijn huidige werkgever: PSI Processturing-in Informatiesystemen B.V. gevestigd in Alphen a.d. Rijn. De voorbeelden zijn echter compleet verzonnen en hebben geen enkele relatie met één van onze opdrachtgevers.

Logistiek

In dit artikel gaan we ons bezig houden met iets dat in de vaktaal Logistiek genoemd wordt. Logistiek houdt zich bezig met het transport en de opslag van goederen. Hieronder vallen dus magazijnen en het beheer van de goederen in deze magazijnen, de distributie van goederen met bijvoorbeeld Van Gend en Loos maar bijvoorbeeld ook het plannen wanneer iets op een bepaalde plaats moet zijn en het uitzoeken waar de goederen het beste vandaan kunnen komen en langs welke weg ze het beste vervoerd kunnen worden. Zelfs het file-probleem zou je een logistiek probleem kunnen noemen.

Zelfs het file-probleem zou je een logistiek probleem kunnen noemen.

Hoogbouwmagazijnen

Misschien heb je ze wel eens gezien. Ze staan meestal op een industrieterrein of anders in ieder geval aan de rand van de stad. Gebouwen zo groot als een flatgebouw, opgebouwd uit een soort golfplaten en verder zonder ramen. Dit zijn de zogenaamde hoogbouwmagazijnen. In figuur 1 staat een afbeelding van een dergelijk magazijn in aanbouw, in dit geval het hoogbouwmagazijn van Centraal Boekhuis te Culemborg met een hoogte van 34 meter, een diepte van 120 meter en een breedte van zo'n 15 meter.

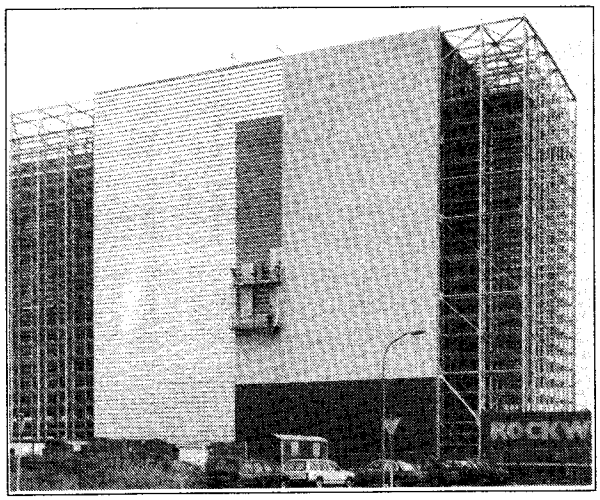


Fig. 1: hoogbouwmagazijn in aanbouw

In een dergelijk hoogbouwmagazijn worden goederen opgeslagen, in geval van Centraal Boekhuis pallets met boeken. In het magazijn staan stellingen waarin de pallets geplaatst kunnen worden waarbij er steeds twee stellingen met de rug tegen elkaar geplaatst zijn. Verder lopen er tussen de stellingen gangpaden waarlangs de pallets worden aan- en afgevoerd. Bij Centraal Boekhuis is ruimte voor ruim 23.000 pallets, verdeeld over 5 gangen. Gemiddeld liggen er in dit magazijn zo'n 2.000.000 boeken.

Als je een hoogbouwmagazijn hebt, kun je natuurlijk de pallets het magazijn in- en uitrijden met een veredelde heftruck. Een dergelijke truck moet dan wel een heel eind met zijn vorken omhoog kunnen. Deze trucks bestaan en worden reachtrucks genoemd. Vaak laat men echter in een dergelijk magazijn zogenaamde automatische kranen of SBA's van Stelling Bedien Apparaat rijden. Dit is een soort automatische reachtruck die door de gang rijdt (op rails) en die een pallet uit de stelling kan pakken en ergens anders neer kan zetten. Uiteraard kan de kraan ook een pallet oppakken en ergens in de stelling, in een lokatie, afzetten. In figuur 2 is het inwendige van een hoogbouwmagazijn afgebeeld. In dit geval gaat het om een magazijn voor bakken. We zien de stellingen met daarin de bakken en, achteraan, een automatische kraan voor het transport van de bakken in en uit de stelling. Verder zien we een aantal transportbanen waarmee de bakken naar de desbetreffende gang vervoerd worden of waarlangs ze worden afgevoerd.

Als er nu een bak het magazijn in moet, moet worden ingeslagen in vaktaal, dan wordt de bak aangevoerd naar een bepaalde gang via de transportbaan die bij die gang hoort. Aan het einde van de transportbaan kan de kraan de bak oppakken waarna de kraan naar de gewenste lokatie gaat en de bak daar afzet. Moet er een bak uit het magazijn gehaald (= uitgeslagen) worden, dan haalt de kraan de bak uit zijn lokatie, brengt de bak naar het begin van de

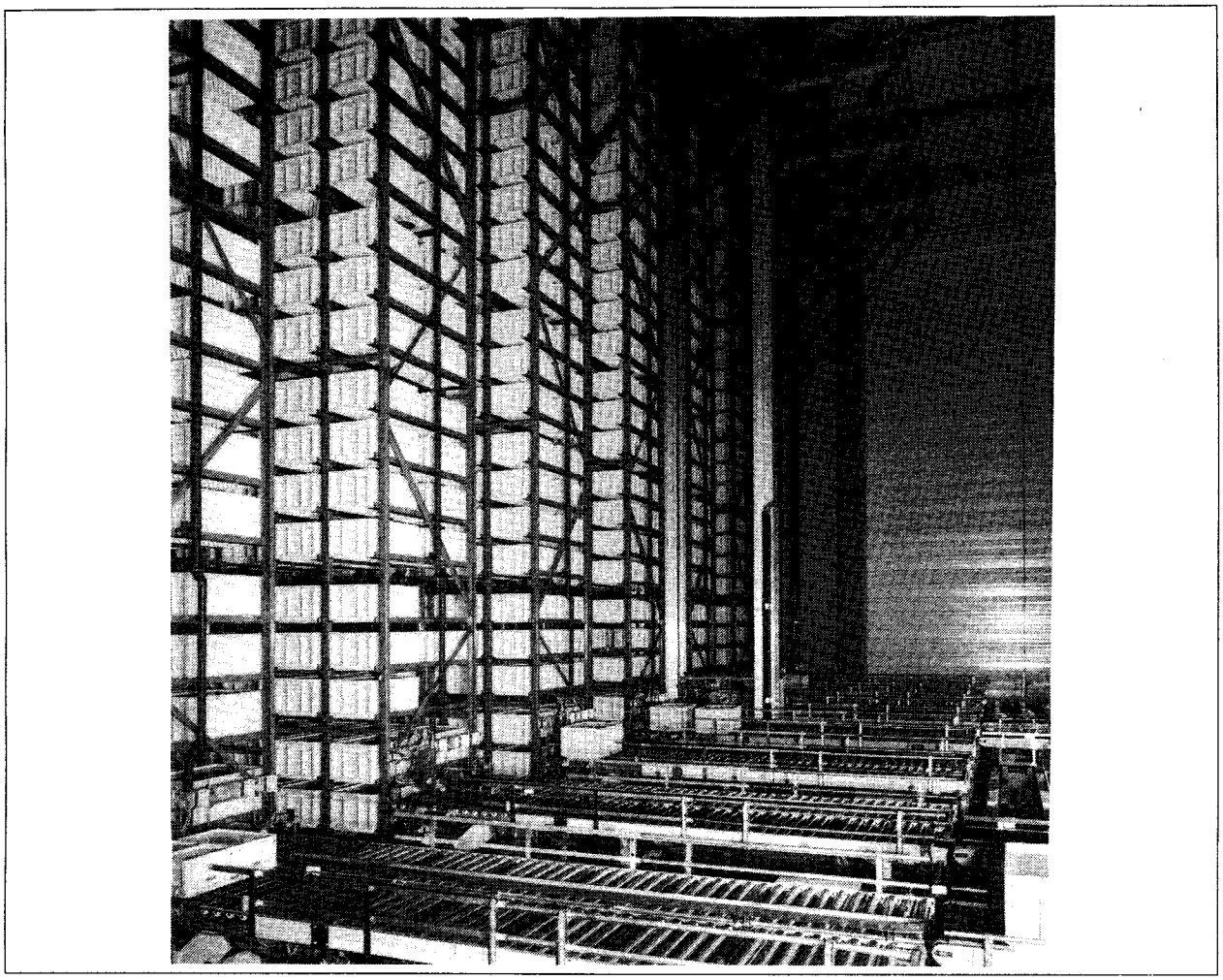


Fig. 2: automatisch bakkenmagazijn

transportbaan en zet de bak daar af. De transportbaan vervoert de bak dan verder.

In het afgebeelde magazijn heeft elke gang zijn eigen automatische kraan. Deze kraan kan een bak verticaal verplaatsen en zelf in horizontale richting bewegen. Verder kan hij zowel links als rechts een bak oppakken of afzetten. Er bestaan ook automatische kranen die, over rails, van de ene naar de andere gang kunnen rijden.

Om een inzicht te geven in de transportcapaciteiten enkele voorbeelden. In een automatisch palletmagazijn kan één enkele kraan per uur zo'n 20 tot 25 pallets per uur in- en uitslaan. Een pallet kan in dat geval tot zo'n 1000 kg wegen. Voor een bakkenmagazijn voor bakken met een gewicht van zo'n 25 kg zijn er capaciteiten haalbaar tot ruim 70 bakken in en uit per kraan per uur. Dit komt uiteraard doordat bakken meestal kleiner en dus lichter zijn.

Enkele logistieke termen

We hebben het in de vorige paragraaf al even over inslag en uitslag gehad. Inslag van goederen is het transport van de goederen naar het magazijn. Uitslag is het transport van de goederen uit het magazijn.

Als een kraan alleen pallets inslaat en geen pallets uit het magazijn haalt, dan rijdt de kraan dus met een pallet van het oppakpunt aan het einde van de transportbaan de gang in, rijdt leeg terug en brengt de volgende pallet weg. Een dergelijke beweging heet "Enkelspel" waarbij niet de hele capaciteit van de kraan gebruikt wordt. Hetzelfde geldt voor een kraan die alleen maar pallets uit het magazijn haalt. Je kunt het echter ook zo uitkienen dat de kraan een pallet inslaat en op de terugweg meteen een pallet uit het magazijn oppakt die uitgeslagen moet worden. Hierbij hoeft een kraan alleen de afstand tussen de lokatie van de ingeslagen pallet en de pallet die uitgeslagen wordt leeg af te leggen. Een dergelijk beweging wordt "Dubbelspel" genoemd. De capaci-

teit van een kraan wordt meestal opgegeven in dubbelspelen per uur.

Voorbeeld van een magazijn

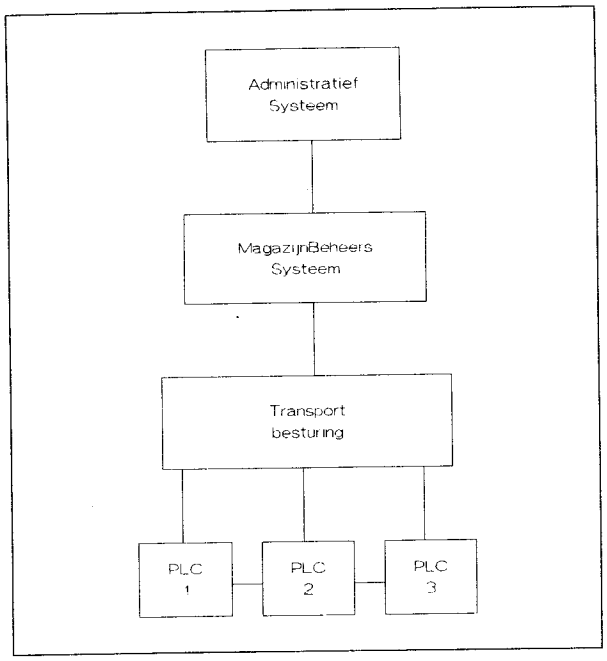


Fig. 3: transport rond een automatisch magazijn

In figuur 3 is het transport van de goederen rond een automatisch magazijn getekend. Links staat het hoogbouwmagazijn met vier gangen. Deze gangen zijn genummerd van 1 tot 4. Elke gang heeft een eigen aanvoerbaan en een eigen afvoerbaan. Aan het einde van de aanvoerbaan kan de kraan de pallet oppakken en naar een lokatie in de gang brengen. Een kraan kan een pallet aan het begin van de afvoerbaan plaatsen waarna de pallet via het transportsysteem afgevoerd wordt.

Rechts boven zijn een aan- en afvoerbaan aangegeven met I van invoerbaan en U van uitvoerbaan. Nieuwe pallets worden van buiten het magazijn aangevoerd en op de invoerbaan I geplaatst. Daar vindt de identificatie van de pallet plaats waarbij er wordt gekeken welke goederen er op de pallet liggen en de pallet wordt opgenomen in de administratie van het magazijn. Om deze reden wordt de plaats waar de I in de tekening staat ook wel I-punt of identificatiepunt genoemd. De uitvoerbaan is voor pallets die compleet afgevoerd worden uit het magazijn. Op deze baan kunnen de pallets worden verzameld waarna ze, bijvoorbeeld met een heftruck, naar een gereedstaande vrachtwagen worden gebracht.

Rechts onder zien we nog twee werkplekken, aangegeven met A en B. Het transportsysteem kan een pallet naar de aanvoerbaan van één van deze werkplekken brengen waarna er "iets" met de pallet gedaan wordt. Vervolgens wordt de pallet overgezet naar de afvoerbaan van de werkplek waarna de pallet terug naar het magazijn kan of bijvoorbeeld naar de uitvoerbaan. Op een dergelijke werkplek kan men bijvoorbeeld tellen hoeveel goederen er nog op de pallet liggen. Deze bewerking wordt inventariseren genoemd. Een andere, veel belangrijkere bewerking, is de zogenaamde orderverzameling. Stel we hebben diverse soorten blikken olie in ons magazijn liggen. Op elke pallet liggen maximaal 250 blikken. Nu wil de voorzitter van de KGN 3 blikken olie A123-51 hebben voor zijn auto. Op dat moment wordt er een pallet met deze soort olie uitgeslagen naar één van de werkplekken waarna iemand, de orderpicker, drie blikken van deze pallet afneemt en in een doos voor de voorzitter doet. De pallet met het restant gaat terug het magazijn in.

Tenslotte is er in het transportsysteem een centraal circuit die de diverse aan- en afvoerbanen met elkaar verbindt.

Automatisering

Een installatie zoals uit het voorbeeld uit de vorige alinea kan niet zonder automatisering. Een mogelijke opbouw van de computersystemen voor een dergelijk geheel staat getekend in figuur 4. In de eerste plaats is er uiteraard besturing van de kranen en het transportsysteem nodig. Dit wordt zonder uitzondering gedaan met behulp van zogenaamde PLC's van Programmable Logical Controllers. In feite zijn dit een soort microcomputers die schakelaars omzetten,

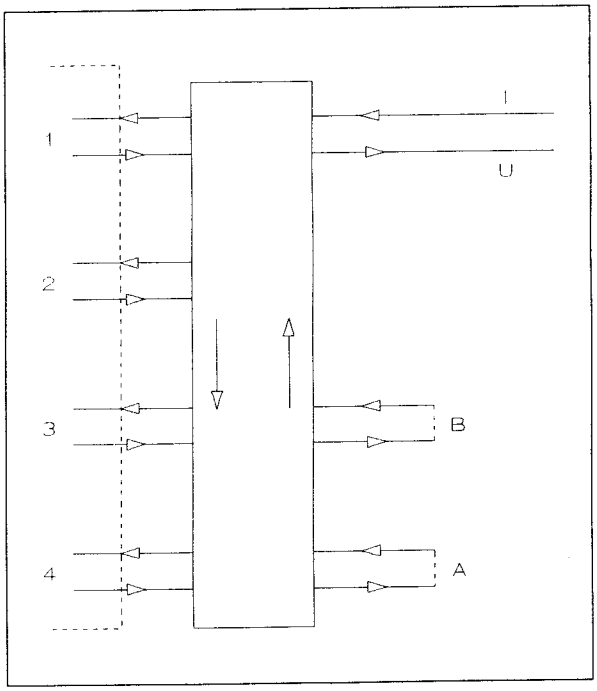


Fig. 4: hiërarchie van de computersystemen

relais uitlezen etc. In een PLC loopt een programma af die op basis van een aantal ingangssignalen een aantal uitgangen activeert. De ingangssignalen kunnen bijvoorbeeld naderingsschakelaars zijn of bijvoorbeeld fotocellen die aangeven dat er op een bepaalde positie op het transportsysteem een pallet staat. Met behulp van de uitgangssignalen worden bijvoorbeeld de motoren van een kraan of het transportsysteem geactiveerd. Verder kunnen PLC's ook met andere PLC's of computers communiceren. In de praktijk zal elke kraan een eigen PLC hebben. Verder zal er minimaal één PLC zijn voor het sturen van het transportsysteem. PLC's worden door een groot aantal leveranciers gemaakt. Bekende voorbeelden zijn onder andere Siemens (S5), Modycom, Satt Control etc.

Op het niveau van de PLC's heb je te maken met eenvoudige opdrachten. Eén van de opdrachten voor de kranen zou bijvoorbeeld kunnen zijn: "Pak de pallet van het oppakpunt en breng deze naar lokatie 03-12". Een opdracht voor de PLC van het transportsysteem zou kunnen zijn "Transporteer de pallet bij het I-punt naar positie 23".

"Pak de pallet van het oppakpunt en breng deze naar lokatie 03-12".

Op een iets hoger niveau, de transportbesturing, krijg je te maken met opdrachten in de vorm "Transporteer de pallet bij het I-punt naar lokatie 03-12 in gang 3". Over het algemeen krijgt deze pallet dan ook een soort identificatie mee in de vorm van een transportnummer. Soms wordt dit transportnummer in barcode op een etiketje afgedrukt zodat de pallet met behulp van een barcode lezer (scanner) geïdentificeerd kan worden. Het systeem dat al deze transporten door de PLC's uit laat voeren en daarbij ook bewaakt waar elke pallet is en er voor zorgt dat er geen aanrijdingen plaatsvinden kan zelf ook weer een PLC zijn. Dit kan echter ook een computersysteem zijn. In de projecten die ik meegemaakt heb, is het soms een PLC die dan ook wel concentrator genoemd wordt en soms een microcomputer met een 6809 of 68000 als processor. De transportbesturing communiceert meestal met de PLC's m.b.v. asynchrone seriële lijnen die ook onderling met elkaar kunnen communiceren. Een nieuwe ontwikkeling is communicatie tussen PLC's m.b.v. een Ethernet netwerk. De transportbesturing is in staat geheel zelfstandig transporten tussen eindpunten uit te voeren. Dat wil zeggen dat de transportbesturing van zijn hoger gelegen systeem opdracht krijgt pallet XYZ te transporteren van A naar B. De transportbesturing verzorgt dit transport en meldt na afloop aan het hogere systeem dat het transport XYZ van A naar B is uitgevoerd.

Om tijdens het transport bij te houden waar een pallet is, zijn er twee technieken in omloop. De eerste baseert zich op zogenaamde backtracking. In de microcomputer wordt de hele installatie afgebeeld en wordt precies bijgehouden waar elke pallet is. Als er aan een PLC opdracht wordt gegeven een pallet te transporteren van positie 23 naar 42 wordt het palletnummer dat in positie 23 in de microcomputer staat doorgeboekt naar positie 42. De tweede techniek werkt met een aantal extra identificatie-punten. Dit zijn bijvoorbeeld barcode lezers die de barcode op het etiket van de pallet kunnen lezen. Bij elke wissel in de installatie staat zo'n lezer en als een pallet daar arriveert wordt zijn barcode gelezen waarna de microcomputer weet welke pallet het is en welke opdrachten hij aan de PLC's moet geven om het transport te laten vervolgen.

Het derde niveau van onderen is het zogenaamde magazijnbeheerssysteem. Dit is het niveau waar PSI specialist in is. Op het magazijnbeheerssysteem wordt onder andere bijgehouden welke lokaties in het magazijn leeg zijn, welke lokaties gevuld zijn en waarmee. Verder doet het magazijnbeheerssysteem de aansturing van de transportbesturing en communiceert ze met bedieningsmensen. Meestal is het magazijnbeheerssysteem ook verantwoordelijk voor het aansturen van de orderverzameling. In de volgende paragraaf zal de rol en functie van het magazijnbeheerssysteem verder beschreven worden. Voor het magazijnbeheerssysteem kunnen, afhankelijk van de gestelde eisen, diverse computers gebruikt worden. PC's, al dan niet in een netwerk, zijn voor niet te grote systemen een goede oplossing. Voor de wat grotere systemen wordt echter meestal voor een minicomputer, bijvoorbeeld een VAX van de firma Digital, gekozen. Op het hoogste niveau binnen de automatisering vinden we meestal een administratief systeem. Dit systeem doet de financiële afhandeling van de transacties. Ook de bestellingen en orders worden meestal op dit systeem ingevoerd waarna het magazijnbeheerssysteem door het administratieve systeem kan worden aangestuurd. Om de scheidslijn tussen administratief systeem en magazijnbeheerssysteem nog meer duidelijk te maken het volgende voorbeeld:

Stel je weer voor dat in het magazijn weer pallets met blikken olie liggen. De kenmerken van de verschillende soorten olie noemen we de stamgegevens en liggen opgeslagen in de database van het administratieve systeem dus: Type X25-12, olie voor dieselmotoren, heeft als kleur licht geel, en kost fl. 23,15

per blik verkoop en fl. 16,12 inkoop. Verder is op het administratieve systeem bekend dat er 1213 blikken in voorraad zijn en dat er een bestelling bij leverancier Vettigheid B.V. geplaatst is voor 5000 blikken. Het administratieve systeem weet absoluut niet waar de blikken olie X25-12 in het magazijn liggen.

Het magazijnbeheerssysteem heeft in zijn database ook een deel van de stamgegevens opgeslagen. Voor het magazijnbeheerssysteem is het niet relevant wat de in- en verkoopprijs van een blok olie is; de omschrijving zoals die op het blik staat wel zodat een orderpicker kan controleren of hij de juiste olie voor zijn neus heeft staan. Wat het MBS (afkorting voor Magazijn Beheers Systeem) wel weet is dat de voorraad olie X25-12 bestaat uit pallet 1001-01 met daarop 600 blikken, pallet 1001-02 met 600 blikken en pallet 0097-14 met 13 blikken. Verder weet MBS dat de pallets op resp. de lokatie 1-12-10 (gang, diepte, hoogte), 3-08-09 en 4-11-07 staan.

Zoals uit figuur 4 blijkt, lopen er communicatielijnen tussen het administratieve systeem en het MBS en tussen de transportbesturing en het MBS. De communicatie naar de transportbesturing is altijd een ONLINE communicatie omdat de transportopdrachten die op het MBS gegenereerd worden met zo weinig mogelijk vertraging uitgevoerd moeten worden. Voor deze communicatie wordt meestal een asynchrone seriële verbinding gebruikt waarbij gebruik gemaakt wordt van een protocol. Als alternatief kan ook gebruik gemaakt worden van een verbinding via een netwerk zoals bijvoorbeeld Ethernet. Voor de communicatie met het administratieve systeem kan, afhankelijk van de situatie, gekozen worden voor een ONLINE verbinding via een netwerk of seriële verbinding (asynchroon of synchroon) of een communicatievorm waarbij de systemen op regelmatige basis via bijvoorbeeld tape of floppy hun informatie uitwisselen. Is een snelle ONLINE communicatie gewenst, dan kan men de twee systemen opnemen in een Local Area Network.

Het magazijnbeheerssysteem

Het magazijnbeheerssysteem is het onderdeel in de hiërarchie van computersystemen waarop we ons zullen concentreren. In de vorige paragraaf is de globale scheiding al aangegeven tussen de transportbesturing en het MBS aan de ene kant en tussen het administratieve systeem en het MBS aan de ander kant. In deze paragraaf zullen we eens wat nauwkeuriger gaan kijken naar de taken van het MBS en welke rol de diverse systemen spelen binnen het magazijnproces. Om dit te illustreren, gaan we uit van een paar praktijkgevallen.

Laten we als voorbeeld maar weer het magazijn met olie-produkten nemen. Op een gegeven moment plaatst de firma een bestelling van een nieuw produkt Y13-12 bij een leverancier. Men bestelt 2000 blikken met een inhoud van 1 liter. Deze bestelling wordt ingebracht in het administratieve systeem. Omdat het produkt nog niet bekend is, moeten ook de stamgegevens van het artikel zoals bijvoorbeeld de omschrijving "Extra fijne olie, speciaal voor sportwagens" in het systeem ingevoerd worden. Het administratieve systeem weet dus nu dat er op een bepaald moment 2000 liter van produkt Y13-12 afgeleverd kan worden.

Men bestelt 2000 blikken met een inhoud van 1 liter. "Extra fijne olie, speciaal voor sportwagens".

Op een gegeven moment komt de vrachtwagen van de leverancier voorrijden met de 2000 blikken van 1 liter. Deze blikken worden uitgeladen en ergens in de buurt van de invoerbaan tijdelijk neergezet. Op het administratieve systeem wordt aangegeven dat de goederen "In huis" zijn en worden eventueel ontbrekende delen van de stamgegevens toegevoegd. Vervolgens stuurt het administratieve systeem twee telegrammen naar het MBS. In het eerste telegram staan de stamgegevens voorzover ze voor het MBS van belang zijn. In het tweede telegram staat de opdracht voor het MBS 2000 blikken van artikel Y13-12 in opslag te nemen en een opdrachtnummer (bijvoorbeeld 001712). Ondertussen hebben de mensen in ontvangst goederen (zo heet de afdeling waar de goederen binnenkomen) de blikken op pallets geplaatst, zodanig dat ze in opslag genomen kunnen worden. Er gaan 250 blikken op een pallet zodat er dus 8 pallets gemaakt zijn. Deze 8 pallets worden op de invoer-transportbaan geplaatst waarna het transportsysteem ze automatisch één voor één transporteert naar het I-punt.

Als de eerste pallet aankomt bij het I-punt, kijkt de bedieningsman (vrouw) met behulp van een terminal die is aangesloten op het MBS welke goederen er op het I-punt verwacht worden. Hij ziet de 2000 blikken Y13-12 staan en kan bepalen dat de pallet die hij voor zijn neus heeft hierbij hoort. Hij maakt aan het MBS, met behulp van dezelfde terminal, bekend dat hij 250 blikken van deze partij voor zijn neus heeft. Het MBS zal dan een zogenaamd partijnummer genereren en een etiket voor één pallet met 250 blik-

ken afdrukken. Deze pallet krijgt dan bijvoorbeeld nummer 1991-0274-01. Uiteraard worden de gegevens van deze pallet (palletnummer, produktnummer en aantal) vastgelegd in de database op het MBS.

Vervolgens kan de bedieningsman via de terminal opdracht geven voor transport van deze pallet in de richting van het magazijn. MBS weet dus nu dat op pallet 1991-0274-01 250 blikken Y13-12 staan uit de opdracht 001712. MBS gaat nu een lokatie bepalen waar de genoemde pallet opgeslagen kan worden. Hiervoor zijn een aantal regels waarvan ik er enkele zal noemen. In de eerste plaats moet de lokatie leeg zijn, er kunnen tenslotte geen twee pallets op één lokatie staan. In de tweede plaats moet de betreffende pallet op die lokatie passen. In een hoogbouwmagazijn zijn namelijk vaak verschillende soorten lokaties, zoals bijvoorbeeld lage voor lage pallets en hoge lokaties voor hoge pallets. In een olie-magazijn speelt brandbaarheid misschien ook een rol waardoor je niet in het wilde weg produkten bij elkaar in de buurt kunt plaatsen. Iets wat ook zeker een rol speelt is dat men geen lokatie in een gang kiest waarvan de kraan langdurig in reparatie is. Nadat deze inslagstrategie doorlopen is, krijgt de transportbesturing opdracht pallet 1991-0274-01 naar de gekozen lokatie te transporteren. De gekozen lokatie wordt in de database op het MBS gereserveerd voor de pallet en in de palletgegevens wordt aangegeven naar welke lokatie de pallet op weg is.

Dus Ordernummer 823: Garagehouder Pietersen bestelt 600 blikken Y13-12.

Met de overige 7 pallets gaat het net zo waarbij de inslagstrategie er over het algemeen voor zal zorgen dat de pallets zo goed mogelijk over de gangen gespreid worden.

De transportbesturing zal de transportopdrachten opbreken in deelopdrachten voor de PLC's en er voor zorgen dat de pallets op de gekozen lokaties worden afgezet. Vervolgens zal de transportbesturing het transport afmelden aan het MBS. Nadat het MBS van de transportbesturing een transportopdracht afgemeld krijgt, zal het MBS in zijn database aangeven dat de betreffende pallet op een bepaalde lokatie staat en dat deze goederen beschikbaar zijn voor uitslag. Nadat alle pallets van een partij zijn afgemeld, zal het MBS aan het administratieve systeem doorgeven dat opdracht 001712 is afgewerkt en dat er 2000 blikken Y13-12 in voorraad zijn genomen. Het administratieve systeem zal dit aantal afboeken van de bestelling en bijboeken bij de voorraad.

Nadat de goederen ingeslagen zijn, staan ter beschikking voor uitslag. Dat wil zeggen dat de goederen verkocht kunnen worden. Dit zal over het algemeen ook geïnitieerd worden op het administratieve systeem. Stel bijvoorbeeld dat een garagehouder 600 blikken Y13-12 bestelt. Deze order wordt, voorzien van een ordernummer ingebracht in het administratieve systeem. Dus Ordernummer 823: Garagehouder Pietersen bestelt 600 blikken Y13-12. Het administratieve systeem weet dat er nog 2000 blikken in voorraad zijn en zal van deze voorraad 600 blikken reserveren. Vervolgens wordt de order doorgestuurd naar het MBS. Het MBS krijgt de order van het administratieve systeem en zal na ontvangst goederen voor deze order gaan reserveren. Dit betekent in het concrete voorbeeld dat hij 2 volle pallets (bijvoorbeeld 1991-0274-01 en 1991-0274-02) compleet zal reserveren plus van pallet 1991-0274-03 100 stuks. Aangezien de goederen niet direct uitgeleverd hoeven te worden, wordt er nog geen transport op gang gebracht. De volgende dag komen er voor dit produkt nog twee orders binnen, ordernummer 824 voor garage Smit voor 75 blikken Y13-12 en ordernummer 825 voor garage De Vries 375 blikken Y13-12. Voor ordernummer 824 worden 75 blikken van pallet 1991-0274-03 gereserveerd en voor ordernummer 825 wordt pallet 1991-0274-04 gereserveerd, 75 blikken op pallet 1991-0274-03 en 75 blikken op pallet 1991-0274-05.

Goed, de vrachtwagen komt er aan en de goederen kunnen worden uitgeslagen. De volle pallets 1991-0274-01, -02 en -04 worden uitgeslagen naar de uitvoerbaan U. Daar aangekomen zal MBS een aantal documenten afdrukken die bij de pallets gevoegd worden. Pallet 1991-0274-03 en 1991-0274-05 worden getransporteerd naar werkplek B. Als pallet 1991-0274-03 op deze werkplek arriveert, zal MBS op de terminal die bij de werkplek staat aangeven dat er 100 blikken voor garage De Vries van deze pallet afgenomen moeten worden. De orderpicker voert deze opdracht uit en bevestigt dit aan het MBS. MBS zal enkele documenten voor deze goederen afdrukken waarna de bedieningsman de 100 blikken in een doos doet of op een pallet stapelt. Vervolgens krijgt de orderpicker opdracht de 75 blikken voor garage Smit af te nemen. Heeft hij dit ook gedaan, dan krijgt hij opdracht voor de 75 blikken voor garage De Vries. Nadat hij deze opdracht

uitgevoerd heeft, is de pallet leeg die vervolgens door MBS, in combinatie met de onderliggende systemen wordt afgevoerd naar bijvoorbeeld de uitvoerbaan. Voor de tweede pallet krijgt de orderpicker opdracht de resterende 75 exemplaren voor garage de Vries af te pakken. Nadat de opdrachten verwerkt zijn, moet de voorraad op pallet 1991-0274-05 nog 175 stuks zijn. De orderpicker ziet echter dat één van de blikken beschadigd is en lekt. Hij kan dan dit blik van de pallet afpakken en op zijn terminal aangeven dat het aantal op de pallet nu 174 stuks is. Vervolgens geeft hij opdracht de pallet af te voeren.

Nadat een order compleet uitgeslagen is, zal MBS de order afmelden bij het administratieve systeem waarbij eventuele afwijkingen worden doorgegeven. Het administratieve systeem zal de factuur aanmaken en de voorraad bijwerken. Ook de voorraadmutatie van 1 stuk, het lekkende blik, zal door MBS aan het administratieve systeem worden doorgegeven.

De pallet die vanaf de werkplek wordt afgevoerd zal weer worden ingeslagen in het hoogbouwmagazijn. Over het algemeen wordt voor deze inslag de inslagstrategie opnieuw doorlopen worden. Het is namelijk niet altijd gewenst de pallet weer op dezelfde lokatie op te slaan. Neem als voorbeeld een magazijn met lokaties met verschillende hoogten. Tengevolge van de orderverzameling kan een hoge pallet laag geworden zijn waarvoor MBS bij voorkeur een lage lokatie moet zoeken. Dit houdt in dat voor elke pallet die uitgeslagen wordt de bijbehorende lokatie wordt vrijgegeven. Komt een pallet terug in het magazijn, dan wordt voor deze pallet door de inslagstrategie één van de lege lokaties gereserveerd.

Wat doe je dan?

Afsluiting

De beschreven voorbeelden schetsen een mogelijke situatie. Uiteraard komt er op de diverse systemen nog wel wat meer kijken dan we beschreven hebben. Neem alleen al de situatie dat de gekozen lokatie niet leeg blijkt te zijn. Op dat moment moeten MBS en de onderliggende systemen een alternatieve lokatie zoeken waarna de lokatie die ten onrechte gevuld blijkt te zijn moet worden geïnventariseerd. Verder wordt de inslagstrategie meestal niet op het moment van afsturen doorlopen maar "ergens" onderweg. In de praktijk betekent dit meestal dat de transportbesturing eerst een voorlopig doeladres krijgt en pas later de definitieve lokatie. Tenslotte is de beschreven wijze van orderverzameling lang niet compleet. Neem bijvoorbeeld het geval dat Pietersen, Smit en De Vries elk twee produkten bestellen, waarvan alleen de bestelling voor Y13-12 gemeenschappelijk is. Wat doe je dan? Sla je uit per klant dus eerst alles voor Pietersen, dan alles voor Smit en tenslotte alles voor De Vries of sla je eerst produkt Y13-12 uit, dan het tweede produkt voor Pietersen etc. Dit zijn allemaal zaken die in de software op het MBS, in dit geval de zogenaamde uitslagstrategie, geregeld worden.

Uiteraard is er nog heel veel over logistieke systemen te vertellen. Ik zou mij voor kunnen stellen dat er mensen zijn die een dergelijk systeem wel eens in bedrijf willen zien. Helaas zie ik momenteel geen mogelijkheden een soort club-excursie te organiseren. Ik heb echter wel enkele video-banden in mijn bezit die ik, in besloten kring, kan vertonen. Zoals het er nu naar uitziet zal dit op de bijeenkomst in Krommenie gebeuren. Misschien daar tot ziens.

Gert van Opbroek

Gevraagd: P.R.-functionaris

voor de KIM Gebruikersclub Nederland.

Het bestuur van de KGN zoekt onder de leden iemand die zich, in de functie van bestuurslid, actief bezig wil gaan houden met het opbouwen en onderhouden van externe contacten en het organiseren van ledenwervingsacties. Wij zoeken hiervoor, bij voorkeur, iemand die ervaring heeft in de Public Relations.

Mocht u interesse hebben in de bovengenoemde bestuursfunctie, laat dit dan even weten aan één van de overige bestuursleden zodat we op de eerstvolgende ledenvergadering uw kandidatuur aan de leden voor kunnen leggen.

De KGN heeft u nodig!

DualBoot, twee operating systems op één harde schijf

Minix is een aardig operating system, maar in de praktijk zijn er (nog?) te weinig applicaties om het als volwaardig stand-alone systeem te kunnen gebruiken. Op de meeste systemen zal Minix/PC broederlijk naast MS-DOS wonen. Wil men Minix gebruiken, dan zal de PC opnieuw geboot moeten worden. Op zich is dat niet zo erg, maar standaard Minix boot helaas niet van harddisk...

Er zijn gelukkig wel Minix-en in omloop die wél van harddisk booten. Zo heeft de NL-MUG ooit eens een versie gehad die zij heel trots de "Advanced Version" doopten. Verder zijn er Shoelace en MX-Boot (de laatste is verkrijgbaar op het BBS) die het beide mogelijk maken direct van de harde schijf te starten. Het grootste probleem van thuis-gemaakte oplossingen én de NL-MUG "advanced" versie is wel dat er slechts gekozen kan worden welk operating system men wil booten door met behulp van fdisk een andere partitie actief te maken. Hieronder volgt een oplossing voor dit vervelende probleem: DualBoot. DualBoot nestelt zich in de master bootrecord op de harddisk. Als de PC dan gestart wordt, verschijnt op het beeldscherm de vraag "Boot <M>inix or <D>OS?". De gebruiker kan dan een D intikken om MS-DOS te starten of een M om Minix te starten. Tikt de gebruiker helemaal niets in, dan wordt de actieve partitie geboot. Is er geen partitie actief, dan blijft DualBoot net zo lang wachten tot de gebruiker wél zijn keuze heeft gemaakt. Met een paar veranderingen in de source is het programma eenvoudig aan te passen aan andere operating systemen.

Vóór de installatie van DualBoot moeten er uiteraard backups gemaakt worden.

Installatie

Vóór de installatie van DualBoot moeten er uiteraard backups gemaakt worden. Verder moet DualBoot aan uw computersysteem aangepast worden. Hiervoor heeft u de source van DualBoot, een editor en masm/link/exe2bin of tasm en tlink nodig. Het aanpassen van de configuratie verloopt als volgt:

1. De tijd die het systeem wacht tot er een toets ingedrukt is wordt bepaald door middel van een eenvoudig lusje. De in de listing gegeven waarde is redelijk goed voor een 16 MHz. PC/AT. Verhoog het nummer in regel 6 voor snellere computers en verlaag dat nummer voor langzamere computers. Er valt niets zinnigs te zeggen over de waarde die voor uw computer het meest geschikt is; simpelweg uitproberen levert de beste oplossing...
2. Partitie #1 boot nu DOS. De letter die gegeven is in regel 7 is de letter die ingetikt moet worden voor DOS. Partitie #2 bevat in de huidige configuratie Minix. De keuze-letter voor partitie twee is gegeven in lijn 8. Deze letters kunnen beide naar eigen inzicht aangepast worden.
3. Er kunnen andere partities gekozen worden door het uitlezen van de partitietabellen in regels 277 en 283 aan te passen. Op deze twee regels staat "lea si,parttable[x]", alwaar de x de offset in de partitietabel aangeeft. Verander deze waarde in 0 voor partitie 1, 10h voor partitie 2, 20h voor partitie #3 of 30h voor partitie nummer 4.
4. Pas tenslotte de teksten in regels 370, 371 en 390 aan aan uw systeem.
5. Save de aangepaste versie en run tasm en tlink: "tasm dualboot,,;" en "tlink dualboot /t". Als alles goed gegaan is heeft u nu de volgende bestanden:

```
dualboot.asm   13182   30-05-91   9:09
dualboot.com    1792    4-01-92  22:23
dualboot.lst   32042    4-01-92  22:23
dualboot.map      99    4-01-92  22:23
dualboot.obj    1421    4-01-92  22:23
                48.536 bytes in 7 file(s)
```

Nu kan DualBoot zichzelf installeren. Tik hiervoor onder de DOS-prompt gewoon "dualboot" in, zonder parameters. Er verschijnt nu een copyright message. Als u DualBoot al eerder gedraaid hebt, verschijnt er dan een vraag of u de oude "savefile" wilt overschrijven. DualBoot schrijft namelijk de originele inhoud van het master bootrecord weg in een bestandje genaamd "DualBoot.Sav" in de root-directory. Als u hier een N intikt, wordt DualBoot niet geïnstalleerd en wordt de savefile niet overschreven.

Vervolgens wordt de inhoud van de master bootrecord overschreven door DualBoot. De partitie informatie wordt daarbij netjes gecopieerd in het nieuwe bootrecord. Kopieer voor de zekerheid de savefile even naar diskette! Mocht alles om wat voor reden dan ook fout gaan, dan kunt u met bijvoorbeeld de Norton utilities de originele partitieinformatie terug zetten op harde schijf! Als u nu opnieuw boot, moet er een copyright message in beeld verschijnen en de vraag "Boot <D>OS or <M>inix?".

Tenslotte

Met dit soort utilities kun je niet voorzichtig genoeg zijn. Maak backups voor je aan de slag gaat! Als u de zaak niet vertrouwt, kunt u natuurlijk uw systeem meenemen naar één van de clubdagen... meestal ben ik daar ook wel aanwezig en kan de installatie altijd begeleiden... Natuurlijk bent u er zich terdege van bewust dat ik geen enkele verantwoordelijkheid op me neem ten aanzien van de werking van DualBoot. Ik garandeer slechts dat het ruimte op papier inneemt...

Och ja, dat zou ik toch bijna vergeten... DualBoot staat natuurlijk ook op "The Ultimate", het BBS van de vereniging... zie de Minix support filearea!

Joost Voorhaar

```
; This program allows multiple operating systems to live on one harddisk
; and lets the user choose what operating system to boot today

; Program specific values

second          equ     5000h           ; Higher for faster machines
part1           equ     'D'             ; Partition 1 boots DOS
part2           equ     'M'             ; Partition 2 boots MINIX

; Commonly used equates

cr              equ     0dh
lf              equ     0ah
msdos           equ     21h

code            segment
                assume cs:code, ds:code
                org 100h

begin:          jmp main

copyright       db      'DB - DualBoot, Version 1.20',cr,lf
                db      'Copyright (c) J.Voorhaar, 1990-1991',cr,lf,lf
                db      'Did you read the docs and backed up your harddisk!?',cr,lf,lf,0

error1          db      'Cannot read master bootrecord from hardddisk #0...',cr,lf,lf,0
error2          db      'Error saving master bootrecord on savefile!',cr,lf,lf,0
error3          db      'Cannot write master bootrecord to hardddisk #0...',cr,lf,lf,0

promptmsg       db      'About to overwrite master bootrecord. Are you sure [y/n] ? '
                db      0
existsmsg       db      'The savefile already exists. Overwrite [y/n] ? ',0
exitmsg         db      lf,'Dualbootrecord successfully installed,',cr,lf
                db      'Press <Ctrl> + <Alt> + <Del> to reboot.',cr,lf,lf,0
noasw           db      'No',cr,lf,0
yesasw          db      'Yes',cr,lf,0
savename        db      'C:\DualBoot.Sav',0

main            proc near

                lea     bx,endprg + 1010h ; 4kB stackspace
                mov     cl,4
```

Fig. 1: sourcetekst van DUALBOOT.ASM

```
                shr     bx,cl
                mov     ah,4ah
                int     msdos               ; Set blocksize
                lea     sp,endprg + 1000h   ; Set stackpointer to new stack
                lea     si,copyright
                call    dispstr             ; Display copyright string
                mov     bp,3                ; 3 retries...
readboot :      push    bp
                mov     ax,0201h            ; Read 1 sector
                lea     bx,diskbuf          ; ES:BX to disk buffer
                mov     cx,1                ; Cylinder 0, sector 1
                mov     dx,80h              ; Harddisk #0, track 0
                int     13h
                pop     bp                  ; Recover retry counter
                jnc     saveboot            ; Read was successfull
                dec     bp
                jne     readboot            ; Retry
                lea     si,error1           ; Error message #1
                call    dispstr             ; Display message
                jmp     ret2dos             ; And terminate program
saveboot :      lea     si,diskbuf[1beh]    ; Source is original partition table
                lea     di,parttable        ; Destination is my master bootrecord
                mov     cx,20h
                cld
                rep     movsw               ; Copy partition info in my bootrecord
                mov     ah,4eh
                lea     dx,savename
                mov     cx,027h             ; Hidden, System, ReadOnly, Archive
                int     msdos               ; Find first
                jc      newfile             ; Not found, must be a new file
                lea     si,existsmsg
                call    dispstr             ; Prompt before overwriting
getkey :        xor     ax,ax
                int     16h                 ; Get key
                call    toupper             ; Convert to uppercase
                cmp     al,'Y'
                je      confirmed           ; User said "Yes"
                cmp     al,'N'
                jne     getkey              ; Illegal keypress, ignore
                lea     si,noasw
                call    dispstr             ; Display "No"
                jmp     ret2dos             ; And return to DOS
confirmed :     lea     si,yesasw
                call    dispstr             ; Say "Yes"
newfile :       mov     ah,3ch              ; Create handle
                lea     dx,savename
                xor     cx,cx
                int     msdos
                jc      save_err            ; Error during "create"
                push    ax                  ; Save handle# on stack
                mov     bx,ax
                mov     cx,512
                lea     dx,diskbuf
                mov     ah,40h
                int     msdos               ; Write the bootsector to file
                jc      save_err            ; Error during "write"
                mov     ah,3eh
```

```
                pop     bx
                int     msdos           ; Close the handle
                jnc     prompt          ; Oke, we may continue
save_err :      lea     si,error2       ; Error #2: Cannot save...
                call    dispstr
                jmp     ret2dos         ; And return to DOS
prompt :        lea     si,promptmsg
                call    dispstr         ; Prompt for continue
getansw :       xor     ah,ah
                int     16h             ; Get a key from keyboard
                call    toupper         ; Convert to uppercase
                cmp     al,'Y'
                je      overwrite       ; Yes, overwrite
                cmp     al,'N'
                jne     getansw         ; Illegal answer, retry
                lea     si,noasw
                call    dispstr         ; Display "No"
                jmp     ret2dos         ; And return to DOS
overwrite :     lea     si,yesasw
                call    dispstr         ; Display "Yes"
                mov     bp,3            ; 3 retries
writeboot :     push    bp
                mov     ax,0301h        ; Write 1 sector
                lea     bx,boot         ; ES:BX to new bootsector
                mov     cx,1            ; Cylinder 0, sector 1
                mov     dx,80h          ; Harddisk #0, track 0
                int     13h
                pop     bp              ; Recover retry counter
                jnc     telluser        ; Write was successful
                dec     bp
                jne     writeboot       ; Retry
                lea     si,error3       ; Error message #3
                call    dispstr         ; Display message
                jmp     ret2dos         ; And terminate program

telluser :      lea     si,exitmsg
                call    dispstr         ; Tell user we're done

ret2dos :       mov     ax,4c00h
                int     msdos           ; Terminate program

main            endp

                org 600h  ; Bootcode will be moved here!

; The bootcode will be loaded at 0000:7C00. One of the first things to do, is
; to move the code to address 0000:0600 so that the bootcode of the desired
; partition won't overwrite the masterbootcode.

boot            proc    near            ; Need near for JUMP-workaround

; Purpose       : Load and execute OS-specific bootcode
; Input         : None
; Output        : DS:SI points to active entry in partition table

                cli                     ; Disable interrupts please!
                xor     ax,ax
```

```
                mov     ss,ax
                mov     sp,7c00h        ; Stack begins at 0000:7c00h
                mov     si,sp
                mov     es,ax
                mov     ds,ax           ; Setup Extra Segment & Data Segment
                sti                     ; Interrupts back on
                cld                     ; Low to high address copy
                mov     di,0600h        ; The destination = 0000:0600
                mov     cx,0100h        ; Copy the whole bootmaster sector
                rep     movsw           ; Move the code to offset 0600

; The following line will generate an error message in TASM. It seems to be
; impossible to specify "FAR" or "NEAR" when *we* want it :-(

;               jmp     far continue    ; Jump to moved code

; So, here is the workaround (assembled "JUMP FAR"):

                db      0EAh            ; JuMP FAR
                dw      offset continue,0 ; To "continue"

continue:       lea     si,menu
                call    dispstr         ; Show menu

                mov     bx,10           ; Wait 10 seconds

waitmore :      push    bx
                mov     cx,second

waitkey :       push    cx
                mov     ah,1
                int     16h             ; Key pressed?
                je      contloop        ; No!

                xor     ah,ah
                int     16h             ; Remove key from buffer

                call    toupper         ; Convert answer to uppercase

                cmp     al,part1        ; Partition 1?
                je      bootpart1       ; Yes, boot it
                cmp     al,part2        ; Partition 2?
                je      bootpart2       ; Yes, boot it

contloop :      pop     cx              ; Recover inner loop counter
                loop    waitkey         ; Check again

outerloop :     pop     bx              ; Recover outer loop counter
                dec     bx
                jne     waitmore        ; Wait more seconds if necessary

                lea     si,parttable[0] ; Partition 0 info
                test    byte ptr [si],80h ; Active?
                jne     active          ; This partition is active

                lea     si,parttable[10h] ; Partition 1info
                test    byte ptr [si],80h ; Active?
```

```
                jne     active          ; This partition is active
                lea     si,parttable[20h] ; Partition 2 info
                test    byte ptr [si],80h ; Active?
                jne     active          ; This partition is active
                lea     si,parttable[30h] ; Partition 3 info
                test    byte ptr [si],80h ; Active?
                je      waitmore        ; No, wait longer!
active :        push    si
                lea     si,bootmsg0
                call    dispstr         ; Display general bootmessage
                pop     si
                jmp     genboot         ; Generic boot

bootpart1 :     lea     si,bootmsg1
                call    dispstr         ; Display bootstring #1
                lea     si,parttable[0] ; Set SI to partition #0
                jmp     genboot         ; Generic boot

bootpart2 :     lea     si,bootmsg2
                call    dispstr         ; Display bootstring #2
                lea     si,parttable[10h] ; Set SI to partition #1

genboot :       push    si

                mov     cx,2[si]        ; Get cylinder & sector info
                mov     dl,80h
                mov     dh,1[si]        ; Get head & disk info

                mov     bp,3            ; 3 retries

readpart :      push    bp
                push    cx
                push    dx
                mov     ax,0201h        ; Read 1 sector
                mov     bx,7c00h        ; At offset 7C00h
                int     13h             ; Read it
                pop     dx
                pop     cx              ; Recover cylinder & sector
                pop     bp              ; Recover retry counter

                jnc     boot_it         ; Read succeeded, boot partition
                dec     bp              ; No, retry
                jnz     readpart

                lea     si,noboot
                call    dispstr         ; Error reading bootrecord
                int     18h             ; Call ROM BASIC
boot_it :       pop     si              ; Recover pointer to partition info

; Once again, stupid MASM/TASM. We can't jump to a immediate value. So...
; we'll take the RET workaround:

                mov     ax,7c00h
                push    ax
                ret
boot            endp
```

```
dispstr     proc    near

; Purpose   : Display a string via BIOS
; Input     : DS:SI points to the string
; Output    : None

            push    ax
            push    si              ; Preserved used vars
            cld
dispstr1 :  lodsb                   ; Get character from string
            cmp     al,0            ; Is it the terminating zero?
            je      dispstr0        ; Yes, we're done
            mov     ah,0eh          ; TTY Write = = function 0eh
            int     10h             ; Display character
            jmp     short dispstr1  ; Handle next char
dispstr0 :  pop     si
            pop     ax              ; Restore the used regs
            ret                     ; And return to caller

dispstr     endp

toupper     proc    near

; Purpose   : convert character to uppercase
; Input     : AL = character
; Output    : AL = uppercase value of character

            cmp     al,'a'
            jb      isupper
            cmp     al,'z'
            ja      isupper
            sub     al,'a' - 'A'    ; Convert to uppercase
isupper :   ret
toupper     endp

bootmsg0    db      7,cr,lf,lf,'Booting active partition',cr,lf,lf,0
bootmsg1    db      'DOS',cr,lf,lf,0
bootmsg2    db      'Minix',cr,lf,lf,0
noboot      db      'Error reading operating system',cr,lf,0
menu        db      cr,'DualBoot, Version 1.20'
            db      ' - Copyright (c) J.Voorhaar, 1990-1991',cr,lf,lf
            db      'Boot <D>OS or <M>inix? ',0

            org     7beh            ; Start of partitiontable
parttable   db      16 dup(?)       ; Entry #1
            db      16 dup(?)       ; Entry #2
            db      16 dup(?)       ; Entry #3
            db      16 dup(?)       ; Entry #4
            db      55h,0aah        ; End of partitiontable

diskbuf     db      512 dup(?)      ; Disk buffer

endprg      label   near

code        ends
            end     begin
```

Methoden en technieken voor datacommunicatie (deel 10)

Inleiding

Dit is al weer de tiende aflevering van deze serie. Dat betekent dat we op de kop af twee jaar bezig zijn. Toen ik met de serie begon, had ik nooit gedacht dat ik zoveel over dit onderwerp zou kunnen schrijven; dat kon ik op dat moment eigenlijk ook niet. Tijdens het schrijven heb ik zelf ook het nodige bijgeleerd en de artikelen volgen zo grofweg mijn eigen ontwikkeling. Nu wil het toeval ook nog dat ik alles wat ik de afgelopen twee jaar over datacommunicatie heb bijgeleerd ook in mijn werk goed kan gebruiken zodat het schrijven van een dergelijke artikelenreeks voor mij eigenlijk alleen maar voordelen heeft.

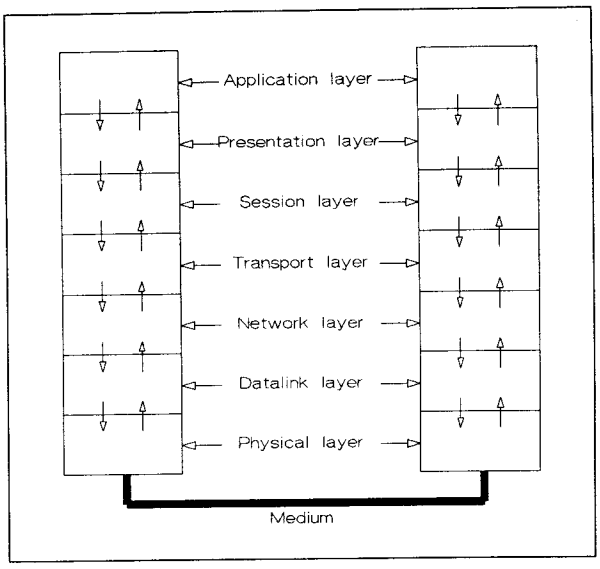


Fig. 1: het OSI referentiemodel

Misschien dat de vorige alinea andere mensen over kan halen ook eens een onderwerp bij de kop te pakken en dat eens helemaal uit te diepen. Zoals aangegeven, heeft het een aantal grote voordelen. Het enige nadeel dat ik zelf ondervind is de morele verplichting elke twee maanden weer een artikel op te hoesten maar dat werkt meteen ook weer als stok achter de deur om de studieboeken toch regelmatig ter hand te nemen.

Goed, terug naar de datacommunicatie. Vorige keer hebben we, na een bijdrage van Hugo van der Kooij, het gehad over het verbinden van netwerken met behulp van repeaters, bridges, routers en gateways. Verder is het protocol voor een computer-telefoonnet, X25, geïntroduceerd. In deze aflevering is wel een fout geslopen waarvoor ik mij als fysicus zeer schaam. De lichtsnelheid, één van de meest belangrijke natuurconstanten, staat verkeerd in het artikel. De lichtsnelheid is niet bij benadering 300.000 meter per seconde maar 300.000 kilometer !! per seconde; dat scheelt dus een factor duizend. Mijn excuses daarvoor.

In deze aflevering pakken we de draad weer op waar we hem twee afleveringen geleden hebben laten liggen.

Deze aflevering

In deze aflevering pakken we de draad weer op waar we hem twee afleveringen geleden hebben laten liggen. We hebben het in die aflevering gehad over de verschillen tussen CMSA/CD (ook wel Ethernet genoemd), Token Ring en Token bus. We zijn toen zo'n beetje blijven steken in het midden van de tweede laag van het OSI referentie model. In deze aflevering wordt de bespreking van deze laag, de zogenaamde datalink layer afgerond zodat we het in de volgende aflevering eens wat hogerop kunnen gaan zoeken.

Een korte herhaling over het OSI referentie model

De laatste afleveringen van deze serie volgen zo ongeveer de opbouw van het OSI-model en de beschrijving van Tanenbaum (ref. 1) hiervan. In figuur 1 is, waarschijnlijk geheel te overvloede, nogmaals dit model getekend.

Het referentie model is opgebouwd in zeven lagen waarbij elke laag aan de ene kant communiceert met dezelfde laag aan de andere kant. Zo communiceert de datalink layer aan de linker kant met de datalink layer aan de rechter kant. Hetzelfde geldt ook voor bijvoorbeeld de network layer en de transport layer. Verder geldt ook dat elke laag een aantal diensten levert aan de laag erboven waarbij je die diensten kunt beschouwen als een set subroutines met behulp waarmee een laag kan communiceren met zijn partner aan de overkant. Neem als voorbeeld de network layer. Deze communiceert met de network layer aan de overkant en maakt daarvoor gebruik van de diensten van de datalink layer. De datalink layer maakt weer gebruik van de diensten van de physical layer maar daarvan is de network layer niet op de hoogte kortom de diensten zijn alleen van belang voor de eerstvolgende laag en zijn transparant voor alle la-

gen erboven. Hetzelfde geldt ook voor de informatie die tussen de lagen in verticale richting wordt uitgewisseld. Als bijvoorbeeld de network layer een pakketje informatie samenstelt, dan zal de datalink layer dit pakketje inhoudelijk niet wijzigen. Het pakketje kan alleen uitgebreid worden met extra informatie aan het begin van het pakketje (een zogenaamde header) en eventueel een aantal bytes aan het einde van het pakketje (de zogenaamde trailer). Aan de inhoud van het pakketje wordt niets gewijzigd (met één uitzondering, maar daarover straks meer).

Eén van de grote voordelen van een dergelijke opbouw is het feit dat de wijze waarop het netwerk aangelegd is slechts bekend is op één of twee niveaus. Zo wordt binnen laag twee, de datalink layer, gedefinieerd of het netwerk gebruik maakt van Token Ring, Token Bus of CMSA/CD (Ethernet). Deze informatie wordt niet doorgegeven aan de network layer waardoor de netwerken vanaf laag 2 exact gelijk zijn. Een ander voorbeeld is de keuze van het medium. In de praktijk kun je kiezen voor diverse uitvoeringen van coax-kabel, twisted pairs of glasvezel. Deze informatie behoort toe aan de physical layer en niet aan de lagen erboven. Dit heeft tot gevolg dat een netwerkprodukt als Novell voor de diverse uitvoeringen aparte netwerkdrivers (zoals printerdrivers voor bijvoorbeeld WordPerfect) levert. Door een bepaalde netwerkdriver te installeren wordt gekozen voor een bepaalde netwerk-uitvoering. De rest van het geheel blijft, net als bij WordPerfect, precies hetzelfde, onafhankelijk van de gebruikte netwerkdriver resp. printerdriver.

Om het verhaal op dit punt volledig te maken nog even het verschil tussen X25 en de IEEE 802 netwerken. X25 is geen Local Area Network meer terwijl de IEEE 802 netwerken dat wel zijn. Verder zijn in X25 de adresseringsmogelijkheden, ten opzichte van IEEE 802 sterk uitgebreid. Toch is een X25 netwerk ook een OSI netwerk waarbij de verschillen ten opzichte van de IEEE 802 netwerken echter op laag 3, de network layer, liggen.

Goed, de voorlaatste aflevering zijn we zo ongeveer blijven steken op het punt waar de drie uitvoeringen van IEEE 802 bij elkaar komen. Dit is in het midden van de Datalink layer op het punt waar de zogenaamde Medium Access Control sublayer overgaat in de Logical Link Control sublayer. Het lijkt mij zinvol hier de draad maar weer op te pakken.

Bij een unacknowledged service gebeurt dit niet; de zender doet zijn uiterste best het packet goed over te sturen.

Connectionless <-> connection oriented services

Eén van de eerste zaken die we nog moeten behandelen is het verschil tussen een zogenaamde "connectionless service" en een "connection oriented service". Bij een connectionless service kan de afzender altijd packets naar de ontvanger sturen. Bij een connection oriented service moet er eerst een verbinding "connection" tussen de twee stations aangebracht worden. Dit laatste is goed vergelijkbaar met een telefoonnetwerk. Als ik informatie via de telefoon naar een ander wil overbrengen, dan moet ik eerst een verbinding tussen mij en de ander tot stand brengen of te wel, ik moet de ander opbellen. Een voorbeeld van een connectionless service is bijvoorbeeld een bulletin board. Als ik iemand een bericht wil sturen, dan kan ik dat bericht op het bulletin board achterlaten waarna de ontvanger het bericht kan lezen. In dat geval is er dus geen rechtstreekse verbinding tussen mij en de ontvanger.

Acknowledged <-> unacknowledged services

Een zaak die nauw samenhangt met het vorige is de acknowledged service versus de unacknowledged service. Bij een acknowledged service laat de ontvanger, na ontvangst van een packet, weten dat het packet goed aangekomen is. Bij een unacknowledged service gebeurt dit niet; de zender doet zijn uiterste best het packet goed over te sturen, maar een vorm van terugkoppeling dat een bericht goed is aangekomen is niet aanwezig. Hoewel de combinatie connectionless en acknowledged technisch wel mogelijk is, wordt ze meestal niet gebruikt. Als er gebruik gemaakt wordt van een acknowledged service, dan zal dit in de praktijk altijd gepaard gaan met een connection oriented service.

Voor het oversturen van een acknowledge zijn een aantal technieken beschikbaar. In de eerste plaats kan de ontvanger na ontvangst van een packet een berichtje terugsturen met als inhoud "Het vorige packet is goed ontvangen". Dit betekent dat de zender pas het volgende packet mag sturen nadat hij de bevestiging van het vorige ontvangen heeft. Een dergelijke manier van werken maakt niet zo'n efficiënt gebruik van het communicatiekanaal; de afzender moet iedere keer zijn transmissie onderbreken en wachten tot het packet door de ontvanger is ingelezen en verwerkt en totdat de acknowledge weer terug ontvangen is. Beter is een protocol waarbij er bijvoorbeeld vier packets onderweg mogen zijn. De zender mag dan maximaal vier packets sturen voor-

dat hij op een acknowledge voor het eerste packet moet wachten. Terwijl de zender het tweede en volgende packet stuurt, kan de ontvanger het eerste packet controleren, opslaan en een acknowledge sturen. Een dergelijk protocol heet een "Sliding Windows" protocol omdat er altijd enkele packets in het window kunnen zitten waarvoor de afzender de bevestiging nog niet heeft ontvangen. In een dergelijk protocol is het meestal ook zo dat een acknowledge niet alleen betekent dat het packet waarvoor de acknowledge gegeven wordt correct ontvangen is maar ook alle voorgaande. In de praktijk betekent dit bijvoorbeeld het volgende:

- De afzender stuurt packet 01 gevolgd door 02, 03 en 04.
- Omdat packet 01 en 02 correct ontvangen worden, stuurt de ontvanger een acknowledge voor packet 01 en 02. Packet 03 wordt niet ontvangen zodat hiervoor geen acknowledge gestuurd wordt. De overige packets worden wel goed ontvangen en worden tijdelijk in de datalink layer van de ontvanger opgeslagen.
- De zender ontvangt de acknowledge voor packet 01 en 02 en stuurt daarom packet 05 en 06. Na verloop van tijd (time-out) ontdekt hij dat de acknowledge voor packet 03 niet komt. Hij stuurt daarom packet 03 nogmaals.
- De ontvanger ontvangt packet 03 correct en stuurt een acknowledge voor ... packet 06!
- De zender ontvangt de acknowledge voor packet 06 en weet dat nu alle packets 01 t/m 06 correct ontvangen zijn. Hij gaat nu verder met het versturen van packet 07 etc.

De ontvanger ontvangt packet 03 correct en stuurt een acknowledge voor ... packet 06!

In het bovenstaande voorbeeld worden alle packets, na packet 03 in de datalink layer van de ontvanger gebufferd. Er zijn ook uitvoeringen van het protocol waarin de ontvanger alle packets na een fout packet eenvoudig negeert. In dat geval moet de zender dus niet alleen packet 03 maar ook packets 04 t/m 06 herhalen.

De voordelen van Sliding Windows zijn overduidelijk. Mensen die wel eens files van een bulletin board halen, weten waarschijnlijk wel dat het ZMODEM protocol veel sneller is dan bijvoorbeeld Kermit of XMODEM. Dit komt omdat ZMODEM sliding windows heeft en Kermit en XMODEM na het versturen van een packet eerst de acknowledge afwachten. De verschillen zijn enorm: bij ZMODEM haal je bij 2400 bps ruim 200 tekens per seconde, bij Kermit ongeveer 90.

Behalve de term Sliding Windows komt ook de term "Piggybacking" nog wel eens voor. Dit heeft ook te maken met de datalink layer en betekent het volgende: In het voorgaande voorbeeld wordt de acknowledge als apart bericht van de ontvanger naar de zender gestuurd. Dit geeft vrij veel overhead omdat in de acknowledge alleen het nummer van het packet hoeft worden doorgegeven en er daarvoor een compleet packet met een capaciteit van enkele kilobytes verstuurd wordt. Om deze reden wordt er in een packet vaak ruimte vrijgehouden om de acknowledge op te nemen. Als er dan een packet met informatie van ontvanger naar zender gaat, dan kan de acknowledge van een packet die in de andere richting gegaan is gratis meerijden (op de rug = back, van het varken = pig). Hierbij mag de ontvanger uiteraard niet te lang wachten met het terugsturen van de acknowledge omdat anders de zender denkt dat het packet niet is aangekomen waarna hij het packet nogmaals zal zenden. Mocht er niet op tijd een packet in tegenovergestelde richting gaan, dan wordt er uiteraard een packet samengesteld met alleen de acknowledge. Duidelijk is dat piggybacking alleen werkt als er tussen twee stations voortdurend informatie in twee richtingen over de lijn gaat. Of dit het geval is hangt uiteraard sterk af van de manier waarop het netwerk ontworpen en gebruikt wordt.

Bit/Character stuffing

In aflevering 5 van deze serie heb ik het al even gehad over een protocol waarbij elk packet wordt afgesloten met de ASCII tekens DLE EXT. Verder start elk packet met het teken STX. Mocht het nu voorkomen dat er midden in het packet ook een DLE voorkomt, dan wordt dit teken voorafgegaan door nog een DLE zodat de ontvanger aan de hand van de combinatie DLE DLE weet dat hier één enkele DLE bedoeld wordt. Deze techniek wordt character stuffing genoemd. De protocollen die in een netwerk gebruikt worden zijn over het algemeen niet byte-georiënteerd maar bit-georiënteerd. Toch komt daar deze problematiek ook voor. Het bitpatroon 01111110 wordt namelijk nog al eens gebruikt om de start en het einde van een frame (in de datalink layer wordt meestal over een frame gesproken in plaats van over een packet) aan te geven. Uiteraard kan deze bitcombinatie ook midden in het frame voorkomen. Om deze reden wordt door de datalink layer na vijf opéénvolgende 1 bits automatisch een 0 bit toegevoegd die er uiteraard door de datalink layer van de ontvanger weer uitgehaald

wordt. Op deze manier kun je elk bitpatroon over een netwerk sturen zonder dat je je druk hoeft te maken over het al dan niet voorkomen van bepaalde bitpatronen.

Taken en diensten van de datalink layer

Binnen het OSI-model heeft de datalink layer van de ontvanger als taak de ontvangen frames in de juiste volgorde en zonder fouten door te geven aan de network layer. De datalink layer van de zender moet er voor zorgen dat de packets die hij (zij?) van de network layer zodanig aan de physical layer aan te bieden dat ze zonder problemen naar de datalink layer van de ontvanger gestuurd kunnen worden. Vooral de eerste eis betekent dat, bij sliding windows, de frames in de datalink layer gebufferd moeten worden omdat de frames strikt in volgorde aan de network layer moeten worden doorgegeven.

In het onderstaande overzicht worden de diensten die in een IEEE 802 protocol door de datalink layer geboden worden opgesomd.

Unacknowledged connectionless service:

1: L_DATA.request(local_address, remote_address, l_sdu, service_class):
2: L_DATA.indication(local_address, remote_address, l_sdu, service_class):

Connection oriented service:

3: L_CONNECT.request(local_address, remote_address, service_class):
4: L_CONNECT.indication(local_address, remote_address, status, service_class):
5: L_CONNECT.response(local_address, remote_address, service_class):
6: L_CONNECT.confirm(local_address, remote_address, status, service_class):

7: L_DISCONNECT.request(local_address, remote_address):
8: L_DISCONNECT.indication(local_address, remote_address, reason):
9: L_DISCONNECT.response(local_address, remote_address):
10: L_DISCONNECT.confirm(local_address, remote_address, status):

11: L_DATA_CONNECT.request(local_address, remote_address, l_sdu):
12: L_DATA_CONNECT.indication(local_address, remote_address, l_sdu):
13: L_DATA_CONNECT.response(local_address, remote_address):
14: L_DATA_CONNECT.confirm(local_address, remote_address, status):

15: L_RESET.request(local_address, remote_address):
16: L_RESET.indication(local_address, remote_address, reason):
17: L_RESET.response(local_address, remote_address):
18: L_RESET.confirm(local_address, remote_address, status):

19: L_CONNECTION_FLOWCONTROL.request(local_address, remote_address, amount):
20: L_CONNECTION_FLOWCONTROL.indication(local_address, remote_address, amount)

Zoals uit het voorgaande blijkt, zijn de services opgebouwd uit een viertal zogenaamde primitieven. Dit zijn "request", "indication", "response" en "confirm". Met behulp van een request geeft de network layer een verzoek aan de datalink layer. Dit verzoek kan onder andere het verzoek zijn om een brokje informatie (l_sdu of link service data unit) naar het remote adres te sturen. Met behulp van de indication geeft de datalink layer aan de network layer aan dat er een zogenaamd "event" heeft plaatsgevonden, bijvoorbeeld dat de verbinding verbroken is. Met behulp van een response kan de network layer van de ontvanger eventueel reageren op een indication van de afzender. Komt een dergelijk response binnen, dan wordt ze als confirm doorgegeven aan de network layer. Dus:

1: De network layer van station A stuurt een request voor station B naar de datalink layer. De datalink layer zorgt ervoor dat dit verzoek ongeschonden bij de datalink layer van station B aankomt.
2: De datalink layer van station B ontvangt de request en stuurt een indication naar de network layer van station B. De network layer of één van de hogere lagen van station B beslist of er een respons moet komen en stuurt, indien dit zo is, deze respons naar de datalink layer van station B.
3: De respons die de datalink layer van station B ontvangt van zijn network layer wordt doorgestuurd naar station A.
4: De datalink layer van station A ontvangt de response en stuurt deze in de vorm van een confirm naar de network layer van station A.

Merk op dat de beschreven afhandeling met response en confirm op een ander niveau plaatsvindt dan de afhandeling van een acknowledged service. In het laatste geval is dit iets wat volledig intern in de datalink layer plaatsvindt terwijl de response/confirm uit het voorbeeld zich op een hoger niveau afspeelt. Op

deze manier kan men vrij kiezen op welk niveau men een vorm van terugkoppeling legt, in de datalink layer of in één van de andere lagen.

Van de parameters in de opgesomde primitieven zijn er enkele die nog wat uitleg behoeven. De informatie wordt overgedragen door middel van de parameter l_sdu en de adressen door middel van local_address en remote_address. De service_class geeft de mogelijkheid een prioriteit (als de lagere lagen dat ondersteunen) mee te geven. De parameters status geeft aan wat er gebeurd is en de parameter reason waarom iets gebeurd is.

De indeling in hoofdgroepen is ook vrijwel vanzelfsprekend. Uiteraard zijn er primitieven voor het versturen van informatie (L_DATA resp. L_DATA_CONNECT), en het openen resp. sluiten van een verbinding (L_CONNECT, L_DISCONNECT). Met behulp van de L_RESET groep kan aangegeven worden dat de verbinding opnieuw gestart moet worden, bijvoorbeeld als er teveel fouten optreden. Met behulp van de L_CONNECTION_FLOWCONTROL groep kunnen de network layer en de datalink layer met elkaar afspreken hoeveel bytes uitgewisseld mogen worden. Op deze manier kan de network layer de datalink layer verzoeken hem niet over zijn toeren te jagen en andersom.

Afsluiting

De inhoud van dit artikel is vrij pittig. Toch hoop ik dat als basis-idee is blijven hangen dat de datalink layer toch een vrij eenvoudig interface heeft naar de network layer. Tenslotte is een interface met slechts 20 mogelijke subroutine calls niet echt iets om je druk over te maken. Uiteraard gebeurt er, op basis van de calls, heel veel binnen de datalink layer maar daar hebben de hogere lagen niets mee te maken.

Het mooie van een OSI-netwerk is het feit dat de interfaces tussen alle lagen gestandaardiseerd zijn. Dit betekent dat de eenvoud waarmee de network layer gebruik kan maken van de diensten van de datalink layer zich doorzet naar de interfaces tussen de overige lagen. Aangezien we in de volgende aflevering de network layer bij de kop pakken, zal dat in dat artikel nogmaals naar voren komen.

Get van Opbroek

Literatuur

Dit artikel is voornamelijk gebaseerd op:
1: Andrew S. Tanenbaum: Computer Networks; Prentice-Hall.

Van de bestuurstafel (1)

Zoals bekend zouden er op 1 januari 1992 drie bestuursleden aftreden. Daarom was er op de agenda van de algemene ledenvergadering een bestuursverkiezing. Hiervoor was door het bestuur Joost Voorhaar bereid gevonden om zich kandidaat te stellen. Verder waren er geen andere aanmeldingen. Omdat een vereniging niet zonder bestuursleden kan heb ik staande de vergadering aangeboden om ook een steentje bij te dragen.

Tijdens de daarop volgende bestuursvergadering zijn de taken opnieuw verdeeld. Jan Derksen wilde wel eens wat anders gaan doen en hij zal zich bezig gaan houden in de projectgroep voor DOS65. Zijn taak is overgenomen door Gert van Opbroek die ook eens wat anders wil doen. De taak van Gert is overgenomen door Joost Voorhaar. Er was en is in ons midden een man die gek is op geld tellen, dat is dus Jacques Banser. Dan is er nog een taak over en dat is die van voorzitter, die is toebedeeld aan Tonny Schäffer. Die laatste is degene die deze letters aan het papier toevertrouwt en zal bij jullie niet zo bekend zijn. Daarom wil ik van deze gelegenheid gebruik maken om mijzelf even voor te stellen.

Ze hebben ooit geprobeerd mij enige educatie bij te brengen.

Nou daar gaat ie dan: ik ben een nog zeer jonge man van slechts 44 jaar, ongehuwd, doch gebonden aan een dochter op een zeer gevaarlijke leeftijd. Jullie zullen begrijpen dat ik haar tijdens de vergaderingen en bijeenkomsten dan ook maar niet meeneem. Ze hebben ooit geprobeerd mij enige educatie bij te brengen, helaas is dat maar ten dele gelukt. Er is door een leger van docenten geprobeerd mij door de LTS, MTS en HTS te loodsen. Tussen die laatste twee heb ik ook nog een poging ondernomen om te voorkomen dat u door de vijand zou worden verrast. Ze zijn niet gekomen dus dat is aardig gelukt. Toen kwam de periode van huisje, boompje, beestje. Mijn ouders waren inmiddels beide overleden en ik wilde mijn dochter niet bij ooms en tantes laten opgroeien. Bovendien liet mijn gezondheid te wensen over. Ik heb toen ons familiebedrijf, een vijftal kermisattracties, aan de kant gedaan. Daarna heb ik nog wel mijn propaedeuse rechten gedaan aan de OU. Op verzoek van mijn huidige werkgever, de gemeente Enschede, heb ik de AMBI-modulen I1, HE1 en 2 en HS4 met redelijk gevolg doorlopen. Mijn functie is daar AOC-leider/systeembeheerder. Het eerste deel in die functie betekent dat ik onderzoek doe bij personen naar hun aanleg, capaciteit, fysieke- en psychische mogelijkheden in automatiseringsfuncties. Een en ander binnen de sociale werkvoorziening, de gemeentelijke diensten en overige bedrijven. Wat systeembeheer is hoef ik hier waarschijnlijk niet uit te leggen. Dat systeem is overigens niet zo groot: een netwerk met 4 terminals en een 5 tal standalone PC's.

Het is al weer bijna 20 jaar geleden dat ik kennis maakte met het fenomeen rekentuig (ik vind dat toch wel een komische vertaling, vooral de laatste lettergreep). Af en toe zit ik nog wel eens heel nostalgisch te kijken naar die TRS-80 Model II. Dat ding had 4 keer zoveel geheugen als zijn voorganger Model I. Tjonge, tjonge wat een geheugen zeg, wel 16KB! Het kon niet op. Na verloop van tijd toch maar eens een stel chippies er bij op gesoldeerd, hallo 32 KB geheugen. Toen nog een expansion-interface, bij de vereniging in onderdelen te koop voor een prikkie, samen gesoldeerd een diskdrive van maar liefst 180 KB er aan. Ja, en nu staat er dan al weer een aantal jaren een XT met 1.5 meg geheugen, een paar diskdrives, twee 20 MB harddisks en een 2400 baud modem. Met als besturing MS-DOS 3.3 en Minix 1.5.

Verder heb ik m'n vrije tijd opgevuld met bestuursfuncties in een buurthuis, bouwcommissie in een renovatiewijk, bewonerscommissie en last but not least 4 jaar als voorzitter van de grootste wijk van Enschede. De laatste 2 jaar heb ik geen bestuurlijke functies verricht, ben dus een beetje uitgerust en heb er weer behoorlijk zin in. Van mijn kant spreek ik dan ook de hoop uit dat we een aantal goede jaren tegemoet gaan met de KGN en dat ik daar het mijne toe kan bijdragen.

Namens het bestuur wil ik de scheidende bestuursleden, Mick Achterberg, Ton Smits en Nico de Vries bedanken voor hun bestuurlijke bijdragen van de afgelopen tijd. Wij wensen ze verder veel succes toe en hopen ze nog regelmatig op de bijeenkomsten te mogen zien.

Als laatste wil het bestuur jullie allemaal een voorspoedig 1992 toewensen. Het liefst zouden we dat willen benadrukken door jullie allemaal de hand te schudden op de volgende bijeenkomst in Assendelft.

De voorzitter in oprichting

Van de bestuurstafel (2)/DOS65 3.0

Daar zit je dan. Als ex-voorzitter. Want nu ik dit stukje schrijf, heeft de club inmiddels een nieuwe voorzitter: Tonny Schäffer. De twee jaar zaten er al weer op. En ik wilde wel eens wat meer tijd voor datgene waar het om draait in ons clubje: hobby. Vandaar mijn vertrek uit het bestuur, en vandaar dat ex-voorzitter. Zoals dat in de statuten staat vermeld, heeft op de afgelopen bestuursvergadering de taakverdeling tussen de bestuursleden plaatsgevonden. Dat ging, zoals gebruikelijk, in goed overleg. Een aantal feiten lagen al enige tijd vast: Gert van Opbroek wilde eens wat anders dan redacteur spelen. En dat Joost Voorhaar waarschijnlijk in het bestuur zou worden gekozen. Zodat er al het een en ander bedacht was. Welnu, Gert van Opbroek heeft nu de taak van secretaris op zich genomen. Tonny Schäffer mag de hele kar trekken. Ik kan u verzekeren: dat komt wel goed, want dat heeft hij bij andere clubs ook al eens gedaan. Jacques Banser blijft doen wat hij nu ook al doet: op de centjes passen, en de grote harde schijven van het BBS vol zien te krijgen en vooral ook heel zien te houden. En Joost Voorhaar is de kersverse redacteur. Niet zo'n vreemde keuze: Joost heeft zich al meer zeer uitgebreid met het blad bemoeid, want hij is er de oorzaak van het blad er zo uitziet als het er uitziet en dat ondergetekende eens per twee maanden uitgebreid met een DTP-pakket in de weer moet. Kat opsluiten, muis tevoorschijn halen...

Hobby dus. Zoals in een eerdere "Van de bestuurstafel" al eens aangekondigd overwoog ik mij te gaan bezighouden met DOS65 3.0 onder het motto: iedereen heeft het erover, maar niemand begint ermee. Toen dat in de ledenvergadering in Almelo opnieuw werd geventileerd hebben we dat geweten: de vergadering was nog niet afgelopen, of er stonden een aantal clubleden die duidelijk last hadden van teveel zin in een nieuw project om zich te melden om ook aan DOS65 3.00 te gaan sleutelen. Met als gevolg dat we er een werkgroep bij hebben: de DOS65 3.0 werkgroep.

Dat clubje bestaat uit Henk Speksnijder, Jaap Prenger, Antoine Megens, Jan Derksen, en ondergetekende. We hebben al eens vergaderd, om de historie eens boven water te krijgen, en een plan de campagne te maken. Dat plan is er inmiddels, en er zijn ideeen genoeg.

Er stonden een aantal clubleden die duidelijk last hadden van teveel zin in een nieuw project.

Als eerste gaan we het al bestaande ontwerp voor de geheugenkaart realiseren, terwijl er ook nagedacht zal gaan worden over een nieuwe geheugenkaart met dynamische RAM erop. Een nieuwe geheugenkaart is namelijk voorwaarde om DOS65 3.0, dat multitasking is, te kunnen draaien. Bovendien kunt u met die nieuwe kaart ook al terecht in uw bestaande DOS65 systeem: namelijk als RAMdisk-kaart. Omdat de kaart SRAMs bevat die byte-gewijs georganiseerd zijn, is het mogelijk de kaart net zo vol te prikken als uw portemonnee dat toelaat: hij werkt al met slechts 1 RAMmetje erop, en met 8 stuks wordt de RAMdisk alleen maar groter. Om die manier hopen we een soort groeipad te kunnen presenteren, waardoor er stap voor stap naar DOS65 3.0 toegewerkt zal kunnen worden. Op die manier blijft het betaalbaar en hoeft de oude machine niet eerst tot de grond toe te worden afgebroken.

Verdere plannen behelzen een SCSI harddisk controller (4 jaar geleden een onmogelijke en vooral dure oplossing, nu zijn SCSI drives net zo duur als hun MFM-broertjes) en een nieuwe CPU kaart met misschien een 65816 erop. De huidige CPU-kaart kan namelijk geen DMA aan. Kortom, er ligt werk genoeg. Heeft u nog ideeen die bij de vorige DOS65 versie niet gerealiseerd werden, maar toch de moeite waard zijn? Laat het eens aan de werkgroep weten! En meedoen mag natuurlijk ook.

Het einde van de pagina komt weer een beetje in zicht. Ik heb aan het begin verzuimd u alvast een uitbundig 1992 toe te wensen, dus bij deze. Dat het nieuwe jaar brengen moge wat u ervan verwacht, zoals gezondheid en voorspoed. Inmiddels is het roer overgegeven aan Tonny Schäffer. Ik wens hem net zo veel plezier toe als ik gehad heb bij de bemoeienissen met de club. Het bestuur heeft weer wat vers bloed in de aderen, en dat is niet slecht: een nieuwe lente, een nieuw geluid nietwaar?

Tot op een bijeenkomst misschien, uw mede-clublid,

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: 053-328506 of 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 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.

T. 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 μP Kenner)
Jekerstraat 23
7523 VP Enschede
Telefoon 053-333483

Jan D.J. Derksen (DOS65 coordinator)
C.P. Soeteliefstraat 41
1785 CC Den Helder
Telefoon 02230-35002

Geert Stappers
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