De µP Kenner

Veertiende jaargang nr. 1
Februari 1990
65

In dit nummer o.a.:

Object georienteerd programmeren
Het Basic I/O System
Communicatie met randapparatuur
Een RAMdisk voor DOS-65
Transputers deel 2

Inhoudsopgave

De μP Kenner
Nummer 65, februari '90
Verschijnt 5 maal per jaar
Oplage: 200 stuks
Druk: FEBO Offset, Enschede

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

Eindredactie:
Gert van Opbroek

Vormgeving:
Joost Voorhaar

Redactieadres:
Gert van Opbroek,
Bateweg 60
2481 AN Woubrugge

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.

Copy voor het blad dient bij voorkeur van de leden afkomstig te zijn. Deze copy kan in papier-, maar liever in machine-leesbare vorm opgestuurd worden aan het redactieadres. Copy 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

Deze uitgave van de μP Kenner is de eerste uitgave van de veertiende jaargang. Ja de veertiende jaargang! De KIM Gebruikersclub Nederland bestaat al ruim dertien jaar en is daarmee de oudste hobbycomputerclub in Nederland. Vorig jaar waren we bezig met de dertiende jaargang van het blad en dat was te merken ook. In de eerste plaats kwam het blad vijf keer uit in plaats van zes keer zoals de laatste jaren gebruikelijk was. In de tweede plaats hebben we vorig jaar een tweetal ledenvergaderingen gehad met als doel de opheffing van de club. Welnu, de dertiende jaargang is voorbij en we beginnen vol goede moed aan een nieuwe jaargang. Ook in dit jaar zal het blad vijf keer uitkomen. Het blijkt namelijk dat door de meeste mensen de computerhobby alleen gedurende de wintermaanden bedreven wordt en dat de clubactiviteiten 's zomers op een vrij laag pitje staan. Dat kan de redactie ook merken aan de hoeveelheid kopij die binnenkomt. De stroom droogt dan vrijwel geheel op en het is zodoende zeer moeilijk in juni/juli een kwalitatief goed blad samen te stellen.

Veranderingen.

In de laatste uitgave van vorig jaar is het al aangekondigd, het blad zou gaan veranderen. Op cover van de voorgaande twee jaargangen prijkte vol trots een foto van de floppy-disk kaart voor DOS-65, tezamen met een door Rinus Vleesch Dubois gemaakte aquarel van een KIM. Het blad waar u nu in aan het lezen bent ziet er totaal anders uit. Dat komt omdat het cover van de vorige jaargang toch iets te prijzig was en verder vonden we het zo langzamerhand weer eens tijd worden voor een nieuw uiterlijk. Het is de bedoeling dat dit cover in ieder geval deze jaargang gebruikt wordt waarbij wel iedere keer een andere (zwart-wit) foto of tekening gebruikt wordt. Het cover is ontworpen door Joost Voorhaar, een zeer actief mede-redactielid die niet alleen een actief schrijver is, maar ook nog eens de hele layout van het blad verzorgt. Het bestuur en de redactie hoopt dat de nieuwe vormgeving die met het nieuwe cover compleet geworden is u aan kan spreken.

Ook de inhoud van het blad is iets veranderd. In de eerste plaats worden bepaalde vaste rubrieken op een nieuwe vaste plaats afgedrukt. Dit verhaal leest u voortaan op pagina 4, de achterkant van de inhoudsopgave. Op pagina 5 staat voortaan altijd de uitnodiging voor de clubbijeenkomst. De informatiepagina en het verhaaltje van de voorzitter zijn verhuisd naar de laatste en de voorlaatste pagina van het binnenwerk. De drie vrije zijden van het cover zijn gereserveerd voor eventuele adverteerders.

Ook de inhoud van het blad zal iets veranderen. We willen namelijk binnen de club een beetje een beleid gaan voeren waarbij we, zoals één van de bestuursleden het uitdrukt, ons presenteren als een club van mensen die van hun eigen systeem precies weten wat er in de behuizing zit en weten hoe dat zo ongeveer (of precies!) werkt en die en passant ook nog even een flinke hoeveelheid geheugen in het systeem van de buurman bijplaatsen (en deze ook nog lid maken?). De inhoud van het blad zal hier voor een belangrijk deel bij gaan aansluiten. We zullen veel meer elementaire artikelen over computersystemen randapparatuur en chips gaan plaatsen. Deze artikelen zullen dan niet alleen over DOS-65 of PC gaan maar ook over andere systemen. Voorbeelden hiervan zijn de reeds gepubliceerde verhalen van Nico de Vries over MS-DOS, van Bram de Bruine over datacommunicatie en van Joost Voorhaar over transputers. Natuurlijk zullen we, net als we altijd al gedaan hebben, ook hardware blijven publiceren en voorlopig blijven we zowel DOS-65 als EC-65(k) ondersteunen. Verder blijft er in het blad ruimte beschikbaar voor software, voor welk systeem dan ook. Alleen ben ik van mening dat listings die langer zijn dan zo'n 10 pagina's eigenlijk te lang zijn omdat er bij dergelijke lange listings een te eenzijdig blad ontstaat.

Eén van de dingen die we graag in het blad zouden willen hebben, is een vaste brievenrubriek. Hierin kan de lezer zijn op- en aanmerkingen over de inhoud van het blad kwijt maar bijvoorbeeld ook zijn technische en algemene vragen. Ik wil u dus allen oproepen van deze service gebruik te maken.

Ledental.

Zoals u wel zult begrijpen, is het voor de club van groot belang dat we voldoende leden hebben. Als we te weinig leden hebben, komt er te weinig geld binnen en hebben we onvoldoende middelen om bijvoorbeeld het blad uit te kunnen geven. We willen dit jaar, met het nieuwe bestuur, een serieuze poging wagen het ledental weer te vergroten. Verder kunt ook u iets voor ons doen. In de eerste plaats is dat natuurlijk een actieve ledenwerving. Verder stellen we het zeer op prijs als u uw eventuele ideeën voor uitbreiding van het ledental bij ons bekend wilt maken.

Tenslotte wens ik u veel plezier met het blad en uw computerhobby,

Gert van Opbroek.

Uitnodiging clubbijeenkomst

Datum: 17 maart 1990
Lokatie: gebouw 't Kruispunt
Slachthuisstraat 22
5664 EP GELDROP
tel: 040-857527

Entreeprijs: *f*10,=

Routebeschrijving

Trein:

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

Auto:

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

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

Programma:

9:30 Zaal open met koffie

10:15 Opening

10:30 Demonstratie door A. vd Hombergh van CP/M op de EC-65k, Voordracht door T. Smits over de grafische kaart van Elektuur met 9365 processor en demonstratie van grafische software op de EC-65.

11:30 Forum en markt

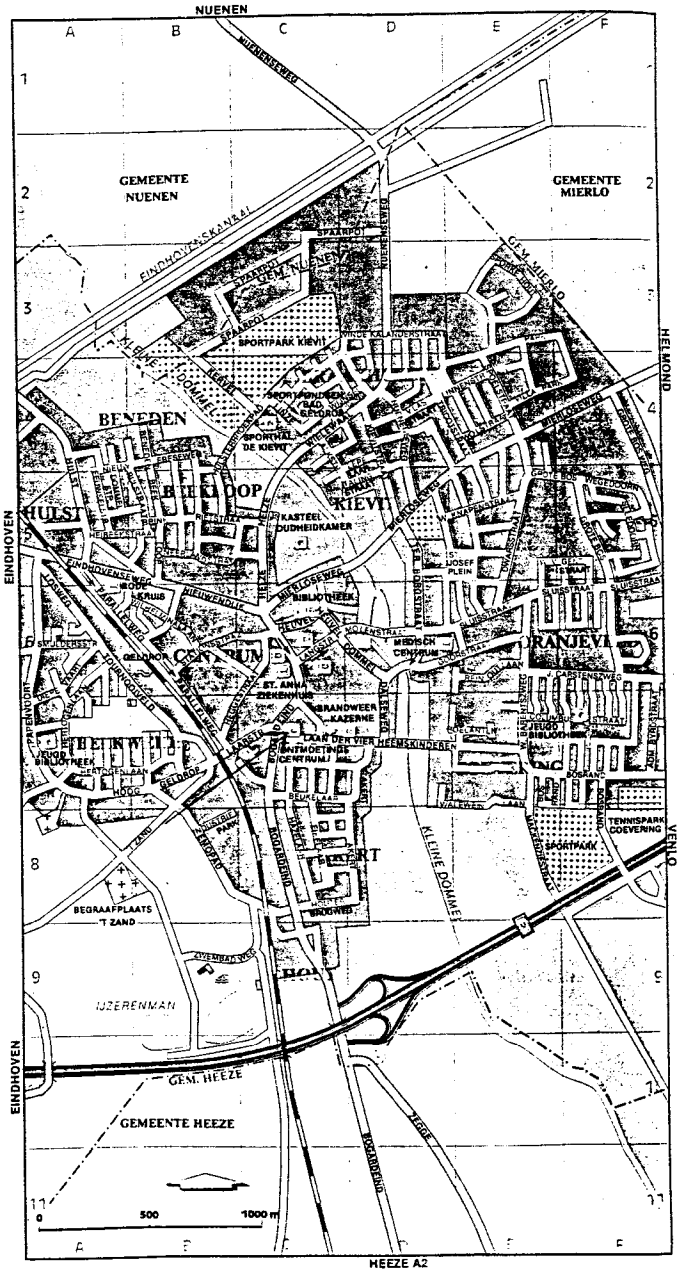
12:00 Lunchpauze

Aansluitend het informele gedeelte bedoeld om kennis, ervaring Public Domain en eigen ontwikkelde software uit te wisselen met uw medeleden. Breng daarom ook uw systeem mee.

17:00 Sluiting.

Attentie

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



Transputers II

Bouwstenen van de toekomst

Vorige keer hebben we heel erg globaal naar de transputer kunnen kijken. Deze aflevering gaat dieper in op de fysieke aspecten van deze opvallende bouwsteen. De processor, de links en de event logica wordt aan een nader onderzoek onderworpen. Om één en ander wat beter in de herinnering te halen heb ik hier het schematisch overzicht van de T414 nog maar eens herhaald.

We beginnen links boven in de hoek, bij de processor. Zoals ik vorige keer al schreef bestaat de processor grofweg uit een zestal registers. Dat zijn de instruction-pointer, de workspace-pointer, een speciaal operand register en de drie rekenregisters, genummerd A t/m C. De workspacepointer wijst een stuk geheugen aan waarin de transputer op dat moment aan het werk is. De process scheduler verplaatst deze pointer eenvoudigweg naar een ander stuk geheugen, waardoor de transputer razendsnel kan schakelen tussen verschillende processen.

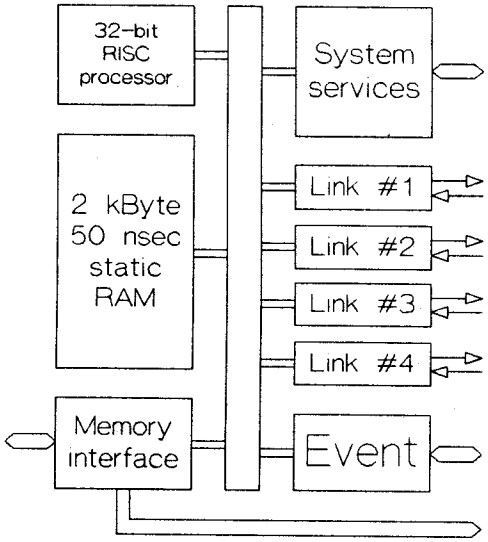


Fig. 4: De T414 van INMOS

De instructionpointer wijst naar een stuk geheugen waarin de instructies opgeslagen staan. Dit geheugen is slechts acht bits breed. De inhoud van ieder byte is onder te verdelen in twee functioneel van elkaar verschillende nibbles. Het ene nibble bevat de instructie, het andere bevat data. Dat lijkt verdacht veel op een rasechte vier-bitter. De truuk is echter deze: de transputer haalt één byte op. Het hoogste nibble is de instructie, de laagste is de operand.

Deze operand wordt in het operand register geplaatst en de instructie wordt uitgevoerd. Als het één van 13 meest benodigde instructies is, is het ver-

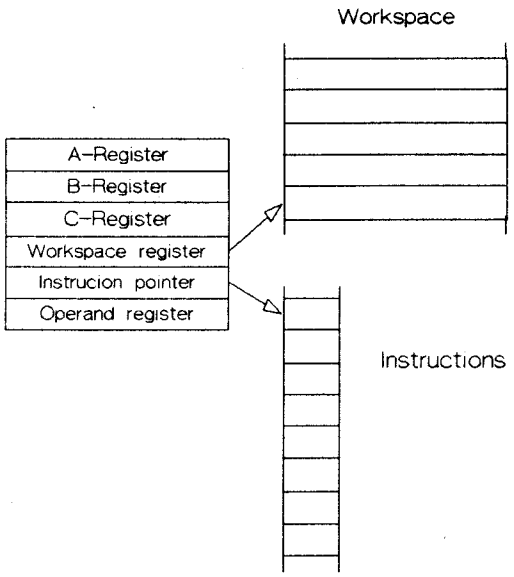


Fig. 5: Werking van work- en instruction pointer

haal hier af. De instructie wordt dan direcct uitgevoerd. Er zijn echter een drietal speciale instructies beschikbaar. Deze instructies maken de rest van de instructieset beschikbaar. Het gaat hier om de instructies "Prefix" (mnemonisch "Pfix"), "Negetive prefix" (Nfix) en "Operate" (Opr). De prefix instructie laadt de operand in het operand register en schuift de inhoud van het register vervolgens een nibble naar links. De nfix instructie doet hetzelfde, maar inverteert de operand eerst. Normaliter wordt na het uitvoeren van een instructie het operand register leeg gemaakt. In het geval van de pfix- en nfix instructies gebeurt dat niet. De opr instructie laadt dan het volgende nibble en voert daarna de instructie uit die door het operand register aangewezen wordt.

Een voorbeeld zal de zaak wel vergemakkelijken. We nemen de instructie "Long Add" (Ladd). Deze instructie telt de waarden die in de rekenregisters A en B staan bij elkaar op en plaatst het resultaat in het A-register. De instructie heeft instructie code 16h. Dit wordt door de assembler van de transputer omgezet in "Pfix #1" en "Opr 6". De resulterende code voor deze instructie wordt dan ook 21F6h. (2X is de code voor Pfix, FX is de code voor Opr). Op deze manier kunnen we drie verschillende soorten instructies onderscheiden. Allereerst dus de 13 standaard-instructies. Dit zijn de meest gebruikte instructies in de meeste processoren. Ze beslaan één nibble voor de code en hebben een operand van ook weer één nibble.

Dan is er een groep waarbij de operand deel uitmaakt van de code. Deze groep begint altijd met "Opr". Alle instructies uit deze tweede groep zijn impliciet en beslaan twee nibbles. De instructie "In" (input message from link) is een voorbeeld van zo'n instructie. De code voor deze instructie is 07, hetgeen gecodeerd wordt door "Opr 7", zonder voorafgaande prefix-instructies. Instructies uit de derde groep beslaan twee bytes per instructie; één byte prefix-instructie en één byte operate instructie.

De pfix- en nfix instructies worden ook gebruikt voor het laden van 32-bits constanten op de rekenstack. Een constante "535h" wordt op de stack gezet door de twee pfix-instructies en één load constant instructie...

In praktische toepassingen wordt deze puzzel gelukkig overgenomen door de compiler en/of assembler. Het zou ondoenlijk zijn voor een mens om bij iedere optelling of vermenigvuldiging steeds een hoop prefix- en nfix-instructies te moeten gebruiken om een constante term te definieren...

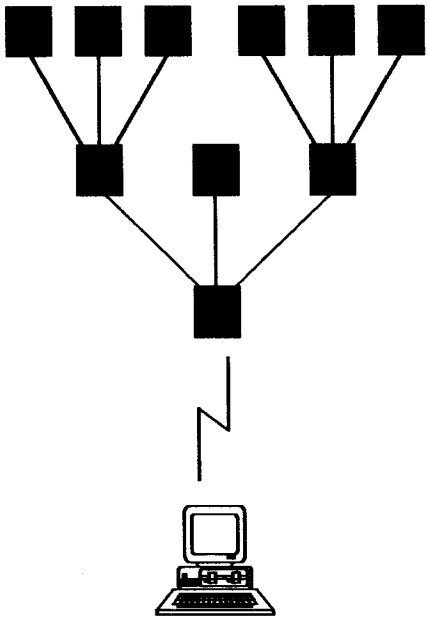


Fig. 6: Boomstructuur met host computer

Maar goed, terug naar de echt fysieke aspecten van de transputer. Laten we eens kijken naar de links. Zoals ik in de vorige aflevering al schreef is een link een razendsnelle seriele verbinding met de buitenwereld. De transputer kan messages over de link naar buiten sturen via speciaal daarvoor gereserveerde instructies. De snelheden van de links worden worden hardwarematig vastgelegd door middel van een drietal aansluitingen. Deze aansluitingen dragen de namen "Link special", "Link0 special" en "Link123 special". De ware hardware freak zal hier onmiddelijk uit concluderen dat er twee pennen nodig zijn om link #1 in te stellen en dat er één pen ("Link special") gecombineerd wordt met de "Link123 special" om de snelheid van de andere drie links in te kunnen stellen. De waarheidstabel (mooi woord he...) ziet er als volgt uit:

Link Special	Link0 Special	MBit/sec	Data rate (kbyte/sec)
0	0	10	400
0	1	5	200
1	0	10	400
1	1	20	800

Voor de links 1, 2 en 3 gelden dezelfde waarden, alleen dan wel i.p.v. "Link0 Special" de pen "Link123 Special". De snelheden van de links 1, 2 en 3 zijn dus niet onafhankelijk van elkaar te regelen. INMOS heeft alle alle externe devices geschikt gemaakt voor 10 Mbit/sec. In een typische boomstructuur (zie fig. 3) wordt link #0 van de root-transputer ingesteld op de maximale snelheid die door de host-computer nog gehaald kan worden. De andere drie links zijn verbonden met andere transputers en kunnen dus op de hoogst mogelijke snelheid ingesteld zijn. Die andere drie transputers zijn aan de root-transputer gekoppeld met hun link #0. Deze kan dan dus ook weer zo snel mogelijk ingesteld worden. De devices die eventueel aan de andere links hangen kunnen dan op een lagere snelheid aangestuurd worden.

Sja, dit besluit het tweede deel in de serie over dit wonderbaarlijke zwarte doosje. Volgende keer meer over de werking van de process-scheduler en een kleine vergelijking met andere systemen.

Joost Voorhaar.

Ontbinden in factoren (tevens priemonderzoek)

Het volgende (Turbopascal-) programma geeft de ontbinding in priemfactoren van getallen beneden een bepaalde waarde. Door niet alle oneven getallen te testen en hieruit de 3-vouden te elimineren wordt het programma ca 30% sneller. Dit wordt bereikt door afwisselend de delers met 2 en 4 te verhogen.

In de while-lus is toegevoegd 'AND deler < 46341, want deler * deler is dan groter dan maxlongint ($=2^{31}-1$), waardoor er geen juist stopcriterium voor priemgetallen tussen 46340^2 en maxlongint is, terwijl de oplossing 'deler < = (maxlongint DIV deler)' het programma ca 50% langzamer maakt.

A.P.Oerlemans

```
PROGRAM ontbinden(input,output);

VAR  faktor,deeltal,
     deler,getal    : longint;
     meer,i,exponent     : integer;
     factor : ARRAY[1..10,0..1] of longint;

FUNCTION spaties(waarde : longint) : string ;
     CONST spatie = '          ';
     VAR  lengte : integer;
          woord: string[10];
BEGIN
     str(waarde,woord);
     lengte := length(woord);
     spaties := copy(spatie,1,lengte);
END; {spaties}

PROCEDURE BepaalFactor(faktor : longint);
BEGIN
     IF (deeltal mod faktor) = 0
     THEN BEGIN
          exponent := 1;
          factor[i,0] := faktor;
WHILE (deeltal MOD faktor) = 0 DO
          BEGIN
               deeltal := deeltal DIV faktor;
               factor[i,1] := exponent;
               exponent := exponent + 1
          END;
          i := i + 1;
     END;
     IF deeltal < > 1
     THEN BEGIN
          factor[i,0] := deeltal;
          factor[i,1] := 1
     END;
END; { BepaalFactor }

PROCEDURE uitvoer;
     VAR  grondtal,exponent : string[40];
          hulpfactor,hulpexp: string[12];
          m : integer;
BEGIN
     str(getal,grondtal);
     grondtal := grondtal + ' = ';
```

```
        exponent := spaties(getal) + ' ';
        m := 1;
        WHILE factor[m,0] < > 0 DO
        BEGIN
              str(factor[m,0],hulpfactor);
              str(factor[m,1],hulpexp);
              IF factor[m,1] > 1
              THEN BEGIN
                    grondtal := grondtal + hulpfactor + spaties(factor[m,1]) + '.';
                    exponent := exponent + spaties(factor[m,0]) + hulpexp + ' '
              END ELSE
              BEGIN
                    grondtal := grondtal + hulpfactor + '.';
                    exponent := exponent + spaties(factor[m,0]) + ' '
              END;
              m := m + 1;
        END;
        writeln;
        writeln('                    ',exponent);
        write('De gevraagde ontbinding is: ');
        writeln(copy(grondtal,1,length(grondtal) - 1));
END; { Uitvoer }

BEGIN { HOOFDDEEL }
        for i := 1 TO 10 DO writeln;
        writeln('                ONTBINDEN');
        writeln('               IN FACTOREN');
        for i := 1 to 10 DO writeln;
        writeln('Dit programma geeft de ontbinding van een getal dat ');
        writeln('moet inliggen tussen 3 en ',maxlongint,'.');
        writeln;
        write('Voer het getal in (een getal < 4 stopt het programma) ');
        readln(getal);
        deeltal := getal;
        WHILE deeltal > 3 DO
        BEGIN
              for i := 1 to 10 DO factor[i,0] := 0;
              i := 1;
              writeln;
              BepaalFactor(2);
              BepaalFactor(3);
              meer := 4;
              deler := 5;
              WHILE (deler * deler < = deeltal) AND (deler < 46341) DO
              BEGIN
                    BepaalFactor(deler);
                    meer := 6 - meer;
                    deler := deler + meer;
              END;
              IF i = 1 THEN writeln('Het getal is een priemgetal')
              ELSE uitvoer;
              writeln;
              write('Voer het getal in (een getal < 4 stopt het programma) ');
              readln(getal);
              deeltal := getal;
        END;
END.
```

Moderne datacommunicatie

Het is nog niet eens zo lang geleden dat ik een modem kocht. eigenlijk is dat zelfs nog maar een half jaartje terug. In die korte tijd ben ik behoorlijk verslingerd geraakt aan de nieuwe Verslaving: datacommunicatie. Verslaving ja, met een hoofdletter V. Mijn huisgenoot denkt er zelfs over om, als hij teruggekeerd is uit Amerika, een modem te kopen. En dan moet er ook maar een computer bij komen...

Wat is er nu zo leuk aan datacommunicatie? Tsja, het antwoord op die vraag is niet simpel te geven. Je zou kunnen zeggen dat het een vrij anonieme manier is om met anderen in contact te treden. Daardoor krijgt de gemiddelde conferentie al heel snel een toon van "ouwe jongens krentebrood". Het formeel bedoelde "u" heeft in de datacommunicatie vrijwel overal plaats moeten maken voor het veel kameraadschappelijker "je", om maar eens wat te noemen. Daar komt dan nog bij dat de vooroordelen die je ten opzichte van een persoon zou kunnen hebben op grond van uiterlijk en/of accent niet bestaan in deze vorm van communicatie.

In die korte tijd ben ik behoorlijk verslingerd geraakt aan de nieuwe Verslaving: datacommunicatie. Verslaving ja, met een hoofdletter V.

Maar goed, genoeg over de voor- en nadelen van deze moderne communicatiemethode. Dit artikel gaat over de technische aspecten van bulletin boards en electronische post. We beginnen bij het bulletin board, het medium waar vrijwel iedere modemgebruiker begonnen is...

Bulletin boards zijn er in soorten en maten. Er zijn verschillende standaard programma's waarmee je een bulletin board kunt opzetten. Eén van de populairste programma's (en zeker ook mijn favoriet) is het programma dat ook gebruikt werd voor het bulletin board van de vereniging: QuickBBS. QuickBBS laat de SysOp toe een heel eigen gezicht aan zijn bulletin board te geven doordat er maar weinig menu's van te voren in het programma vastgelegd zijn. Een heel goede cloon van dit programma is het gloednieuwe "Remote Access" (afgekort tot "RA"). RA is volledig compatible met QuickBBS versie 2.61 (de meest recente versie), maar is sterk verbeterd in het multitasking gebeuren. Een ideale vervanger dus voor bulletin boards die graag met meer dan één telefoonlijn willen gaan werken.

"Maar, hoe gaat nou een telefoonsessie in zijn werk?", zult u wellicht vragen. Wat u allereerst nodig hebt om contact te kunnen leggen met een bulletin board heeft u natuurlijk een computer, een modem en een terminal programma nodig. Heel populair is het telecommunicatie pakket "Telix", een shareware programma met bijzonder veel goede features. Telix zal ik volgende keer bespreken in de shareware rubriek van de μPKenner, dus een preciese beschrijving laat ik hier achterwege. De computer belt voor u het bulletin board en, na wat aftasten van snelheden e.d. krijgt u contact met het bulletin board. Meestal ziet u dan eerst een boodschap in de trant van "FrontDoor 1.99b; non commercial version" en "Hit escape twice for QuickBBS". Het programma FrontDoor (Nederlands: "voordeur") is een zogeheten mailer. Op de preciese functie van die mailer komen we straks nog terug; we taffen nu gewoon even twee maal op de escape-toets en... tataa! Het duurt even, maar daar is het bulletin board al! Even uw naam invullen en, als het de eerste keer is (of als u minstens een x aantal dagen niet meer ingelogd hebt), een hele reeks aan vragen. Deze vragen worden voor een deel door het systeem zelf gesteld om optimaal te kunnen werken met uw configuratie en voor een ander deel zijn het vragen om gegevens die door de SysOp nodig geacht worden voor het checken van uw identiteit. Deze vragen worden vaak ook nog gebruikt om het bulletin board zo goed mogelijk aan te laten sluiten bij de wensen van de bellers. Ze blijven in ieder geval strikt privé, ze worden nooit verkocht, verhuurd of weggegeven aan anderen!

Heeft u de vragenlijst ingevuld, dan komt u in het bulletin board (afgekort tot "BBS") zelf terecht. Soms verschijnen er eerst nog allerlei teksten op uw scherm met nieuws over het BBS of over andere BBS-en. U komt dan terecht in een hoofdmenu waar u kunt kiezen wat u nou precies wil gaan doen. Zo kunt u bijvoorbeeld post gaan lezen en schrijven of bestanden gaan versturen ("uploaden") en ontvangen ("downloaden"). Meestal zijn er op het BBS ook verschillende teksten terug te vinden over van alles en nog wat (de bulletins) en een keur aan verschillende andere programmas's op te starten.

De belangrijkste functie van een BBS is en blijft echter de mogelijkheid tot electronische uitwisseling van post. In grote trekken zijn er twee soorten post: lokale post en de z.g. "echomail". Zoals de naam al

doet vermoeden blijft lokale post op het bulletin board. Echomail daarentegen gaat ook naar een hele hoop andere bulletin boards. 's Nachts bundelt het systeem dat aangesloten is op de echomail (een "node" in BBS terminologie) alle post die voor de echomail bestemd is tot een pakket. Dat pakket wordt 's nachts verstuurd en kan op die manier de volgende dag al op plaats van bestemming zijn (goh, lijkt de P.T.T. wel...). Echomail is vrijwel altijd "public" post. Dat wil zeggen dat anderen op een systeem waar dat bericht aankomt dat bericht ook kunnen lezen. Ook al is het bericht niet aan hun geadresseerd! Echomail is de ideale manier om vragen te stellen aan iedereen die wellicht de kennis en/of zin heeft om u te helpen met een probleem.

Naast echomail is er ook nog netmail. Netmail verschilt van echomail op een aantal punten. In de eerste plaats is netmail meestal private. Dat wil dus zeggen dat alleen de geadresseerde het bericht kan lezen (en de SysOp's van de boards via welke het bericht getransporteerd wordt!). Netmail gaat ook niet naar alle andere aangesloten systemen, zoals echomail, maar wordt zo direct mogelijk naar het geadresseerde systeem gezonden. Meestal komt het er op neer dat het BBS waar het bericht gepost werd, 's nacht direct het BBS van de geadresseerde belt. Daarom is netmail meestal niet gratis!

Mensen die erg veel post versturen, kunnen dat op een snellere manier doen; ze kunnen "point" worden. Een point heeft, net als het BBS, een mailer. Overdag kan iemand die point is zijn berichten intikken op zijn eigen systeem. 's Nachts wordt dan automatisch het systeem gebeld waar men de point-aansluiting heeft aangevraagd. De twee mailers wisselen dan heel snel de berichten met elkaar uit en... er hoeft geen mens meer bij te zijn! De communicatie tussen een point en zijn "boss" (de node waar hij/zij de post naar toe stuurt) lijkt erg veel op de communicatie tussen twee nodes onderling. Het grootste verschil is echter wel dat een point ook bestanden met zijn boss kan uitwisselen; iets dat meestal niet gebeurt tijdens een echomail contact tussen twee nodes. Daarnaast kan een point ook in de lokale postgebieden post uitwisselen.

Er is al enige tijd een pakketje in omloop waarmee men een soort pointsysteem kan opzetten. Met behulp van dat pakket (het "eXpress Response System", oftwel "XRS") kan men post op het BBS selecteren, laten inpakken en downloaden. De zo ontvangen post kan dan thuis in de luie stoel doorgelezen en beantwoord worden, bijna net zo als een point dat doet. De antwoorden worden dan weer gebundeld, en de XRS-gebruiker kan het pakket tijdens een volgende BBS-sessie gewoon weer verzenden aan het BBS. De verschillen tussen een echte point en een dergelijke "fake-point" zijn, dat de XRS-gebruiker iedere keer toch weer het BBS in moet om de post uit te wisselen. Daarnaast wordt de post online gepackt, wat wel eens een minuut of drie à vier kan duren. Een point hoeft het BBS helemaal niet meer in om zijn post af te leveren en nieuwe op te halen. Tevens staat de post op het BBS-systeem al klaar als hij belt; er hoeft dus niet on-line post geselecteerd en gebundeld te worden. Dit zijn slechts een paar van de voordelen die een point heeft t.o.v. een "normale" BBS-er.

Moderne datacommunicatie kan dus veel meer betekenen dan een moeizame manier om aan goedkope software te komen. Heeft u (nog) geen modem? Wellicht is het dan een goed idee de aanschaf van een goed modem te overwegen. Een redelijk goed modem hoeft niet veel te kosten en... je hebt er veel plezier van.

Joost Voorhaar

Ik heb interesse in de KGN en wil

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

Naam : _____

Adres : _____

Postcode en woonplaats : _____

Datum : _____ Handtekening : _____

Dit strookje kunt u ingevuld opsturen aan het secretariaat van:

KIM Gebruikersclub Nederland,
Davidvosstraat 29,
1063 HV Amsterdam.

Programeren in Assembler voor DOS-65 (deel 5)

Na een korte periode van afwezigheid (mijn fans zullen tevergeefs gezocht hebben naar deel 5 in 'de μPKenner' nr. 64) heeft mijn psychiater weer het groene licht gegeven voor wederom een literair hoogstandje, kortom lieve lezers en vooral lezeressen, deel 5! Maar eerst...

Voordat U kunt gaan genotteren, even een korte terugblik op deel 4. Bij mijn exemplaar van nr.63 was in de tekst op pagina 12, de assembler text niet volledig. Er stond tweemaal BNE DLOOP, voor de goede orde dit moet zijn:

```
        BNE  DLOOP2
        DEX
        BNE  DLOOP1
        ...
```

Ook had ik gevraagd om het programma zo te wijzigen dat de letter precies 10x zou knipperen en dan terugkeert naar MON of DOS. Welnu, het zal een ieder duidelijk zijn dat we voor deze truuk een teller nodig hebben. Verder zal die JMP FLASH eruit moeten, want deze, zoals de Fransen zeggen, 'unconditional jump' springt immers ALTIJD. Wat wij nodig hebben is een 'conditional jump', kortom een spring-in-het-veld die verder kijkt dan z'n binaire neus lang is. Maar 'quelle malheur', zoals de Engelsen zeggen, zo'n conditonele jump heeft de 6502 niet aan boord. Gelukkig heeft ie wel conditionele branches, een aantal hebben we al gezien (BCC, BCS, BNE en BEQ). Die branches hebben als nadeel (zie deel 2) dat ze een beperkt bereik hebben van -128 tot +127 geheugenlocaties. Maar voor het EOR_DEMO programma is dat ruimschoots voldoende. Zo'n conditionele branch verlengen tot een conditionele jump is overigens een piece of cake, zoals de Fransen zo treffend weten te zeggen, kijk maar:

```
        ...
        BNE  NOGO
        JMP  GO
NOGO    ...
```

En zie daar, een conditionele jump en wel van het type JEQ (Jump if EQual). Maar goed terug naar ons FLASH probleem, wat we moeten maken is een conditionele jump die alleen springt als de loop teller nog niet klaar is met zijn 10 rondjes knipperen. Zoals ik al had gewaarschuwd kunnen we de X of Y registers niet gebruiken omdat die al in gebruik zijn in de DELAY subroutine. Natuurlijk kun je X of Y eerst bewaren in een geheugenlocatie:

```
        ...             ; begin van EOR_
                            DEMO (zie deel 4)
        LDX  #20        ; 10x knipperen =
                                20 x EOR!!
FLASH   LDA  ...        ; zie deel 4
        ...
        STX  XSAVE      ; bewaar loop teller X
        JSR  DELAY      ; effe wachten, anders
                                gaat het te snel..
        LDX  XSAVE      ; loop teller ophalen
        DEX             ; teller = teller - 1
        BNE  FLASH      ; als NIET nul, doorgaan!
        RTS             ; anders terug naar MON
                                of DOS
XSAVE   FCC 0           ; geheugenlocatie om X
                            tijdelijk te bewaren
```

Natuurlijk zijn er legio andere mogelijkheden, bijvoorbeeld een geheugen (b.v. op de zero-page) locatie gebruiken als TELLER, dit heeft als voordeel dat we X niet hoeven te 'saven'. Gebruik dan LDA #20, STA TELLER als initia- lisatie en DEC TELLER voor de conditionele branch. Ook kun je de teller laten optellen vanaf 0 en dan CMP, CPX of CPY (resp. bij gebruik van een geheugen locatie, index register X of index register Y als teller) gebruiken om te kijken of je al klaar bent. Dit laatste heeft als nadeel dat het programma twee bytes langer wordt, door aftellen te gebruiken maak je namelijk handig gebruik van het feit dat de Z-flag automatisch gezet wordt als je teller de 0 bereikt. Gebruik je optellen, dan heb je de CMP instructie nodig om de Z-flag te zetten. Vraagje, waarom is het (dus) onzin om CMP #0 te gebruiken?

De adresseer mogelijkheden

In dit deel gaan we (eindelijk) eens kijken naar de adresseer mogelijkheden van ons wonderdoosje. Vooral in het begin duizelt het de machinetaal programmeur van de vele mogelijkheden op dat gebied en welke kun je nu het beste gebruiken en waarom? Allereerst maar een opsommimg, de 6502 heeft totaal 13 adresseer- mogelijkheden:

1. Implied
2. Accumulator
3. Absolute
4. Zero Page
5. Immediate
6. Absolute,X
7. Absolute,Y
8. Zero Page,X
9. Zero Page,Y
10. Relative
11. Indirect
12. Indexed Indirect
13. Indirect Indexed

1. Implied

Bij instructies van dit type ligt het adres van de data besloten in de opcode zelf of het zijn instructies die

verder geen data nodig hebben (bijv. NOP). Bij de 6502 zijn dat de instructies BRK, CLC, CLD, CLI, CLV, DEX, DEY, INX, INY, NOP, PHA, PHP, PLA, PLP, RTS, RTI, SEC, SED, SEI, TAX, TAY, TSX, TXA, TXS en TYA (hijg, hijg). Een aantal van deze instructies hebben we al gebruikt in de voorbeelden, een fijnere onderverdeling zou kunnen zijn:

a. instructies voor controle van status register bitjes:
CLC, CLD, CLI, CLV, SEC, SED, SEI

b. instructies voor data transfer tussen registers onderling:
TAX, TAY, TSX, TXA, TXS, TYA

c. instructies voor stack operaties:
PHA, PHP, PLA, PLP, RTS, RTI

d. instructies voor de index registers X en Y:
DEX, DEY, INX, INY

e. overig:
BRK, NOP

Alle implied opcodes zijn slechts een byte lang. De AS syntax is simpel, eenvoudig de 3 karakters voor de opcode intypen en klaar is Clara!

Voorbeeld:

Instructie	Memory
NOP	EA

2. Accumulator

Deze adresseer mogelijkheid is eigenlijk ook Implied, namelijk instructies waarbij in de opcode zelf al vastligt dat ze betrekking hebben op de accu. Bij de 6502 zijn dat er maar vier:

a. ALU bewerkingen:
ASLA, LSRA, ROLA en RORA.

Allemaal schuifoperaties (zie deel 3) die werken met de inhoud van de accu en daar ook het resultaat weer in terugzetten. Ook deze instructies zijn een byte lang. De AS syntax is uitgebreid met een A achter de instructie. Sommige assemblers gebruiken ook wel ASL-A of ASL A. Het onderscheid tussen ASL A en ASL $A is dan echter wat moeilijk (voor de lezer, NIET voor de assembler).

Voorbeeld:

Instructie	Memory
ASLA	0A

3. Absolute

Bij deze methode wordt het adres volledig beschreven in twee bytes die volgen op de opcode. De postbode van de processor weet dus precies waar ie moet wezen want zoals we al zagen in deel 1, de 6502 heeft een adres bereik van 64K en met 16 bits kun je een enkel adres precies beschrijven. De absolute methode wordt gebruikt bij:

a. ALU bewerkingen:
ADC, AND, ASL, BIT, CMP, CPX, CPY, EOR, INC, DEC, LSR, ORA, ROL, ROR, en SBC

b. Registers laden en schrijven:
LDA, LDX, LDY, STA, STX en STY

c. Programma flow instructies:
JMP en JSR

De absolute instructies zijn altijd 3 bytes lang, t.w. de opcode en 2 bytes voor het adres. Zoals we al weten staat het LSB (Least Significant Byte) van het adres in de eerste geheugenlocatie, en het MSB (Most Significant Byte) in de daaropvolgende (zie ook deel 1). De syntax voor de AS assembler is eerst de instructie en vervolgens een 16 bits getal of een label.

Voorbeeld:

instructie	Memory
LDA $1234	AD 34 12

4. Zero Page

Bij deze methode 'weet' de processor (door de opcode) dat het adres op de eerste pagina van het geheugen staat, zoals we al gezien hebben zijn pagina's precies 256 bytes groot en dus heeft de postbode aan een enkel byte genoeg. Je zou kunnen zeggen dat Zero Page adresering een soort wijkpost is terwijl de Absolute adresering voor de hele stad met 65535 inwoners te gebruiken is. De Zero Page methode is bij precies dezelfde instructies als hierboven bij de Abslotute methode is beschreven. Met uitzondering van de JSR en de JMP instructie. Het voordeel van de Zero Page methode is dat het programma sneller wordt, je kunt met je boerenverstand aanvoelen dat de processor minder werkt heeft aan het ophalen van de instructie (een byte korter). Nadeel is dat bij bijna alle 6502 systemen een gedeelte van de Zero Page al in gebruik is bij de systeem software zodat niet alle locaties in jouw flitsende programma gebruikt mogen worden. Bij het DOS65 systeem valt dat gelukkig allemaal mee, omdat hier de systeem software immers voor een groot gedeelte zelf in RAM staat zijn er heel weinig locaties van de Zero Page in gebruik. Voor zover ik weet zijn dat adres 0..3 (PTA en PTB) die (tijdelijk) in gebruik zijn in IO65 (de data in die locaties wordt overigens wel netjes opgeslagen en later terug gezet, zodat je bij normaal gebruik deze locaties best kunt gebruiken, bij programma's die veel interrupts gebruiken is het oppassen geblazen). Verder worden de adressen $FB, $FC, $FE en $FF (tijdelijk) gebruikt tijdens het 'booten' van het systeem. (klopt dat Erwin?). De Zero Page instructies zijn alle 2 bytes lang. De syntax voor AS is net als bij Absolute alleen moet het adres maar 8 bits zijn of het label moet verwijzen naar de zero page.

Voorbeeld:

Instructie	Memory
STA $12	85 12

5. Immediate

Dit is de meest directe vorm om iets aan de processor door te geven, de te gebruiken data staat namelijk direct op de volgende geheugenlocatie na de opcode. Er hoeft dus geen postbode door weer en wind gestuurd te worden om ergens in de binnenlanden van Borne Bokkie een pakketje data op te gaan halen. Je zou verwachten dat deze methode ook toegepast zou kunnen worden bij alle ALU bewerkingen, maar een aantal van die bewerkingen zijn zogenaamde Read-Modify-Write instructies. Dat is Frans voor: data lezen, doe er iets mee en zet vervolgens het resultaat weer terug waar je het gevonden hebt. Het is duidelijk dat deze operaties niet echt bruikbaar zijn bij de immediate instructie, immers iedere keer dat je programma zou starten zou er een ander getal achter de opcode (kunnen) staan. Kortom: de immediate methode wordt alleen gebruikt waarbij een getal gelezen wordt, nooit voor schrijfoperaties. Over blijven dus die ALU bewerkingen waarbij alleen data gelezen wordt en het resultaat alleen in de Accu komt te staan.

a. ALU bewerkingen:
ADC, AND, CMP, CPX, CPY, EOR, ORA en SBC
b. Register laden:
LDA, LDX en LDY

De immediate instructies zijn altijd 2 bytes lang. De syntax voor AS is eerst de instructie, vervolgens een # karakter en dan de 8 bits data of een EQU (ja, ook van 8 bits ja!).

Voorbeeld:

Instructie	Memory
AND #$AA	29 AA

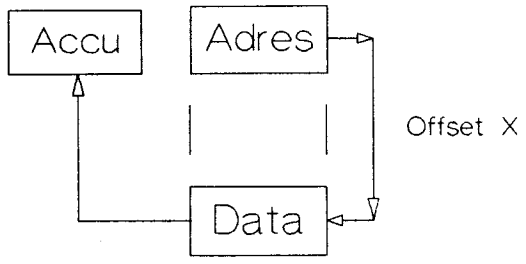


Fig. 1: LDA,X

6. Absolute,X

Deze methode heet ook wel 'indexed adressing', het adres wordt namelijk berekend door bij het 16 bits absolute adres het index register X op te tellen. De postbode krijgt dus een adres door, fietst ermee naar vriend X, krijgt daar zijn offset, zoals de Fransen zeggen, gebruikt vervolgens z'n zakjapanner en weet dan pas waar ie naar toe moet met zijn door hogerhand verstrekte dienstvoertuig. Deze methode is heel handig als je de postbode een tabel wilt laten doorfietsen, in de instructie staat dan het beginadres van de tabel en het index register X doet de rest. Je ziet dat X register kan heel wat meer dan een simpel tellertje. In figuur 1 is dit schematisch weergegeven.

Deze methode kan gebruikt worden bij bijna alle instructies die ook bij de standaard Absolute methode genoemd worden, behalve die instructies die betrekking hebben op het index register X zelf CPX, LDX en STX (kan ik nog begrijpen), de JMP en JSR (kan ik ook nog begrijpen) maar ook op de instructies BIT, CPY en STY (en daar begrijp ik geen hout van, waarom wel LDY en geen CPY of STY meneer Rockwell?). Over blijven dus:

a. ALU instructies:
ADC, AND, ASL, CMP, DEC, EOR, INC, LSR, ORA, ROL, ROR en SBC.
b. Registers laden en schrijven:
LDA, LDY en STA

De instructie lengte is altijd 3 bytes. De syntax voor AS is precies hetzelfde als bij de Absolute methode alleen wordt de operand nu gevolgd door een komma en een X (of x).

Voorbeeld:

Instructie	Memory
STA TABEL,X	9D 00 20 (dus TABEL begint op $2000)

7. Absolute,Y

Nou zul je zeggen, die kennen we al. I.p.v. naar X te fietsen, fietst onze postbode nu eerst langs Y alvorens zijn zakjapanner te voorschijn te halen. Inderdaad, beste lezer (en lieve lezeres) U heeft volkomen gelijk! In figuur 2 is deze adresseermethode schematisch weergegeven.

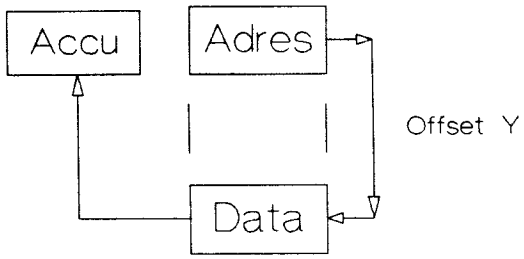


Fig. 2: LDA,Y

Dus alle instructies die bij Absolute,X gebruikt worden kan ik ook bij Absolute,Y gebruiken? Neen, beste lezer (en lieve lezeres) nu heeft U het mis. Meneer Rockwell heeft in zijn oneindige wijsheid beslist dat het niet nodig is om alle instructies zowel in Absolute,X als in Absolute,Y uit te voeren. Wat er over blijft is:

a. ALU instructies:
   ADC, AND, CMP, EOR, ORA en SBC
b. Registers:
   LDA, LDX en STA

De instructielengte en de syntax voor AS is precies hetzelfde als bij de voor gaande methode alleen vervangen we de X (of x) door een Y (of y).

Voorbeeld:

Instructie	Memory
LDX OFFSET,Y	BE 34 12 (OFFSET tabel start op $1234)

8. Zero Page,X

Welnu, even de grijze massa in beweging en je weet gelijk hoe deze instructie groep te werk gaat. Ze geven de postbode een adres in de wijk Zero Page, die fietst (alweer) naar vriend X, gebruikt z'n zakjapanner en weet het huisadres van de gevraagde sjampetter. Voordeel van deze methode is dat de instructie een byte korter is, nadeel is dat de tabel alleen in Zero Page kan staan. Alle instructies die opgesomd zijn bij Absolute,X kunnen ook gebruikt worden met deze adresseermethode en nu wel met STY (how come, mr. Rockwell?) Schematisch is dit verhaal hetzelfde als bij Absolute,X mits je in gedachten houdt dat ADRES nu ergens tussen $0000 en $00FF moet liggen. De instructielengte is 2 bytes en de AS syntax is precies hetzelfde als bij Absolute,X alleen moet de operand nu 8 bits zijn.

Voorbeeld:

Instructie	Memory
STY 50,X	94 50

9. Zero Page,Y

Deze hoef ik niet meer uit te leggen, dacht ik zo. Ook hier heeft meneer Rockwell weer drastisch bezuinigd want bij deze adresseermethode zijn nog slechts twee instructies overgebleven, namelijk LDX en STX. En dat is best wel vervelend soms, je kunt de Accu in de Zero Page dus alleen via X op de verkorte manier (2 bytes) laden e.d. via Y heb je dan gelijk Absolute,Y dus 3 bytes nodig. De instructielengte is 2 bytes, de AS syntax is weer idem als bij Zero Page,X door de X te vervangen met een Y.

Voorbeeld:

Instructie	Memory
STX DATA,Y	96 40 (DATA begint bij adres $0040)

10. Relative

Bij de 6502 wordt deze adresseermethode alleen gebruikt voor de offset van de branches. De data achter de opcode wordt gebruikt om de programmateller ten opzichte van de huidige waarde van de programmateller of PC (zie ook deel 1) te gaan verplaatsen. Het komt er dus op neer dat er een sprong wordt uitgevoerd van -128 tot +127 t.o.v. de huidige locatie. De 6502 kent totaal 8 verschillende branches die alle aan de hand van een bit van het statusregister P wel of niet werken. De conditionele branch dus. Dat zijn:

a. Branches die de C-flag testen:
   BCC, BCS
b. Branches die de Z-flag testen:
   BEQ, BNE
c. Branches die de N-flag testen:
   BMI, BPL
d. Branches die de V-flag testen:
   BVC, BVS

Het zelf berekenen van zo'n relatieve branch gebeurt vanaf de locatie waar de offset moet komen te staan. Voorwaartse referenties zijn eenvoudig uit te tellen. Moeten we bijvoorbeeld over een JMP instructie heen 'branchen' dan is de offset 3 (immers een JMP is 3 bytes lang) (zie figuur 3).

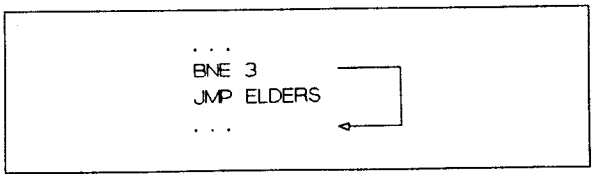


Fig. 3

Een terugwaartste referentie is wat lastiger omdat we nu het two's complement moeten berekenen, bovendien moet de lengte van de branch zelf t.w. 2 bytes ook worden meegeteld. Programeren we nu de kortste oneindige programmaloop die mogelijk is dan moeten we dus als offset -2 opgeven. In 2's complement wordt dat:

```
2 =     00000010
        11111101      (1's complement, alle bits
                                    inverteren)
        00000001  +   (+1)
        11111110      = $FE
```

Ons superkorte nutteloze processortijd verspillende programma wordt dus zoals in figuur 4 afgedrukt is.

```
        CLC         ; Initialisatie
    ┌─> BCC $FE     ; and start
    │                 ; loop ad infinitum
    └─┘
```

Fig. 4

Het effectieve bereik van een branch is gegeven door de 8 bits signed offset dus -128 tot +127. Omdat de instructie zelf 2 bytes lang is wordt dat dus -128 + 2 = -126 (terug) tot 127 + 2 = 129 (vooruit). De instructie lengte is 2 bytes, de AS syntax is de instructie gevolgd door de relatieve offset als je zelf graag relatieve jumps uitrekend, anders een verwijzing naar een label binnen het bereik van de branch.

Voorbeeld:

Instructie	Memory
BNE 3	D0 03

11. Indirect

Dit is een buitenbeentje (of stiefkind) bij de 6502. De methode werkt als volgt, de postbode krijgt een adres door van de processor, hij fietst erheen en krijgt op dat adres te horen waar het pakketje naar toe moet (of moet worden opgehaald). Algemeen: de operand wijst naar adres A1 dat op zijn beurt wijst naar adres A2 waar de data staat. Bij de 6502 wordt deze methode alleen gebruikt voor de JMP instructie. In figuur 5 is de indirecte methode schematisch weergegeven.

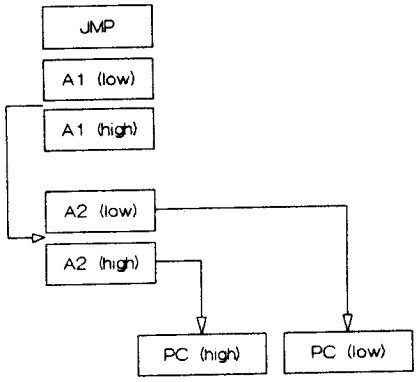


Fig. 5: JMP [A1] ; Jump indirect

Deze instructie is machtig mooi voor gebruik in een menu programma, het programma moet dan immers aan de hand van de invoer steeds een subprogrammaafwerken en dan weer terugkomen in de hoofdlus. Dat kan natuurlijk als volgt:

```
        ...             ; keuze zit in A (0...X)
        CMP  #0         ; was het keuze 0?
        BNE  +6         ; nee, ga door
        JSR  KEUZE0     ; ja, doe subroutine
        JMP  LOOP
        CMP  #'1        ; keuze 1?
        BNE  +6
        JSR  KEUZE1
        JMP  LOOP
        ...
```

Heel vervelend programeren is dat, bovendien heeft het als groot nadeel dat het niet eenvoudig is uit te breiden. Het programma wordt immers bij iedere keuze 10 bytes langer. Het is eigenlijk alleen bruikbaar bij relatief weinig mogelijkheden. En last but not least, iedereen kan dit zo schrijven. De Echte Programmeur doet het met tabellen, en wel als volgt:

```
        ...             ; keuze zit in A (0...X)
        ASLA            ; A = A * 2
        TAX             ; zet A in X
        LDA  KEUZE,0 ; haal subroutine adres
                                        (low)
        STA  ADRES
        LDA  KEUZE+1,X ; en high part
        STA ADRES+1
        JSR  DO_IT      ; doe geselecteerde
                                        functie
        JMP  LOOP       ; en voor de volgende keer
DO_IT   JMP  [ADRES]; ga subroutine uitvoeren

KEUZE FDB  KEUZE0,KEUZE1,KEUZE2,....
ADRES  FDB 0000
```

De tabel KEUZE bevat nu de subroutine adressen. De ASLA is nodig omdat de tabel 16 bits adressen bevat, en de offset dus 2 bytes per entry moet overslaan. Het grote voordeel van deze methode is evident, mijn waarde Watson. Het is leuk programeren, het is eenvoudig uit te breiden door alleen de tabel aan te passen, het programma wordt per entry slechts 2 bytes langer, en het getuigt van genialiteit. De instructie lengte is 3 bytes, de AS syntax is de JMP instructie gevolgt door de operand tussen blokhaken. Bij een aantal andere assemblers wordt ook wel gebruik gemaakt van de JMI (Jump Indirect) instructie, of staat de operand data tussen (normale) ronde haken.

Voorbeeld:

Instructie	Memory
JMP [SUBR]	6C 00 33 (SUBR staat op adr. $3300)

12. Indexed indirect

Dit is een hele rare druif. De postbode krijgt een adres door binnen de wijk Zero Page, hij fietst naar X en krijgt daar een offset. Na gebruik van zijn zakjapanner fietst ie naar de aangegeven locatie en krijgt daar van de bewoner en zijn buurman het adres door van de gezochte knakker. De arme postbode fietst dus heel wat af bij deze methode. En dat is dan ook goed te merken aan de executietijd van deze groep instructies, want hoewel ze alle slechts 2 bytes lang zijn heeft de processor er toch 6 cycles voor nodig. Deze methode is in figuur 6 schematisch weergegeven.

Deze methode gebruikt dus een tabel van 16 bits pointers en het index register is steeds een veelvoud van 2. Eerlijk gezegd heb ik zelf deze methode heel weinig gebruikt, simpel omdat ik er geen toepassing voor wist. Een groot nadeel is ook dat de tabel in Zero Page moet staan. De methode is geimplementeerd voor de volgende instructies:

a. ALU operaties:
ADC, AND, CMP, EOR, ORA en SBC
b. Register operaties:
LDA en STA

De instructie is 2 bytes lang, de AS syntax is eerst de instructie gevolgd door een 8 bits operand met ,X en dat geheel tussen blokhaken (zie voorbeeld).

Voorbeeld:

Instructie	Memory
ADC [PTR,X]	61 20 (PTR staat op $0020)

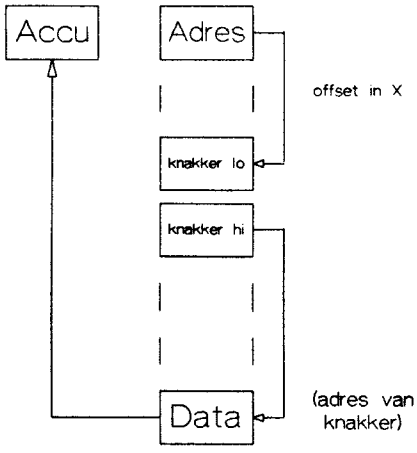


Fig. 6: LDA [ADRES,X]

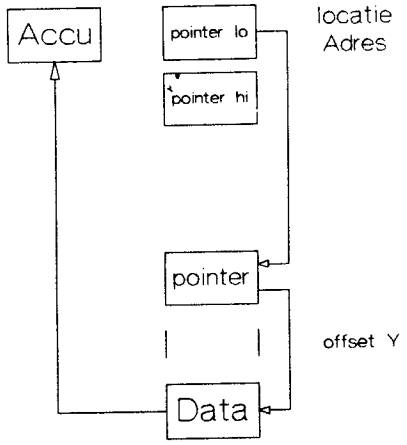


Fig. 7: LDA [ADRES],Y

13. Indirect indexed

Dit is een beauty. De postbode krijgt een adres in de wijk Zero Page, daar krijgt ie van de bewoner en diens buurman een adres door. Vervolgens telt ie met z'n zakjapanner de inhoud van Y bij dat adres en fietst daar vervolgens naar toe. Waarom is dit nu zo'n mooie? Welnu op deze manier heeft de 6502 eigenlijk een 16 bits index register met bovendien daarbovenop nog een 8 bits offset (het Y register). Omdat er in de Zero Page 256 locaties zijn heeft de 6502 dus 128 16 bits index registers!! In figuur 7 is deze methode schematisch weergegeven.

Deze methode werkt voor de volgende instructies:

a. ALU operaties:
ADC, AND, CMP, EOR, ORA en SBC
b. Register operaties:
LDA en STA

De instructie lengte is 2 bytes, de AS syntax is de instructie gevolgd door een 8 bits adres (in Zero Page dus) tussen blok haken en dit weer gevolgd door ,Y (zie voorbeeld).

Voorbeeld:

Instructie	Memory
LDA [PTR],Y	B1 20
	(PTR staat op $0020)

Tenslotte

Zo, dat zijn ze. Hiermee zul je het moeten doen, meer heeft de 6502 niet aan boord. Nog een kort woord over de te kiezen methode. Eigenlijk is dat helemaal vrij. Dat is juist het leuke van programeren. Je kunt er een eigen stijl in ontwikkelen, maar pas op voor broddelwerk. Ik hoop dat dit verhaal een beetje duidelijk is geworden, helaas dit keer geen voorbeeld programma van de hand van uw nederige dienaar. Voor de volgende keer heb ik eigenlijk nog geen onderwerp, dus laat eens wat van je horen. Wat wil je graag behandeld hebben? Subroutines? Interrupts? Vectoren? I/O technieken? Graag wat reacties dus, of zit ik toch in een zwart gat te schrijven, is dit inderdaad alleen wat bezigheidstherapie zoals mijn psychiater me voorschreef om van de spanningen af te komen? Reacties liefst via ons BBS (053-303902), via de redactie Gert van Opbroek, maar je kunt mij ook direct bellen ('s-avonds 038-537073).

Antoine Megens

Desktop Publishing; vloek of zegen?

De automatisering heeft alweer toegeslagen in het redactie-kot van de μP Kenner. werd tot voor kort uw lijfblad nog met behulp van schaar en lijmpot in elkaar geplakt, nu wordt dat werk geheel met behulp van de computer gedaan. De teksten worden door de verschillende redacteurs aangeleverd aan de hoofdredacteur die er voor zorgt dat de gecorrigeerde teksten in ASCII- of WordPerfect formaat bij de eindredactie terechtkomen. Deze worden dan door de lay-outer met behulp van een desktop publisher (een tekst-opmaak pakket) netjes omgevormd tot het eindresultaat. Daarna hoeft alleen de drukker nog maar te zorgen dat de μP Kenner in een oplage van ca. 200 stuks vermenigvuldigd wordt.

Het DTP-pakket dat voor de lay-out van de μP Kenner gebruikt wordt heeft een winkelwaarde van ca. *f* 2300,-. Er zijn echter een heleboel goedkopere DTP-pakketten in de handel. Hiervan zijn de programma's NewsRoom, ClickArt en Publish-it! wel de bekendste. Een duurdere, maar helaas niet veel betere is PageMaker, het pakket dat Windows populair maakte voor de PC. Deze pakketten (m.u.v. PageMaker) kunnen over het algemeen echter maar een beperkt aantal (matrix-) printers aansturen en de beschikbare lettertypen liggen vast in het pakket. Kortom: ze zijn niet flexibel, beperkt in het gebruik, traag en leveren dientengevolge een zeer slecht eindresultaat af.

Zoals je geen boekhoudpakket kunt gebruiken zonder dat je kennis hebt van boekhouden, kun je ook niet lay-outen zonder dat je kennis hebt van vormgeving.

Kernigan en Richie schreven over de door hun ontworpen taal al eens: "C gives you enough rope to hang yourself". Analoog hieraan zou je kunnen stellen: "DTP gives you enough power for a nuclear meltdown". Een beetje DTP-pakket heeft al gauw de beschikking over 3 of 4 totaal verschillende lettertypen (of "fonts", zoals de vakterm luidt), vaak in 6 of 7 verschillende formaten, in normaal-, vet- én schuinschrift ("Italics" zegt de drukker). Gaat men zonder vakkennis en/of inzicht met een dergelijk pakket aan de slag, dan wordt het al snel een rommeltje. Zoals je geen boekhoudpakket kunt gebruiken zonder dat je kennis hebt van boekhouden, kun je ook niet lay-outen zonder dat je kennis hebt van vormgeving.

De zegswijze "overdaad schaadt" gaat heel sterk op bij lay-outen. Voor de μPkenner maken we gebruik van slechts twee lettertypen, in drie verschillende groottes. Eén font wordt uitsluitend gebruikt in de accenten (titels en paginakoppen), het andere wordt voor de rest gebruikt.

Met professioneel lay-outen kan je een leuke cent bijverdienen. Maar dan dient het eindresultaat wel van dusdanige kwaliteit en prijs te zijn dat het aantrekkelijk is voor commerciële ondernemingen om een amateur te verkiezen boven een commercieel werkende vormgever. Dat betekent dat er veel geïnvesteerd moet worden in apparatuur en dat men heel wat verschillende voorbeelden moet hebben om aan de potentiële klant te kunnen laten zien. Een totaal lay-out pakket is duur. Erg duur zelfs. Om mee te beginnen volstaat een XT met harde schijf en een muis misschien nog wel, maar bij wat grotere publicaties (meer dan 10 pagina's) is een liefst snelle AT met een snelle harde schijf noodzaak. Voeg daar een goede laserprinter aan toe met de benodigde software en je zit al gauw aan een bedrag van ca. *f* 25.000,-. Wil je ook illustraties gaan verwerken, dan moet er ook nog een goed foto-copieer apparaat en een scanner met een hoge resolutie bij komen. Een lichtbak met melkwit glas completeert de uitrusting. En dan kun je nog maar een beperkt aantal kleuren aan...

Wat ik nu eigenlijk met dit artikel wil zeggen is het volgende: naar aanleiding van de facelift die de μP Kenner ondergaan heeft zijn er wellicht lezers die in hun diskettebak zijn gedoken en op een paar verdwaalde diskettes een backup van een DTP-pakket in bewaring hebben gekregen van een goede vriend. Wat spelevaren kan natuurlijk nooit kwaad, maar bezint eer ge begint. De weg van een mislukte uitnodiging voor een feest tot aan een flitsend ogend, 50-pagina's tellend document is als een lang en vooral smal bergpad. Plenty mogelijkheden om onderweg te struikelen dus. Met fatale gevolgen voor uw vriendenkring, ben ik bang. Het zelf maken van een kerst-, nieuwjaars- of paaskaart is best leuk werk. Maar gebruik er geen DTP-pakket voor met een standaard illustratie uit de (meegeleverde) illustratie library. Want, wees eerlijk... een kaartje met een uit blokjes opgebouwde afbeelding komt lang niet zo persoonlijk over als een misschien wel iets mislukte maar met de hand ontworpen groet. Toch?

Joost Voorhaar.

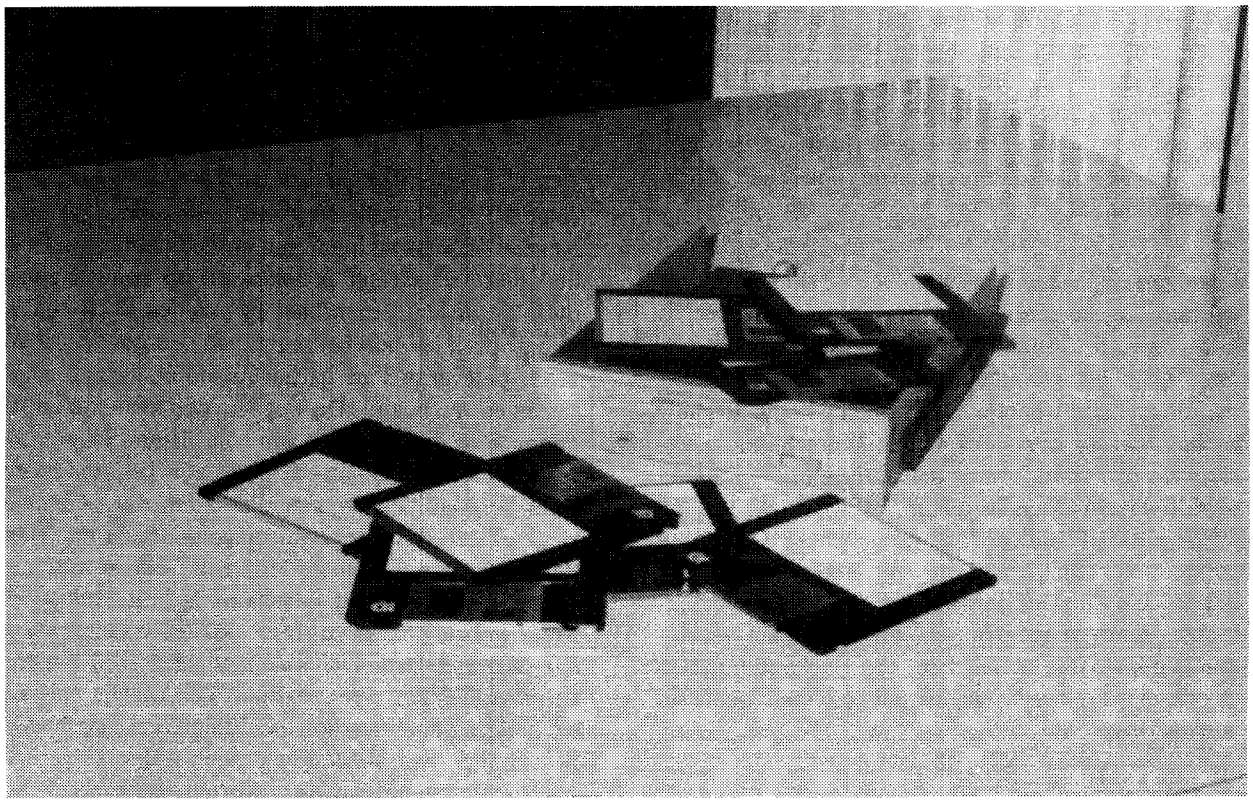
Een 3.5 inch diskettedrive

De diskette drives worden steeds goedkoper. Zo kon je een paar jaar geleden een 5.25 inch diskette drive kopen voor zo'n zeshonderd gulden, nu is de prijs gehalveerd en op het moment zie je zelfs aanbiedingen van 3.5 inch drives voor onder de tweehonderd gulden (80 track / dual sided, bijv. van Timtronix). Deze prijzen gelden voor fabrieksnieuwe drives, tweedehands drives zijn uiteraard nog goedkoper. Kortom, het loont de moeite om te zien wat de mogelijkheden zijn op de diverse computersystemen. Het antwoord is eenvoudig, de drives zijn zo goed als uitwisselbaar met bestaande 5.25 inch drives. Er zijn echter een paar verschillen.

- De connectoren zijn verschillend. De voedingsconnector van de 5.25 inch drive is een moeilijk verkrijgbaar geval met vier holle pennen die resp. zorgen voor +12V, gnd, gnd, +5V. Bij de 3.5 inch drives is deze connector vervangen door een type waarop zelfs de standaard flatcable connectors passen. De aan te bieden spanningen zijn identiek aan die van de 5.25 inch drive, het stroomverbruik kan flink lager zijn.

- Voor de interface-connector geldt hetzelfde, de 5.25 inch connector is een zogenaamde 'edge' type, een connector die op de rand van een printplaat geschoven moet worden. Bij de 3.5 inch drive is deze connector vervangen door een standaard flatcable connector van een paar gulden (34 pin).

- De interface-signalen komen bijna overeen. Alle oneven pennen liggen aan massa. Het enige probleem dat we tegenkwamen was het ready signaal (pin 34). Dit signaal geeft aan dat er een diskette in de drive aanwezig is en dat de drive "op toeren" is. Onder DOS-65 en op de MC68000 verwacht de controller een ready signaal en deze werd niet door een 3.5 inch drive afgegeven. Op de TEAC fd35fgn blijken een aantal strap-mogelijkheden aanwezig en bij één van die mogelijkheden staat 'RY'. Door hier een kortsluitstripje op te steken, wordt het probleem verholpen en geeft de drive het Ready-signaal af. Op een ander type drive, de ALPS S/N1A zijn geen strapmogelijkheden gevonden. Deze drives zijn ingebouwd in de MC68000 van Gert van Opbroek. Door in dit systeem op de controller een verbinding te maken tussen gnd en pen 34, is hier het probleem verholpen. De drive geeft nu altijd Ready en dat werkt.



3.5" diskdrive, ook in de PC

In het volgende overzicht zijn de diverse signalen weergegeven.

De interface signalen:

2		reserved
4	in	in use / head load
6	in	drive select 3
8	out	index / sector
10	in	drive select 0
12	in	drive select 1
14	in	drive select 2
16	in	motor on
18	in	direction (track steppermotor)
20	in	step
22	in	write data
24	in	write gate
26	out	track 00
28	out	write protect
30	out	read data
32	in	side select
34	out	ready

Alle oneven pennen liggen aan massa.
In = signaal van controller naar drive.

De voedingsplug:

1	+ 12 Volt
2	massa
3	massa
4	+ 5 Volt

Voor mensen die op een eenvoudige manier hun 5.25 inch drives uit willen wisselen voor een 3.5 inch drive, zijn er bij de meeste merken zogenaamde inbouwframes te koop. Een 3.5 inch drive in een dergelijk frame heeft dezelfde afmetingen als een 5.25 inch drive. Verder verzorgt het frame ook de overgang van de connectoren voor een 5.25 drive naar die van een 3.5 inch drive.

Een 3.5 inch drive heeft verschillende voordelen. Ze is kleiner dus gemakkelijker (verticaal!) in de kast te passen, heeft minder stroomopname en is vooral geruislozer. Bovendien zijn de floppy's wat handzamer.

Een MS-DOS machine heeft meestal de beschikking over 360 kB of over 1.2 MB 5.25 inch floppy drives. Wisselt men een 360 kB drive in voor een 3.5 inch drive, dan ruilt men meestal een 40 tracks drive in voor een 80 tracks drive. Bij de XT-compatible systemen, kan men dan 720 kB in plaats van 360 kB op een floppy kwijt. Voor de 1.44 MB die bij een AT-compatible systeem op een 3.5 inch floppy geschreven kan worden heeft men een speciale drive nodig. Bovendien wordt dit formaat niet door de XT ondersteund. Wil men een 3.5 inch floppy drive in een XT-compatible systeem gebruiken, dan moet men (bij mijn weten) een MS-DOS versie 3.0 of hoger gebruiken. Bovendien moet in de CONFIG.SYS-file deze drive gespecificeerd worden. Voor een echte kloon is er echter ook een andere mogelijkheid. Het door Nico de Vries ontwikkelde BIOS is in staat een eventuele 3.5 inch drive te herkennen en definieert deze dan ook als een drive van 720 kB. Dit BIOS is via de penningmeester te bestellen (zie elders in dit nummer).

Ernst Elderenbosch

Over objecten

"Object georiënteerd" is de kreet waarmee veel nieuwe implementaties van allerlei verschillende programmeertalen worden aangekondigd. Maar wat betekent die zinsnede nu eigenlijk en wat brengt "object georiënteerd" nu voor voordelen met zich mee?

Iedereen weet wat er met het woord "object" nu eigenlijk bedoeld wordt. Om de betekenis van een dergelijk woord exact te definieren is echter een stuk moeilijker. Het woordenboek der Nederlandse taal van Koenen omschrijft het woord object als een voorwerp, zaak of persoon der beschouwing of handeling. Of... als een voorwerp. In de computertechniek gebruiken we de term "object" eigenlijk te pas en te onpas om allerlei uiteenlopende zaken aan te duiden. Wel, de term "object georiënteerd" is deze keer eens wat duidelijker dan meestal het geval is in de informatica. Het gaat om een taal, operating system, database of een willekeurig ander computergereedschap waarbij de objecten tijdens de bouwfase van het behandelende programma niet bekend hoeven te zijn. "Leuke theoretische omschrijving", zult u nu misschien denken. Een voorbeeld zal het één en ander waarschijnlijk wel verduidelijken.

Men neme een willekeurige IBM-compatible PC. Op die PC draait als operating system "MS-DOS". De gebruiker van die PC kan het commando "type" gebruiken om de inhoud van een bestand te zien. Wil hij echter een directory bekijken, dan moet'ie het commando "dir" gebruiken. Dit noopt de beginnende gebruiker nogal eens tot wanhoop, zeker als hij nog niet goed bekend is met de verschillen tussen een directory en een ASCII- of een binair bestand. Was MS-DOS nu opgezet als object-georiënteerd operating system, dan bestonden deze problemen niet. Hij kon dan bijvoorbeeld een commando "SHOW" gebruiken voor al deze problemen. De werking van dit commando is dan afhankelijk van het object waarop het betrekking heeft. Het object geeft op aanvraag informatie over zichzelf. Het show-request zou bij een directory resulteren in een stroom ASCII-tekens die de inhoud van de directory weergeven. Een zelfde request aan een ASCII-bestand geeft de inhoud van het ASCII-bestand als stream. Een binair bestand zou dan bijvoorbeeld het request honoreren met een stroom ASCII-tekens die de hex-dump van dat bestand vertegenwoordigt. De enige taak van het commando is dan nog de stream op redelijke wijze naar het output-device te sturen. Een commando als "SHOW A:" geeft dan dus de current directory van drive A: weer, "SHOW \AUTOEXEC.BAT" geeft de inhoud van een bestand met de naam "\AUTOEXEC.BAT". Het interpreteren van parameters is daarmee ook een stuk beter georganiseerd. In het commando "DIR /P" geeft de /P aan dat er na iedere volle pagina gewacht moet worden tot een toets ingedrukt is. "TYPE /P" resulteert helaas in een foutmelding...

```
{ Onderstaande definities vormen de basis van de symbol table. Een symbol kan verschillende waarden
aannemen met verschillende betekenissen. De field-identifier "LblType" geeft aan wat voor betekenis aan
het symbol gehecht moet worden. }
TYPE    Size = (Byt,Wor,Double);              { Definieer byte, word en Doubleword classes }
        SymbolPtr = ^Symbol;
        Symbol = RECORD                       { Entry in de symbol table }
           Name : string[12];                 { Naam van het symbol, max. 12 chars }
           Next : SymbolPtr;                  { Pointer naar het volgende symbol }
           CASE LblType : Byte OF
              1 : (Adres : Longint);          { Een adres is een 32 bits waarde }
              2 : (Equate : LongInt);         { Een constante waarde }
              3 : (CType : Size;              { Pre-defined variabele }
                   CValue : Pointer;          { Pointer naar waarden }
                   CAdr : LongInt);           { Adres van het gereserveerde geheugen }
              4 : (MemType : Size;            { Storage allocation }
                   MemSize : Word;            { Grootte van gereserveerde geheugen }
                   MemAdr : Longint);         { Bijbehorende adres }
              5 : (Macro : Pointer);          { Pointer naar een macro }
           END;                               { CASE & RECORD }
```

Fig. 1: semie-RECORD definitie van een object

Programmeren

Object georiënteerd programmeren lijkt nieuw, maar is eigenlijk al zou oud als de weg naar Rome. We hoeven daarbij alleen maar de denken aan het pascal-statement "write" of "writeln" (BASIC: print). Het statement "writeln(value)" geeft, naar gelang het type van de gebruikte variabele, heel verschillende resultaten. Op het moment dat het om een character gaat wordt de geretourneerde waarde van de variabele heel anders geïnterpreteerd dan als de variabele als integer gedeclareerd was geweest. Het lijkt dus alsof er niets nieuws aan de taal toegevoegd is. De kneep zit 'em echter hier in, dat de compiler van tevoren precies weet hoe hij de waarde moet gaan interpreteren. Dat heeft hij netjes uitgezocht op het moment dat'ie het commando ging omzetten naar P-code of naar objectcode. Er is in dit voorbeeld dus niet echt sprake van object georiënteerd programmeren.

In de nieuwe versies van verschillende talen is er een speciaal object-type ingevoerd. Als voorbeeld nemen we TurboPascal 5.5. De decalaratie van een object ziet er op het eerste gezicht net zo uit als de definitie van een record. Het voorbeeld in figuur 1 is een record declaratie uit een assembler voor de Motorola 68000 die in TurboPascal geschreven is. Het zou zich uitstekend lenen om in een object te van-

```
; Dit programma demonstreert de werking van de symbol-table. Het leest een ASCII-Z string, converteert de ; string naar hoofdletters en voert de string uit via een system-call

; System calls
SYSTEM      EQU       1                    ; Function dispatcher van het systeem
DOSRET      EQU       0                    ; Return to DOS function code
STROUT      EQU       9                    ; String output function code
START
            LEA.L     STRING,A0            ; A0 naar tekst
            LEA.L     STORAGE,A1           ; A1 naar resulterende tekst
            BSR       UPCASE               ; Converteer naar uppercase
            LEA.L     STORAGE,A0           ; A0 naar resulterende tekst
            BSR       DISPLAY              ; Geef resultaat weer op scherm
            MOVE.W #DOSRET,-(SP)
            TRAP      #SYSTEM              ; Einde programma, terug naa DOS
; Converteer string naar hoodletters
UPCASE
            MOVE.B    (A0)+,D0             ; Character - D0
            CMPI.B    #'z',D0              ; D0 'z'?
            BHI.S     STORECHAR            ; Ja, dus is het geen kleine letter
            CMPI.B    #'a',D0              ; D0 'a'
            BLT.S     STORECHAR            ; Ja, dus... geen kleine letter
            ANDI.B    #$5F,D0              ; Kleine letter - grote letter
STORECHAR
            MOVE.B    D0,(A1)+             ; En save het resultaat
            CMPI.B    #0,D0                ; Was het het einde van de string?
            BNE       UPCASE               ; Nee, ga verder met het volgende char
            RTS                            ; Klaar, return to caller
; Display tekst
DISPLAY
            MOVE.L    A0,-(SP)             ; Tekst adres op de stack
            MOVE.W    #STROUT,-(SP)        ; String out functie
            TRAP      #SYSTEM              ; Display via DOS
            ADDQ.L    #6,SP                ; Corrigeer de stackpointer
            RTS                            ; En return to caller
STRING      DC.B      'Hallo wereld!',0    ; Tekstje voor uitvoer
STORAGE     DS.B      80                   ; 80 bytes werkruimte
            END
```

Fig. 2: Mogelijke assembler labels in de praktijk

gen. Het gaat hier om een record dat een label definieert. Een label kan betrekking hebben op de in figuur 2 gebruikte assembler-structuren.

De assembler kan een list-file aanmaken met een cross-reference tabel aan het einde van de listing. Daarin wordt niet alleen de waarde van het label opgenomen, maar ook een korte omschrijving van het label. De voorgaande, als voorbeeld gebruikte assemblerlisting resulteert in de cross-reference tabel van figuur 3. Hiervoor wordt in de assembler een speciale routine gebruikt die steeds uitzoekt wat er nu eigenlijk geprint moet gaan worden. Veel handiger zou het zijn als we een object konden definiëren dat zelf "weet" hoe het de waarde van het object geprint moet worden. De assembler, die geschreven is in TurboPascal 5.0, maakt gebruik van een truuk. Het record werd uitgebreid met een pointer naar een functie die de waarde van het record zelf interpreteerde en het resultaat als een string teruggaf aan het aanroepende programma. Stiekum dus toch een beetje object-georiënteerd? Inderdaad. Stiekum toch... De nieuwe implementatie van TurboPascal (versie 5.5) lost het echter allemaal een stuk eleganter op dan ik dat in mijn assemblertje heb gedaan. Het gebruik van objecten komt in dit geval met name de leesbaarheid ten goede, wat zeker belangrijk is vanuit onderhoudstechnisch oogpunt. Het is wellicht toch de moeite waard om eens te gaan experminteren met objecten; zeker als je van plan bent om ooit nog eens een data-base achtige applicatie te gaan bouwen.

Joost Voorhaar.

```
*** ASM68k Crossassembler, version 1.02
Copyright (C) J.Voorhaar, 1988-1990

Symbol table
Symbol Name   Type      Value
------------------------------------------------------------------------
@CPU          TEKST     "MC68000 "
@DATE         TEKST     "12-02-90"
@FILENAME     TEKST     "EXAMPLE1.ASM"
@SYSTEM       TEKST     "NEC V20"
@TIME         TEKST     "15:13:40"
DISPLAY       Code      0000003C
DOSRET        Number    0
START         Code      00000000
STORAGE       Byte      00000056
STORECHAR     Code      00000032
STRING        Byte      00000048
STROUT        Number    9
SYSTEM        Number    1
UPCASE        Code      00000020

Lines assembled: 52
None of the errors was found.
```

Fig. 3: De gegenereerde cross-reference lijst

Een ramdisk voor DOS-65

Het DOS65 systeem heeft heel veel ingebouwde mogelijkheden. Zo kun je standaard 4 diskdrives aansluiten, floppy, winchester of virtual disk. De drivers die nodig zijn om floppy drives aan te sturen zijn al aanwezig en ook voor de virtual disk, die het gewone geheugen van de 6502 gebruikt (weliswaar alleen het gedeelte boven 56k), is de complete software ondersteuning al 'ingebakken' in dos65. Alleen voor de winchester disk zijn de drivers nog niet standaard aanwezig. Als het dos voor winchester geassembleerd wordt, moeten hierin twee adressen vastgelegd worden van de twee driver routines. Deze twee routines zorgen resp. voor het schrijven en voor het lezen van een sector van harde schijf. Verder wordt het aan de user overgelaten hoe die routines er uit moeten zien.

Ik heb voor mijn systeem gekozen voor een leesroutine die begint op $e200 en een schrijfroutine met een startpunt op $e203. Op het moment dat dos65 een sector wil lezen of schrijven wordt het tracknr doorgegeven in het x register (max. 256 tracks) en het sectornr in het y register (max. 128 sectors). De meeste drives maken gebruik van 32 sectors per track (x 256 bytes).

De vrijheid die dos65 biedt om zelf diskdrivers te schrijven maken het systeem zeer flexibel. Zo is het helemaal niet zo moeilijk om een driver te maken die een harddisk simuleert terwijl gebruik wordt gemaakt van een heel ander medium, zoals een z80 kaart of zelfs een ander computersysteem. Ook een harddisk driver is niet moeilijk te maken, bijvoorbeeld voor een SASI drive of SCSI drive. In een volgend artikel wil ik graag de listing geven van een driver voor een SASI harddisk.

De z80 kaart heeft in de eenvoudigste uitvoering maar 64k. Hiervan kunnen we ruim 4k niet gebruiken want dit wordt ingenomen door het operating system van de z80. Er blijft dus krap 60 k over. Dit lijkt heel weinig, maar blijkt ruim voldoende voor een redelijk grote assembler source plus de vertaalde versie. De source van deze driver, de vertaalde versie, de listing en de assembler zelf vonden gemakkelijk een plaatsje in de z80 'harddisk'.

We zijn gewend om een floppy of harddisk eenmaal te formatteren en daarna nooit meer. Op de z80 kaart wordt gebruik gemaakt van dynamische rams en deze moeten telkens weer opnieuw geladen worden met zinnige informatie nadat het systeem uit geweest is. Daarom bevat het eerste deel van de software (zdriver.mac) een initialisatie routine die niet alleen de 6821 registers de juiste startwaarde geeft maar ook de eerste sectors van de z80 disk vult met een lege directory en systeem sector. Er zijn maximaal 239 blocks beschikbaar, inclusief de system en directory sectors. De z80 schijf is volledig als schijf aan te spreken, bijv. met discdoctor, dir, sdir, cat etc. Voor het initialisatie deel gebruik ik $0200-$0470. Dit kan op elke willekeurige plaats in het geheugen gelegd worden. Normaal wordt dit door login.com geladen en uitgevoerd, waarna het kan worden overschreven. De driver routines worden gelijktijdig door login.com geladen (het is gewoon een ander deel van het zelfde programma) maar deze moeten altijd beschikbaar blijven. Ze moeten dan ook geladen worden op een lokatie die niet door andere programma's overschreven wordt. Als je hiervoor bijv. $e400 kiest, dan hoef je verder niets anders te doen dan dit adres in de source file te zetten, het adres van de z80 kaart in de source file te zetten en een versie van dos65 te gebruiken waarin staat dat de harddisk drivers op $e400 beginnen. Deze versie is eenvoudig te maken als je de source files hebt (vraag het de dos65 distributeur of mij). Hierna werkt de z80 kaart zonder verdere hardware wijzigingen als ramdisk. Het systeem ziet 'm alleen als harddisk maar daar merk je verder niets van. Bij mij draait de z80 als drive 2.

Als je net als ik vindt dat elektuur wel wat slordig met de I/O adressen is omgesprongen dan stel ik een kleine wijziging voor. De tapekaart zit bij elektuur op $e280-$e283, de z80 kaart op $e300-$e303 en verder zijn deze twee pagina's leeg. Dat vond ik nogal slordig en daarom heb ik de tape naar $e150 gezet en de z80 kaart naar $e158-$e15b.

De vrijgekomen ruimte op $e200-$e3ff heb ik benut door een stukje ram van de cpu kaart op deze plaats te zetten (deze werd maar voor de helft gebruikt). Hiervoor heb ik een 74ls02 gebruikt op de volgende wijze: A9 naar pin 8, A10 naar pin 9, pin 10 naar pin 11 en naar pin 12, pin 13 naar ingang (pin 11) van n60 (74ls30). Deze ingang moet eerst worden losgemaakt van de 5 Volt. De 74ls02 heb ik met pin 7 en pin 14 bovenop n60 gesoldeerd en de rest van de pennen ingekort en opzij laten steken. De adreslijnen zijn dan vlakbij beschikbaar op de cpu print. Hiermee wordt de 6116 dan voor driekwart gebruikt ipv. voor de helft, maar belangrijker is dat er 512 bytes beschikbaar zijn die door geen enkel programma worden gebruikt. Dit is de ideale plaats voor bijvoorbeeld winchester drivers (of z80 kaart drivers).

De z80 kaart is nog eenvoudiger te modificeren (die kan nu niet op pagina $e3xx blijven zitten). Om deze op $e158 te krijgen kun je het volgende doen: Snij de verbindingen door naar ic23 (74ls688) pin 15, pin 6 en leg deze pennen aan +5Volt (pin 20). Snij de

verbinding door naar pen 18 en verbind pin 18 met 22a van de busconnector. Snij de verbinding door met pin 13 van ic.. en leg daarna pin 13 van ic.. aan 23c van de busconnector. De eerste twee onderbrekingen heb ik aan de soldeer zijde gemaakt en de laatste twee aan de componenten zijde van de print (eps 86238).

Na deze wijzigingen kun je de drivers laden op $e200 ipv. $e400 en de z80 kaart zit nu op $e158. Kijk wel even uit met een elektuur tape kaart. Deze moet nu ook gemodificeerd worden voor je 'm in de bus steekt, anders krijg je dubbele adressering.

Na het laden van zdriver.bin en het uitvoeren van de initialisering (met GO 0200) is de 'silicon disk' gereed voor gebruik. Enkele gemeten tijden om een indruk te geven van de snelheidsverschillen:

AS	zdriver.mac	zdriver.bin	tijd ca.(sec.)
floppy 1	floppy 2	floppy 2	13.25
floppy 1	z80 disk	z80 disk	5.70
z80 disk	z80 disk	z80 disk	3.80
LC (ipv.AS)	z80 disk	z80 disk	2.85

Hierbij werd de source van zdriver geassembleerd met de dos65 assembler op mijn 2 MHz 65C02 systeem naar een .bin file zonder listing. In het laatste geval werd gewoon LC gegeven, waarbij de assembler niet opnieuw geladen wordt, maar slechts opnieuw wordt gestart. De tijden varieren in de praktijk nogal, hier geef ik de snelste tijden weer. Omdat de driver bij mij in het geheugengebied $e200-$e2ff geladen wordt, is het mogelijk om te meten m.b.v. exetime, dat op $e400-$e5ff draait. Het is mogelijk dat ik nog eens uitvis hoe de driver er uit moet zien om optimaal gebruik te kunnen maken van een 256k z80 kaart. Je zou dan ca. 240k 'silicon disk' tot je beschikking hebben. Wie weet. Voor nu in ieder geval: succes!

Ernst Elderenbosch

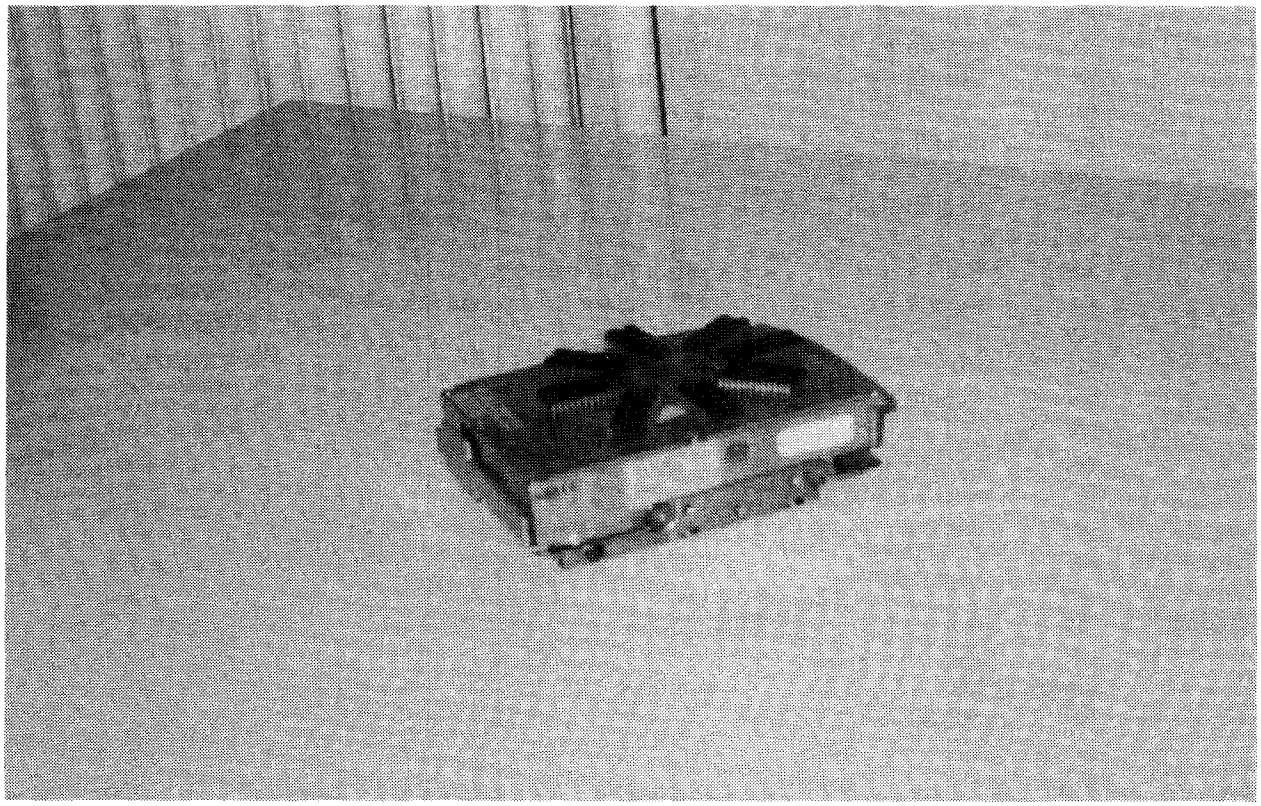
```
; file                  zdriver.mac
;
; purpose     use z80 pcb as harddisk
; system      DOS65 with Z80 card (64k or more)
;
; use                   copy zdriver.bin to disk 0:
;                       include in your login.com:
;
;                       LO S:ZDRIVER.BIN
;                       GO 0200
;
; size                  239 blocks x 256 bytes
;
; date                  21-jan-90           60k silicon disk (z80 card)
;
; author      Ernst R. Elderenbosch
;= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

;WARNING !!
;
;             The following two addresses are system dependant:
;             worg is a free memory area of 256 bytes. In a normal
;             dos65 system, you could use $e400-$e4ff. In that case,
;             do not use EXETIME or my window routines at $e400.
;   -         z80bas is normally $e300, the address of the z80 card.
;
worg          equ     $e200               ; drivers memory $e200-e3ff
z80bas        equ     $e158               ; z80 base address
;= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
zofs        equ     $0100           ; base address in z80 memory to be used
zsiz        equ     $ef00           ; silicon disk size
memp        equ     $e8             ; dos65 r/w memory pointer
prtbyt      equ     $d038           ; dos65 print accu hex routine
prtext      equ     $d03b           ; dos65 print text routine
; z80 commands
cread       equ     $03             read sector
cwrite      equ     $01             write sector
            org     $0200

silicon     jmp     iniz80          ; initialize z80 card
;
;           z80 card initialisation
;
iniz80      lda     #40
            ldy     #44
            ldx     #0              ; set port directions
            sta     z80bas+1
            stx     z80bas
            sty     z80bas+1
            dex
            sta     z80bas+3
            stx     z80bas+2
            sty     z80bas+3
            ldx     #20
1           lda     z80bas          ; purge z80 fifo buffer
            dex
            bne     1.b
            jsr     dirsec
            rts
;
dirsec      lda     memp
            pha
            lda     memp+1
            pha
            ldy     #1
            ldx     #0
            lda     #buffer1&255
            sta     memp
            lda     #buffer1>8
            sta     memp+1
            jsr     winwrite
            ldy     #2
            ldx     #0
            lda     #buffer2&255
            sta     memp
            lda     #buffer2>8
            sta     memp+1
            jsr     winwrite
            ldy     #3
            ldx     #0
            lda     #buffer2&255
            sta     memp
            lda     #buffer2>8
            sta     memp+1
            jsr     winwrite
            ldy     #4
            ldx     #0
```

```
            lda     #buffer2&255
            sta     memp
            lda     #buffer2>8
            sta     memp+1
            jsr     winwrite
            pla
            sta     memp+1
            pla
            sta     memp
            rts
;
buffer1
            fcc     $01,$02,$03,$04,$05,$06,$07,$08 ; 00
            fcc     $09,$0a,$0b,$0c,$0d,$0e,$0f,$10
            fcc     $11,$12,$13,$14,$15,$16,$17,$18 ; 10
            fcc     $19,$1a,$1b,$1c,$1d,$1e,$1f,$20
            fcc     $20,$08,$20,$20,$ff,$ff,$ff,$ff ; 20
            fcc     $01,$01,$00,$c0,$01,$01,$00,$00
            fcc     $00,$00,$00,$00,$00,$00,$00,$00 ; 30
            fcc     $00,$00,$00,$00,$00,$00,$00,$00
            fcc     $53,$69,$6c,$69,$63,$6f,$6e,$20 ; 40
            fcc     $64,$69,$73,$6b,$20,$7a,$38,$30
            fcc     $00,$00,$00,$00,$00,$00,$00,$00 ; 50
            fcc     $09,$08,$e9,$d8,$12,$34,$12,$34
            fcc     $7f,$ff,$ff,$ff,$ff,$ff,$ff,$ff ; bitmap 239 sectors
            fcc     $ff,$ff,$ff,$ff,$ff,$ff,$ff,$ff ;  = z80 mem 0100-efff
            fcc     $ff,$ff,$ff,$ff,$ff,$ff,$ff,$ff ;
            fcc     $ff,$ff,$ff,$ff,$fe ;
            res     bufend-*,0              ;
;
            org     buffer1+256
bufend
;
buffer2     res     256,0                   ; directory
;
bufend2
;
;           z80 card driver (read write sector)
            org     worg
winread     jmp     zread
winwrite jmp zwrite
woff        fdb     zofs                    offset address
wmax        fdb     zofs+zsiz               max address

wcmdb       fcc     0                       command buffer, 1=write, 3=read
            fcc     0,0                     z80 address default $100 (=zofs)
            fcc     1,0                     sector length 256 bytes
ytemp       fcb     0
;
;read sector from silicon drive
zread       lda     #cread                  ; get read sector command
            jsr     wrwsec                  ; start read sector procedure
            bcs     9.f                     ; carry set = error
            ldy     #0
read1       sty     ytemp
            jsr     getz80
            sta     [memp],y
            ldy     ytemp
            iny
```

```
            bne     read1
9           rts
;
;
;write sector to silicon drive
;
zwrite      lda     #cwrite         ; get write sector command
            jsr     wrwsec          ; start write sector procedure
            bcs     9.f             ; carry set = error
            ldy     #0
write1      sty     ytemp
            lda     [memp],y
            jsr     outz80
            ldy     ytemp
            iny
            bne     write1
9           rts
;
;
wrwsec      sty     ytemp
            sta     wcmdb           ; command (r/w)
            dey                     ; calculate address
            cpy     #32
            bcs     werr
            tya
            asla                    ; 32 sectors/track
            asla                    ; shift 3 bits left
            asla
            stx     wcmdb+1         ; rotate right and
            lsr     wcmdb+1         ; shift in right 3 bits
            rora                    ; of track nr (max ff)
            lsr     wcmdb+1
            rora
            lsr     wcmdb+1
            rora
            adc     woff            ; offset (this partition)
            sta     wcmdb+1         ; **** sector nr absoluut
            tay
            lda     #0
            sta     wcmdb+2
            tya
            sbc     wmax+1
            bcs     werr
            jsr     cmdout
            rts
                    ;
cmdout      ldy     #5
            ldx     #0
            cmd1    lda             wcmdb,x
            jsr     outz80
            inx
            dey
            bne     cmd1
            rts
;
werr        sec
            rts
;
getz80
```

```
1       lda     z80bas+1
        bpl     1.b
        lda     z80bas          ; get data byte
        rts
;
outz80  pha
1       lda     z80bas+3
        bpl     1.b             ; wait until last byte accepted
        lda     z80bas+2        ; reset pbc flag
        pla
        sta     z80bas+2        ; send byte to z80
        rts
;
res     fill-*,$ff
        org     $e2ff
fill
;
        end     silicon
```



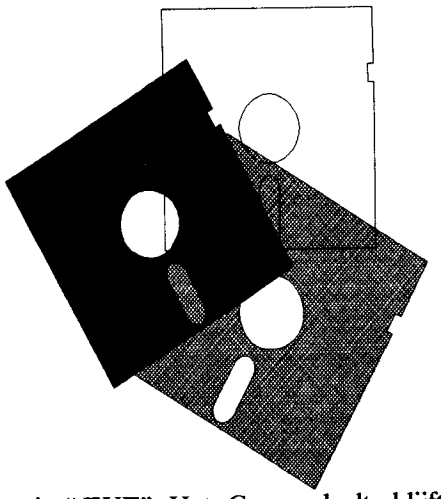
To Share Or Not To Share, That's The Question

Aj, een Engelstalige titel... nou ja, dat mag hopelijk uw aandacht gevangen hebben want dit is het eerste stukje van een nieuwe rubriek. "Een nieuwe rubriek?" Inderdaad, want als alles gaat zoals we zouden willen wordt dit een nieuw vast punt in de μP Kenner. Een rubriek waarin we de interessante nieuwe shareware en public domain programma's gaan bespreken die op het bulletin board van de vereniging binnengekomen zijn. Geen modem? Dat is jammer, maar de sysop van "The Ultimate", Jacques Banser, zal vast wel genegen zijn om aan speciale verzoeken gehoor te geven. Een diskette, een briefje erbij welke software en een enveloppe met postzegel en adres zou voldoende moeten zijn om toch in het bezit van de begeerde software te komen.

Helaas heb ik alleen de beschikking over een PC om programmatuur mee te testen. Omdat ik graag tegemoet wil komen aan de vraag om ook andere systemen aan bod te laten komen moet ik dus een beroep doen op u, de lezer dezes. Misschien heeft u wel een andere computer waarvoor ook leuke shareware en/of public domain software beschikbaar is. Schroom niet, en schrijf!

Shareware en public domain programma's... soms zitten er werkelijk juweeltjes tussen. Zoals het programma "4DOS", een vervanger van "COMMAND.COM", de command interpreter van MS- en PC-DOS. Het gaat niet om een shell waarin de gebruiker nog maar twee toetsjes in hoeft te tikken om WordPerfect op te starten. Nee, een gebruiker van 4DOS kan ook prima uit de voeten met een "kaal" MS-DOS systeem. Het gaat hier namelijk om een bijna exacte kloon van de oude vertrouwde commandline interface van DOS. En dan kan hij nog een beetje meer... Bij mij heeft de toepassing van 4DOS mijn harddisk al flink ontlast; er zijn een hele hoop kleine utilities die ik niet meer nodig heb. De opvallendste en handigste uitbreidingen betreffen zonder twijfel de uitbreidingen van het wild-card systeem. Stel dat alle files behalve die, die op .EXE eindigen weggegooid moeten worden uit de current directory. Met DOS is dat een ware crime. 4DOS kent voor dit probleem het commando "Except". De tegenhanger ervan is het commando "Global". Met dit commando kun je een commando laten uitvoeren over alle (sub-) directories. Het commando "Global del *.*" heeft dan meteen een mooie opruiming van uw harddisk tot gevolg...

4DOS bestaat, in tegenstelling tot Command.Com uit twee verschillende files. Een Com-file en één met



de extensie ".EXE". Het .Com-gedeelte blijft altijd in het geheugen van de PC zitten. Het andere deel wordt bij het laden van een applicatie netjes naar (hard-) disk weggeschreven. Op die manier is het mogelijk dat 4DOS veel meer kan dan Command.Com, en toch niet veel meer geheugenruimte inneemt tijdens het gebruik van een applicatie.

Verder zijn er nog verschillende versies voor de PC/XT en de AT. De AT-versie zal waarschijnlijk een stuk sneller zijn dan die, voor de XT. Deze AT-versie kan trouwens ook gebruikt worden indien men over een PC/XT beschikt met een V20 uP.

Het installeren van 4DOS gaat heel eenvoudig. In de Config.Sys file moet de regel "Shell=C:\4DOS.Com /P /E=1024" voorkomen. Helaas zet dit commando de ComSpec van het systeem dan nog niet goed. Daarvoor moet dan een regel in AutoExec.Bat opgenomen worden waarin deze naar de executable gezet wordt: "Set ComSpec=C:\System\4Dos88.Exe" en klaar is Clara!

Naast de genoemde commando's heeft u dan meteen de beschikking over een uitstekende commandline editor ("Ced" kan dus wel richting NUL-device), een lister (Hoppa, weg met "List"), een on-line popup DOShelp, en nog veel meer. Een aanrader van de eerste orde...

Versie 2.21 van 4DOS is beschikbaar op het bulletin board van de vereniging. U kunt hem terugvinden onder de naam "4DOS221.LZH" in de IBM-PC area.

J.Voorhaar

De IBM-PC en z'n klonen (Deel 7)

Inmiddels hebben we zo ongeveer alles in de systeemkast van de PC(/XT) bekeken, op de software na. Alle software in een PC(/XT) is opgeslagen in ROM. In de meeste klonen is dit gedaan in een EPROM. De software op het PC-moederboard bestaat bij een echte IBM uit twee delen: het systeem BIOS en de Cassette-BASIC. Dit laaste is een erfenis uit de oer-PC-tijd. De oer-PC had namelijk geen disk drives, maar wel een cassette-aansluiting, en: BASIC in ROM. Herinnert u dat niet onmiddellijk aan PET's, Apple's, C-64's en Acorn Atoms? Er was dus niet nieuws onder de zon. De BASIC was zoals gebruikelijk weer van Microsoft, zij het dat het een behoorlijk uitgebreide versie betreft. De klonen hebben zonder uitzondering wel (een) voetje(s) voor de Cassette-BASIC ROM(s) maar die zit er, om ruzie met IBM te voorkomen, nooit in. Tot zover de BASIC. De andere software is veel interessanter. Dat is:

Het systeem-BIOS

U heeft het natuurlijk al gezien: het woord BIOS wordt met uitsluitend hoofdletters geschreven. Het zal dus wel een afkorting zijn. Dat klopt. Het betekent: Basic Input/Output System. Het BIOS is dus het stuk programma dat er voor zorgt dat er invoer en uitvoer van gegevens kan plaatsvinden. Zoals het woord Basic al aangeeft betreft het hier alleen de functies op het allerlaagste niveau.

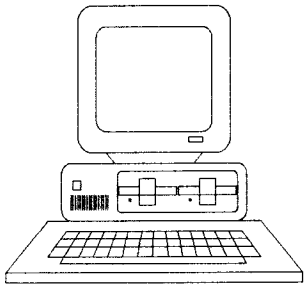
Het systeem BIOS is opgeslagen in een (EP)ROM van 8 kbyte, meestal een 2764. De (EP)ROM is en adresmap te vinden van adres F000:E000 t/m adres F000:FFFF. Binnen dit adresbereik valt, Du raadt het al, de reset routine op FFFF:0000. De software in de (EP)ROM is te verdelen in twee stukken:

- Het eigenlijke BIOS.
- De power-on initialisatie (reset).

Over het eigenlijke BIOS zullen we het in het volgende deel gaan hebben. Eerst gaan we kijken naar hetgeen de PC allemaal doet nadat hij is ingeschakeld, maar voordat de DOS-prompt verschijnt. En dat is heel wat.

De Power-On Self Test

De titel hierboven geeft al aan wat de belangrijkste activiteit van de PC is voordat de DOS-prompt verschijnt: zichzelf controleren. In de oudere IBM-documentatie is trouwens geen sprake van een POST



(de afkorting dus) maar van de Power-On Diagnostics of POD.

De POST zoals we hem verder maar zullen noemen, voert twee taken uit, die een beetje door elkaar heen lopen. De eerste taak is het controleren en testen van de aanwezige hardware. De tweede, daaraan verwante taak is het bepalen van de systeemconfiguratie.

Van de machinetest merkt de doorsnee-gebruiker eigenlijk niet zoveel: hij ziet alleen maar een RAM-test en in sommige gevallen wordt ook floppy drive A: getest. Als alles verder in orde is, wordt gewoon DOS geboot. Is er echter iets mis, dan gebeurt er meestal wat meer: er verschijnt minimaal een foutmelding op het scherm.

Sommige kloon-BIOSsen maken zich er met een Jantje van Leiden vanaf: met een gammele RAM-test heb je het dan wel gehad.

De machinetest verschilt zeer sterk van BIOS tot BIOS. Het IBM-origineel controleert werkelijk van alles: er wordt zelfs gecontroleerd of de CPU werkt! (Of dat zinvol is, waag ik nog altijd te betwijfelen: hoe kan een CPU nu zichzelf controleren? Als 'ie stuk is komt de machine sowieso niet overeind, doet 'ie het wel, dan is de kans op een defecte CPU meteen tot vrijwel nul gereduceerd.) Sommige kloon-BIOSsen maken zich er met een Jantje van Leiden vanaf: met een gammele RAM-test heb je het dan wel gehad. Om een indruk te geven wat een IBM BIOS allemaal doet een lijstje:

- CPU test (HALT)
- Disable alle video kaarten
- BIOS ROM checksum test (HALT)
- Refresh Timer 1 test (HALT)
- DMA controller test (HALT)
- Refresh timer initialisatie
- Test of er refresh optreedt (HALT)
- Enable expansion box

- RAM test, eerste 16k (HALT)
- Bepaling hoeveelheid RAM
- Stack initialisatie
- Interrupt controller initialisatie
- Initialisatie interrupt vectoren (dummy interrupt handler)
- Uitlezen moederbord switches
- Bepaling power-up video (CGA, MDA of EGA).
- Indien MDA/CGA, VRAM test (401/501)
- Video initialisatie, clear screen
- ROMscan van C000-C800 (EGA/VGA BIOS)
- Test op spontane interrupts (101)
- Systeemtijd timer test (101)
- Test of keyboard aanwezig (301)
- Test op vastzittende toetsen (301)
- Initialisatie interrupt vectoren (Echte routines)
- Enable timer0 (systeemtijd) IRQ
- Expansion box test (1801)
- Indien geen re-boot: uitgebreide RAM-test, print hoeveelheid op scherm. (201)
- ROMscan van C800-F600
- Checksum test BASIC ROMs
- Enable FDC interrupts
- Test floppy drive A: (601)
- Keyboard buffer setup
- Enable keyboard IRQ
- Print POST errors
- Bepaal printerpoorten
- Bepaal COM-poorten
- Test op joystickpoort
- Enable parity check
- Boot DOS.

Dit lijstje levert ongeveer 1.8 kbyte code op, eigenlijk best compact gezien de hoeveelheid werk die verzet wordt. Achter iedere test staat steeds wat er gebeurt als de test faalt. In de beginfase wordt de CPU stilgezet, maar verderop, als het video getest en geinitialiseerd is, wordt eenvoudig een getal afgedrukt dat de foutcode aangeeft. IBM vond het niet nodig de getallen uit te breiden met een voor de leek meer begrijpelijke tekstmededeling. Een heleboel zaken in de lijst zijn eigenlijk wel voor de hand liggend, zoals de intialisatie van de interrupt vectoren en de I/O chips op het moederbord.

De expansion box is allang uit de mode, en wordt in de meeste kloon-BIOSsen ook alleen maar ge-enabled, er wordt niet meer op getest. De expansion box is eigenlijk overbodig geworden met het verschijnen van de moederborden die 640 kbyte RAM kunnen bergen, en die acht slots hebben. Wordt dan ook nog een multi I/O kaart toegepast, dan zijn er altijd slots genoeg over.

In dezelfde volgorde als de POST-lijst zullen we nu naar een aantal zaken in detail gaan kijken.

ROM checksum

Om enige controle over de integriteit van de ROM-inhoud te hebben wordt al vroeg in de POST de checksum van de BIOS-ROM zelf, en later ook van de bij IBM altijd aanwezige BASIC ROMs bepaald. Om niet van alle ROMs de checksum te hoeven onthouden, is in ieder ROM een byte gereserveerd, dat gebruikt moet worden om de byte-checksum van het hele ROM precies nul te maken. In vrijwel alle BIOS ROMs wordt hiervoor het laatste byte gebruikt. In een loopje worden alle bytes van het ROM bij elkaar opgeteld, waarbij steeds de carry wordt vergeten. De som van alle 8192 bytes van het ROM moet dan nul worden. Ook de vier BASIC ROMs zijn ieder 8 kbyte groot, en hier geldt dezelfde procedure. Wordt een BIOS ROM checksum ongelijk nul gevonden, dan start de machine niet op.

De moederbord switches

Op het moederbord van een PC/XT zit een DIP switch met acht schakelaartjes, waarmee een aantal zaken worden ingesteld. De schakelaartjes worden door het BIOS uitgelezen via de 8255 en in een geheugenlocatie bewaard. De schakelaartjes hebben de volgende betekenis:

1 ON: Loop POST, OFF: Boot DOS
2 ON: Geen 8087 aanwezig
3 & 4: Hoeveelheid RAM op het moederbord, zie onder
5 & 6: Power-on default video display, zie onder
7 & 8: Aantal floppy disk drives, zie onder

Hoeveelheid RAM op het moederbord:

3	4	
ON	ON	64 kbyte
OFF	ON	128 kbyte
ON	OFF	192 kbyte
OFF	OFF	256 kbyte of meer

Default power-on display:

MDA	5 OFF	6 OFF
CGA 80 koloms	5 ON	6 OFF
CGA 40 koloms	5 OFF	6 ON
EGA, VGA of ander	5 ON	6 ON

De IBM documentatie geeft verder aan, dat wanneer er een MDA en een CGA in 1 machine zitten, dat dan de MDA als power-up display moet worden gekozen, om schade aan de MDA monitor te voorkomen.

EGA en VGA kaarten initialiseren verdere display-adapters in de machine zelf (zie bij ROMscan), zodat dan switches 5 en 6 beide op ON moeten staan, ook bij een EGA/VGA en een MDA adapter in het systeem.

Aantal floppy disk drives:

1	7 ON	8 ON
2	7 OFF	8 ON
3	7 ON	8 OFF
4	7 OFF	8 OFF

Deze twee switches zijn alleen van betekenis als switch 1 op OFF (boot DOS) staat. Ze bepalen onder meer de drive letter van de eerste winchester in het systeem. (De laagste driveletter voor een winchester is altijd C:, ook bij 1 floppy drive).

In een oude PC zitten twee DIPswitches, waarvan de tweede gebruikt wordt om de hoeveelheid geheugen dat in slots zit, in te stellen. De oer-PC komt echter zo weinig voor dat dit een beetje buiten het bestek van dit verhaal valt.

ROM Scanning

Dit is weer iets nieuws. Door de opzet van BIOS (zie deel 8) is het mogelijk BIOS routines te vervangen door andere, of om het BIOS uit te breiden. Dit wordt gedaan door extra ROMs in de memory map op te nemen die ook BIOS functies bevatten. Er zijn twee geheugengebieden gereserveerd voor dergelijke ROMs, die aaneensluitend zijn:

C0000 t/m C7FFF - Extra video BIOS
C8000 t/m F5FFF - Ander extra BIOS

Men kan zoveel extra BIOSsen opnemen als men zelf wil. Het systeem BIOS spoort de extra BIOSsen in de machine op door de zogenaamde ROM scan.

De oer-PC bezit een beperktere ROM scan, waardoor in deze machines geen EGA kaarten kunnen worden gebruikt.

De ROM scan wordt mogelijk, omdat een extra BIOS ROM aan twee criteria moet voldoen:

Het woord op adres 0 moet 55AAH zijn
De checksum van het ROM moet nul zijn

Om te weten hoeveel bytes er nodig zijn voor de checksum, staat in het byte op adres 2 in het ROM, groot het ROM is, in eenheden van 512 bytes.

De ROM scan geschiedt als volgt. Eerst wordt het woord op adres C0000/C8000 gelezen. Staat hier 55AAH, dan wordt het byte op adres C0002/C8002 gelezen. Vervolgens wordt uitgerekend wat het eindadres van het ROM is, en de checksum bepaald. Is de checksum nul, dan betreft het een geldig extra BIOS ROM, en wordt het ROM aangeroepen met een CALL FAR naar adres 3 in het ROM, zodat het zijn extra functies kan initialiseren. In dit geval wordt na terugkeer van de CALL FAR met de ROM scan verdergegaan op het eindadres van het gevonden ROM plus 1.

Wordt op adres C0000/C8000 niet 55AAH gevonden dan wordt het huidige adres (C0000/C8000) met 2k (0800) verhoogd, en wordt de ROM scan voortgezet.

Is de checksum niet nul, dan wordt het ROM niet aangeroepen, doch wordt de ROM scan voortgezet op het berekende ROM eindadres plus 1.

Het hele ROM scan spelletje wordt volgehouden totdat het eindadres van een bepaald gebied (C8000 of F6000) bereikt is.

Twee typische voorbeelden van extra BIOSsen:
Op de EGAkaart zit een BIOS dat alle videofuncties overneemt van het systeem BIOS, maar dan speaciaal toegesneden op de EGA kaart. Hierbij ondersteunt het EGA BIOS automatisch een tweede displayadapter in het systeem, die dan meestal een MDA of Hercules kaart is. Via schakelaartjes op de EGAkaart wordt dan bepaald welk display het power-up display is.

Op iedere winchestercontrollerkaart voor een PC(/XT) zit ook een extra BIOS, dat de diskroutines voor de harde schijf bevat.

Andere kaarten die een extra BIOS kunnen hebben zijn LIM-EMS geheugekaarten en speciale floppy controllers die 1.2M en 1.44M floppy drives in een PC/XT aankunnen.

De oer-PC bezit een beperktere ROM scan, waardoor in deze machines geen EGA kaarten kunnen worden gebruikt. Er is voor deze machines een update voor het BIOS verkrijgbaar om dit wel mogelijk te maakt.

Keyboard test

Het toetsenbord heeft, zoals in deel 6 uitgelegd, een eigen ingebouwde test, die geactiveerd wordt door zowel de clock- als datalijn gedurende twintig milliseconde laag te maken. Dit is dan ook precies wat er in het BIOS gebeurt. Na dit laag maken van deze twee lijnen gaat het BIOS wachten op een toetsenbord interrupt, met daarop een time out. Verloopt de time out, dat verschijnt er een '301' fout, die dan

geinterpreteerd moet worden als 'toetsenbord niet aanwezig'.

Is er wel een interrupt opgetreden, dan heeft het toetsenbord een scancode gestuurd, die in principe de afloop van de ingebouwde test aangeeft. Is de scancode niet AA, dan verschijnt ook de '301' fout: maar nu betekent het 'toetsenbord defect'.

Is de scancode wel AA, dan wordt de toetsenbordtest voorgezet met het opnieuw wachten op een interrupt van het toetsenbord, wederom met een time out. Omdat het toetsenbord thans geen activiteit behoort te ontplooien, dient een interrupt uit te blijven en de time out te verstrijken. Gebeurt dit, dan is de test geslaagd.

Treedt er toch een toetsenbord interrupt op, dan wordt de scan code gelezen en in hex op het scherm gezet, gevolgd door de bekende '301'. In geval wordt aangenomen dat de toets met de getoonde scancode vastzit en spontaan aanleiding is tot scancode transmissie.

Geheugentest

De geheugentest is het meest in het oog lopende deel van de POST, en meestal ook de meest tijdrovende. Een originele IBM test zijn geheugen buitengewoon grondig. Een 640 kbyte machine is dan ook bijna anderhalve minuut met de geheugentest bezig. Nu wordt daar weleens op gevloekt, maar men dient zich te bedenken, dat een machine die start met een geheugen dat niet 100% betrouwbaar is, ook niet nuttig is.

Een originele IBM test zijn geheugen buitengewoon grondig. Een 640 kbyte machine is dan ook bijna anderhalve minuut met de geheugentest bezig.

De geheugentest van de eerste 16 kbyte RAM geschiedt helemaal voorin de POST. Dit is nodig om er zeker van te zijn, dat er plaats is voor een stack, voor de interruptvectoren en werkgeheugen voor het BIOS. Deze test is geen aanleiding tot mededelingen op het scherm, want dat is nog niet geinitialiseerd. Als de basis 16 kbyte RAM blijkt te werken, wordt bepaald hoeveel RAM geheugen de machine bevat. IBM gebruikt hiervoor een methode die ook dubbeladressering kan ontdekken. Vrijwel alle niet IBM BIOSsen ontdekken dubbeladressering niet. De gevonden hoeveelheid wordt bewaard in een geheugenlocatie in het BIOS werkgeheugen.

Nadat het video is geinitialiseerd en het toetsenbord is getest, wordt aan de grote geheugentest begonnen, te beginnen bij 16 kbyte, want de eerste 16 kbyte waren al getest en bevatten stack en werkgeheugen. Het geheugen wordt in stappen van 16 kbyte gevuld met 0FFh, 0AAh, 055H en 0, en vervolgns weer teruggelezen. Blijkt een blok te werken, dan wordt de nieuwe totale hoeveelheid RAM op het scherm gemeld.

Wordt echter een fout gevonden, dat wordt de RAMtest gestopt, en verschijnt de '201' foutmelding, gevolgd door het adres waar het misging.

Merk op, dat dit alles meteen ook alle pariteitsbits in het RAM goed zet. De pariteitcontrole wordt echter pas aan het einde van de POST ingeschakeld. Tenslotte is het zo, dat bij een re-boot (met Ctrl-Alt-Del) de geheugentest niet wordt gedaan: de routine die de hoeveelheid geheugen bepaalt in het begin van de POST zorgt er in zo'n geval voor dat het gehele RAM gevuld wordt met nullen.

Floppy drive test

De floppy disk drive test heeft als doel om te bepalen of de floppy disk controller er is, of deze werkt en interrupts genereert, en of er tenminste 1 drive is (drive A:) waarvan geboot kan worden.

De test verloopt als volgt. Eerst wordt de floppy disk controller geinitialiseerd met de benodigde drive parameters, waaronder de step rate en de head-settling tijd. Antwoordt de controller niet met een interrupt, dan verschijnt er '601' op het scherm: de controller is er niet of is defect.

Gaat de initialisatie echter goed, dan wordt een recalibrate commando gegeven voor drive A:, waarbij niet gelet wordt op het drive ready signaal. De kop van drive A: wordt dan, onafhankelijk van het feit of er een disk in de drive zit, op track nul gezet. Meldt de controller nu geen track0 vanaf de drive dan is de drive defect, of niet aangesloten. Ook dit geeft aanleiding tot een '601'-melding.

Reageert de drive wel, dat wordt de kop nog naar track 20 verplaatst om een mogelijk op track nul vastzittende kop los te laten schrikken.

Bepaling verdere systeem configuratie

Als laatste wordt, na de tweede ROM scan, bepaald hoeveel printer- en RS-232 poorten er in de machine te vinden zijn, en of er een joystick adapter is.

De test op een printerpoort is nogal triviaal: er wordt gekeken of de printer-datapoort geschreven, en ook weer teruggelezen kan worden. Het BIOS zal dit proberen voor drie verschillende adressen: 03BCh, 0378h en 0278h. Ofschoon ze in hardware manuals wel consequent zo genoemd worden, liggen de printers LPT1, LPT2 en LPT3 niet noodzakelijk op die addressen.

De adressen worden in de genoemde volgorde gescand. De eerst gevonden poort wordt toegewezen aan LPT1, de tweede aan LPT2, en als er een derde poort wordt aangetroffen, dan wordt dit LPT3. Merk op, dat poort 03BCh alleen in het systeem voorkomt als er een MDA of Hercules display kaart is gemonteerd. Zit zo'n kaart in de machine, dan is de printerpoort daarop dus altijd LPT1. Dit ligt anders bij een poort op adres 0378h. Is er geen MDA/Hercules in het systeem, dan is deze poort LPT1, anders wordt hij LPT2. Een analoog verhaal geldt voor de poort op 0278h. Is het de enige poort, dat wordt hij LPT1. Is er een MDA/Hercules, maar geen poort op 0378h, dat is de poort LPT2. Hetzelfde geldt als de poort op 0378h wel in de machine zit, maar er geen MDA/Hercules kaart is. Tenslotte wordt de poort LPT3 als zowel een monochrome display adapter als een poort op 0378h in de machine voorkomen. Even puzzelen soms, maar als men weet hoe de LPT-telling tot stand komt, is het toch echt niet moeilijk.

Vrijwel analoog zijn de gebeurtenissen rond de RS-232 poorten COM1 en COM2. Het BIOS zal hier de adressen 03F8h en 02F8h scannen op aanwezigheid van een poort. De test is zo mogelijk nog simpeler dan bij de printer: er wordt in het statusrgister gelezen, en gebruik gemaakt van het feit dat bepaalde bits hiervan altijd nul zijn. Ook hier wordt de eerst gevonden poort COM1. In de praktijk is dit adres 03F8h. Wordt een tweede poort op 02F8h gevonden, dan wordt die COM2. Is de poort op 02F8h de enige in de machine, dan is hij weer COM1.

Oplettende lezers zullen hebben opgemerkt, dat in dit verhaal COM3 en COM4 niet voorkomen. Dat klopt. De PC(/XT) en ook de AT kennen officieel maar twee mogelijke RS-232 poorten. Pas bij de introductie van de PS/2 modellen verscheen IBM met machines die een derde en een vierde COM-poort konden hebben. In deze machines is het derde gescande adres 03E8h, het vierde 02E8h. Sommige recente BIOSsen scannen ook deze addressen.

Ook de joystickpoort wordt herkend doordat bij diens aanwezigheid een aantal bits van poortadres 0201h nul zullen zijn. En daarmee is de gehele configuratie bepaald.

Hardware Interrupts

Er is dit verhaal al heel wat keertjes het woord interrupt gevallen. Hiermee werd tot dusver steeds een hardware interrupt bedoeld. De hardware interrupts zitten in een PC(/XT) zo in elkaar:

IRQ0:	Timer0, systeemtijd (18.2/s)
IRQ1:	Toetsenbord, elke toetsactie
IRQ2:	Gereserveerd
IRQ3:	RS-232 poort op 02F8h ('COM2')
IRQ4:	RS-232 poort op 03F8h ('COM1')
IRQ5:	Winchester controller
IRQ6:	Floppy disk controller
IRQ7:	Parallel printer

Zoals eerder vermeld in de serie wordt de NMI gebruikt om pariteitsfouten te melden.

De volgende keer...

Gaan we het hebben over die andere interrupts, de software interrupts. Want daarmee kun je het BIOS aan het werk zetten. Let maar op.
Tot dan.

Nico de Vries

Methoden en technieken voor datacommunicatie

Inleiding

In het algemeen zal men op een computer zogenaamde randapparatuur aan willen sluiten. Door middel van deze randapparatuur kan de computer communiceren met zijn omgeving en aangezien een computer met data omgaat, heet dit datacommunicatie. Hoewel strikt genomen een beeldscherm en toetsenbord ook randapparaten zijn, wil ik deze in dit artikel buiten beschouwing laten.

Behalve het communiceren met randapparaten, beslaat datacommunicatie een veel groter gebied. Ook het uitwisselen van gegevens tussen computers onderling valt onder datacommunicatie. Datacommunicatie beslaat het gebied vanaf het uit laten printen van documentjes op een printer tot en met de grote internationale netwerken, al dan niet met sattelietverbindingen. In dit artikel zullen we ons echter voornamelijk bezig gaan houden met de meer eenvoudiger vormen van datacommunicatie.

Het coderen van lettertekens

Binnen een computer worden gegevens opgeslagen in de vorm van bits, dit zijn elementen die de waarde 0 of 1 kunnen bevatten. In een computer wordt de waarde van een bit weergegeven door een spanning van +5 Volt (1) of door een spanning van 0 Volt (0). Deze bits worden samengevoegd tot rijtjes van 8 bits, een byte. Uit bytes kunnen nu weer nieuwe groepen bits opgebouwd worden, bijvoorbeeld een groepje van 2 byte of 16 bits etc.

Door 8 bits te gebruiken, kunnen 256 verschillende combinaties van 0 en 1 aangegeven worden, te beginnen bij 00000000 en eindigend op 11111111. Gebruikt met slechts 7 bits, dan zijn dit 128 verschillende combinaties.

Als we nu door een printer letters af willen laten drukken, dan ontstaat er een probleem. Een computer kan slechts omgaan met 0 en 1 en degene die de uitvoer moet lezen, wil graag dat er gebruik gemaakt wordt van letters. Hiervoor heeft men afgesproken dat voor elke letter een bepaalde code van 7 bits geldt. Bovendien heeft men aan de normale, printbare letters een aantal niet-printbare (besturings-) tekens toegevoegd. Dit zijn bijvoorbeeld het "Wagen Terug" en "Papier Opvoer" teken (Carriage return en Line feed). In tabel 1 is de belangrijkste standaard voor de codering weergegeven. Dit is de zogenaamde ASCII (Naar American Standard Code for Information Interchange) die door vrijwel de hele computerwereld gebruikt wordt. Behalve deze standaard bestaat er ook nog de EBCDIC (Extended Binairy Coded Decimal Interchange Code) die door IBM voor zijn mainframes gebruikt wordt.

Door het sturen van het juiste bitpatroon naar een printer zal de printer het bijbehorende teken afdrukken of het bijbehorende commando uitvoeren. Zoals al is aangegeven, worden in computers de bits samengevoegd tot bytes, wat groepjes van 8 bits zijn. In de ascii-standaard worden per letter slechts 7 bits gebruikt. Om toch steeds een byte per teken te kun-

			b7	0	0	0	0	1	1	1	1
			b6	0	0	1	1	0	0	1	1
			b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1								
0	0	0	0	NUL	DLE	SP	0	@	P	‘	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	"	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	’	7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	–	=	M	]	m	}
1	1	1	0	SO	RS	.		N	^	n	~
1	1	1	1	SI	US	/	?	O	_	o	DEL

Tabel 1: De ASCII tekenset

nen gebruiken, wordt er meestal nog een bit voor gezet. Elk teken is dan 8 bits. Als gebruik gemaakt wordt van de 7-bits code, dan doet het er niet toe wat de waarde van dit bit is. In de praktijk wordt echter meestal gebruik gemaakt van een 8-bits codering. Dit wil zeggen dat als dit 8e bit een 0 is, de codering uit tabel 1 gebruikt wordt, dus de normale ascii-codering met een extra 0 er voor. Is echter dit voorste bit een 1, dan drukt de printer (of het beeldscherm) een ander teken af. Dit kunnen bijvoorbeeld grafische tekens zijn. Helaas bestaat er voor deze tekens geen standaard codering. Het hangt van de printerfrabrikant af welk teken een printer af zal drukken als hij de code 1001 0011 toegestuurd krijgt. Als er gebruik gemaakt wordt van een 7 bit codering, dan wordt het 8e bit vaak gevuld met het zogenaamde pariteitsbit. Hieraan kan bij bepaalde vormen van communicatie geconstateerd worden of de letter goed over gekomen is.

Paralelle overdracht

Als u op uw computer een printer aangesloten hebt, dan zal dat in de meeste gevallen via een paralelle aansluiting gebeurd zijn. Dat wil zeggen dat de gecodeerde lettertekens over een snoer naar de printer gestuurd worden waarbij er voor elk van de 8 bits een apart draadje gereserveerd is. Als de computer nu een teken naar de printer wil sturen, dan zorgt hij er voor dat op elk draadje de juiste spanning aanwezig is en geeft dan over een negende draadje een teken aan de printer dat er weer een letterteken voor

Pin No.	Signal	Direction	Description
1	DATA STROBE	To printer	Samples input data when changing from low level to high level.
2	DATA BIT 1	To printer	Indicate input data. High level indicates 1 and low level, 0.
3	DATA BIT 2		
4	DATA BIT 3		
5	DATA BIT 4		
6	DATA BIT 5		
7	DATA BIT 6		
8	DATA BIT 7		
9	DATA BIT 8		
10	ACKNOWLEDGE	From printer	Low level indicates character input completion, or function operation end.
11	BUSY	From printer	High level indicates data cannot be received. Low level indicates data can be input.
12	PAPER OUT	From printer	High level indicates paper end.
13	SELECT	From printer	High level indicates the select (online) condition.
14, 16, 33	0V	–	Signal ground
17	CHASSIS GROUND	–	Frame ground
18	+5V	From printer	+5V supply (50mA maximum)
19 to 30	0V	–	Return for the twisted-pair wires of pins 1 to 11
31	INPUT-PRIME	To printer	Controller is initialized at low level. Pulse width more than 5.0 ms.
32	FAULT	From printer	This signal changes from high to low level when printer runs out of paper.
15, 34, 35, 36		–	Unused

Note: Connector pin arrangement:

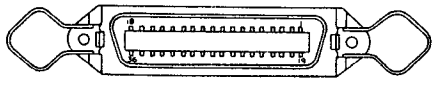


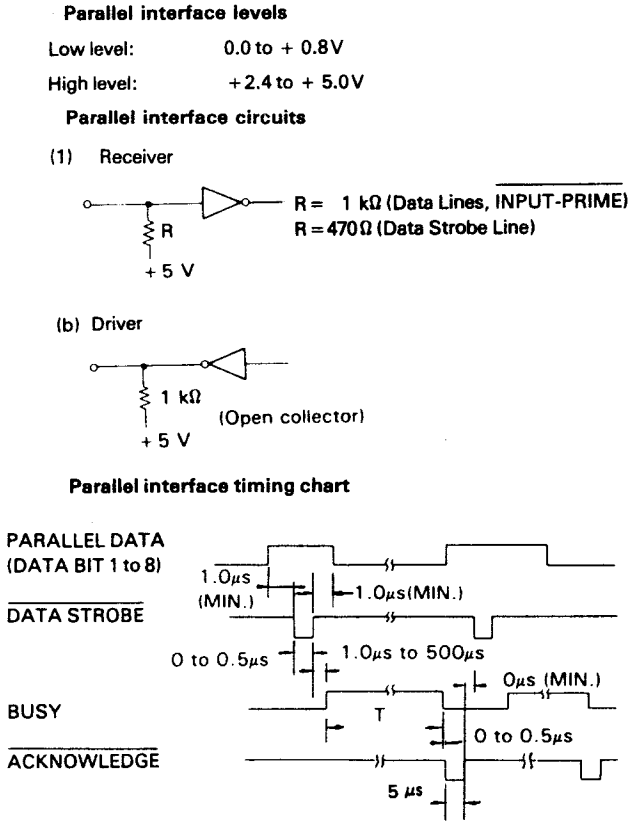
Fig. 1: Signalen van de Centronics definitie

hem gereed staat. Dit teken bestaat uit een omschakeling van 1 naar 0 en even later weer naar 1. Dit signaal wordt de data strobe genoemd. Behalve de bovengenoemde signalen zijn er in het snoer meestal ook nog een paar draadjes voor andere signalen aanwezig. Een voorbeeld hiervan is een signaal waarmee aangegeven wordt dat het papier op is. Het kenmerkende van een paralelle aansluiting is het feit dat er voor elk bit een apart draadje in het snoer aanwezig is. De bits worden dus paralel doorgegeven aan de printer. Een normale printerkabel bestaat uit ongeveer 30 draadjes waarvan de helft voor afscherming bedoeld zijn en geen signalen overbrengen. In figuur 1 zijn de signalen van de zogenaamde centronics standaard weergegeven die normaal gesproken voor het aansluiten van printers gebruikt wordt. Deze standaard maakt gebruik van een 36-polige connector. Verder zijn in figuur 2 nog enkele andere gegevens van de centronics standaard afgedrukt. De spanningen die bij een paralelle overdracht volgens de centronics standaard gebruikt worden, zijn + 5 Volt voor een 1 en 0 Volt voor een 0, dus precies zoals in de computer zelf.

Op IBM-compatibles worden printers meestal ook via een paralelle verbinding aangesloten. Hierbij wordt gebruik gemaakt van de centronics standaard waarbij aan de computerkant in plaats van een 36-polige connector een 25 polige (female) connector gebruikt wordt. De gebruikte signalen zijn echter hetzelfde.

Bij paralelle overdracht worden de bits als het ware naast elkaar gezet. Dit heeft als voodeel dat de overdracht van tekst naar een printer zeer snel gedaan kan worden, in principe wordt de overdracht begrenst door de snelheid waarmee de computer de tekens aan kan leveren. Daar echter een printer niet met deze hoge snelheid kan printen, heeft elke printer de beschikking over een buffer. Als dit buffer (bijna) vol is, wordt dit door middel van het BUSY-signaal aan de computer doorgegeven. Hoewel er in dit voorbeeld steeds wordt uitgegegaan van de overdracht van data van een computer naar een printer, zijn er nog legio andere voorbeelden van een paralelle overdracht van data. Paralelle overdracht wordt meestal ingezet als de te overbruggen afstanden kort zijn en de vereiste of gewenste snelheid

hoog is. Het nadeel van paralelle overdracht is het feit dat er zeer veel signalen gelijktijdig door het snoer moeten lopen en dat daardoor de kostprijs van de kabels (zeker bij de wat grotere afstanden) nogal hoog is. Bovendien kan een overdracht met hoge snelheid alleen bedrijfszeker gebeuren bij kabellengtes van maximaal enkele meters. Wil men informatie over grotere afstanden overbrengen, dan moet er gezocht worden naar andere methoden.



Note: T Minimum: 150μs
Maximum: Printing, carriage return, and line spacing time

Fig. 2: Signaal-niveaus en timing in de Centronics definitie

Seriële overdracht

Een van de alternatieven voor paralelle overdracht bestaat uit het achter elkaar zetten van de bits waaruit een letterteken bestaat waarna deze bits over een snoer van slechts enkele draadjes gestuurd wordt. Voor de gegevens (data) heeft men dan namelijk slechts twee draadjes nodig. Dit is een draadje voor het sigaal en een nul-draadje. Als men twee kanten uit wil communiceren, dan kan men één draadje als gemeenschappelijke nul of massa gebruiken, één draadje voor de communicatie van A naar B en één draadje voor de communicatie van B naar A. Men kan echter ook besluiten de signaaldraaden elk hun eigen massa- of nuldraad te geven. Welke mogelijkheid men kiest, is afhankelijk van de toespassing en de omgeving waar de kabel ligt. Ik kom daar straks nog op terug.

Hoe kun je nu de bits in een byte achter elkaar over één zo'n draadje sturen? Welnu, dat gaat als volgt, evenals bij de paralelle overdracht heeft een bit 1 een andere spanning (of stroomsterkte, maar daarover straks) als een bit 0. Bij de 1 is dat bijvoorbeeld een spanning van - 12 Volt en bij een 0 een spanning van + 12 Volt.

Verder is het van groot belang dat de afzender en de ontvanger met elkaar afgesproken hebben hoe lang de bij een bit behorende spanning op het draadje aanwezig is. Dit wordt de "bittijd" genoemd. Delen we één seconde door de bittijd, dan krijgen we het aantal bits per seconde of te wel de bitrate, uitgedruikt in b.p.s. Meestal wordt voor de bitrate de term baudrate gebruikt. Strikt genomen is dit onjuist. Een modem met een bitrate van 2400 b.p.s. (V22bis) heeft namelijk een baudrate van 600 baud. Het verschil tussen bitrate en baudrate zal later (in het vervolg over modems) uitgelegd worden.

Als we even uitgaan van 2400 b.p.s., dan is elk bit dus 1/2400 = 0.0004 seconde = 0.4 milliseconde (ms) op de lijn aanwezig (vergeef mij de afronding). Als de zender er nu maar voor zorgt dat er elke 0.4 ms een nieuw bit op de lijn gezet wordt en de ontvanger elke 0.4 ms even kijkt wat de spanning op de lijn is, dan zou het in theorie allemaal goed moeten gaan.

In de praktijk zijn er echter twee problemen. In de eerste plaats kan het best zo zijn dat er geen continuë stroom van bits van A naar B gaat en in de tweede plaats lopen de klokken op en A en B meestal niet gelijke snel. Als de intervallen waarmee A schrijft iets korter zijn dan de intervallen waarmee B leest, dan gaat B vroeg of laat een bit missen en dat willen we niet. In de praktijk worden er twee manieren gebruikt worden om deze problemen op te lossen.

De eerste methode is de zogenaamde synchrone communicatie waarbij de informatie in blokken van bijvoorbeeld 1000 bit opgedeeld is. Aan het begin van zo'n blok worden er extra bitpatronen meegestuurd waaraan de ontvanger kan zien dat er weer een blok aankomt en om er voor te zorgen dat de klok van de ontvanger precies gelijk gaat lopen aan de klok van de afzender. Verder worden aan het eind ook nog extra bitpatronen overgestuurd waarmee gecontroleerd kan worden of alles goed over gekomen is. Als er geen dataoverdracht is, dan is de spanning op de lijn constant en heeft meestal de waarde van een bit 1.

De tweede methode is de asynchrone communicatie. Hier wordt iedere keer een blokje van 8 bits (7 of 9 komt ook voor) met informatie overgestuurd. Om de ontvanger duidelijk te maken dat er weer wat aan komt, is afgesproken dat er voor het eerste bit een extra 0 bit gestuurd wordt. Aangezien een lijn die in rust is, het spanningsniveau van een bit 1 aanneemt, krijgen we hier dus een duidelijke spanningsval waarmee de start van een blokje data aangegeven wordt. Dit extra bit heet startbit en is altijd 0. Na de 8 data-bits wordt de lijn teruggebracht in de rusttoestand en wordt de spanning weer gelijk aan bit 1. Tussen twee brokjes data wordt dit niveau minimaal 1 bittijd (of 1.5 of 2, afhankelijk van de in-

stelling) aangehouden. Dit zijn de zogenaamde stopbits. Doordat er iedere keer maar een paar bitjes overgestuurd worden, krijgen de afzender en ontvanger, ook als hun klokken niet helemaal gelijk lopen, niet de kans ten opzichte van elkaar uit de pas te gaan lopen. Door middel van het startbit wordt de klok van de ontvanger namelijk iedere keer weer gelijk gezet aan de klok van de afzender. Het hele blokje bits dus vanaf startbit tot en met het stopbit wordt frame genoemd. In figuur 3 is schematisch weergegeven hoe asynchrone overdracht in zijn werk gaat. Bij personal en micro-computers is de asynchrone communicatie de meest gebruikte vorm van seriële dataoverdracht.

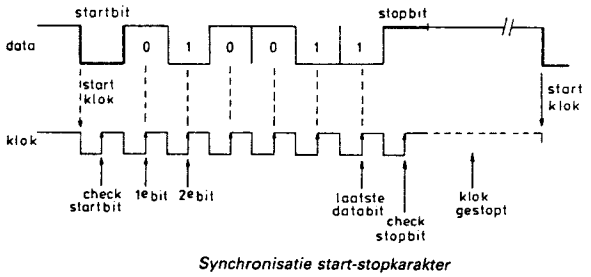


Fig. 3: Tijddiagram voor seriele communicatie

Als we van een byte de bits na elkaar versturen, dan moet afgesproken worden welk bit als eerste over de lijn verstuurd wordt. In het algemeen is dat het minst significante (laagste) bit van het byte. Het hoogste databit in een frame kan ook een speciale betekenis hebben. Hierin kan een pariteits-bit doorgegeven worden. Wordt er met pariteit gewerkt, dan kan dit bit gebruikt worden om te controleren of de databits goed over gekomen zijn. We kennen vier vormen van pariteit. Dit zijn een even (even) of oneven (odd) pariteit waarbij het hele bitpatroon een even of oneven aantal enen bevat. Verder kan men ook afspreken dat het pariteitsbit altijd 0 (space) of 1 (mark) is. Bij de eerste twee vormen van pariteit kan men aan de kant van de ontvanger detecteren of er een bit verkeerd over gekomen is. Fouten in meer dan 1 bit kunnen helaas niet altijd met één pariteitsbit geconstateerd worden. Bovendien heeft het werken met slechts één pariteitsbit het nadeel dat men een geconstateerde fout niet kan herstellen. Door het oversturen van meer controlebits dan het ene pariteitsbit kan men fouten beter constateren en vaak ook corrigeren. Bij het oversturen van enkele bytes wordt deze techniek echter niet gebruikt. Wil men een controlemogelijkheid, dan wordt er meestal in een mode gewerkt waarbij 7 databits overgestuurd worden plus één pariteitsbit. Meestal werkt men echter met 8 databits zonder pariteit.

Vormen van seriële data-overdracht

In het bovenstaande wordt uitgegaan van een zender van informatie en een ontvanger. Dit suggereert een één-richting-verkeer. Dit heet ook wel een Simplex-verbinding, er is slechts een enkelvoudige communicatie mogelijk. Over het algemeen willen we echter in twee richtingen informatie overbrengen, denk maar eens aan de communicatie met een bulletin board. Dat kan op twee manieren: Half-Duplex en Full-Duplex. Full-duplex wil zeggen dat beide partijen gelijktijdig informatie over kunnen sturen. Full-Duplex verbindingen bestaan daarom dan ook uit twee Simplex verbindingen, één van A naar B en één van B naar A. Bij een Half-Duplex verbinding kan zowel A als B als zender optreden, maar niet tegelijker tijd. Een dergelijke verbinding kan het beste vergeleken worden met mobilefoonverkeer waarin d.m.v. het woordje "OVER" aangegeven wordt dat de andere partij iets mag zenden. Na het woordje "OVER" schakelt de zender over op ontvangen en de ontvanger schakelt over op zenden. Iets dergelijks gebeurt ook bij een Half-Duplex verbinding waarbij door middel van stuursignalen dit overschakelen geregeld wordt. In het algemeen worden tegenwoordig Full-Duplex verbindingen gebruikt.

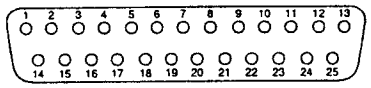
De meest gebruikte manier om serieel data over te brengen is volgens de zogenaamde V24 of RS-232 standaard. Binnen deze standdaard wordt gebruik gemaakt van kabels met 25 polige connectoren. In figuur 4 is een overzicht gegeven welke signalen binnen deze standaard gebruikt worden en op welke pennen ze aangesloten zijn. Behalve de drie zogenaamde datalijnen Transmit Data, Receive Data en Signal Ground zijn er ook nog een aantal stuursignalen gedefinieerd. Door middel van deze signalen laat de beide partijen elkaar weten of er gegevens uitgewisseld kunnen worden. Daar RS-232 ook gebruikt wordt voor modem-verbindingen, is er een signaal waarmee het modem laat weten dat hij contact heeft met een ander modem en dat de modems elkaar begrijpen. Dit is het signaal Carrier Detect. Het is niet altijd noodzakelijk alle stuursignalen te gebruiken. Vaak kan volstaan worden met slechts de signalen TD, RD en SG aan te sluiten. Hier volstaat dan dus een verbindingskabel met slechts drie draden.

RS-232 zonder tussenschakeling van modems kan men gebruiken voor afstanden van enkele tientallen meters in omgevingen waar niet al te grote storingssignalen (aan en uit schakelen van machines) aanwezig zijn.

Het nadeel van een RS-232 verbinding zonder tussenschakeling van een modem is het feit dat, tengevolge van de gemeenschappelijke massa voor zenden en ontvangen, het niet mogelijk is een zoge-

COMM Connector Pin Assignments

PIN	FUNCTION	NOTES
1	Signal Ground (SGND)	Chassis and reference ground.
2	Transmit Data (TXD)	Transmits serial data from the terminal.
3	Receive Data (RXD)	Receives serial data into the terminal.
4	Request to Send (RTS)	Asks the host to transmit.
5	Clear to Send (CTS)	Tells the terminal that the host is ready to transmit.
6	Data Set Ready (DSR)	Tells the terminal the host is in the data mode and is ready to exchange RTS, CTS and CD.
7	Signal Ground (SGND)	Chassis and reference ground.
8	Carrier Detect (CD)	Tells the terminal that the signal received on the communication line is of adequate quality to ensure proper demodulation of data. If off, indicates no signal received or signal not suited for demodulation.
20	Data Terminal Ready (DTR)	Tells the host that the terminal is ready to transmit or receive.



COMM Connector Pin Locations

Fig. 4: De signalen volgens de RS-232 standaard

naamde galvanische scheiding tussen zender en ontvanger aan te brengen. Dit houdt in, dat bij een RS-232 verbinding de massa's van de beide partijen met elkaar verbonden worden. Dit geeft, bij grotere afstanden en in een storende omgeving, een onnodig grote kans op fouten in de communicatie. Om dit op te lossen zijn er een tweetal andere mogelijkheden in gebruik, de wat verouderde current loop verbinding en de modernere RS-422 definitie.

Bij een current loop verbinding, wordt de eigenlijke verbinding gevormd door een viertal draden. Twee hiervan zijn bedoeld voor het verzenden van informatie van A naar B en twee voor het verzenden van informatie van B naar A. In tegenstelling tot de RS-232 standaard (en ook de RS-422 standaard) is het bij current loop niet zo dat de spanning op een draad bepalend is voor de waarde van een bit maar de stroom door de draad. Als er door de draad geen stroom loopt, dan wil dit zeggen dat het bijbehorende bit ook 0 is. Loopt er een stroom van zo'n 20 mA, dan is het bijbehorende bit een 1. In figuur 5 is aangegeven hoe dat in zijn werk gaat. In deze figuur is de schakeling voor een passieve ontvanger en een passieve zender weergegeven. dat wil zeggen dat de andere partij de stroomlussen van stroom voorziet.

Als we eerst het ontvangst-circuit bekijken, dan zien we dat de led in de opto coupler gaat branden als er stroom door de lus loopt. Door middel van de bijbehorende foto-transistor kan de ontvanger detecteren of de led brand (bit is 1) of dat de led uit is. De zender schakelt dus een stroombron op de lus aan als hij een bit 1 wil sturen en schakelt hem weer van de lijn af als dit bit 0 is.

Bekijken we nu de schakeling voor de zender, dan werkt deze in grote lijnen als volgt. Als de zender een bit 1 wil sturen, dan gaat de led in de opto coupler branden waardoor de bijbehorende transistor in geleiding gebracht wordt. Hierdoor wordt de spanning tussen de basis en de emissor van de 2SA952 zodanig groot dat ook deze transistor gaat geleiden waardoor de stroomlus gesloten wordt. Er loopt dan dus stroom (20 mA) door de stroomlus waardoor de ontvanger kan zien dat er een bit 1 overgestuurd wordt.

We zien verder dat er geen elektrische verbinding tussen het passieve apparaat en de stroomlus is waardoor er een galvanische scheiding tussen de apparaten bestaat. Het andere apparaat levert de stroom en is uiteraard wel elektrisch met de stroomlous verbonden. Het grote voordeel van een galvanische scheiding is dat hiermee zogenaamde aardlussen voorkomen worden. Vooral deze aardlussen zijn de oorzaak van een grote storingsgevoeligheid. Met behulp van current loop verbindingen kunnen afstanden van honderden meters tot ongeveer een kilometer zonder modems overbrugd worden.

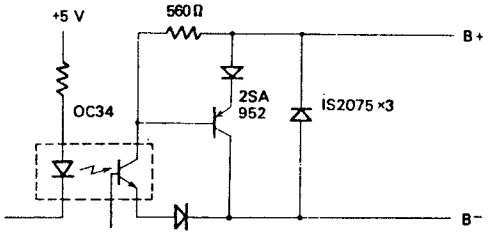
De derde manier om een seriële verbinding te leggen is de moderne RS-422 standaard. Ook hier wordt gebruik gemaakt van twee keer twee lijnen voor de data, alleen wordt hier wel weer met spanningen voor 0 en 1 gewerkt. Behalve het feit dat er voor zenden en ontvangen een aparte draad is, is er ook een vorm van galvanische scheiding aangebracht. In dit geval wordt echter niet met opto couplers gewerkt maar met verschilversterkers (opamps). Beide signaallijnen worden namelijk aangesloten op een verschilversterker waardoor alleen het zuivere verschil in spanning bekeken wordt. Aangezien storingen over het algemeen een gelijke stoorspanning op beide draden zullen geven, worden deze dus niet doorgegeven aan de ontvanger. Aangezien een opamp wel een weerstand van enkele megaOhms naar massa kan hebben, is de isolatie van beide partijen te opzichte van elkaar ook zeer groot. In de praktijk betkent dit dat ook RS-422 geschikt is voor verbindingen over grotere afstanden en in een storende omgeving.

Current Loop

Signal standards

Division	Item	Minimum	Maximum	Logic
Transmission Circuit	Mark current	15 mA	20 mA	1
	Mark voltage drop	0.8 V	1.4 V	
	Space current	0 mA	1 mA	0
Receiving circuit	Mark current	15 mA	20 mA	1
	Mark voltage drop	0 V	0.4 V	
	Space current	0 mA	1 mA	0

Transmission circuit



Receiving circuit

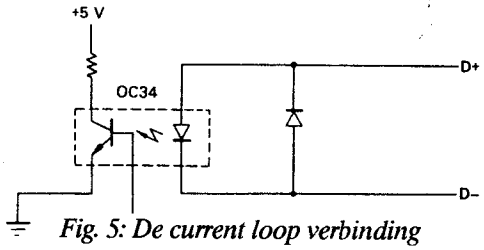


Fig. 5: De current loop verbinding

Afsluiting

In dit artikel ben ik ingegaan op enkele zaken die bij datacommunicatie naar voren komen. Ik heb geschreven over het verschil tussen een parallele overdracht van gegevens en over seriële overdracht. Verder heb ik het gehad over enkele manieren waarop in de praktijk een seriële overdracht gerealiseerd wordt. In het vervolg op dit artikel wil ik het gaan hebben over de werking van modems. In dat artikel wordt ook het verschil tussen bits per seconde en baud uitgelegd.

(Wordt vervolgd).

Gert van Opbroek

Aan de redactie

"Stereometrische figuren op de computer", mP Kenner no. 64 (december '89)

Na bestudering van het programma over ruimtelijk figuren, je weet wel waar je een 3D bril voor nodig hebt zijn er wat vragen over gebleven. Het betreft de syntax van enkele BASIC commando's. "SCREEN" in regel 960 heeft drie parameters, terwijl hetzelfde commando in regel 2530 er maar n heeft. De regel 2640: LINE (XL(0),YL(0)) - (XL(1), YL(1)),4 : LINE - (XL(2), YL(2)),4. Zou het kunnen dat de syntax als volgt is: LINE [(x_beg, y_beg)] -(x_end, y_end),color Wat doet Locate 7? Wat doet Color 7,1? De kleuren 4 & 5 zijn rood en groen, maar welke is welke?

Groeten, Geert

Balans per 31 december 1989

AKTIVA	1989	1988
Inventaria	1.800,00	4304,20
Voorraden	pm	pm
Te ontvangen posten	0,00	0,00
Geldmiddelen	16.449,58	22.199,88
TOTAAL	18.249,58 +	26.504,08 +

PASSIVA		
Vrije reserve	11.963,14	19.798,20
Vooruitontvangen contributie	4950,00	4.200,00
Te betalen posten	1336,44	2.505,88
TOTAAL	18.249,58 +	26.504,08 +

STAAT VAN BATEN EN LASTEN OVER 1989

LASTEN	1989	1988
Druk kosten 6502 kenner	10.400,89	16.288,20
Verzendkosten	0,00	0,00
Redactie kosten	371,74	414,03
Bestuur kosten	858,50	1.030,52
HCC dagen	0,00	0,00
Afschrijving inventaris	2.504,00	2.504,-
Eprom Programmer	905,06	0,00
Sysop	1.050,35	0,00
TOTAAL	16.090,54 +	20.236,77 +

BATEN		
Contributie	8550,00	11.708,00
Opbrengst advertentie	0,00	900,00
Verkoop losse 6502 kenner	8,00	40,00
Verkoop cas. en manuals	0,00	0,00
Bijeenkomsten	95,00	486,00
Rente bank en giro	443,72	454,62
Dos 65	0,00	1.519,30
Div (o.a. ohio disk)	0,00	0,00
Epromprogrammer	275,00	0,00
TOTAAL	9.371,72 +	15.107,92 +

Opgemaakt door J.H.G.M. Banser
Haaksbergerstraat 199
7513 EM Enschede

INVENTARIS *f* 1.800,00 (v.j.) *f* 4.304,20

Te specifiren als volgt:

	aanschaffings jaar	en waarde	boek-waarde t/m 1988	afschrijf waarde 1989	boek-waarde 31-12-89
Schrijfmachine	1981	1.224,00	0,00	0,00	0,00
Printer OKI-80	1981	590,00	0,00	0,00	0,00
Tractorfeed	1981	175,00	0,00	0,00	0,00
Apple	1982	4.000,00	0,00	0,00	0,00
Kim	1980	1.294,00	0,00	0,00	0,00
Junior	1981	399,00	0,00	0,00	0,00
Flipover	1980	310,00	0,00	0,00	0,00
Twee recorders	1983	394,00	0,00	0,00	0,00
Printer OKI-84	1983	2.613,00	0,00	0,00	0,00
Monitor	1983	299,00	0,00	0,00	0,00
Apple like	1985	4.000,00	1.000,00	1000,00	0,00
Daisyw printer	1987	608,40	304,20	304,20	0,00
MS-DOS computer	1987	2.400,00	1.800,00	600,00	1200,00
Modem	1988	1.800,00	1.200,00	600,00	600,00
TOTAAL		20.106,40	4.304,20	2.504,20	1.800,00

Het afschrijvings percentage bedraagt 25% van de aanschaffingswaarde.
Het modem zal in drie jaar worden afgeschreven.

GELDMIDDELEN *f* 16.449.08. (v.j.) *f* 22.199.88

Dit betreffen direkt opneembare saldi bij :

	1989	1988
Post giro	6.842,28	13.017,20
Post sterrekening	8.072,73	7.661,00
Post renterekening	215,15	208,84
Algemene bank	1.319,42	1.312,84
Totaal per 31 december	16.449,58 +	22.199,88 +

VRIJE RESERVE *f* 11963,14 (v.j.) *f* 19798,20

	1989	1988
Stand per 1 januari	19.798,20	26.157,60
Toevoeging batig saldo verdeling	-7.835,06	-6.359,40
Totaal per 31 december	11.963,14 +	19.798,20 +

TE BETALEN POSTEN: *f* 1.336,41 (v.j.) *f* 2.505,88

1988	1987	
PTT verzendkosten 6502 kenner	344,09	0,00
Drukkers kosten 6502 kenner nov	992,32	2.139,68
Zaal huur almelo	0,00	100,00
Redaktie (g.v.Opbroek)	0,00	266,20
Totaal per 31 december	1.336,41 +	2.505,88 +

Diverse specificatie

CONTRIBUTIE *f* 8.550 (v.j.) *f* 11.800

	1989	1988	1989	1988
Vooruit ontvangen per 1 jan.	99	84	4.950,00	4.200,00
ontvangen in het verenigingsjaar	171	236	8.550,00	11.800,00
	270	320	13.500,00	16.000,00
af vooruitontvangen per 31 dec.	-99	-84	-4.950,00	-4.200,00
	171	236	8.550,00	11.800,00

VOORUITONTVANGEN CONTRIBUTIE *f* 4.950,00 (v.j.) *f* 4.200,00

In het jaar 1989 werd van 99 leden de contributie voor 1990 vooruitontvangen.
In 1988 bedroeg dit aantal 84 voor het verenigingsjaar 1989.

VERKOOP CASSETTES EN MANUALS *f* 0,00 (v.j.) *f* 0,00

DRUKKOSTEN μP KENNER *f* 10.400,89 (v.j.) *f* 16.288,22

In 1989 werden vijf edities verzonden.

REDAKTIE KOSTEN *f* 371,74 (v.j.) *f* 414,03

	1989	1988
porto kosten	9,00	47,00
reis kosten	0,00	54,00
telefoon kosten	0,00	134,43
materiaal	187,74	78,60
div	175,00	0,00
TOTAAL	371,74 +	414,03 +

OSTEN *f* 858,50 (v.j.) *f* 1.030,52

	1989	1988
porto kosten	406,50	487,50
bestuurs kosten	168,75	76,70
reis kosten	120,00	80,00
telefoon kosten	0,00	0,00
kamer v koophandel	71,50	60,00
materiaal	91,75	326,32
totaal	858,50	1.030,52

DOS65 *f* -630,06 (v.j.) *f* 1.519,30

	1989	1988
printen dos65	0,00	300,00
eproms	0,00	300,00
porto kosten	0,00	-139,40
materiaal	0,00	-190,93
manuals	0,00	555,00
dos65 div	0,00	694,63
Eprom programmer	-630,06	
TOTAAL	*f* -630,06 +	*f* 1.519,30 +

Er zijn nog verschillende printen in voorraad maar die worden niet opgenomen in de balans.
Dit zijn: Diverse Floppy-disk controllers voor de Dos65 en de Eprom-programmer printen.

Van de layout computer

Naar aanleiding van de gewijzigde lay-out van de μP Kenner is mij gevraagd een aantal regels op te stellen waar de aangeleverde teksten aan moeten voldoen om voor plaatsing in aanmerking te komen. De grondregel waar alle verderop genoemde regels in feite op neer komen, luidt: lever alle tekst zo schoon mogelijk aan. Wat dat precies inhoud zal ik hieronder puntsgewijs toelichten.

- In principe kunnen alleen teksten in WordPerfect formaat (versie 4.2) of schone ASCII verwerkt worden. Het lay-out programma kan veel meer formaten aan, maar wij beschikken niet over al die verschillende tekstverwerkers. Gebruikt u WordPerfect 5.0, voer de teksten dan toch uit in 4.2-formaat!

- Gebruik geen vet, onderstrepen of andere "opmaak" codes in uw teksten. Het is een bijzonder tijdrovend en vervelend karwei ze er allemaal weer uit te moeten slopen. Accenten en trema's zijn wel toegestaan, ook in ASCII (IBM character set!).

- De harde kern in de redactie heeft al eens beweerd dat auteurs, die spaties gebruiken om teksten uit te lijnen, eigenlijk voor een speciaal tribunaal zouden moeten verschijnen. Dit tribunaal kan volstaan met een wetboek dat op één pagina A5 past. Er staat dan ook maar één straf op: de doodstraf. Gebruik ook geen spaties om tabellen en/of listings uit te lijnen! Als u een PC bezit, dan heeft u daarvoor een speciale toets die iets groter uitgevallen is dan de meeste: de TAB-toets (Hmm... wetenschappelijk verantwoord onderzoek heeft aangetoond dat zelfs de ZX Spectrum speelcomputer een TAB toets heeft).

- Een harde return geeft in het opmaak pakket het einde van een alinea aan. Het pakket zet er automatisch een extra grote witruimte achter. Gebruik dus geen twee harde returns achter elkaar en zeker geen harde return, gevolgd door een spatie en nog een harde return (daar strandt onze super-de-luxe sloopmacro ook op).

- Lever listings gaarne los aan. Neem ze vooral niet op in uw tekst! Wilt u aangeven waar ze in de tekst ongeveer geplaatst zouden moeten worden, dan kunt u een commentaar voor de layouter opnemen. Commentaar graag vet en/of onderstrepen (het is de ENIGE keer dat dat mag!!!!).

- Assembler listings zijn vervelende dingen om te layouten. Maak het ons niet moeilijker dan het al is en lever de ORIGINELE SOURCE files aan! Gebruik tabs om een veld uit te lijnen. Dus: een tab na (of in plaats van) een label, na de opdracht, en eentje na de operand(s).

- Gebruik geen afbreekstreepjes en zet de afbreekfunctie van uw tekstverwerker UIT. Breek zeker niet iets met de hand af (ook hiervoor is een speciaal tribunaal in het leven geroepen).

- Tenslotte horen er in een "schone" tekst ook geen kolommen thuis, geen aanwijzingen over de breedte van het papier etc.

Illustraties kunnen uw tekst(en) op aardige wijze verlevendigen. Als u illustraties hebt in een goede kwaliteit (foto's, schilderijen of bankbiljetten), kunt u het beste de illustratie zelf opsturen. De drukker kan de illustraties dan in de hoogst haalbare kwaliteit verwerken. Er zijn ook een aantal elektronische formaten die ingelezen kunnen worden in het DTP-pakket. Een volledige lijst hiervan kunt u opvragen bij de redactie. Maakt u schema's in bijv. OrCad, lever deze dan in HPGL formaat aan! Tekeningen kunt u in dat formaat aanmaken door de HP-plotter driver te installeren en vervolgens met het bijgeleverde programma "PlotAll" een plot-file te genereren. Lever in geen geval de originele schema-files aan, daar kunnen we NIETS mee beginnen.

Indien mogelijk zullen we altijd proberen ergens een oplossing te vinden voor kopij in afwijkend formaat. Ook de redactie kan echter geen wonderen verrichten (al lijkt het er af en toe wel op...). Voor kopij die niet in het juiste formaat aangeleverd is kan geen enkele garantie gegeven worden dat de betreffende kopij ook geplaatst wordt!

Awel, dat was het wel zo'n beetje. Voor vragen kunt u altijd (liefst schriftelijk) contact opnemen met de redactie van uw lijfblad. Suggesties aangaande de layout en/of inhoud van de μP Kenner zijn altijd welkom.

Joost Voorhaar.

Een speciaal club-BIOS voor XT's

Ondergetekende heeft al een paar jaar een BIOS voor XT-compatibelen in het pakket. Dat BIOS is inmiddels uitgegroeid tot een zeer goed doordacht produkt waar nagenoeg alle foutjes en bugs uit zijn verdwenen.

Een aantal mensen opperde de mogelijkheid om het BIOS aan de leden van de club aan te bieden tegen een bescheiden prijs. Dat kan natuurlijk. Het BIOS is inmiddels voorzien van de naam van de club. Nu wilt u natuurlijk weten wat er zo bijzonder aan dat BIOS is, want in uw PC zit waarschijnlijk al een goed BIOS van een gerenommeerde fabrikant, zoals IBM, AMI, Phoenix, Award of DTK. Het KIM Club-BIOS biedt echter meer. Kijk maar:

- Een snelle, maar niet minder grondige RAM-test
- Automatische herkenning van COM3: en COM4:
- Gedetailleerd rapport van de machine-configuratie
- Floppy disk BIOS met versnelde toegang: 21 seconden winst bij het formatteren van een 360kbyte floppy
- Volledig getest met een V20 CPU en clocksnelheden tot 10 MHz
- Compatibiliteit staat inmiddels volledig buiten kijf
- Automatische herkenning en ondersteuning van 3.5 inch 720k drives

Dit laatste is vrijwel een unicum: er hoeft niet meer in de CONFIG.SYS geknoeid te worden met DRIVPARM's of DRIVER.SYSsen, en er hoeft geen speciaal DOS voor gebruikt te worden: gewoon drives aansluiten en booten, klaar.

Het BIOS is geschikt voor vrijwel iedere compatibele XT-achtige machine die een 8 kbyte BIOS in EPROM heeft. Het is niet geschikt voor de pseudo compatibelen, waaronder helaas alle Philips PC's, vrijwel alle Tulips en de Vendex Headstart machines vallen. Deze machines zijn namelijk wel op DOS-niveau redelijk compatibel maar niet op BIOS-niveau.

Het BIOS niet een gejat BIOS van een ander met een veranderde naam. Het wordt vàn het begin geassembleerd en bevat uitsluitend nette code. Van oorsprong is het een zeer slecht uit Taiwan of Korea afkomstig BIOS zonder merknaam of copyright statement, dat wemelde van de spelfouten in de meldingen. Door de jaren heen is er zoveel aan het BIOS veranderd dat er gerust van een eigen produkt gesproken kan worden. De meest drastische wijzigingen hebben zich in het floppy disk gedeelte (3.5 inch support, snelheidswinst) en de power-on self-test (geheel opnieuw geschreven) voorgedaan. De source file bevat op vrijwel iedere regel commentaar en is 180 kbyte groot. Het BIOS is geschreven in MASM 4.0.

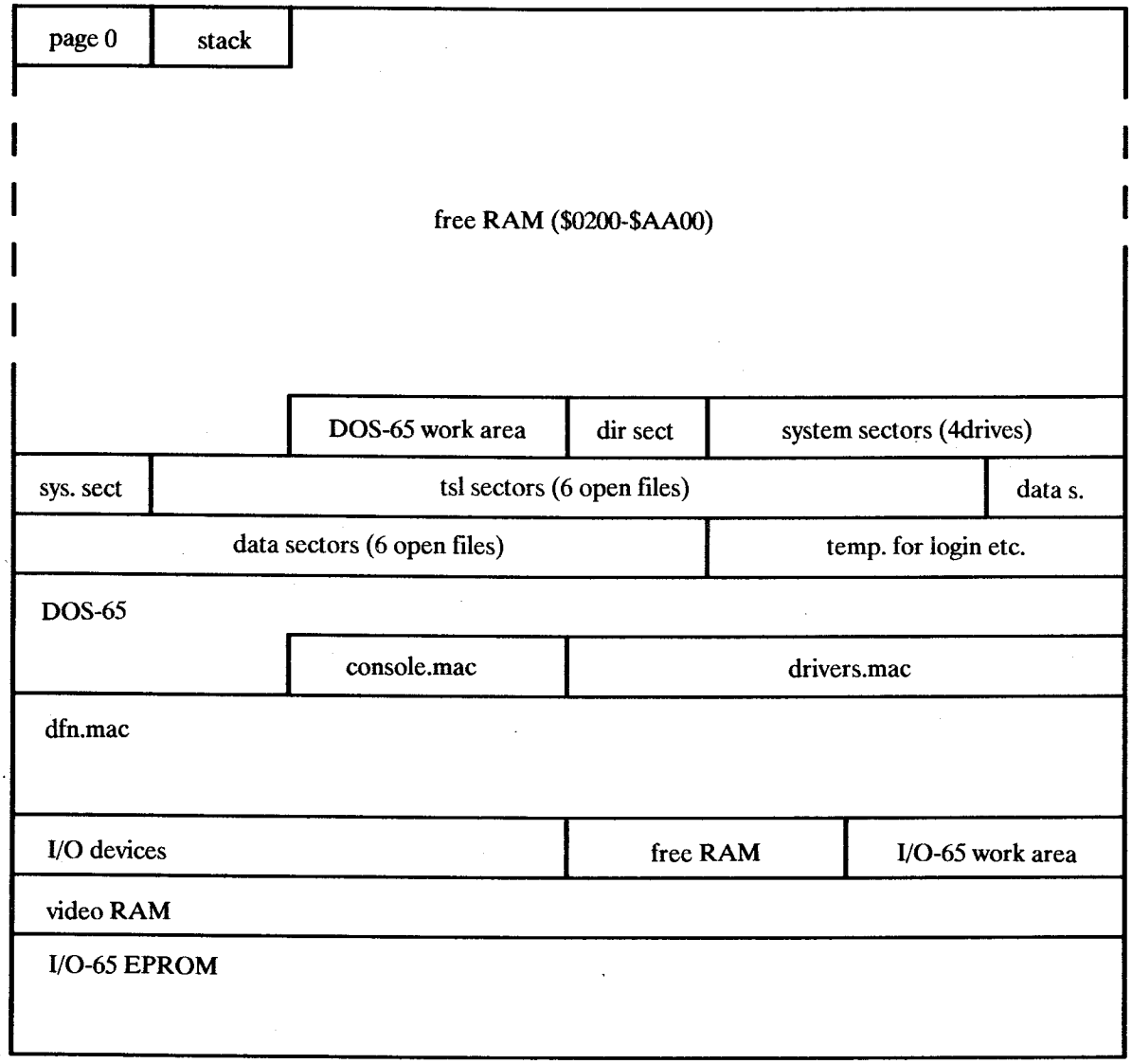
U kunt het BIOS bestellen door *f* 25,- over te maken aan de penningmeester onder vermelding van 'KIM XT-BIOS'. U ontvangt dan een geprogrammeerde 2764 EPROM en een Engelstalige handleiding over de installatie en de eigenschappen van BIOS. Voor de ware freaks is er ook nog een assembler-listing te bestellen. De listing is ruim 120 pagina's dik en geeft een gedetailleerde kijk in de BIOS-keuken. De prijs van de listing moet nog nader bepaald worden: dit hangt mede af van de belangstelling ervoor.

Nico de Vries

Dos65 memory map $0000-$FFFF

Op verzoek hierbij een grafische voorstelling van de geheugenindeling in een dos65 computer. Voor een fijnere onderverdeling kun je terecht bij de diverse artikelen en listings die bij het Dos65 pakket geleverd worden. Zo is er een listing van alle I/O65 entries, de Dos65 entries, en een goede beschrijving van wat je verwachten kunt van de Dos65 en I/O65 subroutines. Iedere regel representeert 2 kbyte.

Ernst Elderenbosch



De meeste stukken geheugen zullen wel bekend voorkomen maar hier een korte uitleg van de 'moeilijke gevallen':

directory sector	momenteel gebruikte drive directory sector.
system sectors	kopie van de system sectors (sect.0) van de max. vier drives (floppy, winchester of virtual).
tsl sectors	kopie van de track sector list (file info) van de max. zes tegelijk geopende files.
data sectors	kopie van de momenteel gebruikte datasectors van deze max. zes geopende files.
temp. for login	dit gebied wordt tijdelijk gebruikt voor het booten.
dos65	bevat de altijd in Dos65 aanwezige commando's, zoals de redirection commando's, dir, go, load, etc...
console	bevat de terminal routines.
drivers	de drivers voor floppy, ramdisk, winchester.
dfn	disk functies open, read, write, seek etc...

Van de voorzitter

Het vorige jaar werd afgesloten met een verrassing: een blad dat geheel met Desktop Publishing op een PC werd gemaakt. Daar was al wat van uitgelekt op de ledenvergadering in Almelo, maar dat het zo mooi en prettig leesbaar zou worden had ik niet verwacht. Verantwoordelijk voor deze vooruitgang was Joost Voorhaar, geassisteerd door onze penningmeester Jacques Banser. Beiden gefeliciteerd met het resultaat!

Joost blijkt echter heel wat in zijn mars te hebben: hij heeft op de afgelopen bijeenkomst in Krommenie een uiterst verhelderend verhaal gehouden over Public Relations, en wat dat inhoudt en zou moeten inhouden voor onze vereniging. Er kwamen behoorlijk wat ideeën over de tafel, en onze P.R.-man Geert Stappers heeft er aardig wat van opgeschreven.

Zo stelde Joost voor om het blad nog fraaier te maken (kan dat nog dan?) en van een meer functionele cover te voorzien. Of dat nu al gebeurd is, heeft u inmiddels al gezien. Ook ideeën over hoe wij als vereniging nieuwe leden zouden kunnen aantrekken werden geopperd. Want dat is toch wel dringend nodig willen we als vereniging blijven bestaan.

Een ander leuk idee van Joost was het beginnen van een brievenrubriek in het blad. Schrijf uw probleempjes op computergebied eens op, en gooi het in de groep. Met z'n allen weten we meer dan u en ik alleen. Er gaat wellicht nog heel wat veranderen, zowel in het blad als in de vereniging. We zullen daarbij zeker in de gaten houden wat de doelstellingen van onze club zijn.

Even iets anders: uw voorzitter kijkt sinds een paar weken gekleurd naar z'n computer. Grafisch is het mooi maar ik moet nog steeds wennen aan al die kleurtjes bij het tekstverwerken. Ook is het behelpen met een EGA kaart: hij onderstreept niet op het scherm. Maar toch vind ik het een vooruitgang, temeer omdat de kleurenbuis een zogenaamde multisync is: je stopt er een redelijk willekeurig video signaal in en je hebt een plaatje, of dat nu CGA, MDA, Hercules, EGA, VGA of hi-res EGA is. Handig hoor. Nog veel lees- en mogelijk overtikplezier gewenst, en tot de volgende keer maar weer (in Geldrop dus).

Nico de Vries

Binnenkort in de µP Kenner

In de µP Kenner kunt u binnenkort onder andere de volgende artikelen aantreffen:

De aankondiging van software voor DOS-65 waarmee u op een floppy het formaat van MS-DOS kunt lezen en schrijven. Met behulp van deze software is uitwisseling van gegevens met een MS-DOS computer (en met de redactie!) een fluitje van een cent.

Een programmaatje voor MS-DOS die automatisch! tot maximaal 4 seriële poorten herkent en installeert. U kunt dus nu gebruik maken van COM1 t/m COM4.

Een stukje hardware (inclusief print) waarmee op DOS-65 de SID 6581 (bekend van de Commodore 64) aangesloten kan worden.

Een artikel dat beschrijft hoe modems werken en hoe we in plaats van de 300 bit per seconde van een paar jaar geleden nu tot 9600 bits per seconde (en hoger) over een telefoonleiding kunnen sturen.

Alweer een aflevering in de MS-DOS serie van Nico de Vries.

Vraag: wie schrijft een serie over de Atari ST of de Amiga zoals Nico dat voor over IBM en zijn klonen doet? Als u dit wilt gaan doen, dan kunt u contact opnemen met de redactie.

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 beant- woorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board beschikbaar gesteld.

Het telefoonnummer is: 053-303902.

Software Bibliotheek en Technisch Forum:

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

Het Bestuur:

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

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

Mick Agterberg (secretaris)
Davidvosstraat 29
1063 HV Amterdam
Telefoon 020-131538

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

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

Jan D.J. Derksen
Ed Verkadestraat 9-1
7558 TH Hengelo
Telefoon 074-770970

Geert Stappers
Engelseweg 7
5825 BT Overloon
Telefoon 04788-1279

Ton Smits
De Meren 39
4731 WB Oudebosch

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 Mueller